

Olena Reference Manual

0.10

Generated by Doxygen 1.3.6-20040222

Thu Apr 15 20:13:04 2004

Contents

Chapter 1

Olena Namespace Index

1.1 Olena Namespace List

Here is a list of all documented namespaces with brief descriptions:

internal (Abstract internal)	??
io (Io namespace)	??
oln (Oln namespace)	??
oln::abstract (Oln::abstract namespace)	??
oln::arith (Arithmetic implementation)	??
oln::convert (Conversion implementation (for example cast, color, or neighborhood to window))	??
oln::convert::abstract (Base classes for conversion)	??
oln::convert::abstract::internal (Internal purpose only)	??
oln::convert::internal (Internal purpose only)	??
oln::convol (Algorithms related to convolution)	??
oln::convol::fast (Implementation of algorithms for large structuring elements)	??
oln::convol::fast::internal (Internal purpose only)	??
oln::convol::slow (Convolution algorithms)	??
oln::impl (Representation of the image in memory)	??
oln::internal (Internal purpose only)	??
oln::io::gz (Functions for gz files)	??
oln::level (Level algorithm implementation)	??
oln::math (Useful functions)	??
oln::math::internal (Internal purpose only)	??
oln::morpher (Contain all the morpher relative declarations and functions)	??
oln::morpher::abstract (Implementation of oln::abstract::generic_morpher)	??
oln::morpho (Algorithm based on morphological mathematic)	??
oln::morpho::attr (Implementation of attributes)	??
oln::morpho::attr::tools (Useful tools for morphological math)	??
oln::morpho::env (Implementation of environments used by attributes)	??
oln::morpho::env::abstract	??
oln::morpho::fast (Algorithm enhanced for large structuring elements)	??
oln::morpho::fast::tarjan (Oln::morpho::tarjan implementation)	??
oln::morpho::fast::tarjan::internal (Internal purpose only)	??
oln::morpho::hybrid (Hybrid (sure and sequential) algorithm implementation)	??
oln::morpho::internal (Internal purpose only)	??
oln::morpho::sequential (Sequential algorithm implementation)	??

oln::morpho::slow (Algorithm that are slow (but need less memory), or that are slow if it is used with a large structuring element)	??
oln::morpho::sure (Reference algorithm implementation: sure but slow)	??
oln::snakes (Oln::snakes::snake implementation)	??
oln::topo (Topological algorithms)	??
oln::topo::combinatorial_map (Namespace for combinatorial map)	??
oln::topo::combinatorial_map::internal (Oln::combinatorial_map::internal)	??
oln::topo::inter_pixel (Oln::topo::inter_pixel::interpixel implementation)	??
oln::topo::inter_pixel::internal (Internal purpose only)	??
oln::topo::tarjan (Implementation of tarjan set)	??
oln::topo::tarjan::abstract (Abstract classes for tarjan based algorithms)	??
oln::topo::tarjan::obsolete (Namespace containing obsolete implementation of flat zone)	??
oln::transforms (Transform algorithm implementation)	??
oln::utils (Utilities, such as statistics)	??
oln::utils::abstract (Abstract utilities)	??
oln::utils::internal (Internal purpose only)	??

Chapter 2

Olena Hierarchical Index

2.1 Olena Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

```
oln::internal::_fake
ntg::internal::_from_float< n, ncomps, qbits, color_system > . . . . . ??
ntg::internal::_from_float< ncomps, ncomps, qbits, color_system >
ntg::cast::internal::_round< Tdest, Tsrc >
ntg::cast::internal::_round< float_d, float_s >
ntg::cast::internal::_round< float_d, float_value< Tsrc > >
ntg::cast::internal::_round< float_s, float_d >
ntg::cast::internal::_round< float_s, float_value< Tsrc > >
ntg::cast::internal::_round< float_value< Tdest >, float_value< Tsrc > >
ntg::cast::internal::_round< Tdest, double >
ntg::cast::internal::_round< Tdest, float_s >
ntg::cast::internal::_round< Tdest, float_value< Tsrc > >
ntg::internal::_to_float< n, ncomps, qbits, color_system > . . . . . ??
ntg::internal::_to_float< ncomps, ncomps, qbits, color_system >
ntg::builtin::abstract_trait< T >
ntg::builtin::abstract_trait< char >
ntg::builtin::abstract_trait< signed char >
ntg::builtin::abstract_trait< signed int >
ntg::builtin::abstract_trait< signed long >
ntg::builtin::abstract_trait< signed short >
all_double_traits
    ntg::internal::operator_traits< operator_cmp, double, double >
all_double_traits
    ntg::internal::operator_traits< operator_minus, double, float >
all_double_traits
    ntg::internal::operator_traits< operator_times, double, double >
all_double_traits
    ntg::internal::operator_traits< operator_plus, double, double >
all_double_traits
    ntg::internal::operator_traits< operator_max, double, double >
all_double_traits
    ntg::internal::operator_traits< operator_plus, double, int_u< nbits, B > >
all_double_traits
```

```

    ntg::internal::operator_traits< operator_max, double, int_u< nbits, B > >
all_double_traits
    ntg::internal::operator_traits< operator_div, double, float >
all_double_traits
    ntg::internal::operator_traits< operator_div, double, int_s< nbits, B > >
all_double_traits
    ntg::internal::operator_traits< operator_min, double, int_u< nbits, B > >
all_double_traits
    ntg::internal::operator_traits< operator_cmp, double, int_s< nbits, B > >
all_double_traits
    ntg::internal::operator_traits< operator_times, double, int_s< nbits, B > >
all_double_traits
    ntg::internal::operator_traits< operator_cmp, double, float >
ntg::internal::all_double_traits
all_double_traits
    ntg::internal::operator_traits< operator_minus, double, int_s< nbits, B > >
all_double_traits
    ntg::internal::operator_traits< operator_min, double, float >
all_double_traits
    ntg::internal::operator_traits< operator_minus, double, double >
all_double_traits
    ntg::internal::operator_traits< operator_times, double, float >
all_double_traits
    ntg::internal::operator_traits< operator_max, double, float >
all_double_traits
    ntg::internal::operator_traits< operator_max, double, int_s< nbits, B > >
all_double_traits
    ntg::internal::operator_traits< operator_plus, double, int_s< nbits, B > >
all_double_traits
    ntg::internal::operator_traits< operator_plus, double, float >
all_double_traits
    ntg::internal::operator_traits< operator_div, double, int_u< nbits, B > >
all_double_traits
    ntg::internal::operator_traits< operator_min, double, int_s< nbits, B > >
all_double_traits
    ntg::internal::operator_traits< operator_cmp, double, int_u< nbits, B > >
all_double_traits
    ntg::internal::operator_traits< operator_times, double, int_u< nbits, B > >
all_double_traits
    ntg::internal::operator_traits< operator_div, double, double >
all_double_traits
    ntg::internal::operator_traits< operator_min, double, double >
all_double_traits
    ntg::internal::operator_traits< operator_minus, double, int_u< nbits, B > >
ntg::internal::all_float_traits
all_float_traits
    ntg::internal::operator_traits< operator_div, float, int_s< nbits, B > >
all_float_traits
    ntg::internal::operator_traits< operator_max, float, int_u< nbits, B > >
all_float_traits
    ntg::internal::operator_traits< operator_minus, float, float >
all_float_traits
    ntg::internal::operator_traits< operator_times, float, int_u< nbits, B > >
all_float_traits

```

ntg::internal::operator_traits< operator_times, float, float >	
all_float_traits	
ntg::internal::operator_traits< operator_min, float, int_s< nbits, B > >	
all_float_traits	
ntg::internal::operator_traits< operator_div, float, float >	
all_float_traits	
ntg::internal::operator_traits< operator_minus, float, int_s< nbits, B > >	
all_float_traits	
ntg::internal::operator_traits< operator_cmp, float, int_u< nbits, B > >	
all_float_traits	
ntg::internal::operator_traits< operator_max, float, int_s< nbits, B > >	
all_float_traits	
ntg::internal::operator_traits< operator_plus, float, int_u< nbits, B > >	
all_float_traits	
ntg::internal::operator_traits< operator_div, float, int_u< nbits, B > >	
all_float_traits	
ntg::internal::operator_traits< operator_cmp, float, float >	
all_float_traits	
ntg::internal::operator_traits< operator_times, float, int_s< nbits, B > >	
all_float_traits	
ntg::internal::operator_traits< operator_min, float, float >	
all_float_traits	
ntg::internal::operator_traits< operator_min, float, int_u< nbits, B > >	
all_float_traits	
ntg::internal::operator_traits< operator_plus, float, float >	
all_float_traits	
ntg::internal::operator_traits< operator_minus, float, int_u< nbits, B > >	
all_float_traits	
ntg::internal::operator_traits< operator_cmp, float, int_s< nbits, B > >	
all_float_traits	
ntg::internal::operator_traits< operator_max, float, float >	
all_float_traits	
ntg::internal::operator_traits< operator_plus, float, int_s< nbits, B > >	
oln::topo::combinatorial_map::internal::alpha< U >	??
ntg::any< E >	??
ntg::internal::any_ntg_< E >	??
ntg::any_ntg< E >	??
ntg::value< E >	??
ntg::enum_value	
ntg::real_value	
ntg::vect_value	
any	
oln::abstract::point< point1d >	??
oln::point1d	??
any	
oln::abstract::point< point2d >	??
oln::point2d	??
any	
oln::abstract::point< point3d >	??
oln::point3d	??
any	
oln::abstract::neighborhood< neighborhood3d >	??
oln::abstract::window_base< neighborhood< neighborhood3d >, neighborhood3d > . .	??

	oln::abstract::neighborhoodnd< neighborhood3d >	??
	oln::neighborhood3d	??
any		
	oln::abstract::neighborhood< neighborhood2d >	??
	oln::abstract::window_base< neighborhood< neighborhood2d >, neighborhood2d > . .	??
	oln::abstract::neighborhoodnd< neighborhood2d >	??
	oln::neighborhood2d	??
any		
	oln::abstract::neighborhood< neighborhood1d >	??
	oln::abstract::window_base< neighborhood< neighborhood1d >, neighborhood1d > . .	??
	oln::abstract::neighborhoodnd< neighborhood1d >	??
	oln::neighborhood1d	??
any		
	oln::convert::abstract::conversion< mlc::exact_vt< conversion_to_type< ntg::color< ocomps, oqbits, ocolor >, mlc::exact_vt< conversion_from_type_to_type< ntg::color< icomps, iqbits, icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::final, mlc::exact_vt< color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_xyz_to_nrgb< inbits, outbits > >, f_xyz_to_nrgb< inbits, outbits > >::ret >, mlc::exact_vt< color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_xyz_to_nrgb< inbits, outbits > >, f_xyz_to_nrgb< inbits, outbits > >::ret >::ret >, mlc::exact_vt< conversion_from_type_to_type< ntg::color< icomps, iqbits, icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::final, mlc::exact_vt< color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_xyz_to_nrgb< inbits, outbits > >, f_xyz_to_nrgb< inbits, outbits > >::ret >, mlc::exact_vt< color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_xyz_to_nrgb< inbits, outbits > >, f_xyz_to_nrgb< inbits, outbits > >::ret >, mlc::exact_vt< conversion_to_type< ntg::color< ocomps, oqbits, ocolor >, mlc::exact_vt< conversion_from_type_to_type< ntg::color< icomps, iqbits, icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::final, mlc::exact_vt< color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_xyz_to_nrgb< inbits, outbits > >, f_xyz_to_nrgb< inbits, outbits > >::ret >, mlc::exact_vt< color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_xyz_to_nrgb< inbits, outbits > >, f_xyz_to_nrgb< inbits, outbits > >::ret >, mlc::exact_vt< conversion_from_type_to_type< ntg::color< icomps, iqbits, icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::final, mlc::exact_vt< color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_xyz_to_nrgb< inbits, outbits > >, f_xyz_to_nrgb< inbits, outbits > >::ret >, mlc::final >::ret >::ret >	??
any		

Generated on Thu Apr 15 20:17:29 2004 for Olena by Doxygen


```

oln::convert::abstract::conversion< mlc::exact_vt< conversion_to_type< Output,
mlc::exact_vt< bound< Output, Exact >, Exact >::ret >, mlc::exact_vt<
bound< Output, Exact >, Exact >::ret >::ret, mlc::exact_vt< conversion_to_
type< Output, mlc::exact_vt< bound< Output, Exact >, Exact >::ret >, mlc::final
>::ret > . . . . . ??
any

```

```

oln::convert::abstract::conversion< mlc::exact_vt< conversion_to_type< ntg::color<
ocomps, oqbits, ocolor >, mlc::exact_vt< conversion_from_type_to_type<
ntg::color< icomps, iqbits, icolor >, ntg::color< ocomps, oqbits, ocolor >,
mlc::final, mlc::exact_vt< color_conversion< icomps, iqbits, icolor, ocomps,
oqbits, ocolor, f_rgb_to_xyz< inbits, outbits > >, f_rgb_to_xyz< inbits, outbits
> >::ret >, mlc::exact_vt< color_conversion< icomps, iqbits, icolor, ocomps,
oqbits, ocolor, f_rgb_to_xyz< inbits, outbits > >, f_rgb_to_xyz< inbits, outbits
> >::ret >::ret >, mlc::exact_vt< conversion_from_type_to_type< ntg::color<
icomps, iqbits, icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::final,
mlc::exact_vt< color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor,
f_rgb_to_xyz< inbits, outbits > >, f_rgb_to_xyz< inbits, outbits > >::ret
>, mlc::exact_vt< color_conversion< icomps, iqbits, icolor, ocomps, oqbits,
ocolor, f_rgb_to_xyz< inbits, outbits > >, f_rgb_to_xyz< inbits, outbits >
>::ret >::ret >::ret, mlc::exact_vt< conversion_to_type< ntg::color< ocomps,
oqbits, ocolor >, mlc::exact_vt< conversion_from_type_to_type< ntg::color<
icomps, iqbits, icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::final,
mlc::exact_vt< color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor,
f_rgb_to_xyz< inbits, outbits > >, f_rgb_to_xyz< inbits, outbits > >::ret >,
mlc::exact_vt< color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor,
f_rgb_to_xyz< inbits, outbits > >, f_rgb_to_xyz< inbits, outbits > >::ret >::ret
>, mlc::exact_vt< conversion_from_type_to_type< ntg::color< icomps, iqbits,
icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::final, mlc::exact_vt< color_
conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_rgb_to_xyz< inbits,
outbits > >, f_rgb_to_xyz< inbits, outbits > >::ret >, mlc::final >::ret >::ret > . ??
any

```

```

oln::convert::abstract::conversion< mlc::exact_vt< conversion_to_type< Result_Type,
mlc::exact_vt< conversion_from_type_to_type< Argument_Type, Result_Type,
Base, Exact >, Exact >::ret >, mlc::exact_vt< conversion_from_type_to_type<
Argument_Type, Result_Type, Base, Exact >, Exact >::ret >::ret, mlc::exact_vt<
conversion_to_type< Result_Type, mlc::exact_vt< conversion_from_type_to_
type< Argument_Type, Result_Type, Base, Exact >, Exact >::ret >, mlc::exact_
vt< conversion_from_type_to_type< Argument_Type, Result_Type, Base, Exact
>, Base >::ret >::ret > . . . . . ??
any

```

```

oln::convert::abstract::conversion<  mlc::exact_vt<  conversion_to_type<  ntg::color<
  ocomps, oqbits, ocolor >, mlc::exact_vt<  conversion_from_type_to_type<
  ntg::color<  icomps, iqbits, icolor >, ntg::color<  ocomps, oqbits, ocolor >,
  mlc::final, mlc::exact_vt<  color_conversion<  icomps, iqbits, icolor, ocomps,
  oqbits, ocolor, f_yuv_to_rgb<  inbits, outbits > >, f_yuv_to_rgb<  inbits, outbits
  > >::ret >, mlc::exact_vt<  color_conversion<  icomps, iqbits, icolor, ocomps,
  oqbits, ocolor, f_yuv_to_rgb<  inbits, outbits > >, f_yuv_to_rgb<  inbits, outbits
  > >::ret >::ret >, mlc::exact_vt<  conversion_from_type_to_type<  ntg::color<
  icomps, iqbits, icolor >, ntg::color<  ocomps, oqbits, ocolor >, mlc::final,
  mlc::exact_vt<  color_conversion<  icomps, iqbits, icolor, ocomps, oqbits, ocolor,
  f_yuv_to_rgb<  inbits, outbits > >, f_yuv_to_rgb<  inbits, outbits > >::ret
  >, mlc::exact_vt<  color_conversion<  icomps, iqbits, icolor, ocomps, oqbits,
  ocolor, f_yuv_to_rgb<  inbits, outbits > >, f_yuv_to_rgb<  inbits, outbits >
  >::ret >::ret >::ret, mlc::exact_vt<  conversion_to_type<  ntg::color<  ocomps,
  oqbits, ocolor >, mlc::exact_vt<  conversion_from_type_to_type<  ntg::color<
  icomps, iqbits, icolor >, ntg::color<  ocomps, oqbits, ocolor >, mlc::final,
  mlc::exact_vt<  color_conversion<  icomps, iqbits, icolor, ocomps, oqbits, ocolor,
  f_yuv_to_rgb<  inbits, outbits > >, f_yuv_to_rgb<  inbits, outbits > >::ret >,
  mlc::exact_vt<  color_conversion<  icomps, iqbits, icolor, ocomps, oqbits, ocolor,
  f_yuv_to_rgb<  inbits, outbits > >, f_yuv_to_rgb<  inbits, outbits > >::ret
  >::ret >, mlc::exact_vt<  conversion_from_type_to_type<  ntg::color<  icomps,
  iqbits, icolor >, ntg::color<  ocomps, oqbits, ocolor >, mlc::final, mlc::exact_vt<
  color_conversion<  icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_yuv_to_rgb<
  inbits, outbits > >, f_yuv_to_rgb<  inbits, outbits > >::ret >, mlc::final >::ret
  >::ret > . . . . .

```

any

??

```

oln::convert::abstract::conversion<  mlc::exact_vt<  conversion_to_type<  Output,
  mlc::exact_vt<  stretch<  Output, Exact >, Exact >::ret >, mlc::exact_vt<
  stretch<  Output, Exact >, Exact >::ret >::ret, mlc::exact_vt<  conversion_to_-
  type<  Output, mlc::exact_vt<  stretch<  Output, Exact >, Exact >::ret >, mlc::final
  >::ret > . . . . .

```

any

??


```

oln::convert::abstract::conversion<  mlc::exact_vt<  conversion_to_type<  ntg::color<
  ocomps, oqbits, ocolor >, mlc::exact_vt<  conversion_from_type_to_type<
  ntg::color<  icomps, iqbits, icolor >, ntg::color<  ocomps, oqbits, ocolor >,
  mlc::final, mlc::exact_vt<  color_conversion<  icomps, iqbits, icolor, ocomps,
  oqbits, ocolor, f_rgb_to_hsl<  inbits, outbits > >, f_rgb_to_hsl<  inbits, outbits
  > >::ret >, mlc::exact_vt<  color_conversion<  icomps, iqbits, icolor, ocomps,
  oqbits, ocolor, f_rgb_to_hsl<  inbits, outbits > >, f_rgb_to_hsl<  inbits, outbits
  > >::ret >::ret >, mlc::exact_vt<  conversion_from_type_to_type<  ntg::color<
  icomps, iqbits, icolor >, ntg::color<  ocomps, oqbits, ocolor >, mlc::final,
  mlc::exact_vt<  color_conversion<  icomps, iqbits, icolor, ocomps, oqbits, ocolor,
  f_rgb_to_hsl<  inbits, outbits > >, f_rgb_to_hsl<  inbits, outbits > >::ret >,
  mlc::exact_vt<  color_conversion<  icomps, iqbits, icolor, ocomps, oqbits, ocolor,
  f_rgb_to_hsl<  inbits, outbits > >, f_rgb_to_hsl<  inbits, outbits > >::ret >::ret
  >::ret, mlc::exact_vt<  conversion_to_type<  ntg::color<  ocomps, oqbits, ocolor
  >, mlc::exact_vt<  conversion_from_type_to_type<  ntg::color<  icomps, iqbits,
  icolor >, ntg::color<  ocomps, oqbits, ocolor >, mlc::final, mlc::exact_vt<
  color_conversion<  icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_rgb_to_hsl<
  inbits, outbits > >, f_rgb_to_hsl<  inbits, outbits > >::ret >, mlc::exact_vt<
  color_conversion<  icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_rgb_to_hsl<
  inbits, outbits > >, f_rgb_to_hsl<  inbits, outbits > >::ret >::ret >, mlc::exact_vt<
  conversion_from_type_to_type<  ntg::color<  icomps, iqbits, icolor >, ntg::color<
  ocomps, oqbits, ocolor >, mlc::final, mlc::exact_vt<  color_conversion<  icomps,
  iqbits, icolor, ocomps, oqbits, ocolor, f_rgb_to_hsl<  inbits, outbits > >, f_rgb_-
  to_hsl<  inbits, outbits > >::ret >, mlc::final >::ret >::ret > . . . . . ??
any

```

```

oln::convert::abstract::conversion<  mlc::exact_vt<  conversion_to_type<  Output,
  mlc::exact_vt<  force<  Output, Exact >, Exact >::ret >, mlc::exact_vt<
  force<  Output, Exact >, Exact >::ret >::ret, mlc::exact_vt<  conversion_to_type<
  Output, mlc::exact_vt<  force<  Output, Exact >, Exact >::ret >, mlc::final >::ret > ??
any

```

```

oln::convert::abstract::conversion<  mlc::exact_vt<  conversion_to_type<  Output,
  mlc::exact_vt<  cast<  Output, Exact >, Exact >::ret >, mlc::exact_vt<  cast<
  Output, Exact >, Exact >::ret >::ret, mlc::exact_vt<  conversion_to_type<  Output,
  mlc::exact_vt<  cast<  Output, Exact >, Exact >::ret >, mlc::final >::ret > . . . . ??
any

```

```

oln::convert::abstract::conversion<  mlc::exact_vt<  conversion_to_type<  ntg::color<
  ocomps, oqbits, ocolor >, mlc::exact_vt<  conversion_from_type_to_type<
  ntg::color<  icomps, iqbits, icolor >, ntg::color<  ocomps, oqbits, ocolor >,
  mlc::final, mlc::exact_vt<  color_conversion<  icomps, iqbits, icolor, ocomps,
  oqbits, ocolor, f_yiq_to_rgb<  inbits, outbits > >, f_yiq_to_rgb<  inbits, outbits
  > >::ret >, mlc::exact_vt<  color_conversion<  icomps, iqbits, icolor, ocomps,
  oqbits, ocolor, f_yiq_to_rgb<  inbits, outbits > >, f_yiq_to_rgb<  inbits, outbits
  > >::ret >::ret >, mlc::exact_vt<  conversion_from_type_to_type<  ntg::color<
  icomps, iqbits, icolor >, ntg::color<  ocomps, oqbits, ocolor >, mlc::final,
  mlc::exact_vt<  color_conversion<  icomps, iqbits, icolor, ocomps, oqbits, ocolor,
  f_yiq_to_rgb<  inbits, outbits > >, f_yiq_to_rgb<  inbits, outbits > >::ret >,
  mlc::exact_vt<  color_conversion<  icomps, iqbits, icolor, ocomps, oqbits, ocolor,
  f_yiq_to_rgb<  inbits, outbits > >, f_yiq_to_rgb<  inbits, outbits > >::ret >::ret
  >::ret, mlc::exact_vt<  conversion_to_type<  ntg::color<  ocomps, oqbits, ocolor
  >, mlc::exact_vt<  conversion_from_type_to_type<  ntg::color<  icomps, iqbits,
  icolor >, ntg::color<  ocomps, oqbits, ocolor >, mlc::final, mlc::exact_vt<
  color_conversion<  icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_yiq_to_rgb<
  inbits, outbits > >, f_yiq_to_rgb<  inbits, outbits > >::ret >, mlc::exact_vt<
  color_conversion<  icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_yiq_to_rgb<
  inbits, outbits > >, f_yiq_to_rgb<  inbits, outbits > >::ret >::ret >, mlc::exact_vt<
  conversion_from_type_to_type<  ntg::color<  icomps, iqbits, icolor >, ntg::color<
  ocomps, oqbits, ocolor >, mlc::final, mlc::exact_vt<  color_conversion<  icomps,
  iqbits, icolor, ocomps, oqbits, ocolor, f_yiq_to_rgb<  inbits, outbits > >,
  f_yiq_to_rgb<  inbits, outbits > >::ret >, mlc::final >::ret >::ret > . . . . . ??
any

```

```

oln::convert::abstract::conversion<  mlc::exact_vt<  conversion_to_type<  point3d,
  mlc::exact_vt<  conversion_from_type_to_type<  ntg::color<  3, Qbits, S >,
  point3d, mlc::final, mlc::exact_vt<  value_to_point<  ntg::color<  3, Qbits, S
  >, Exact >, Exact >::ret >, mlc::exact_vt<  value_to_point<  ntg::color<  3,
  Qbits, S >, Exact >, Exact >::ret >, mlc::exact_vt<  conversion_from_-
  type_to_type<  ntg::color<  3, Qbits, S >, point3d, mlc::final, mlc::exact_vt<
  value_to_point<  ntg::color<  3, Qbits, S >, Exact >, Exact >::ret >, mlc::exact_-
  vt<  value_to_point<  ntg::color<  3, Qbits, S >, Exact >, Exact >::ret >::ret
  >::ret, mlc::exact_vt<  conversion_to_type<  point3d, mlc::exact_vt<  conversion_-
  from_type_to_type<  ntg::color<  3, Qbits, S >, point3d, mlc::final, mlc::exact_vt<
  value_to_point<  ntg::color<  3, Qbits, S >, Exact >, Exact >::ret >, mlc::exact_-
  vt<  value_to_point<  ntg::color<  3, Qbits, S >, Exact >, Exact >::ret >::ret
  >, mlc::exact_vt<  conversion_from_type_to_type<  ntg::color<  3, Qbits, S >,
  point3d, mlc::final, mlc::exact_vt<  value_to_point<  ntg::color<  3, Qbits, S >,
  Exact >, Exact >::ret >, mlc::final >::ret >::ret > . . . . . ??
any

```


<code>oln::convert::abstract::conversion< mlc::exact_vt< conversion_to_type< point1d, mlc::exact_vt< conversion_from_type_to_type< Argument_type, point1d, mlc::final, mlc::exact_vt< value_to_point< Argument_type, Exact >, Exact >::ret >, mlc::exact_vt< value_to_point< Argument_type, Exact >, Exact >::ret >::ret >, mlc::exact_vt< conversion_from_type_to_type< Argument_type, point1d, mlc::final, mlc::exact_vt< value_to_point< Argument_type, Exact >, Exact >::ret >, mlc::exact_vt< value_to_point< Argument_type, Exact >, Exact >::ret >::ret >::ret, mlc::exact_vt< conversion_to_type< point1d, mlc::exact_vt< conversion_from_type_to_type< Argument_type, point1d, mlc::final, mlc::exact_vt< value_to_point< Argument_type, Exact >, Exact >::ret >, mlc::exact_vt< value_to_point< Argument_type, Exact >, Exact >::ret >::ret >, mlc::exact_vt< conversion_from_type_to_type< Argument_type, point1d, mlc::final, mlc::exact_vt< value_to_point< Argument_type, Exact >, Exact >::ret >, mlc::final >::ret >::ret ></code>	??
any	
<code>oln::utils::abstract::histogram< mlc::exact_vt< histogram< T, CPT, V2P, mlc::exact_vt< histogram_minmax< T, CPT, V2P, Exact >, Exact >::ret >, mlc::exact_vt< histogram_minmax< T, CPT, V2P, Exact >, Exact >::ret >::ret ></code>	??
any	
<code>oln::utils::abstract::histogram< mlc::exact_vt< histogram< T, CPT, V2P, mlc::exact_vt< histogram_min< T, CPT, V2P, Exact >, Exact >::ret >, mlc::exact_vt< histogram_min< T, CPT, V2P, Exact >, Exact >::ret >::ret ></code>	??
any	
<code>oln::utils::abstract::histogram< mlc::exact_vt< histogram< T, CPT, V2P, Exact >, Exact >::ret ></code>	??
any	
<code>oln::utils::abstract::histogram< mlc::exact_vt< histogram< T, CPT, V2P, mlc::exact_vt< histogram_max< T, CPT, V2P, Exact >, Exact >::ret >, mlc::exact_vt< histogram_max< T, CPT, V2P, Exact >, Exact >::ret >::ret ></code>	??
any	
<code>oln::abstract::dpoint< dpoint1d ></code>	??
<code>oln::dpoint1d</code>	??
any	
<code>oln::abstract::dpoint< dpoint2d ></code>	??
<code>oln::dpoint2d</code>	??
any	
<code>oln::abstract::dpoint< dpoint3d ></code>	??
<code>oln::dpoint3d</code>	??
any	
<code>oln::morpho::env::abstract::env< NullEnv ></code>	??
<code>oln::morpho::env::NullEnv</code>	??
any	
<code>oln::morpho::env::abstract::env< ParentEnv< I > ></code>	??
<code>oln::morpho::env::ParentEnv< I ></code>	??
any	
<code>oln::morpho::env::abstract::env< OtherImageEnv< I > ></code>	??
<code>oln::morpho::env::OtherImageEnv< I ></code>	??
any	
any	
<code>oln::abstract::struct_elt< w_window2d< T > ></code>	??
<code>oln::abstract::w_window< w_window2d< T > ></code>	??
<code>oln::abstract::window_base< w_window< w_window2d< T > >, w_window2d< T > ></code>	??

	oln::abstract::w_windownd< w_window2d< T > >	??
	oln::w_window2d< T >	??
any	oln::abstract::struct_elt< window1d >	??
	oln::abstract::window< window1d >	??
	oln::abstract::window_base< window< window1d >, window1d >	??
	oln::abstract::windownd< window1d >	??
	oln::window1d	??
any	oln::abstract::struct_elt< window3d >	??
	oln::abstract::window< window3d >	??
	oln::abstract::window_base< window< window3d >, window3d >	??
	oln::abstract::windownd< window3d >	??
	oln::window3d	??
any	oln::abstract::struct_elt< window2d >	??
	oln::abstract::window< window2d >	??
	oln::abstract::window_base< window< window2d >, window2d >	??
	oln::abstract::windownd< window2d >	??
	oln::window2d	??
any	oln::abstract::struct_elt< w_window1d< T > >	??
	oln::abstract::w_window< w_window1d< T > >	??
	oln::abstract::window_base< w_window< w_window1d< T > >, w_window1d< T > >	??
	oln::abstract::w_windownd< w_window1d< T > >	??
	oln::w_window1d< T >	??
any		
any	oln::abstract::struct_elt< w_window3d< T > >	??
	oln::abstract::w_window< w_window3d< T > >	??
	oln::abstract::window_base< w_window< w_window3d< T > >, w_window3d< T > >	??
	oln::abstract::w_windownd< w_window3d< T > >	??
	oln::w_window3d< T >	??
any		
	oln::morpho::attr::attribute< mlc::exact_vt< height_type< T, Exact >, Exact >::ret >	??
any		
	oln::morpho::attr::attribute< mlc::exact_vt< box_type< I, Exact >, Exact >::ret >	??
any		
	oln::morpho::attr::attribute< mlc::exact_vt< card_type< T, mlc::exact_vt< card_full_type< I, T, Exact >, Exact >::ret >, mlc::exact_vt< card_full_type< I, T, Exact >, Exact >::ret >::ret >	??
any		
	oln::morpho::attr::attribute< mlc::exact_vt< integral_type< T, Exact >, Exact >::ret >	??
any		
	oln::morpho::attr::attribute< mlc::exact_vt< card_type< T, Exact >, Exact >::ret >	??
any		
	oln::morpho::attr::attribute< mlc::exact_vt< ball_parent_change< I, Exact >, Exact >::ret >	??
any		
	oln::morpho::attr::attribute< mlc::exact_vt< maxvalue_type< T, Exact >, Exact >::ret >	??
any		

oln::morpho::attr::attribute< mlc::exact_vt< minvalue_type< T, Exact >, Exact >::ret > . . .	??
any	
oln::morpho::attr::attribute< mlc::exact_vt< ball_type< I, Exact >, Exact >::ret >	??
any	
oln::morpho::attr::attribute< mlc::exact_vt< dist_type< I, Exact >, Exact >::ret >	??
any	
oln::abstract::image_size< image1d_size >	??
oln::image1d_size	??
any	
oln::abstract::image_size< image2d_size >	??
oln::image2d_size	??
any	
oln::abstract::image_size< image3d_size >	??
oln::image3d_size	??
any	
oln::impl::image_impl< image_array1d< T > >	??
oln::impl::image_array< T, image_array1d< T > >	??
oln::impl::image_array1d< T >	??
any	
oln::impl::image_impl< image_array2d< T > >	??
oln::impl::image_array< T, image_array2d< T > >	??
oln::impl::image_array2d< T >	??
any	
oln::impl::image_impl< image_array3d< T > >	??
oln::impl::image_array< T, image_array3d< T > >	??
oln::impl::image_array3d< T >	??
any	
oln::topo::tarjan::abstract::tarjan< mlc::exact_vt< tarjan_with_attr< Exact >, Exact >::ret >	??
any	
oln::topo::tarjan::abstract::tarjan< mlc::exact_vt< tarjan_with_attr< mlc::exact_vt< flat_zone< T, DestType, A, Exact >, Exact >::ret >, mlc::exact_vt< flat_zone< T, DestType, A, Exact >, Exact >::ret >::ret >	??
any	
oln::abstract::iter< mlc::exact_vt< fwd_iter3d< Exact >, Exact >::ret >	??
any	
oln::abstract::iter< mlc::exact_vt< fwd_iter1d< Exact >, Exact >::ret >	??
any	
oln::abstract::iter< mlc::exact_vt< fwd_iter2d< Exact >, Exact >::ret >	??
any	
oln::abstract::iter< mlc::exact_vt< bkd_iter3d< Exact >, Exact >::ret >	??
any	
oln::abstract::iter< mlc::exact_vt< bkd_iter1d< Exact >, Exact >::ret >	??
any	
oln::abstract::behavior< mlc::exact_vt< value_behavior< T, Exact >, Exact >::ret >	??
any	
oln::abstract::behavior< mlc::exact_vt< mirror_behavior< Exact >, Exact >::ret >	??
any	
oln::morpho::attr::attribute< mlc::exact_vt< cube_type< I, Exact >, Exact >::ret >	??
any	
oln::abstract::iter< mlc::exact_vt< bkd_iter2d< Exact >, Exact >::ret >	??
any	
oln::abstract::behavior< mlc::exact_vt< replicate_behavior< Exact >, Exact >::ret >	??
any	

oln::abstract::point< Exact >	??
any	
oln::abstract::neighborhood< Exact >	??
oln::abstract::window_base< neighborhood< Exact >, Exact >	??
oln::abstract::neighborhoodnd< Exact >	??
any	
oln::convert::abstract::conversion< Exact, Base >	??
oln::convert::abstract::conversion_to_type< Result_Type, Exact, Base >	??
oln::convert::abstract::conversion_from_type_to_type< Argument_Type, Result_Type, Exact, Base >	??
oln::convert::abstract::color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, Exact >	??
oln::convert::abstract::color_conversion< 3, inbits, hsi_traits, 3, outbits, rgb_traits, f_hsi_to_rgb< inbits, outbits > >	??
oln::convert::f_hsi_to_rgb< inbits, outbits >	??
oln::convert::abstract::color_conversion< 3, inbits, hsl_traits, 3, outbits, rgb_traits, f_hsl_to_rgb< inbits, outbits > >	??
oln::convert::f_hsl_to_rgb< inbits, outbits >	??
oln::convert::abstract::color_conversion< 3, inbits, hsv_traits, 3, outbits, rgb_traits, f_hsv_to_rgb< inbits, outbits > >	??
oln::convert::f_hsv_to_rgb< inbits, outbits >	??
oln::convert::abstract::color_conversion< 3, inbits, nrgb_traits, 3, outbits, rgb_traits, f_nrgb_to_rgb< inbits, outbits > >	??
oln::convert::f_nrgb_to_rgb< inbits, outbits >	??
oln::convert::abstract::color_conversion< 3, inbits, nrgb_traits, 3, outbits, xyz_traits, f_nrgb_to_xyz< inbits, outbits > >	??
oln::convert::f_nrgb_to_xyz< inbits, outbits >	??
oln::convert::abstract::color_conversion< 3, inbits, rgb_traits, 3, outbits, hsi_traits, f_rgb_to_hsi< inbits, outbits > >	??
oln::convert::f_rgb_to_hsi< inbits, outbits >	??
oln::convert::abstract::color_conversion< 3, inbits, rgb_traits, 3, outbits, hsl_traits, f_rgb_to_hsl< inbits, outbits > >	??
oln::convert::f_rgb_to_hsl< inbits, outbits >	??
oln::convert::abstract::color_conversion< 3, inbits, rgb_traits, 3, outbits, hsv_traits, f_rgb_to_hsv< inbits, outbits > >	??
oln::convert::f_rgb_to_hsv< inbits, outbits >	??
oln::convert::abstract::color_conversion< 3, inbits, rgb_traits, 3, outbits, nrgb_traits, f_rgb_to_nrgb< inbits, outbits > >	??
oln::convert::f_rgb_to_nrgb< inbits, outbits >	??
oln::convert::abstract::color_conversion< 3, inbits, rgb_traits, 3, outbits, xyz_traits, f_rgb_to_xyz< inbits, outbits > >	??
oln::convert::f_rgb_to_xyz	
oln::convert::abstract::color_conversion< 3, inbits, rgb_traits, 3, outbits, yiq_traits, f_rgb_to_yiq< inbits, outbits > >	??
oln::convert::f_rgb_to_yiq	
oln::convert::abstract::color_conversion< 3, inbits, rgb_traits, 3, outbits, yuv_traits, f_rgb_to_yuv< inbits, outbits > >	??
oln::convert::f_rgb_to_yuv	
oln::convert::abstract::color_conversion< 3, inbits, xyz_traits, 3, outbits, nrgb_traits, f_xyz_to_nrgb< inbits, outbits > >	??
oln::convert::f_xyz_to_nrgb< inbits, outbits >	??

```

oln::convert::abstract::color_conversion< 3, inbits, xyz_traits, 3, outbits, rgb_
    traits, f_xyz_to_rgb< inbits, outbits > > . . . . . ??
oln::convert::f_xyz_to_rgb
oln::convert::abstract::color_conversion< 3, inbits, yiq_traits, 3, outbits, rgb_
    traits, f_yiq_to_rgb< inbits, outbits > > . . . . . ??
oln::convert::f_yiq_to_rgb
oln::convert::abstract::color_conversion< 3, inbits, yuv_traits, 3, outbits, rgb_
    traits, f_yuv_to_rgb< inbits, outbits > > . . . . . ??
oln::convert::f_yuv_to_rgb
oln::convert::value_to_point< Argument_type, Exact > . . . . . ??
oln::convert::value_to_point< ntg::color< 3, Qbits, S >, Exact > . . . . . ??
oln::convert::abstract::conversion_from_type_to_type< Argument_type, point1d,
    mlc::exact_vt< value_to_point< Argument_type, Exact >, Exact >::ret > ??
oln::convert::abstract::conversion_from_type_to_type< ntg::color< 3, Qbits, S >,
    point3d, mlc::exact_vt< value_to_point< ntg::color< 3, Qbits, S >, Exact
    >, Exact >::ret > . . . . . ??
oln::convert::abstract::conversion_from_type_to_type< ntg::color< icomps, iqbits,
    icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::exact_vt< color_
    conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, Exact >, Exact
    >::ret > . . . . . ??
oln::convert::abstract::conversion_from_type_to_type< ntg::color< icomps, iqbits,
    icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::exact_vt< color_
    conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_hsi_to_rgb<
    inbits, outbits > >, f_hsi_to_rgb< inbits, outbits > >::ret > . . . . . ??
oln::convert::abstract::conversion_from_type_to_type< ntg::color< icomps, iqbits,
    icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::exact_vt< color_
    conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_hsl_to_rgb<
    inbits, outbits > >, f_hsl_to_rgb< inbits, outbits > >::ret > . . . . . ??
oln::convert::abstract::conversion_from_type_to_type< ntg::color< icomps, iqbits,
    icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::exact_vt< color_
    conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_hsv_to_rgb<
    inbits, outbits > >, f_hsv_to_rgb< inbits, outbits > >::ret > . . . . . ??
oln::convert::abstract::conversion_from_type_to_type< ntg::color< icomps, iqbits,
    icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::exact_vt< color_
    conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_nrgb_to_rgb<
    inbits, outbits > >, f_nrgb_to_rgb< inbits, outbits > >::ret > . . . . . ??
oln::convert::abstract::conversion_from_type_to_type< ntg::color< icomps, iqbits,
    icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::exact_vt< color_
    conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_nrgb_to_xyz<
    inbits, outbits > >, f_nrgb_to_xyz< inbits, outbits > >::ret > . . . . . ??
oln::convert::abstract::conversion_from_type_to_type< ntg::color< icomps, iqbits,
    icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::exact_vt< color_
    conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_rgb_to_hsi<
    inbits, outbits > >, f_rgb_to_hsi< inbits, outbits > >::ret > . . . . . ??
oln::convert::abstract::conversion_from_type_to_type< ntg::color< icomps, iqbits,
    icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::exact_vt< color_
    conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_rgb_to_hsl<
    inbits, outbits > >, f_rgb_to_hsl< inbits, outbits > >::ret > . . . . . ??
oln::convert::abstract::conversion_from_type_to_type< ntg::color< icomps, iqbits,
    icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::exact_vt< color_
    conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_rgb_to_hsv<
    inbits, outbits > >, f_rgb_to_hsv< inbits, outbits > >::ret > . . . . . ??

```

oln::convert::abstract::conversion_from_type_to_type< ntg::color< icoamps, iqbits, icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::exact_vt< color_-conversion< icoamps, iqbits, icolor, ocomps, oqbits, ocolor, f_rgb_to_nrgb< inbits, outbits > >, f_rgb_to_nrgb< inbits, outbits > >::ret >	??
oln::convert::abstract::conversion_from_type_to_type< ntg::color< icoamps, iqbits, icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::exact_vt< color_-conversion< icoamps, iqbits, icolor, ocomps, oqbits, ocolor, f_rgb_to_xyz< inbits, outbits > >, f_rgb_to_xyz< inbits, outbits > >::ret >	??
oln::convert::abstract::conversion_from_type_to_type< ntg::color< icoamps, iqbits, icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::exact_vt< color_-conversion< icoamps, iqbits, icolor, ocomps, oqbits, ocolor, f_rgb_to_yiq< inbits, outbits > >, f_rgb_to_yiq< inbits, outbits > >::ret >	??
oln::convert::abstract::conversion_from_type_to_type< ntg::color< icoamps, iqbits, icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::exact_vt< color_-conversion< icoamps, iqbits, icolor, ocomps, oqbits, ocolor, f_rgb_to_yuv< inbits, outbits > >, f_rgb_to_yuv< inbits, outbits > >::ret >	??
oln::convert::abstract::conversion_from_type_to_type< ntg::color< icoamps, iqbits, icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::exact_vt< color_-conversion< icoamps, iqbits, icolor, ocomps, oqbits, ocolor, f_xyz_to_nrgb< inbits, outbits > >, f_xyz_to_nrgb< inbits, outbits > >::ret >	??
oln::convert::abstract::conversion_from_type_to_type< ntg::color< icoamps, iqbits, icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::exact_vt< color_-conversion< icoamps, iqbits, icolor, ocomps, oqbits, ocolor, f_xyz_to_rgb< inbits, outbits > >, f_xyz_to_rgb< inbits, outbits > >::ret >	??
oln::convert::abstract::conversion_from_type_to_type< ntg::color< icoamps, iqbits, icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::exact_vt< color_-conversion< icoamps, iqbits, icolor, ocomps, oqbits, ocolor, f_yiq_to_rgb< inbits, outbits > >, f_yiq_to_rgb< inbits, outbits > >::ret >	??
oln::convert::abstract::conversion_from_type_to_type< ntg::color< icoamps, iqbits, icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::exact_vt< color_-conversion< icoamps, iqbits, icolor, ocomps, oqbits, ocolor, f_yuv_to_rgb< inbits, outbits > >, f_yuv_to_rgb< inbits, outbits > >::ret >	??
oln::convert::bound< Output, Exact >	??
oln::convert::cast< Output, Exact >	??
oln::convert::force< Output, Exact >	??
oln::convert::stretch< Output, Exact >	??
oln::convert::abstract::conversion_to_type< ntg::color< ocomps, oqbits, ocolor >, mlc::exact_vt< conversion_from_type_to_type< ntg::color< icoamps, iqbits, icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::final, mlc::exact_vt< color_conversion< icoamps, iqbits, icolor, ocomps, oqbits, ocolor, Exact >, Exact >::ret >, mlc::exact_vt< color_conversion< icoamps, iqbits, icolor, ocomps, oqbits, ocolor, Exact >, Exact >::ret >::ret, mlc::exact_vt< conversion_from_type_to_type< ntg::color< icoamps, iqbits, icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::final, mlc::exact_vt< color_conversion< icoamps, iqbits, icolor, ocomps, oqbits, ocolor, Exact >, Exact >::ret >, mlc::final >::ret > . .	??

```

oln::convert::abstract::conversion_to_type< ntg::color< ocomps, oqbits, ocolor >,
    mlc::exact_vt< conversion_from_type_to_type< ntg::color< icomps, iqbits,
icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::final, mlc::exact_vt<
color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_hsi_to_
rgb< inbits, outbits > >, f_hsi_to_rgb< inbits, outbits > >::ret >, mlc::exact_
vt< color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_hsi_
to_rgb< inbits, outbits > >, f_hsi_to_rgb< inbits, outbits > >::ret >::ret,
mlc::exact_vt< conversion_from_type_to_type< ntg::color< icomps, iqbits,
icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::final, mlc::exact_vt<
color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_hsi_to_
rgb< inbits, outbits > >, f_hsi_to_rgb< inbits, outbits > >::ret >, mlc::final
>::ret > . . . . . ??

```

```

oln::convert::abstract::conversion_to_type< ntg::color< ocomps, oqbits, ocolor >,
    mlc::exact_vt< conversion_from_type_to_type< ntg::color< icomps, iqbits,
    icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::final, mlc::exact_vt<
    color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_hsl_to_
    rgb< inbits, outbits > >, f_hsl_to_rgb< inbits, outbits > >::ret >, mlc::exact_vt<
    color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_hsl_
    to_rgb< inbits, outbits > >, f_hsl_to_rgb< inbits, outbits > >::ret >::ret,
    mlc::exact_vt< conversion_from_type_to_type< ntg::color< icomps, iqbits,
    icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::final, mlc::exact_vt<
    color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_hsl_to_
    rgb< inbits, outbits > >, f_hsl_to_rgb< inbits, outbits > >::ret >, mlc::final
    >::ret > . . . . .

```

```

oln::convert::abstract::conversion_to_type< ntg::color< ocomps, oqbits, ocolor >,
    mlc::exact_vt< conversion_to_type_to_type< ntg::color< icomps, iqbits,
    icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::final, mlc::exact_vt<
    color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_hsv_to_
    rgb< inbits, outbits > >, f_hsv_to_rgb< inbits, outbits > >::ret >, mlc::exact_
    vt< color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_hsv_
    to_rgb< inbits, outbits > >, f_hsv_to_rgb< inbits, outbits > >::ret >::ret,
    mlc::exact_vt< conversion_from_type_to_type< ntg::color< icomps, iqbits,
    icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::final, mlc::exact_vt<
    color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_hsv_to_
    rgb< inbits, outbits > >, f_hsv_to_rgb< inbits, outbits > >::ret >, mlc::final
    >::ret >

```

```

oln::convert::abstract::conversion_to_type< ntg::color< ocomps, oqbits, ocolor >,
    mlc::exact_vt< conversion_from_type_to_type< ntg::color< icomps, iqbits,
icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::final, mlc::exact_vt<
color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, f_nrgb_to_rgb<
inbits, outbits > >, f_nrgb_to_rgb< inbits, outbits > >::ret >,
mlc::exact_vt< color_conversion< icomps, iqbits, icolor, ocomps, oqbits,
ocolor, f_nrgb_to_rgb< inbits, outbits > >, f_nrgb_to_rgb< inbits, outbits
> >::ret >::ret, mlc::exact_vt< conversion_from_type_to_type< ntg::color<
icomps, iqbits, icolor >, ntg::color< ocomps, oqbits, ocolor >, mlc::final,
mlc::exact_vt< color_conversion< icomps, iqbits, icolor, ocomps, oqbits,
ocolor, f_nrgb_to_rgb< inbits, outbits > >, f_nrgb_to_rgb< inbits, outbits
> >::ret >, mlc::final >::ret > . . . . .

```


oln::convert::abstract::conversion_to_type< Output, mlc::exact_vt< stretch< Output, Exact >, Exact >::ret >	??
oln::convert::abstract::conversion_to_type< point1d, mlc::exact_vt< conversion_from_type_to_type< Argument_type, point1d, mlc::final, mlc::exact_vt< value_to_point< Argument_type, Exact >, Exact >::ret >, mlc::exact_vt< value_to_point< Argument_type, Exact >, Exact >::ret >::ret, mlc::exact_vt< conversion_from_type_to_type< Argument_type, point1d, mlc::final, mlc::exact_vt< value_to_point< Argument_type, Exact >, Exact >::ret >, mlc::final >::ret >	??
oln::convert::abstract::conversion_to_type< point3d, mlc::exact_vt< conversion_from_type_to_type< ntg::color< 3, Qbits, S >, point3d, mlc::final, mlc::exact_vt< value_to_point< ntg::color< 3, Qbits, S >, Exact >, Exact >::ret >, mlc::exact_vt< value_to_point< ntg::color< 3, Qbits, S >, Exact >, Exact >::ret >::ret, mlc::exact_vt< conversion_from_type_to_type< ntg::color< 3, Qbits, S >, point3d, mlc::final, mlc::exact_vt< value_to_point< ntg::color< 3, Qbits, S >, Exact >, Exact >::ret >, mlc::final >::ret >	??
oln::convert::abstract::conversion_to_type< Result_Type, mlc::exact_vt< conversion_from_type_to_type< Argument_Type, Result_Type, Base, Exact >, Exact >::ret, mlc::exact_vt< conversion_from_type_to_type< Argument_Type, Result_Type, Base, Exact >, Base >::ret >	??
any	
oln::utils::abstract::histogram< Exact >	??
oln::utils::histogram< T, CPT, V2P, Exact >	??
oln::utils::histogram_max< T, CPT, V2P, Exact >	??
oln::utils::histogram_min< T, CPT, V2P, Exact >	??
oln::utils::histogram_minmax< T, CPT, V2P, Exact >	??
oln::utils::histogram< T, CPT, V2P, mlc::exact_vt< histogram_max< T, CPT, V2P, Exact >, Exact >::ret >	??
oln::utils::histogram< T, CPT, V2P, mlc::exact_vt< histogram_min< T, CPT, V2P, Exact >, Exact >::ret >	??
oln::utils::histogram< T, CPT, V2P, mlc::exact_vt< histogram_minmax< T, CPT, V2P, Exact >, Exact >::ret >	??
any	
oln::abstract::dpoint< Exact >	??
any	
oln::morpho::env::abstract::env< Exact >	??
any	
oln::abstract::struct_elt< Exact >	??
oln::abstract::w_window< Exact >	??
oln::abstract::window_base< w_window< Exact >, Exact >	??
oln::abstract::w_windownd< Exact >	??
oln::abstract::window< Exact >	??
oln::abstract::window_base< window< Exact >, Exact >	??
oln::abstract::windownd< Exact >	??
any	
oln::morpho::attr::attribute< Exact >	??
oln::morpho::attr::ball_parent_change	
oln::morpho::attr::ball_type< I, Exact >	??
oln::morpho::attr::box_type< I, Exact >	??
oln::morpho::attr::card_type< T, Exact >	??
oln::morpho::attr::card_full_type	
oln::morpho::attr::card_type< T, mlc::exact_vt< card_full_type< I, T, Exact >, Exact >::ret >	??

oln::morpho::attr::cube_type< I, Exact >	??
oln::morpho::attr::dist_type< I, Exact >	??
oln::morpho::attr::height_type< T, Exact >	??
oln::morpho::attr::integral_type< T, Exact >	??
oln::morpho::attr::maxvalue_type< T, Exact >	??
oln::morpho::attr::minvalue_type< T, Exact >	??
any	
oln::abstract::image_size< Exact >	??
any	
oln::impl::image_impl< Exact >	??
oln::impl::image_array< T, Exact >	??
any	
oln::topo::tarjan::abstract::tarjan< Exact >	??
oln::topo::tarjan::abstract::tarjan_with_attr< Exact >	??
oln::topo::tarjan::flat_zone< T, DestType, A, Exact >	??
oln::topo::tarjan::abstract::tarjan_with_attr< mlc::exact_vt< flat_zone< T, DestType, A, Exact >, Exact >::ret >	??
any	
oln::abstract::iter< Exact >	??
oln::abstract::iter1d< Exact >	??
oln::bkd_iter1d< Exact >	??
oln::fwd_iter1d< Exact >	??
oln::abstract::iter1d< mlc::exact_vt< bkd_iter1d< Exact >, Exact >::ret >	??
oln::abstract::iter1d< mlc::exact_vt< fwd_iter1d< Exact >, Exact >::ret >	??
oln::abstract::iter2d< Exact >	??
oln::bkd_iter2d< Exact >	??
oln::fwd_iter2d< Exact >	??
oln::abstract::iter2d< mlc::exact_vt< bkd_iter2d< Exact >, Exact >::ret >	??
oln::abstract::iter2d< mlc::exact_vt< fwd_iter2d< Exact >, Exact >::ret >	??
oln::abstract::iter3d< Exact >	??
oln::bkd_iter3d< Exact >	??
oln::fwd_iter3d< Exact >	??
oln::abstract::iter3d< mlc::exact_vt< bkd_iter3d< Exact >, Exact >::ret >	??
oln::abstract::iter3d< mlc::exact_vt< fwd_iter3d< Exact >, Exact >::ret >	??
any	
oln::abstract::behavior< Exact >	??
oln::mirror_behavior< Exact >	??
oln::replicate_behavior< Exact >	??
oln::value_behavior< T, Exact >	??
oln::topo::combinatorial_map::internal::any< Inf >	??
oln::topo::combinatorial_map::internal::anyfunc< U, V, Inf >	??
mlc_hierarchy::any< E >	
oln::topo::combinatorial_map::internal::any< beta< U, V > >	??
oln::topo::combinatorial_map::internal::anyfunc< U, V, beta< U, V > >	??
oln::topo::combinatorial_map::internal::beta< U, V >	??
oln::topo::combinatorial_map::internal::any< lambda< U, V > >	??
oln::topo::combinatorial_map::internal::anyfunc< U, V, lambda< U, V > >	??
oln::topo::combinatorial_map::internal::lambda	
oln::topo::combinatorial_map::internal::any< sigma< U > >	??
oln::topo::combinatorial_map::internal::anyfunc< U, U, sigma< U > >	??
oln::topo::combinatorial_map::internal::sigma	

```

ntg::any_class< T >
ntg::any_const_class< T >
any_with_diamond
any_with_diamond
  oln::abstract::image< Exact > . . . . . ??
    oln::abstract::image_with_dim< 1, Exact > . . . . . ??
    oln::abstract::image_with_dim< 2, Exact > . . . . . ??
    oln::abstract::image_with_dim< 3, Exact > . . . . . ??
    oln::abstract::image_with_type< T, Exact > . . . . . ??
    oln::abstract::data_type_image< Exact > . . . . . ??
      oln::abstract::data_type_image_with_dim< Dim, Exact > . . . . . ??
        oln::abstract::non_vectorial_image_with_dim< Dim, Exact > . . . . . ??
        oln::abstract::binary_image_with_dim< Dim, Exact > . . . . . ??
        oln::abstract::decimal_image_with_dim< Dim, Exact > . . . . . ??
        oln::abstract::integer_image_with_dim< Dim, Exact > . . . . . ??
        oln::abstract::vectorial_image_with_dim
      oln::abstract::non_vectorial_image< Exact > . . . . . ??
        oln::abstract::binary_image< Exact > . . . . . ??
        oln::abstract::binary_image_with_dim< Dim, Exact > . . . . . ??
        oln::abstract::decimal_image< Exact > . . . . . ??
        oln::abstract::decimal_image_with_dim< Dim, Exact > . . . . . ??
        oln::abstract::integer_image< Exact > . . . . . ??
        oln::abstract::integer_image_with_dim< Dim, Exact > . . . . . ??
        oln::abstract::non_vectorial_image_with_dim< Dim, Exact > . . . . . ??
      oln::abstract::vectorial_image< Exact > . . . . . ??
        oln::abstract::vectorial_image_with_dim
    oln::abstract::image_with_type< image_id< Exact >::value_type, Exact > . . . . . ??
mlc_hierarchy::any_with_diamond< E >
  oln::io::internal::anything . . . . . ??
  oln::arith::arith_return_type_proxy_cst_div_< I, T >
  oln::arith::arith_return_type_proxy_cst_max_< I, T >
  oln::arith::arith_return_type_proxy_cst_min_< I, T >
  oln::arith::arith_return_type_proxy_cst_minus_< I, T >
  oln::arith::arith_return_type_proxy_cst_plus_< I, T >
  oln::arith::arith_return_type_proxy_cst_times_< I, T >
  oln::arith::arith_return_type_proxy_div_< I1, I2 >
  oln::arith::arith_return_type_proxy_max_< I1, I2 >
  oln::arith::arith_return_type_proxy_min_< I1, I2 >
  oln::arith::arith_return_type_proxy_minus_< I1, I2 >
  oln::arith::arith_return_type_proxy_plus_< I1, I2 >
  oln::arith::arith_return_type_proxy_times_< I1, I2 >
  mlc::array1d< Info_, T_ >
  mlc::internal::array1d_elt_< T, Info >
  mlc::array1d_info< card_, center_, i_ >
  mlc::array1d_info< card_, internal::unknown_, i_ >
  mlc::internal::array1d_start_< T >
  mlc::array2d< Info_, T_ >
  mlc::internal::array2d_elt_< T, Info >
  mlc::array2d_info< nrows_, ncols_, center_, i_ >
  mlc::array2d_info< nrows_, ncols_, internal::unknown_, i_ >
  mlc::internal::array2d_start_< T >
  mlc::array3d< Info_, T_ >
  mlc::internal::array3d_elt_< T, Info >

```

```

mlc::array3d_info< nplanes_, nrows_, ncols_, center_, i_ >
mlc::array3d_info< nplanes_, nrows_, ncols_, internal::unknown_, i_ >
mlc::internal::array3d_start< T >
mlc::assign_exact_offset< Exact, Final >
mlc::assign_exact_offset< mlc::final, mlc::final >
oln::morpho::attr::attr_traits< ball_parent_change< I, Exact > > . . . . . ??
oln::morpho::attr::attr_traits< ball_type< I, Exact > > . . . . . ??
oln::morpho::attr::attr_traits< box_type< I, Exact > > . . . . . ??
oln::morpho::attr::attr_traits< card_full_type< I, T, Exact > > . . . . . ??
oln::morpho::attr::attr_traits< card_type< T, Exact > > . . . . . ??
oln::morpho::attr::attr_traits< cube_type< I, Exact > > . . . . . ??
oln::morpho::attr::attr_traits< dist_type< I, Exact > > . . . . . ??
oln::morpho::attr::attr_traits< height_type< T, Exact > > . . . . . ??
oln::morpho::attr::attr_traits< integral_type< T, Exact > > . . . . . ??
oln::morpho::attr::attr_traits< maxvalue_type< T, Exact > > . . . . . ??
oln::morpho::attr::attr_traits< minvalue_type< T, Exact > > . . . . . ??
oln::morpho::attr::attr_traits< other_image< Dad, I, Exact > > . . . . . ??
mlc::begin_type
mlc::bool_case_< Cond, Ret, Cases >
mlc::bool_switch_< bool_case_< Cond, Ret >, Default >
mlc::bool_switch_< bool_case_< Cond, Ret, Cases >, Default >
ntg::bounded< T, i_min, i_max >
bounded
    ntg::bounded_s
bounded
    ntg::bounded_u
oln::box< PointType >
oln::utils::buffer . . . . . ??
ntg::C_for_float_d
ntg::C_for_float_s
ntg::C_for_int_s< nbits >
ntg::C_for_int_s< 16 >
ntg::C_for_int_s< 32 >
ntg::C_for_int_s< 8 >
ntg::C_for_int_u< nbits >
ntg::C_for_int_u< 16 >
ntg::C_for_int_u< 32 >
ntg::C_for_int_u< 8 >
mlc::case_< Cond, Ret, Cases >
oln::topo::chamfer< T > . . . . . ??
ret
    oln::morpho::attr::other_image< Dad, I, Exact > . . . . . ??
oln::morpho::attr::change_exact< integral_type< T, OldExact >, NewExact > . . . . . ??
oln::topo::combinatorial_map::cmap< I > . . . . . ??
oln::morpho::cmp_queue_elt< T > . . . . . ??
oln::morpher::color_mute< T, N > . . . . . ??
oln::morpher::color_mute< ntg::color< nbcomps_, nbits_, color_system >, N > . . . . . ??
oln::convert::abstract::conversion< Exact, Base >::output< T >
oln::convert::convoutput< ConvType, Base, InputType > . . . . . ??
ntg::builtin::cumul_trait< T >
ntg::builtin::cumul_trait< char >
ntg::builtin::cumul_trait< signed char >
ntg::builtin::cumul_trait< signed short >
ntg::builtin::cumul_trait< unsigned char >

```

ntg::builtin::cumul_trait< unsigned short >	
ntg::cycle_behavior	??
ntg::cycle_behavior::get< T >	
ntg::cycle_behavior::get< T >::cycle_fmod	
ntg::cycle_behavior::get< T >::cycle_mod	
ntg::data_type	??
ntg::vectorial	
ntg::internal::deduce_from_traits< Op, T, U >	??
ntg::internal::deduce_op_behavior< B1, B2 >	??
ntg::internal::deduce_op_behavior< B, B >	
ntg::internal::default_less< T >	??
default_less	
oln::internal::default_less< dpoint1d >	??
default_less	
oln::internal::default_less< point2d >	??
default_less	
oln::internal::default_less< point3d >	??
default_less	
oln::internal::default_less< dpoint2d >	??
default_less	
oln::internal::default_less< point1d >	??
default_less	
oln::internal::default_less< dpoint3d >	??
oln::internal::default_less< abstract::dpoint< Exact > >	??
oln::internal::default_less< abstract::point< Exact > >	??
oln::internal::default_less< coord >	??
ntg::internal::default_less< ntg::color< ncomps, qbits, color_system > >	??
mlc::form::desc< id >	
oln::morpher::abstract::dest_type< 1, T >	
oln::morpher::abstract::dest_type< 2, T >	
oln::morpher::abstract::dest_type< 3, T >	
oln::internal::dim_iterate_rec< dim, skip >	??
oln::internal::dim_iterate_rec< dim, 0 >	??
oln::internal::dim_skip_iterate_rec< dim, skip, current >	??
oln::internal::dim_skip_iterate_rec< dim, skip, 0 >	??
oln::dim_traits< Dim, T, Exact >	??
oln::dim_traits< 1, T, Exact >	??
oln::dim_traits< 2, T, Exact >	??
oln::dim_traits< 3, T, Exact >	??
dir_iter_	
oln::topo::inter_pixel::bkd_dir_iter< Dim, Exact >	??
dir_iter_	
oln::topo::inter_pixel::fwd_dir_iter< Dim, Exact >	??
oln::topo::inter_pixel::internal::dir_traits< Dim >	??
oln::topo::inter_pixel::internal::dir_traits< 2 >	??
oln::topo::dmap< T, T2 >	??
dpoint_traits	
oln::dpoint_traits< dpoint1d >	??
dpoint_traits	
oln::dpoint_traits< dpoint2d >	??
dpoint_traits	
oln::dpoint_traits< dpoint3d >	??
oln::dpoint_traits< abstract::dpoint< Exact > >	??

oln::transforms::dwt< I, K >	??
oln::topo::tarjan::empty_class	
mlc::end_type	
oln::snakes::energy< I >	??
oln::snakes::continuity_energy< I >	??
oln::snakes::curvature_energy< I >	??
oln::snakes::dummy_energy< I >	??
oln::snakes::image_energy< I >	??
enum_value	
ntg::bin	
enumerated	
ntg::binary	
mlc::eq< i, j >	
mlc::exact< T >	
mlc::exact< const T >	
mlc::exact< T & >	
mlc::exact< T * >	
mlc::exact_vt< T, Exact >	
mlc::exact_vt< Exact, final >	
oln::math::f_abs< T >	??
f_div	
oln::arith::default_f_div	
oln::math::internal::f_dot_product_nv< DestValue, I, J >	??
oln::math::internal::f_dot_product_v< DestValue, I, J >	??
f_max	
oln::arith::default_f_max	
f_min	
oln::arith::default_f_min	
f_minus	
oln::arith::default_f_minus	
f_plus	
oln::arith::default_f_plus	
oln::math::f_sqr< T >	??
oln::morpho::slow::f_tarjan_map< I, D, Env >	??
f_times	
oln::arith::default_f_times	
oln::utils::internal::f_to_float_< DestT, SrcT >	??
oln::utils::internal::f_to_float_< typename ntg::color< ncomps, qbits, color_system >::float_- vec_type, ntg::color< ncomps, qbits, color_system > >	??
false_type	
mlc::returns_bool_< false >	
mlc::false_type	
oln::morpho::internal::fast_morpho_inner< NP1, Dim, I, S, H, B, P, O >	??
oln::morpho::internal::fast_morpho_inner< Dim, Dim, I, S, H, B, P, O >	
mlc::final	
mlc::internal::find_pow2sup< N, false_type >	
mlc::internal::find_pow2sup< N, true_type >	
oln::topo::tarjan::obsolete::flat_zone< I >	??
ntg::force	??
ntg::force::get< T >	
oln::convol::fast::internal::gaussian_< dim >	??
oln::convol::fast::internal::gaussian_< 1 >	??
oln::convol::fast::internal::gaussian_< 2 >	??

oln::convol::fast::internal::gaussian_< 3 >	??
oln::io::internal::get_it< V >	
ntg::internal::get_order< T1, T2 >	
ntg::internal::get_order_inv< T1, T2 >	
oln::io::internal::get_pnm_type< I >	??
mlc::greater< i, j >	
mlc::greatereq< i, j >	
oln::snakes::greedy< N, I, external_energy >	??
oln::utils::hist_traits< Exact >	??
oln::utils::hist_traits< histogram< T, CPT, V2P, Exact > >	??
oln::utils::hist_traits< histogram_max< T, CPT, V2P, Exact > >	
oln::utils::hist_traits< histogram_min< T, CPT, V2P, Exact > >	
oln::utils::hist_traits< histogram_minmax< T, CPT, V2P, Exact > >	
oln::level::hlut< T, T2 >	??
oln::level::hlut_def< T, T2 >	??
ntg::hsi_traits< hsi_I >	
ntg::hsi_traits< hsi_S >	
point_type	
oln::snakes::node< I >	??
id_< T >	
mlc::if_< Cond, if_true_type, if_false_type >	
mlc::if_< false, if_true_type, if_false_type >	
oln::image1d< T, Exact >::mute< U >	??
oln::image2d< T, Exact >::mute< U >	??
oln::image3d< T, Exact >::mute< U >	??
oln::image_id< image1d< T, Exact > >	??
oln::image_id< image2d< T, Exact > >	??
oln::image_id< image3d< T, Exact > >	??
oln::image_id< image< Dim, T, Impl, Exact > >	??
oln::image_id< morpher::color_morpher< I, Exact > >	??
oln::image_id< morpher::iter_morpher< SrcType, IterType, Exact > >	??
oln::image_id< morpher::piece_morpher< SrcType, Exact > >	??
oln::image_id< morpher::slicing_morpher< const SrcType, Exact > >	??
oln::image_id< morpher::slicing_morpher< SrcType, Exact > >	??
oln::image_id< morpher::super_piece_morpher< SrcType, Exact > >	??
oln::image_id< morpher::super_slicing_morpher< const SrcType, Exact > >	??
oln::image_id< morpher::super_slicing_morpher< SrcType, Exact > >	??
oln::image_id< oln::morpher::subq_morpher< SrcType, N, Exact > >	??
oln::io::internal::image_reader< F, I >	??
oln::image_size_traits< abstract::image_size< Exact > >	??
oln::image_size_traits< image1d_size >	??
oln::image_size_traits< image2d_size >	??
oln::image_size_traits< image3d_size >	??
oln::image_traits	??
oln::image_traits< abstract::image_with_dim< 1, Exact > >	??
oln::image_traits< abstract::image_with_dim< 2, Exact > >	??
oln::image_traits< abstract::image_with_dim< 3, Exact > >	??
oln::image_traits< abstract::image_with_impl< Impl, Exact > >	??
oln::image_traits< abstract::image_with_impl< Impl, Exact > >	??
oln::image_traits< abstract::image_with_type< T, Exact > >	??
oln::image_traits< image1d< T, Exact > >	??
oln::image_traits< image2d< T, Exact > >	??
oln::image_traits< image3d< T, Exact > >	??

```

oln::image_traits< image< Dim, T, Impl, Exact > > . . . . . ??
oln::image_traits< morpher::abstract::generic_morpher< SrcType, Exact > >
oln::image_traits< morpher::color_morpher< SrcType, Exact > > . . . . . ??
oln::image_traits< morpher::iter_morpher< SrcType, IterType, Exact > > . . . . . ??
oln::image_traits< morpher::piece_morpher< SrcType, Exact > > . . . . . ??
oln::image_traits< morpher::slicing_morpher< SrcType, Exact > > . . . . . ??
oln::image_traits< oln::morpher::subq_morpher< SrcType, N, Exact > > . . . . . ??

oln::image_traits< abstract::image< Exact > >
oln::image_traits< abstract::image< Exact > > . . . . . ??
oln::image_traits< abstract::image_with_dim< image_id< Exact >::dim, Exact > > . . . . . ??
oln::image_traits< abstract::image_with_impl< image_id< Exact >::impl_type, image_id<
    Exact >::exact_type > > . . . . . ??
oln::image_traits< abstract::image_with_impl< image_id< image< Dim, T, Impl, Exact >
    >::impl_type, image_id< image< Dim, T, Impl, Exact > >::exact_type > > . . . . . ??
oln::image_traits< abstract::image_with_type< image_id< Exact >::value_type, Exact > > . . . . . ??
oln::image_traits< image< image_id< image1d< T, Exact > >::dim, image_id< image1d< T,
    Exact > >::value_type, image_id< image1d< T, Exact > >::impl_type, image_id<
    image1d< T, Exact > >::exact_type > > . . . . . ??
oln::image_traits< image< image_id< image2d< T, Exact > >::dim, image_id< image2d< T,
    Exact > >::value_type, image_id< image2d< T, Exact > >::impl_type, image_id<
    image2d< T, Exact > >::exact_type > > . . . . . ??
oln::image_traits< image< image_id< image3d< T, Exact > >::dim, image_id< image3d< T,
    Exact > >::value_type, image_id< image3d< T, Exact > >::impl_type, image_id<
    image3d< T, Exact > >::exact_type > > . . . . . ??
oln::image_traits< morpher::abstract::generic_morpher< SrcType, image_id< morpher::color_-
    morpher< SrcType, Exact > >::exact_type > > . . . . . ??
oln::image_traits< morpher::abstract::generic_morpher< SrcType, image_id< morpher::iter_-
    morpher< SrcType, IterType, Exact > >::exact_type > > . . . . . ??
oln::image_traits< morpher::abstract::generic_morpher< SrcType, image_id<
    morpher::piece_morpher< SrcType, Exact > >::exact_type > > . . . . . ??
oln::image_traits< morpher::abstract::generic_morpher< SrcType, image_id<
    morpher::slicing_morpher< SrcType, Exact > >::exact_type > > . . . . . ??
oln::image_traits< oln::morpher::abstract::generic_morpher< SrcType, image_id<
    oln::morpher::subq_morpher< SrcType, N, Exact > >::exact_type > > . . . . . ??
image_with_dim
    oln::abstract::data_type_image_with_dim< Dim, Exact > . . . . . ??
oln::abstract::image_with_type_with_dim_switch< Exact > . . . . . ??
ret
    oln::abstract::image_with_impl< image_id< image_id< slicing_morpher< SrcType, Exact
        > >::exact_type >::impl_type, image_id< slicing_morpher< SrcType, Exact >
        >::exact_type > . . . . . ??
ret
    oln::abstract::image_with_impl< image_id< image< Dim, image_id< image3d< T, Ex-
        act > >::value_type, image_id< image3d< T, Exact > >::impl_type, image_id<
        image3d< T, Exact > >::exact_type > >::impl_type, image_id< image< Dim,
        image_id< image3d< T, Exact > >::value_type, image_id< image3d< T, Exact
        > >::impl_type, image_id< image3d< T, Exact > >::exact_type > >::exact_type > ??
ret
    oln::abstract::image_with_impl< image_id< image_id< piece_morpher< SrcType, Exact
        > >::exact_type >::impl_type, image_id< piece_morpher< SrcType, Exact >
        >::exact_type > . . . . . ??
ret

```

oln::abstract::image_with_impl< image_id< image_id< slicing_morpher< const SrcType, Exact > >::exact_type >::impl_type, image_id< slicing_morpher< const SrcType, Exact > >::exact_type >	??
ret	
oln::abstract::image_with_impl< image_id< image_id< color_morpher< const SrcType, Exact > >::exact_type >::impl_type, image_id< color_morpher< const SrcType, Exact > >::exact_type >	??
ret	
oln::abstract::image_with_impl< image_id< image< Dim, T, Impl, Exact > >::impl_type, image_id< image< Dim, T, Impl, Exact > >::exact_type >	??
ret	
oln::abstract::image_with_impl< image_id< image_id< piece_morpher< const SrcType, Exact > >::exact_type >::impl_type, image_id< piece_morpher< const SrcType, Exact > >::exact_type >	??
ret	
oln::abstract::image_with_impl< image_id< image_id< iter_morpher< const SrcType, IterType, Exact > >::exact_type >::impl_type, image_id< iter_morpher< const SrcType, IterType, Exact > >::exact_type >	??
ret	
oln::abstract::image_with_impl< image_id< oln::image_id< subq_morpher< SrcType, N, Exact > >::exact_type >::impl_type, oln::image_id< subq_morpher< SrcType, N, Exact > >::exact_type >	??
ret	
oln::abstract::image_with_impl< image_id< image< Dim, image_id< image1d< T, Exact > >::value_type, image_id< image1d< T, Exact > >::impl_type, image_id< image1d< T, Exact > >::exact_type > >::impl_type, image_id< image< Dim, image_id< image1d< T, Exact > >::value_type, image_id< image1d< T, Exact > >::impl_type, image_id< image1d< T, Exact > >::exact_type > >::exact_type >	??
ret	
oln::abstract::image_with_impl< Impl, Exact >	??
oln::image< Dim, T, Impl, Exact >	??
oln::image1d< T, Exact >	??
oln::image2d< T, Exact >	??
oln::image3d< T, Exact >	??
oln::image< image_id< image1d< T, Exact > >::dim, image_id< image1d< T, Exact > >::value_type, image_id< image1d< T, Exact > >::impl_type, image_id< image1d< T, Exact > >::exact_type >	??
oln::image< image_id< image3d< T, Exact > >::dim, image_id< image3d< T, Exact > >::value_type, image_id< image3d< T, Exact > >::impl_type, image_id< image3d< T, Exact > >::exact_type >	??
oln::morpher::abstract::generic_morpher< SrcType, Exact >	??
oln::morpher::iter_morpher	
oln::morpher::iter_morpher< const SrcType, IterType, Exact >	??
oln::morpher::subq_morpher< SrcType, N, Exact >	??
oln::morpher::super_color_morpher< SrcType, Exact >	??
oln::morpher::color_morpher< SrcType, Exact >	??
oln::morpher::color_morpher< const SrcType, Exact >	??
oln::morpher::super_color_morpher< const SrcType, image_id< color_morpher< const SrcType, Exact > >::exact_type >	??
oln::morpher::super_color_morpher< SrcType, image_id< color_morpher< SrcType, Exact > >::exact_type >	??
oln::morpher::super_piece_morpher< SrcType, Exact >	??
oln::morpher::piece_morpher< SrcType, Exact >	??
oln::morpher::piece_morpher< const SrcType, Exact >	??

oln::morpher::super_piece_morpher< const SrcType, image_id< piece_morpher< const SrcType, Exact > >::exact_type >	??
oln::morpher::super_piece_morpher< SrcType, image_id< piece_morpher< SrcType, Exact > >::exact_type >	??
oln::morpher::super_slicing_morpher< SrcType, Exact >	??
oln::morpher::slicing_morpher< SrcType, Exact >	??
oln::morpher::slicing_morpher< const SrcType, Exact >	??
oln::morpher::super_slicing_morpher< const SrcType, image_id< slicing_morpher< const SrcType, Exact > >::exact_type >	??
oln::morpher::super_slicing_morpher< SrcType, image_id< slicing_morpher< SrcType, Exact > >::exact_type >	??
oln::morpher::abstract::generic_morpher< const , image_id< color_morpher< const SrcType, Exact > >::exact_type >	??
oln::morpher::abstract::generic_morpher< const , image_id< piece_morpher< const SrcType, Exact > >::exact_type >	??
oln::morpher::abstract::generic_morpher< const , image_id< slicing_morpher< const SrcType, Exact > >::exact_type >	??
oln::morpher::abstract::generic_morpher< const SrcType, image_id< iter_morpher< const SrcType, IterType, Exact > >::exact_type >	??
oln::morpher::abstract::generic_morpher< SrcType, image_id< color_morpher< SrcType, Exact > >::exact_type >	??
oln::morpher::abstract::generic_morpher< SrcType, image_id< iter_morpher< SrcType, IterType, Exact > >::exact_type >	??
oln::morpher::abstract::generic_morpher< SrcType, image_id< piece_morpher< SrcType, Exact > >::exact_type >	??
oln::morpher::abstract::generic_morpher< SrcType, image_id< slicing_morpher< SrcType, Exact > >::exact_type >	??
oln::morpher::abstract::generic_morpher< SrcType, oln::image_id< subq_morpher< SrcType, N, Exact > >::exact_type >	??
ret	
ret	
oln::abstract::image_with_impl< image_id< image_id< color_morpher< SrcType, Exact > >::exact_type >::impl_type, image_id< color_morpher< SrcType, Exact > >::exact_type >	??
ret	
oln::abstract::image_with_impl< image_id< image_id< iter_morpher< SrcType, IterType, Exact > >::exact_type >::impl_type, image_id< iter_morpher< SrcType, IterType, Exact > >::exact_type >	??
ret	
ret	
oln::abstract::image_with_impl< image_id< image< Dim, image_id< image2d< T, Exact > >::value_type, image_id< image2d< T, Exact > >::impl_type, image_id< image2d< T, Exact > >::exact_type > >::impl_type, image_id< image2d< T, Exact > >::value_type, image_id< image2d< T, Exact > >::impl_type, image_id< image2d< T, Exact > >::exact_type > >::exact_type >	??
ret	
oln::abstract::image_with_impl< image_id< Exact >::impl_type, Exact >	??
oln::io::internal::image_writer< F, I >	??
oln::utils::internal::img_max_size< T >	??
oln::utils::internal::img_max_size< ntg::color< 3, Qbits, S > >	
impl_traits	
oln::impl_traits< impl::image_array3d< T > >	??
impl_traits	
oln::impl_traits< impl::image_array1d< T > >	??

impl_traits	
oln::impl_traits< impl::image_array2d< T > >	??
impl_traits	
oln::impl_traits< impl::image_array< T, Exact > >	??
oln::impl_traits< impl::image_impl< Exact > >	??
int_value	
ntg::uint_value	
int_value	
ntg::sint_value	
integer	
ntg::unsigned_integer	
integer	
ntg::signed_integer	
oln::topo::inter_pixel::interpixel< I >	??
ntg::interval< lval, uval >	??
ntg::interval< 0, 1 >	??
ntg::hsl_traits< hsl_L >	
ntg::hsl_traits< hsl_S >	
ntg::hsv_traits< hsv_S >	
ntg::hsv_traits< hsv_V >	
ntg::rgb_traits< rgb_B >	
ntg::rgb_traits< rgb_G >	
ntg::rgb_traits< rgb_R >	
ntg::xyz_traits< xyz_X >	
ntg::xyz_traits< xyz_Y >	
ntg::xyz_traits< xyz_Z >	
ntg::yiq_traits< yiq_Y >	
ntg::yuv_traits< yuv_Y >	
ntg::interval< 0, 360 >	??
ntg::hsi_traits< hsi_H >	
ntg::hsl_traits< hsl_H >	
ntg::hsv_traits< hsv_H >	
mlc::invalid	
mlc::internal::is_a< form::class_ >	
mlc::internal::is_a< form::class_ >::helper< T, U >	
mlc::internal::is_a< form::template_1_class_class_g_class_ >	
mlc::internal::is_a< form::template_1_class_class_g_class_ >::helper< T, U >	
mlc::internal::is_a< form::template_1_class_g_class_ >	
mlc::internal::is_a< form::template_1_class_g_class_ >::helper< T, U >	
ntg::internal::is_defined< T >	
ntg::internal::is_defined< undefined_traits >	
mlc::is_false< false >	
mlc::internal::is_pow2< N >	
mlc::internal::is_pow2< 16 >	
mlc::internal::is_pow2< 32 >	
mlc::internal::is_pow2< 64 >	
mlc::internal::is_pow2< 8 >	
mlc::is_true< true >	
iter_traits	
oln::iter_traits< bkd_iter2d< Exact > >	??
iter_traits	
oln::iter_traits< fwd_iter2d< Exact > >	??
iter_traits	

oln::iter_traits< abstract::iter2d< Exact > >	??
iter_traits	
oln::iter_traits< topo::inter_pixel::fwd_dir_iter< Dim, Exact > >	
iter_traits	
oln::iter_traits< abstract::iter3d< Exact > >	??
iter_traits	
oln::iter_traits< topo::inter_pixel::internal::dir_iter< 2, Exact > >	??
iter_traits	
oln::iter_traits< topo::inter_pixel::bkd_dir_iter< Dim, Exact > >	
iter_traits	
oln::iter_traits< topo::inter_pixel::internal::dir_iter< 1, Exact > >	??
iter_traits	
oln::iter_traits< abstract::iter1d< Exact > >	??
iter_traits	
oln::iter_traits< bkd_iter1d< Exact > >	??
iter_traits	
oln::iter_traits< fwd_iter1d< Exact > >	
iter_traits	
oln::iter_traits< topo::inter_pixel::internal::dir_iter< 3, Exact > >	??
iter_traits	
oln::iter_traits< bkd_iter3d< Exact > >	
iter_traits	
oln::iter_traits< fwd_iter3d< Exact > >	
super_type	
oln::topo::inter_pixel::internal::dir_iter_	
oln::iter_traits< abstract::iter< Exact > >	??
oln::utils::key	??
ntg::builtin::largest_trait< T >	
ntg::builtin::largest_trait< char >	
ntg::builtin::largest_trait< signed char >	
ntg::builtin::largest_trait< signed short >	
ntg::builtin::largest_trait< unsigned char >	
ntg::builtin::largest_trait< unsigned short >	
mlc::internal::lbrk_	
mlc::less< i, j >	
mlc::lesseq< i, j >	
oln::topo::combinatorial_map::internal::level< U >	
mlc::max< i, j >	
max_accumulator< T >	??
max_accumulator< coord >	??
mlc::maxN< i, j, N >	
oln::utils::MD5	??
ntg::internal::meta_undefined_traits< T >	
mlc::min< i, j >	
oln::mute< I, T >	??
oln::mute< input_type, comp_type >	??
mlc::neq< i, j >	
mlc::internal::no_	
oln::topo::combinatorial_map::internal::node< U >	
oln::topo::inter_pixel::node< I >	??
non_vectorial	
ntg::enumerated	
non_vectorial	
ntg::real	

```

ntg::non_vectorial
mlc::not_implemented_yet
ntg::nrgb_traits< nrgb_B >
ntg::nrgb_traits< nrgb_G >
ntg::nrgb_traits< nrgb_R >
ntg::internal::operator_cmp
ntg::internal::operator_div
ntg::internal::operator_logical
ntg::internal::operator_max
ntg::internal::operator_min
ntg::internal::operator_minus
ntg::internal::operator_mod
ntg::internal::operator_plus
ntg::internal::operator_times
ntg::internal::operator_traits< Op, T, U > . . . . . ??

    ntg::internal::operator_traits< Op, cycle< T, I >, U >
    ntg::internal::operator_traits< Op, cycle< T1, I1 >, cycle< T2, I2 > >
    ntg::internal::operator_traits< Op, range< T, I, B >, U >
    ntg::internal::operator_traits< Op, range< T1, I1, B1 >, range< T2, I2, B2 > >
    ntg::internal::operator_traits< Op, U, cycle< T, I > >
    ntg::internal::operator_traits< Op, U, range< T, I, B > >

ntg::internal::operator_traits< Op, traits_lhs_type, traits_rhs_type > . . . . . ??

ntg::internal::operator_traits< operator_cmp, bin, bin >
ntg::internal::operator_traits< operator_cmp, cplx< R1, T1 >, cplx< R2, T2 > >
ntg::internal::operator_traits< operator_cmp, int_s< nbits, B1 >, int_s< mbits, B2 > >
ntg::internal::operator_traits< operator_cmp, int_s< nbits, B1 >, int_u< mbits, B2 > >
ntg::internal::operator_traits< operator_cmp, int_u< nbits, B1 >, int_u< mbits, B2 > >
ntg::internal::operator_traits< operator_cmp, vec< N, T1, S1 >, vec< N, T2, S2 > >
ntg::internal::operator_traits< operator_div, cplx< R1, T1 >, cplx< R2, T2 > >
ntg::internal::operator_traits< operator_div, cplx< R1, T1 >, T2 >
ntg::internal::operator_traits< operator_div, int_s< nbits, B1 >, int_s< mbits, B2 > >
ntg::internal::operator_traits< operator_div, int_s< nbits, B1 >, int_u< mbits, B2 > >
ntg::internal::operator_traits< operator_div, int_u< mbits, B2 >, int_s< nbits, B1 > >
ntg::internal::operator_traits< operator_div, int_u< nbits, B1 >, int_u< mbits, B2 > >
ntg::internal::operator_traits< operator_div, vec< N, T1 >, T2 >
ntg::internal::operator_traits< operator_logical, bin, T >
ntg::internal::operator_traits< operator_max, bin, bin >
ntg::internal::operator_traits< operator_max, int_s< nbits, B1 >, int_s< mbits, B2 > >
ntg::internal::operator_traits< operator_max, int_u< nbits, B1 >, int_u< mbits, B2 > >
ntg::internal::operator_traits< operator_min, bin, bin >
ntg::internal::operator_traits< operator_min, int_s< nbits, B1 >, int_s< mbits, B2 > >
ntg::internal::operator_traits< operator_min, int_u< nbits, B1 >, int_u< mbits, B2 > >
ntg::internal::operator_traits< operator_minus, cplx< R1, T1 >, cplx< R2, T2 > >
ntg::internal::operator_traits< operator_minus, cplx< R1, T1 >, T2 >
ntg::internal::operator_traits< operator_minus, cplx< rect, T1 >, vec< 2, T2 > >
ntg::internal::operator_traits< operator_minus, int_s< nbits, B1 >, int_s< mbits, B2 > >
ntg::internal::operator_traits< operator_minus, int_s< nbits, B1 >, int_u< mbits, B2 > >
ntg::internal::operator_traits< operator_minus, int_u< 32, B1 >, int_u< mbits, B2 > >
ntg::internal::operator_traits< operator_minus, int_u< nbits, B1 >, int_u< mbits, B2 > >
ntg::internal::operator_traits< operator_minus, vec< N, T1 >, vec< N, T2 > >
ntg::internal::operator_traits< operator_mod, int_s< nbits, B1 >, int_s< mbits, B2 > >
ntg::internal::operator_traits< operator_mod, int_s< nbits, B1 >, int_u< mbits, B2 > >
ntg::internal::operator_traits< operator_mod, int_u< nbits, B1 >, int_u< mbits, B2 > >

```

```

ntg::internal::operator_traits< operator_mod, vec< N, T1 >, T2 >
ntg::internal::operator_traits< operator_plus, cplx< R1, T1 >, cplx< R2, T2 > >
ntg::internal::operator_traits< operator_plus, cplx< R1, T1 >, T2 >
ntg::internal::operator_traits< operator_plus, cplx< rect, T1 >, vec< 2, T2 > >
ntg::internal::operator_traits< operator_plus, int_s< nbits, B1 >, int_s< mbits, B2 > >
ntg::internal::operator_traits< operator_plus, int_s< nbits, B1 >, int_u< mbits, B2 > >
ntg::internal::operator_traits< operator_plus, int_u< nbits, B1 >, int_u< mbits, B2 > >
ntg::internal::operator_traits< operator_plus, vec< N, T1, S1 >, vec< N, T2, S2 > >
ntg::internal::operator_traits< operator_times, cplx< R1, T1 >, cplx< R2, T2 > >
ntg::internal::operator_traits< operator_times, cplx< R1, T1 >, T2 >
ntg::internal::operator_traits< operator_times, int_s< nbits, B1 >, int_s< mbits, B2 > >
ntg::internal::operator_traits< operator_times, int_s< nbits, B1 >, int_u< mbits, B2 > >
ntg::internal::operator_traits< operator_times, int_u< nbits, B1 >, int_u< mbits, B2 > >
ntg::internal::operator_traits< operator_times, vec< N, T1 >, T2 >
ntg::internal::operator_traits< operator_times, vec< N, T1 >, vec< N, T2 > >
ntg::internal::optraits< T > . . . . . ??
    ntg::internal::optraits< cycle< T, interval > >
    ntg::internal::optraits< enum_value< E > >
    ntg::internal::optraits< range< T, interval, behavior > >
    ntg::internal::optraits< real_value< E > > . . . . . ??
        ntg::internal::optraits< float_value< E > >
        ntg::internal::optraits< int_value< E > > . . . . . ??
    ntg::internal::optraits< sint_value< E > >
    ntg::internal::optraits< uint_value< E > >
    ntg::internal::optraits< vect_value< E > >
    ntg::type_traits< T > . . . . . ??
ntg::internal::optraits< bool >
ntg::internal::optraits< color< ncomps, qbits, color_system > >
ntg::internal::optraits< cplx< polar, T > >
ntg::internal::optraits< enum_value< bin > > . . . . . ??
    ntg::internal::optraits< bin >
ntg::internal::optraits< float_value< double > > . . . . . ??
    ntg::internal::optraits< double >
ntg::internal::optraits< float_value< float > > . . . . . ??
    ntg::internal::optraits< float >
ntg::internal::optraits< int_value< E > >
ntg::internal::optraits< sint_value< char > > . . . . . ??
    ntg::internal::optraits< char >
ntg::internal::optraits< sint_value< int_s< nbits, behavior > > > . . . . . ??
    ntg::internal::optraits< int_s< nbits, behavior > >
ntg::internal::optraits< sint_value< signed char > > . . . . . ??
    ntg::internal::optraits< signed char >
ntg::internal::optraits< sint_value< signed int > > . . . . . ??
    ntg::internal::optraits< signed int >
ntg::internal::optraits< sint_value< signed long > > . . . . . ??
    ntg::internal::optraits< signed long >
ntg::internal::optraits< sint_value< signed short > > . . . . . ??
    ntg::internal::optraits< signed short >
ntg::internal::optraits< uint_value< int_u< nbits, behavior > > > . . . . . ??
    ntg::internal::optraits< int_u< nbits, behavior > >

```


ntg::internal::optraits< uint_value< unsigned char > >	??
ntg::internal::optraits< unsigned char >	
ntg::internal::optraits< uint_value< unsigned int > >	??
ntg::internal::optraits< unsigned int >	
ntg::internal::optraits< uint_value< unsigned long > >	??
ntg::internal::optraits< unsigned long >	
ntg::internal::optraits< uint_value< unsigned short > >	??
ntg::internal::optraits< unsigned short >	
ntg::internal::optraits< value< E > >	
ntg::internal::optraits< value< E > >	??
ntg::internal::optraits< vec< 2, T > >	??
ntg::internal::optraits< cplx< rect, T > >	
ntg::internal::optraits< vect_value< vec< N, T, E > > >	??
ntg::internal::optraits< vec< N, T, E > >	
oln::morpho::attr::other_point	??
oln::convert::abstract::internal::output< Base, T >	??
oln::convert::abstract::internal::output< conversion_from_type_to_type< Argument_Type, Result_Type, Exact, Base >, Argument_Type >	
oln::convert::abstract::internal::output< conversion_to_type< Result_Type, Exact, Base >, T >	
mlc::internal::pbrk_	
oln::io::internal::pnm2d_info	
oln::io::internal::pnm_read_data< V, R >	??
oln::io::internal::pnm_read_data< PnmBinary, ReadPnmPlain >	??
oln::io::internal::pnm_read_data< PnmBinary, ReadPnmRaw >	??
oln::io::internal::pnm_read_data< PnmInteger, ReadPnmPlain >	??
oln::io::internal::pnm_read_data< PnmInteger, ReadPnmRaw >	??
oln::io::internal::pnm_read_data< PnmVectorial, ReadPnmPlain >	??
oln::io::internal::pnm_read_data< PnmVectorial, ReadPnmRaw >	
oln::io::internal::pnm_reader< R, Dim, V, I >	??
oln::io::internal::image_reader< ReadPnmPlain, I >	??
oln::io::internal::image_reader< ReadPnmRaw, I >	??
oln::io::internal::pnm_reader< R, 2, PnmBinary, I >	??
oln::io::internal::pnm_reader< R, 2, PnmInteger, I >	??
oln::io::internal::pnm_reader< R, 2, PnmVectorial, I >	??
oln::io::internal::pnm_reader< ReadPnmPlain, I::dim, get_pnm_type< I >::ret, I >	??
oln::io::internal::pnm_reader< ReadPnmRaw, 3, P, I >	??
oln::io::internal::pnm_reader< ReadPnmRaw, I::dim, get_pnm_type< I >::ret, I >	??
oln::io::internal::pnm_write	??
oln::io::internal::pnm_write_data< V, R >	??
oln::io::internal::pnm_write_data< PnmBinary, WritePnmPlain >	??
oln::io::internal::pnm_write_data< PnmBinary, WritePnmRaw >	??
oln::io::internal::pnm_write_data< PnmInteger, WritePnmPlain >	??
oln::io::internal::pnm_write_data< PnmInteger, WritePnmRaw >	??
oln::io::internal::pnm_write_data< PnmVectorial, WritePnmPlain >	??
oln::io::internal::pnm_write_data< PnmVectorial, WritePnmRaw >	??
pnm_writer	
oln::io::internal::image_writer< WritePnmRaw, I >	
pnm_writer	
oln::io::internal::image_writer< WritePnmPlain, I >	
oln::io::internal::pnm_writer< W, Dim, V, I >	
oln::io::internal::pnm_writer< W, 2, PnmBinary, I >	??
oln::io::internal::pnm_writer< W, 2, PnmInteger, I >	??

oln::io::internal::pnm_writer< W, 2, PnmVectorial, I >	??
oln::io::internal::pnm_writer< WritePnmRaw, 3, P, I >	??
point_traits	
oln::point_traits< point2d >	??
point_traits	
oln::point_traits< point1d >	??
point_traits	
oln::point_traits< point3d >	??
oln::point_traits< abstract::point< Exact > >	??
mlc::pow2sup< N >	
oln::io::internal::readers_trier	??
real	
ntg::integer	
real	
ntg::decimal	
real_value	
ntg::int_value	
real_value	
ntg::float_value	
oln::convol::fast::internal::recursivefilter_coef< FloatT >	??
ntg::internal::ret_behavior_if< need_check, Ret >	??
ntg::internal::ret_behavior_if< false, Ret >	
returns_bool_	
mlc::type_eq	
returns_bool_	
mlc::internal::is_a< form::class_ >::check	
returns_bool_	
mlc::type_eq< T, T >	
returns_bool_	
mlc::internal::is_a< form::template_l_class_class_g_class_ >::check	
returns_bool_	
mlc::internal::is_a< form::template_l_class_g_class_ >::check	
ntg::saturate	??
ntg::saturate::get< T >	
mlc::saturateN< i, N >	
oln::convol::internal::se_array< Size, S >	??
oln::utils::se_stat< Sum, Var >	??
oln::snakes::segment< I >	??
oln::utils::select_distrib_sort< reverse >	??
oln::utils::select_distrib_sort< true >	
ntg::builtin::signed_cumul_trait< T >	
ntg::builtin::signed_cumul_trait< char >	
ntg::builtin::signed_cumul_trait< signed char >	
ntg::builtin::signed_cumul_trait< unsigned char >	
ntg::builtin::signed_cumul_trait< unsigned long >	
ntg::builtin::signed_largest_trait< T >	
ntg::builtin::signed_largest_trait< unsigned long >	
ntg::builtin::signed_trait< T >	
ntg::builtin::signed_trait< unsigned char >	
ntg::builtin::signed_trait< unsigned int >	
ntg::builtin::signed_trait< unsigned long >	
ntg::builtin::signed_trait< unsigned short >	
mlc::simple_for< from, Cmp, to, by >	
mlc::internal::simple_for_statement< i, Cmp, to, by, false >	

mlc::internal::simple_for_statement< i, Cmp, to, by, true >	
sint_value	
ntg::int_s	
oln::snakes::snake< algorithm >	??
oln::morpho::sort_dimensions< E >	??
oln::morpho::internal::stat_< I, E, V >	??
oln::morpho::internal::stat_< I, E, ntg::bin >	
binary_function	
oln::arith::f_min	
binary_function	
oln::internal::compose_ub_< F1, F2 >	??
binary_function	
oln::arith::f_minus	
binary_function	
oln::internal::compose_bu_< F1, F2 >	??
binary_function	
oln::arith::f_logic_and	??
binary_function	
oln::arith::f_plus	
binary_function	
oln::convert::internal::compconv2_< C, BF >	??
binary_function	
oln::arith::f_logic_and_not	??
binary_function	
oln::arith::f_div	
binary_function	
oln::arith::f_max	
binary_function	
oln::arith::f_logic_or	??
binary_function	
oln::arith::f_times	
ios	
oln::io::gz::zfstream_common	??
oln::io::gz::zifstream	??
oln::io::gz::zofstream	
istream	
oln::io::gz::zifstream	??
ostream	
oln::io::gz::zofstream	
streambuf	
oln::io::gz::zfilebuf	??
unary_function	
oln::level::f_invert< T >	??
unary_function	
oln::internal::compose_uu_< F1, F2 >	??
unary_function	
oln::arith::f_minus_cst	
unary_function	
oln::arith::f_max_cst	
unary_function	
oln::utils::f_minmax< T >	??
oln::utils::f_moments< T, C >	??
unary_function	
oln::arith::f_logic_and_cst	

unary_function	
oln::convert::internal::compconv1_< C, UF >	??
unary_function	
oln::arith::f_times_cst	
unary_function	
oln::arith::f_min_cst	
unary_function	
oln::level::threshold< Input, Output, Exact >	??
unary_function	
oln::arith::f_logic_or_cst	
unary_function	
oln::arith::f_logic_and_not_cst	
unary_function	
oln::arith::f_logic_not	??
unary_function	
oln::arith::f_div_cst	
unary_function	
oln::f_identity< T >	??
unary_function	
oln::arith::f_plus_cst	
oln::io::internal::stream_wrapper< W >	??
oln::io::internal::stream_wrapper< StreamFile >	??
oln::io::internal::stream_wrapper< StreamGz >	??
oln::io::internal::stream_wrappers_find_files< W >	??
oln::io::internal::stream_wrappers_find_files< StreamNone >	??
ntg::strict	??
ntg::strict::get< T >	
struct_elt_traits	
oln::struct_elt_traits< abstract::w_windownd< Exact > >	??
struct_elt_traits	
oln::struct_elt_traits< abstract::window< Exact > >	??
struct_elt_traits	
oln::struct_elt_traits< window1d >	??
struct_elt_traits	
oln::struct_elt_traits< w_window3d< T > >	??
struct_elt_traits	
oln::struct_elt_traits< window3d >	??
struct_elt_traits	
oln::struct_elt_traits< w_window1d< T > >	??
struct_elt_traits	
oln::struct_elt_traits< abstract::window_base< Sup, Exact > >	??
struct_elt_traits	
oln::struct_elt_traits< w_window2d< T > >	??
struct_elt_traits	
oln::struct_elt_traits< abstract::w_window< Exact > >	??
struct_elt_traits	
oln::struct_elt_traits< abstract::windownd< Exact > >	??
struct_elt_traits	
oln::struct_elt_traits< neighborhood1d >	??
struct_elt_traits	
oln::struct_elt_traits< window2d >	??
struct_elt_traits	
oln::struct_elt_traits< neighborhood3d >	??
struct_elt_traits	

oln::struct_elt_traits< abstract::neighborhoodnd< Exact > >	??
struct_elt_traits	
oln::struct_elt_traits< neighborhood2d >	??
oln::struct_elt_traits< abstract::neighborhood< Exact > >	??
oln::struct_elt_traits< abstract::struct_elt< Exact > >	??
mlc::switch_< Cond, case_< Compare, Ret >, Default >	
mlc::switch_< Cond, case_< Compare, Ret, Cases >, Default >	
oln::topo::tarjan::tarjan_set< I, aux_data_type >	??
oln::morpho::fast::tarjan::tarjan_set< T, ATTRIBUTE, Env >	??
oln::topo::tarjan::tarjan_traits< flat_zone< T, DestType, A, Exact > >	??
oln::utils::timer	??
ntg::builtin::to_ntg< T >	
ntg::builtin::to_ntg< char >	
ntg::builtin::to_ntg< signed char >	
ntg::builtin::to_ntg< signed int >	
ntg::builtin::to_ntg< signed long >	
ntg::builtin::to_ntg< signed short >	
ntg::builtin::to_ntg< unsigned char >	
ntg::builtin::to_ntg< unsigned int >	
ntg::builtin::to_ntg< unsigned long >	
ntg::builtin::to_ntg< unsigned short >	
mlc::top	
mlc::true_type	
true_type	
mlc::returns_bool_< true >	
oln::io::internal::try_readers< R, T >	??
oln::io::internal::try_readers< ReadNone, T >	??
oln::io::internal::try_stream_wrappers_in< W, T, Reader >	??
oln::io::internal::try_stream_wrappers_in< StreamNone, T, Reader >	??
oln::io::internal::try_stream_wrappers_out< W, T, Writer >	??
oln::io::internal::try_stream_wrappers_out< StreamNone, T, Reader >	??
oln::io::internal::try_writers< W, T >	??
oln::io::internal::try_writers< WriteNone, T >	??
ret	
ntg::range< T, interval, behavior >	??
ret	
ntg::cycle	
mlc::typeadj< T >	
mlc::typeadj< const T & >	
mlc::typeadj< const T >	
mlc::typeadj< T & >	
ntg::internal::typetraits< T >	??
ntg::internal::typetraits< cycle< T, interval > >	
ntg::internal::typetraits< enum_value< E > >	
ntg::internal::typetraits< float_value< E > >	
ntg::internal::typetraits< int_value< E > >	??
ntg::internal::typetraits< range< T, interval, behavior > >	
ntg::internal::typetraits< real_value< E > >	??
ntg::internal::typetraits< sint_value< E > >	
ntg::internal::typetraits< uint_value< E > >	
ntg::internal::typetraits< vect_value< E > >	
ntg::type_traits< T >	??
ntg::internal::typetraits< bin >::build_value_type< E >	

```

ntg::internal::typetraits< bool >
ntg::internal::typetraits< char >::build_value_type< E >
ntg::internal::typetraits< color< ncomps, qbits, color_system > >
ntg::internal::typetraits< double >::build_value_type< E >
ntg::internal::typetraits< enum_value< bin > > . . . . . ??
    ntg::internal::typetraits< bin >
ntg::internal::typetraits< float >::build_value_type< E >
ntg::internal::typetraits< float_value< double > > . . . . . ??
    ntg::internal::typetraits< double >
ntg::internal::typetraits< float_value< float > > . . . . . ??
    ntg::internal::typetraits< float >
ntg::internal::typetraits< int_s< nbits, behavior > >::build_value_type< E >
ntg::internal::typetraits< int_u< nbits, behavior > >::build_value_type< E >
ntg::internal::typetraits< int_value< E > >
ntg::internal::typetraits< real_value< E > >
ntg::internal::typetraits< signed char >::build_value_type< E >
ntg::internal::typetraits< signed int >::build_value_type< E >
ntg::internal::typetraits< signed long >::build_value_type< E >
ntg::internal::typetraits< signed short >::build_value_type< E >
ntg::internal::typetraits< sint_value< char > > . . . . . ??
    ntg::internal::typetraits< char >
ntg::internal::typetraits< sint_value< int_s< nbits, behavior > > > . . . . . ??
    ntg::internal::typetraits< int_s< nbits, behavior > >
ntg::internal::typetraits< sint_value< signed char > > . . . . . ??
    ntg::internal::typetraits< signed char >
ntg::internal::typetraits< sint_value< signed int > > . . . . . ??
    ntg::internal::typetraits< signed int >
ntg::internal::typetraits< sint_value< signed long > > . . . . . ??
    ntg::internal::typetraits< signed long >
ntg::internal::typetraits< sint_value< signed short > > . . . . . ??
    ntg::internal::typetraits< signed short >
ntg::internal::typetraits< uint_value< int_u< nbits, behavior > > > . . . . . ??
    ntg::internal::typetraits< int_u< nbits, behavior > >
ntg::internal::typetraits< uint_value< unsigned char > > . . . . . ??
    ntg::internal::typetraits< unsigned char >
ntg::internal::typetraits< uint_value< unsigned int > > . . . . . ??
    ntg::internal::typetraits< unsigned int >
ntg::internal::typetraits< uint_value< unsigned long > > . . . . . ??
    ntg::internal::typetraits< unsigned long >
ntg::internal::typetraits< uint_value< unsigned short > > . . . . . ??
    ntg::internal::typetraits< unsigned short >
ntg::internal::typetraits< unsigned char >::build_value_type< E >
ntg::internal::typetraits< unsigned int >::build_value_type< E >
ntg::internal::typetraits< unsigned long >::build_value_type< E >
ntg::internal::typetraits< unsigned short >::build_value_type< E >
ntg::internal::typetraits< value< E > >
ntg::internal::typetraits< value< E > > . . . . . ??
ntg::internal::typetraits< vec< 2, T, cplx< R, T > > > . . . . . ??
    ntg::internal::typetraits< cplx< R, T > >

```

ntg::internal::typetraits< vec< N, T, Self > >::build_value_type< E >	
ntg::internal::typetraits< vect_value< vec< N, T, Self > > >	??
ntg::internal::typetraits< vec< N, T, Self > >	
uint_value	
ntg::int_u	
mlc::undefined	
ntg::internal::undefined_traits	
ntg::unsafe	??
ntg::unsafe::get< T >	
ntg::builtin::unsigned_cumul_trait< T >	
ntg::builtin::unsigned_cumul_trait< char >	
ntg::builtin::unsigned_cumul_trait< signed char >	
ntg::builtin::unsigned_cumul_trait< signed long >	
ntg::builtin::unsigned_cumul_trait< unsigned char >	
ntg::builtin::unsigned_largest_trait< T >	
ntg::builtin::unsigned_largest_trait< signed long >	
ntg::builtin::unsigned_trait< T >	
ntg::builtin::unsigned_trait< char >	
ntg::builtin::unsigned_trait< signed char >	
ntg::builtin::unsigned_trait< signed int >	
ntg::builtin::unsigned_trait< signed long >	
ntg::builtin::unsigned_trait< signed short >	
mlc::utest< N >	
oln::io::internal::utils	??
oln::convert::value_to_point< Argument_type, Exact >::doit_binary< O, I >	??
oln::convert::value_to_point< Argument_type, Exact >::doit_not_binary< O, I >	??
ntg::builtin::value_type< T, E >	
ntg::builtin::value_type< char, E >	
ntg::builtin::value_type< signed short, E >	
ntg::builtin::value_type< signedchar, E >	
ntg::builtin::value_type< signedint, E >	
ntg::builtin::value_type< signedlong, E >	
vec	
ntg::cplx< rect, T >	
vect_value	
ntg::vec	
vect_value	
ntg::cplx< polar, T >	
vect_value	
ntg::color< ncomps, qbits, color_system >	??
oln::morpho::internal::watershed_con_point_handler_	
oln::morpho::internal::watershed_seg_point_handler_	
oln::internal::wavelet_coeffs_< T, N, Self >	??
oln::transforms::coiflet2	??
oln::transforms::coiflet4	??
oln::transforms::coiflet6	??
oln::transforms::daub10	??
oln::transforms::daub12	??
oln::transforms::daub20	??
oln::transforms::daub4	??
oln::transforms::daub6	??
oln::transforms::daub8	??
oln::transforms::haar	??

```

oln::internal::wavelet_coeffs_< ntg::float_d, 10, daub10 > . . . . . ??
oln::internal::wavelet_coeffs_< ntg::float_d, 12, coiflet4 > . . . . . ??
oln::internal::wavelet_coeffs_< ntg::float_d, 12, daub12 > . . . . . ??
oln::internal::wavelet_coeffs_< ntg::float_d, 18, coiflet6 > . . . . . ??
oln::internal::wavelet_coeffs_< ntg::float_d, 2, haar > . . . . . ??
oln::internal::wavelet_coeffs_< ntg::float_d, 20, daub20 > . . . . . ??
oln::internal::wavelet_coeffs_< ntg::float_d, 4, daub4 > . . . . . ??
oln::internal::wavelet_coeffs_< ntg::float_d, 6, coiflet2 > . . . . . ??
oln::internal::wavelet_coeffs_< ntg::float_d, 6, daub6 > . . . . . ??
oln::internal::wavelet_coeffs_< ntg::float_d, 8, daub8 > . . . . . ??
oln::window_base_friend_traits< abstract::neighborhood< Exact > > . . . . . ??
oln::window_base_friend_traits< abstract::w_window< Exact > > . . . . . ??
oln::window_base_friend_traits< abstract::window< Exact > > . . . . . ??
oln::winitr< Win >
oln::winneighb< Win >
ntg::internal::with_arith< T > . . . . . ??
oln::io::internal::writers_trier . . . . . ??
mlc::internal::x_< T >
mlc::internal::x_< void >
ntg::yiq_traits< yiq_I >
ntg::yiq_traits< yiq_Q >
ntg::yuv_traits< yuv_U >
ntg::yuv_traits< yuv_V >
oln::topo::combinatorial_map::internal::zeta
oln::io::gz::zomanip< T > . . . . . ??
const const &
coord
vector< U >
vector< V >
Sup
  oln::abstract::window_base< Sup, Exact > . . . . . ??
T
  mlc::internal::wrap

```


Chapter 3

Olena Class Index

3.1 Olena Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

ntg::internal::_from_float< n, ncomps, qbits, color_system >	??
ntg::internal::_to_float< n, ncomps, qbits, color_system >	??
oln::topo::combinatorial_map::internal::alpha< U >	??
ntg::any< E > (Top of static hierarchy)	??
oln::topo::combinatorial_map::internal::any< Inf >	??
ntg::any_ntg< E > (Top of the ntg types hierarchy)	??
ntg::internal::any_ntg< E > (Bridge to internal for ntg type hierarchy)	??
oln::topo::combinatorial_map::internal::anyfunc< U, V, Inf >	??
oln::io::internal::anything (Anything)	??
oln::morpho::attr::attr_traits< ball_parent_change< I, Exact > > (Trait specialization for the ball_parent_change attribute)	??
oln::morpho::attr::attr_traits< ball_type< I, Exact > > (Trait specialization for the ball attribute)	??
oln::morpho::attr::attr_traits< box_type< I, Exact > > (Trait specialization for the box attribute)	??
oln::morpho::attr::attr_traits< card_full_type< I, T, Exact > > (Trait specialization for card_full attribute)	??
oln::morpho::attr::attr_traits< card_type< T, Exact > > (Trait specialization for card attribute)	??
oln::morpho::attr::attr_traits< cube_type< I, Exact > > (Trait specialization for the cube attribute)	??
oln::morpho::attr::attr_traits< dist_type< I, Exact > > (Trait specialization for the dist attribute)	??
oln::morpho::attr::attr_traits< height_type< T, Exact > > (Trait specialization for the height attribute)	??
oln::morpho::attr::attr_traits< integral_type< T, Exact > > (Trait specialization for the integral attribute)	??
oln::morpho::attr::attr_traits< maxvalue_type< T, Exact > > (Trait specialization for the max-value attribute)	??
oln::morpho::attr::attr_traits< minvalue_type< T, Exact > > (Trait specialization for the min-value attribute)	??
oln::morpho::attr::attr_traits< other_image< Dad, I, Exact > > (Trait specialization for the other_image attribute)	??
oln::morpho::attr::attribute< Exact > (Attribute abstract class)	??
oln::morpho::attr::ball_type< I, Exact > (Ball attribute)	??
oln::abstract::behavior< Exact >	??
oln::topo::combinatorial_map::internal::beta< U, V > (This function must be built using assign)	??

oln::abstract::binary_image< Exact >	??
oln::abstract::binary_image_with_dim< Dim, Exact >	??
oln::topo::inter_pixel::bkd_dir_iter< Dim, Exact > (Backward iterator on direction)	??
oln::bkd_iter1d< Exact > (Backward Iterator of 1 dimension)	??
oln::bkd_iter2d< Exact >	??
oln::bkd_iter3d< Exact >	??
oln::convert::bound< Output, Exact >	??
oln::morpho::attr::box_type< I, Exact > (Box attribute)	??
oln::utils::buffer (Buffer used for MD5 data type abstraction)	??
oln::morpho::attr::card_type< T, Exact > (Cardinal attribute)	??
oln::convert::cast< Output, Exact >	??
oln::topo::chamfer< T >	??
oln::morpho::attr::change_exact< integral_type< T, OldExact >, NewExact > (Change the exact type of an attribute)	??
oln::topo::combinatorial_map::cmap< I >	??
oln::morpho::cmp_queue_elt< T >	??
oln::transforms::coiflet2 (Coifman wavelet coefficients)	??
oln::transforms::coiflet4 (Coifman wavelet coefficients)	??
oln::transforms::coiflet6 (Coifman wavelet coefficients)	??
ntg::color< ncomps, qbits, color_system > (Generic type for color)	??
oln::convert::abstract::color_conversion< icomps, iqbits, icolor, ocomps, oqbits, ocolor, Exact >	??
oln::morpher::color_morpher< SrcType, Exact >	??
oln::morpher::color_morpher< const SrcType, Exact > (The specialized version for 'const' declared images)	??
oln::morpher::color_mute< T, N >	??
oln::morpher::color_mute< ntg::color< nbcomps_, nbits_, color_system >, N > (Specialized version for ntg::color)	??
oln::convert::internal::compconv1_< C, UF >	??
oln::convert::internal::compconv2_< C, BF >	??
oln::internal::compose_bu_< F1, F2 >	??
oln::internal::compose_ub_< F1, F2 >	??
oln::internal::compose_uu_< F1, F2 >	??
oln::snakes::continuity_energy< I >	??
oln::convert::abstract::conversion< Exact, Base >	??
oln::convert::abstract::conversion_from_type_to_type< Argument_Type, Result_Type, Exact, Base > (Base class if both input and output types of the conversion are fixed)	??
oln::convert::abstract::conversion_to_type< Result_Type, Exact, Base > (Base class for the conversion to a specific type)	??
oln::convert::convoutput< ConvType, Base, InputType >	??
oln::morpho::attr::cube_type< I, Exact > (Cube attribute)	??
oln::snakes::curvature_energy< I >	??
ntg::cycle_behavior (Apply a modulus when an overflow occurs)	??
ntg::data_type (Top of the hierarchy)	??
oln::abstract::data_type_image< Exact >	??
oln::abstract::data_type_image_with_dim< Dim, Exact >	??
oln::transforms::daub10 (Daubechies wavelet coefficients)	??
oln::transforms::daub12 (Daubechies wavelet coefficients)	??
oln::transforms::daub20 (Daubechies wavelet coefficients)	??
oln::transforms::daub4 (Daubechies wavelet coefficients)	??
oln::transforms::daub6 (Daubechies wavelet coefficients)	??
oln::transforms::daub8 (Daubechies wavelet coefficients)	??
oln::abstract::decimal_image< Exact >	??
oln::abstract::decimal_image_with_dim< Dim, Exact >	??

<code>ntg::internal::deduce_from_traits< Op, T, U ></code> (Find the good <code>operator_traits</code> , following a simple algorithm)	??
<code>ntg::internal::deduce_op_behavior< B1, B2 ></code> (Determine the resulting behavior of an operator return type)	??
<code>ntg::internal::default_less< T ></code>	??
<code>oln::internal::default_less< abstract::dpoint< Exact > ></code>	??
<code>oln::internal::default_less< abstract::point< Exact > ></code>	??
<code>oln::internal::default_less< coord ></code> (Default_less<coord>)	??
<code>oln::internal::default_less< dpoint1d ></code>	??
<code>oln::internal::default_less< dpoint2d ></code>	??
<code>oln::internal::default_less< dpoint3d ></code>	??
<code>ntg::internal::default_less< ntg::color< ncomps, qbits, color_system > ></code>	??
<code>oln::internal::default_less< point1d ></code>	??
<code>oln::internal::default_less< point2d ></code>	??
<code>oln::internal::default_less< point3d ></code>	??
<code>oln::internal::dim_iterate_rec_< dim, skip ></code> (Iterate over all dimensions except one)	??
<code>oln::internal::dim_iterate_rec_< dim, 0 ></code> (Specialization of <code>dim_iterate_rec_</code> with skip dimension set to 0)	??
<code>oln::internal::dim_skip_iterate_rec_< dim, skip, current ></code> (Functions used to iterate over all dimensions except one)	??
<code>oln::internal::dim_skip_iterate_rec_< dim, skip, 0 ></code> (Specialization of <code>dim_skip_iterate_rec_</code> with current dimension set to 0)	??
<code>oln::dim_traits< Dim, T, Exact ></code>	??
<code>oln::dim_traits< 1, T, Exact ></code>	??
<code>oln::dim_traits< 2, T, Exact ></code>	??
<code>oln::dim_traits< 3, T, Exact ></code>	??
<code>oln::topo::inter_pixel::internal::dir_traits< Dim ></code> (Provides the enum <code>dir</code>)	??
<code>oln::topo::inter_pixel::internal::dir_traits< 2 ></code> (Provides the enum <code>dir</code> for 2D)	??
<code>oln::morpho::attr::dist_type< I, Exact ></code> (Dist attribute)	??
<code>oln::topo::dmap< T, T2 ></code>	??
<code>oln::abstract::dpoint< Exact ></code>	??
<code>oln::dpoint1d</code>	??
<code>oln::dpoint2d</code>	??
<code>oln::dpoint3d</code>	??
<code>oln::dpoint_traits< abstract::dpoint< Exact > ></code>	??
<code>oln::dpoint_traits< dpoint1d ></code>	??
<code>oln::dpoint_traits< dpoint2d ></code>	??
<code>oln::dpoint_traits< dpoint3d ></code>	??
<code>oln::snakes::dummy_energy< I ></code> (Default external energy)	??
<code>oln::transforms::dwt< I, K ></code> (Object to compute dwt transforms)	??
<code>oln::snakes::energy< I ></code>	??
<code>oln::morpho::env::abstract::env< Exact ></code> (Top of environment hierarchy)	??
<code>oln::math::f_abs< T ></code> (Absolute value fctor)	??
<code>oln::math::internal::f_dot_product_nv< DestValue, I, J ></code> (Dot product for non-vectorial types)	??
<code>oln::math::internal::f_dot_product_v< DestValue, I, J ></code> (Dot product for vectorial types)	??
<code>oln::convert::f_hsi_to_rgb< inbits, outbits ></code>	??
<code>oln::convert::f_hsl_to_rgb< inbits, outbits ></code>	??
<code>oln::convert::f_hsv_to_rgb< inbits, outbits ></code>	??
<code>oln::f_identity< T ></code>	??
<code>oln::level::f_invert< T ></code> (Fctor to invert a value)	??
<code>oln::arith::f_logic_and</code> (Functor AND operators)	??
<code>oln::arith::f_logic_and_not</code> (Functor AND NOT operators)	??
<code>oln::arith::f_logic_not</code> (Functor NOT operator)	??
<code>oln::arith::f_logic_or</code> (Functor OR operators)	??

<code>oln::utils::f_minmax< T ></code> (Unary function that stores the min and the max)	??
<code>oln::utils::f_moments< T, C ></code> (Computes the mean, the variance and store the min, the max)	??
<code>oln::convert::f_nrgb_to_rgb< inbits, outbits ></code>	??
<code>oln::convert::f_nrgb_to_xyz< inbits, outbits ></code>	??
<code>oln::convert::f_rgb_to_hsi< inbits, outbits ></code>	??
<code>oln::convert::f_rgb_to_hsl< inbits, outbits ></code>	??
<code>oln::convert::f_rgb_to_hsv< inbits, outbits ></code>	??
<code>oln::convert::f_rgb_to_nrgb< inbits, outbits ></code>	??
<code>oln::math::f_sqr< T ></code> (Square factor)	??
<code>oln::morpho::slow::f_tarjan_map< I, D, Env ></code>	??
<code>oln::utils::internal::f_to_float< DestT, SrcT ></code>	??
<code>oln::utils::internal::f_to_float< typename ntg::color< ncomps, qbits, color_system >::float_vec_type, ntg::color< ncomps, qbits, color_system > ></code> (Specialization of the <i>f_to_float</i> struct for the colors)	??
<code>oln::convert::f_xyz_to_nrgb< inbits, outbits ></code>	??
<code>oln::morpho::internal::fast_morpho_inner< NP1, Dim, I, S, H, B, P, O ></code>	??
<code>oln::topo::tarjan::flat_zone< T, DestType, A, Exact ></code>	??
<code>oln::topo::tarjan::obsolete::flat_zone< I ></code>	??
<code>ntg::force</code> (Force the value to be assigned without checks)	??
<code>oln::convert::force< Output, Exact ></code>	??
<code>oln::topo::inter_pixel::fwd_dir_iter< Dim, Exact ></code> (Backward iterator on direction)	??
<code>oln::fwd_iter1d< Exact ></code> (Forward Iterator on image 1 dimension)	??
<code>oln::fwd_iter2d< Exact ></code> (Backward Iterator on image 2 dimension)	??
<code>oln::fwd_iter3d< Exact ></code> (Backward Iterator on image 3 dimension)	??
<code>oln::convol::fast::internal::gaussian< dim ></code> (Compute the gaussian filter)	??
<code>oln::convol::fast::internal::gaussian< 1 ></code>	??
<code>oln::convol::fast::internal::gaussian< 2 ></code>	??
<code>oln::convol::fast::internal::gaussian< 3 ></code>	??
<code>oln::morpher::abstract::generic_morpher< SrcType, Exact ></code>	??
<code>oln::io::internal::get_pnm_type< I ></code>	??
<code>oln::snakes::greedy< N, I, external_energy ></code>	??
<code>oln::transforms::haar</code> (Haar wavelet coefficients)	??
<code>oln::morpho::attr::height_type< T, Exact ></code> (Attribute working on height between components)	??
<code>oln::utils::hist_traits< Exact ></code> (Traits for <code>oln::utils::abstract::histogram</code> hierarchy)	??
<code>oln::utils::hist_traits< histogram< T, CPT, V2P, Exact > ></code> (Traits for <code>oln::utils::abstract::histogram</code> hierarchy)	??
<code>oln::utils::abstract::histogram< Exact ></code>	??
<code>oln::utils::histogram< T, CPT, V2P, Exact ></code>	??
<code>oln::utils::histogram_max< T, CPT, V2P, Exact ></code>	??
<code>oln::utils::histogram_min< T, CPT, V2P, Exact ></code>	??
<code>oln::utils::histogram_minmax< T, CPT, V2P, Exact ></code>	??
<code>oln::level::hlut< T, T2 ></code> (Look up table "id" version)	??
<code>oln::level::hlut_def< T, T2 ></code> (Look up table "default" version)	??
<code>oln::abstract::image< Exact ></code>	??
<code>oln::image< Dim, T, Impl, Exact ></code>	??
<code>oln::image1d< T, Exact ></code>	??
<code>oln::image1d< T, Exact >::mute< U ></code> (Define ret equal to <code>image1d<U></code>)	??
<code>oln::image1d_size</code>	??
<code>oln::image2d< T, Exact ></code>	??
<code>oln::image2d< T, Exact >::mute< U ></code> (Define ret equal to <code>image2d<U></code>)	??
<code>oln::image2d_size</code>	??
<code>oln::image3d< T, Exact ></code>	??
<code>oln::image3d< T, Exact >::mute< U ></code> (Define ret equal to <code>image3d<U></code>)	??
<code>oln::image3d_size</code>	??

<code>oln::impl::image_array< T, Exact ></code>	??
<code>oln::impl::image_array1d< T ></code>	??
<code>oln::impl::image_array2d< T ></code>	??
<code>oln::impl::image_array3d< T ></code>	??
<code>oln::snakes::image_energy< I ></code>	??
<code>oln::image_id< image1d< T, Exact > ></code>	??
<code>oln::image_id< image2d< T, Exact > ></code>	??
<code>oln::image_id< image3d< T, Exact > ></code>	??
<code>oln::image_id< image< Dim, T, Impl, Exact > ></code>	??
<code>oln::image_id< morpher::color_morpher< I, Exact > ></code> (Retrieve types and dimension of the color_morpher)	??
<code>oln::image_id< morpher::iter_morpher< SrcType, IterType, Exact > ></code> (Informations about the iter morpher)	??
<code>oln::image_id< morpher::piece_morpher< SrcType, Exact > ></code> (Informations about the piece morpher)	??
<code>oln::image_id< morpher::slicing_morpher< const SrcType, Exact > ></code> (Informations about the const slicing morpher)	??
<code>oln::image_id< morpher::slicing_morpher< SrcType, Exact > ></code> (Informations about the slicing morpher)	??
<code>oln::image_id< morpher::super_piece_morpher< SrcType, Exact > ></code> (Informations about the super piece morpher)	??
<code>oln::image_id< morpher::super_slicing_morpher< const SrcType, Exact > ></code> (Informations about the const super slicing morpher)	??
<code>oln::image_id< morpher::super_slicing_morpher< SrcType, Exact > ></code> (Informations about the super slicing morpher)	??
<code>oln::image_id< oln::morpher::subq_morpher< SrcType, N, Exact > ></code>	??
<code>oln::impl::image_impl< Exact ></code>	??
<code>oln::io::internal::image_reader< F, I ></code> (Read an image of type reader_id)	??
<code>oln::io::internal::image_reader< ReadPnmPlain, I ></code>	??
<code>oln::io::internal::image_reader< ReadPnmRaw, I ></code>	??
<code>oln::abstract::image_size< Exact ></code>	??
<code>oln::image_size_traits< abstract::image_size< Exact > ></code>	??
<code>oln::image_size_traits< image1d_size ></code>	??
<code>oln::image_size_traits< image2d_size ></code>	??
<code>oln::image_size_traits< image3d_size ></code>	??
<code>oln::image_traits</code>	??
<code>oln::image_traits< abstract::image_with_dim< 1, Exact > ></code>	??
<code>oln::image_traits< abstract::image_with_dim< 2, Exact > ></code>	??
<code>oln::image_traits< abstract::image_with_dim< 3, Exact > ></code>	??
<code>oln::image_traits< abstract::image_with_impl< Impl, Exact > ></code>	??
<code>oln::image_traits< abstract::image_with_type< T, Exact > ></code>	??
<code>oln::image_traits< image1d< T, Exact > ></code>	??
<code>oln::image_traits< image2d< T, Exact > ></code>	??
<code>oln::image_traits< image3d< T, Exact > ></code>	??
<code>oln::image_traits< image< Dim, T, Impl, Exact > ></code>	??
<code>oln::image_traits< morpher::color_morpher< SrcType, Exact > ></code> (Specialized version for color_morpher)	??
<code>oln::image_traits< morpher::iter_morpher< SrcType, IterType, Exact > ></code> (Traits for iter morpher)	??
<code>oln::image_traits< morpher::piece_morpher< SrcType, Exact > ></code> (Traits for piece morpher)	??
<code>oln::image_traits< morpher::slicing_morpher< SrcType, Exact > ></code> (Traits for slicing morpher)	??
<code>oln::image_traits< oln::morpher::subq_morpher< SrcType, N, Exact > ></code>	??
<code>oln::abstract::image_with_dim< 1, Exact ></code> (The specialized version for <code>image1d</code>)	??
<code>oln::abstract::image_with_dim< 2, Exact ></code> (The specialized version for <code>image2d</code>)	??

oln::abstract::image_with_dim< 3, Exact > (The specialized version for image3d)	??
oln::abstract::image_with_impl< Impl, Exact >	??
oln::abstract::image_with_type< T, Exact >	??
oln::abstract::image_with_type_with_dim_switch< Exact >	??
oln::io::internal::image_writer< F, I > (Write an image of type writer_id)	??
oln::utils::internal::img_max_size< T > (Return the size of the space needed to explore the type T)	??
oln::impl_traits< impl::image_array1d< T > >	??
oln::impl_traits< impl::image_array2d< T > >	??
oln::impl_traits< impl::image_array3d< T > >	??
oln::impl_traits< impl::image_array< T, Exact > >	??
oln::impl_traits< impl::image_impl< Exact > >	??
oln::abstract::integer_image< Exact >	??
oln::abstract::integer_image_with_dim< Dim, Exact >	??
oln::morpho::attr::integral_type< T, Exact > (Integral attribute)	??
oln::topo::inter_pixel::interpixel< I >	??
ntg::interval< lval, uval >	??
oln::abstract::iter< Exact > (Iterator)	??
oln::abstract::iter1d< Exact > (Iterator on image of 1 dimension)	??
oln::abstract::iter2d< Exact > (Iterator on image of 2 dimensions)	??
oln::abstract::iter3d< Exact >	??
oln::morpher::iter_morpher< const SrcType, IterType, Exact > (The specialized version for 'const' declared images)	??
oln::iter_traits< abstract::iter1d< Exact > > (Traits for iter::iter1d)	??
oln::iter_traits< abstract::iter2d< Exact > > (Traits for abstract::iter2d)	??
oln::iter_traits< abstract::iter3d< Exact > > (Traits for abstract::iter3d)	??
oln::iter_traits< abstract::iter< Exact > > (Traits for abstract::iter)	??
oln::iter_traits< bkd_iter1d< Exact > > (Traits for bkd_iter1d)	??
oln::iter_traits< bkd_iter2d< Exact > >	??
oln::iter_traits< fwd_iter2d< Exact > >	??
oln::iter_traits< topo::inter_pixel::internal::dir_iter_< 1, Exact > > (Traits for iterator for 1D directions)	??
oln::iter_traits< topo::inter_pixel::internal::dir_iter_< 2, Exact > > (Traits for iterator for 2D directions)	??
oln::iter_traits< topo::inter_pixel::internal::dir_iter_< 3, Exact > > (Traits for iterator for 3D directions)	??
oln::utils::key (16 bytes key)	??
max_accumulator< T >	??
oln::morpho::attr::maxvalue_type< T, Exact > (Max value attribute)	??
oln::utils::MD5 (Class used to compute a MD5 digest)	??
oln::morpho::attr::minvalue_type< T, Exact > (Min value attribute)	??
oln::mirror_behavior< Exact >	??
oln::mute< I, T >	??
oln::abstract::neighborhood< Exact > (Neighborhood)	??
oln::neighborhood1d (Neighborhood 1 dimension)	??
oln::neighborhood2d (Neighborhood 2 dimensions)	??
oln::neighborhood3d (Neighborhood 3 dimensions)	??
oln::abstract::neighborhoodnd< Exact > (Neighborhood N dimensions)	??
oln::snakes::node< I >	??
oln::topo::inter_pixel::node< I >	??
oln::abstract::non_vectorial_image< Exact >	??
oln::abstract::non_vectorial_image_with_dim< Dim, Exact >	??
oln::morpho::env::NullEnv (Useless environment)	??

<code>ntg::internal::operator_traits< Op, T, U ></code> (Give return type for operators, depending on the input types)	??
<code>ntg::internal::optraits< T ></code> (Associates functions to types)	??
<code>ntg::internal::optraits< real_value< E > ></code> (Implement common operators for scalars)	??
<code>oln::morpho::attr::other_image< Dad, I, Exact ></code> (Metaclass used to change attribute behavior)	??
<code>oln::morpho::attr::other_point</code> (Metaclass used to change attribute behavior)	??
<code>oln::morpho::env::OtherImageEnv< I ></code> (Environment containing image)	??
<code>oln::convert::abstract::internal::output< Base, T ></code> (Retrieve the result type of a conversion)	??
<code>oln::morpho::env::ParentEnv< I ></code> (Environment containing point)	??
<code>oln::morpher::piece_morpher< SrcType, Exact ></code> (The default piece morpher class)	??
<code>oln::morpher::piece_morpher< const SrcType, Exact ></code> (The specialized version for 'const' images)	??
<code>oln::io::internal::pnm_read_data< V, R ></code>	??
<code>oln::io::internal::pnm_read_data< PnmBinary, ReadPnmPlain ></code>	??
<code>oln::io::internal::pnm_read_data< PnmBinary, ReadPnmRaw ></code>	??
<code>oln::io::internal::pnm_read_data< PnmInteger, ReadPnmPlain ></code>	??
<code>oln::io::internal::pnm_read_data< PnmInteger, ReadPnmRaw ></code>	??
<code>oln::io::internal::pnm_read_data< PnmVectorial, ReadPnmPlain ></code>	??
<code>oln::io::internal::pnm_reader< R, Dim, V, I ></code>	??
<code>oln::io::internal::pnm_reader< R, 2, PnmBinary, I ></code>	??
<code>oln::io::internal::pnm_reader< R, 2, PnmInteger, I ></code>	??
<code>oln::io::internal::pnm_reader< R, 2, PnmVectorial, I ></code> (Class <code>pnm_reader<R, 2, PnmVectorial, I></code>)	??
<code>oln::io::internal::pnm_reader< ReadPnmRaw, 3, P, I ></code>	??
<code>oln::io::internal::pnm_write</code>	??
<code>oln::io::internal::pnm_write_data< V, R ></code>	??
<code>oln::io::internal::pnm_write_data< PnmBinary, WritePnmPlain ></code>	??
<code>oln::io::internal::pnm_write_data< PnmBinary, WritePnmRaw ></code>	??
<code>oln::io::internal::pnm_write_data< PnmInteger, WritePnmPlain ></code>	??
<code>oln::io::internal::pnm_write_data< PnmInteger, WritePnmRaw ></code>	??
<code>oln::io::internal::pnm_write_data< PnmVectorial, WritePnmPlain ></code>	??
<code>oln::io::internal::pnm_write_data< PnmVectorial, WritePnmRaw ></code>	??
<code>oln::io::internal::pnm_writer< W, 2, PnmBinary, I ></code>	??
<code>oln::io::internal::pnm_writer< W, 2, PnmInteger, I ></code>	??
<code>oln::io::internal::pnm_writer< W, 2, PnmVectorial, I ></code>	??
<code>oln::io::internal::pnm_writer< WritePnmRaw, 3, P, I ></code>	??
<code>oln::abstract::point< Exact ></code>	??
<code>oln::point1d</code>	??
<code>oln::point2d</code>	??
<code>oln::point3d</code>	??
<code>oln::point_traits< abstract::point< Exact > ></code>	??
<code>oln::point_traits< point1d ></code>	??
<code>oln::point_traits< point2d ></code>	??
<code>oln::point_traits< point3d ></code>	??
<code>ntg::range< T, interval, behavior ></code> (Restrict the interval of a type)	??
<code>oln::io::internal::readers_trier</code> (Readers trier functor, helper for stream_wrappers)	??
<code>oln::convol::fast::internal::recursivefilter_coef< FloatT ></code> (Data structure for coefficients used for a recursive filter call)	??
<code>oln::replicate_behavior< Exact ></code> (Replicate_behavior)	??
<code>ntg::internal::ret_behavior_if< need_check, Ret ></code> (Determine the behavior to use depending on check requirements)	??
<code>ntg::saturate</code> (Bound values to the nearest limit when an overflow occurs)	??
<code>oln::convol::internal::se_array< Size, S ></code>	??
<code>oln::utils::se_stat< Sum, Var ></code> (Compute the variance and the mean within a window)	??

<code>oln::snakes::segment< I ></code>	??
<code>oln::utils::select_distrib_sort< reverse ></code>	??
<code>oln::morpher::slicing_morpher< SrcType, Exact ></code> (The default slicing morpher class)	??
<code>oln::morpher::slicing_morpher< const SrcType, Exact ></code> (The specialized version for 'const' images)	??
<code>oln::snakes::snake< algorithm ></code>	??
<code>oln::morpho::sort_dimensions< E ></code> (Functor to sort dimensions)	??
<code>oln::morpho::internal::stat< I, E, V ></code> (Min and Max on a structuring element)	??
<code>oln::io::internal::stream_wrapper< W ></code>	??
<code>oln::io::internal::stream_wrapper< StreamFile ></code>	??
<code>oln::io::internal::stream_wrapper< StreamGz ></code>	??
<code>oln::io::internal::stream_wrappers_find_files< W ></code> (Find files compatible with the given root, extension free filename)	??
<code>oln::io::internal::stream_wrappers_find_files< StreamNone ></code> (Do nothing because of the type of stream)	??
<code>oln::convert::stretch< Output, Exact ></code>	??
<code>ntg::strict</code> (Strict checking, abort in there is a problem)	??
<code>oln::abstract::struct_elt< Exact ></code>	??
<code>oln::struct_elt_traits< abstract::neighborhood< Exact > ></code> (Traits for <code>abstract::neighborhood</code>)	??
<code>oln::struct_elt_traits< abstract::neighborhoodnd< Exact > ></code> (Traits for <code>abstract::neighborhoodnd</code>)	??
<code>oln::struct_elt_traits< abstract::struct_elt< Exact > ></code> (Traits for <code>abstract::struct_elt</code>)	??
<code>oln::struct_elt_traits< abstract::w_window< Exact > ></code> (Traits for <code>abstract::w_window</code>)	??
<code>oln::struct_elt_traits< abstract::w_windownd< Exact > ></code> (Traits for <code>abstract::w_windownd</code>)	??
<code>oln::struct_elt_traits< abstract::window< Exact > ></code>	??
<code>oln::struct_elt_traits< abstract::window_base< Sup, Exact > ></code> (Traits for <code>abstract::window_base</code>)	??
<code>oln::struct_elt_traits< abstract::windownd< Exact > ></code> (Traits for <code>abstract::windownd</code>)	??
<code>oln::struct_elt_traits< neighborhood1d ></code> (Traits for <code>neighborhood1d</code>)	??
<code>oln::struct_elt_traits< neighborhood2d ></code> (Traits for <code>neighborhood2d</code>)	??
<code>oln::struct_elt_traits< neighborhood3d ></code> (Traits for <code>neighborhood3d</code>)	??
<code>oln::struct_elt_traits< w_window1d< T > ></code> (Traits for <code>w_windownd1d</code>)	??
<code>oln::struct_elt_traits< w_window2d< T > ></code> (Traits for <code>w_windownd2d</code>)	??
<code>oln::struct_elt_traits< w_window3d< T > ></code> (Traits for <code>w_windownd3d</code>)	??
<code>oln::struct_elt_traits< window1d ></code> (Traits for <code>window1d</code>)	??
<code>oln::struct_elt_traits< window2d ></code> (Traits for <code>window2d</code>)	??
<code>oln::struct_elt_traits< window3d ></code> (Traits for <code>window3d</code>)	??
<code>oln::morpher::subq_morpher< SrcType, N, Exact ></code> (Sub quantify an image)	??
<code>oln::morpher::super_color_morpher< SrcType, Exact ></code> (Abstract <code>color_morpher</code> class used for code factorization)	??
<code>oln::morpher::super_piece_morpher< SrcType, Exact ></code> (Abstract piece morpher class used for code factorization)	??
<code>oln::morpher::super_slicing_morpher< SrcType, Exact ></code> (Abstract slicing morpher class used for code factorization)	??
<code>oln::topo::tarjan::abstract::tarjan< Exact ></code> (Top of tarjan hierarchy)	??
<code>oln::topo::tarjan::tarjan_set< I, aux_data_type ></code>	??
<code>oln::morpho::fast::tarjan::tarjan_set< T, ATTRIBUTE, Env ></code> (Struct that contains everything to compute an attribute opening or closing)	??
<code>oln::topo::tarjan::tarjan_traits< flat_zone< T, DestType, A, Exact > ></code> (Traits specialization for <code>flat_zone</code>)	??
<code>oln::topo::tarjan::abstract::tarjan_with_attr< Exact ></code> (Abstract class to perform a tarjan algorithm on an image with attribute computing)	??
<code>oln::level::threshold< Input, Output, Exact ></code> (Threshold the value of the image)	??
<code>oln::utils::timer</code>	??

<code>oln::io::internal::try_readers< R, T ></code> (Image's reader)	??
<code>oln::io::internal::try_readers< ReadNone, T ></code> (Reader for image of type None)	??
<code>oln::io::internal::try_stream_wrappers_in< W, T, Reader ></code> (Read a stream)	??
<code>oln::io::internal::try_stream_wrappers_in< StreamNone, T, Reader ></code> (Read a stream of type None)	??
<code>oln::io::internal::try_stream_wrappers_out< W, T, Writer ></code> (Write a stream)	??
<code>oln::io::internal::try_stream_wrappers_out< StreamNone, T, Reader ></code> (Write a stream of type None)	??
<code>oln::io::internal::try_writers< W, T ></code> (Image's writer)	??
<code>oln::io::internal::try_writers< WriteNone, T ></code> (Reader for image of type None)	??
<code>ntg::type_traits< T ></code> (Associates properties and methods to types)	??
<code>ntg::internal::typetraits< T ></code> (Associates types to types)	??
<code>ntg::unsafe</code> (No check performed)	??
<code>oln::io::internal::utils</code> (Utils for io (get extension of a file))	??
<code>ntg::value< E ></code> (Concrete value storage class)	??
<code>oln::value_behavior< T, Exact ></code>	??
<code>oln::convert::value_to_point< Argument_type, Exact ></code>	??
<code>oln::convert::value_to_point< Argument_type, Exact >::doit_binary< O, I ></code> (Convert a binary to a point)	??
<code>oln::convert::value_to_point< Argument_type, Exact >::doit_not_binary< O, I ></code> (Convert a non vectorial to a point)	??
<code>oln::convert::value_to_point< ntg::color< 3, Qbits, S >, Exact ></code>	??
<code>oln::abstract::vectorial_image< Exact ></code>	??
<code>oln::abstract::w_window< Exact ></code> (Weight Window)	??
<code>oln::w_window1d< T ></code> (Window 1 dimension with weight)	??
<code>oln::w_window2d< T ></code> (Window 2 dimensions with weight)	??
<code>oln::w_window3d< T ></code> (Window 3 dimensions with weight)	??
<code>oln::abstract::w_windownd< Exact ></code> (Weight Window N dimensions)	??
<code>oln::internal::wavelet_coeffs< T, N, Self ></code> (Wavelet coefficient data structure)	??
<code>oln::abstract::window< Exact ></code> (Window)	??
<code>oln::window1d</code> (Window 1 dimension)	??
<code>oln::window2d</code> (Window 2 dimensions)	??
<code>oln::window3d</code> (Window 3 dimensions)	??
<code>oln::abstract::window_base< Sup, Exact ></code> (Window Base)	??
<code>oln::window_base_friend_traits< abstract::neighborhood< Exact > ></code>	??
<code>oln::window_base_friend_traits< abstract::w_window< Exact > ></code>	??
<code>oln::window_base_friend_traits< abstract::window< Exact > ></code>	??
<code>oln::abstract::windownd< Exact ></code> (Window N dimensions)	??
<code>ntg::internal::with_arith< T ></code> (Return a type which supports inc and dec)	??
<code>oln::io::internal::writers_trier</code> (Writers trier functor, helper for stream_wrappers)	??
<code>oln::io::gz::zfilebuf</code> (Performs operation on compressed files)	??
<code>oln::io::gz::zfilestream_common</code> (Define an interface for compressed file stream manipulation)	??
<code>oln::io::gz::zifstream</code> (Read only zstream)	??
<code>oln::io::gz::zomanip< T ></code> (Define a pair func / val to perform manipulation on zofstream)	??

Chapter 4

Olena File Index

4.1 Olena File List

Here is a list of all documented files with brief descriptions:

1d.hh	??
1d.hxx	??
2d.hh	??
2d.hxx	??
3d.hh	??
3d.hxx	??
abstract_hierarchy.hh	??
accum.hh	??
integre/ntg/all.hh	??
metallic/mlc/array/all.hh	??
allfunc.hh	??
alpha.hh	??
anyfunc.hh	??
apply.hh	??
attribute_closing_opening.hh	??
attribute_closing_opening_map.hh	??
attribute_closing_opening_map.hxx	??
attribute_union_find.hh	??
attributes.hh	??
base.hh	??
olena/oln/basics.hh	??
olena/oln/convert/basics.hh	??
olena/oln/io/basics.hh	??
integre/ntg/basics.hh	??
basics1d.hh	??
basics2d.hh	??
basics3d.hh	??
olena/oln/core/abstract/behavior.hh	??
olena/oln/core/behavior.hh	??
integre/ntg/real/behavior.hh	??
beta.hh	??
bin.hh	??
enum/bin.hh	??

binary_image.hh	??
bkd-dir-iter.hh	??
bkd_iter1d.hh	??
bkd_iter2d.hh	??
bkd_iter3d.hh	??
bool.hh	??
bound.hh	??
box.hh	??
box.hxx	??
buffer.hh	??
buffer.hxx	??
builtin_bool.hh	??
builtin_float.hh	??
builtin_int.hh	??
builtin_properties.hh	??
olena/oln/convert/cast.hh	??
integre/ntg/utls/cast.hh	??
cc.hh	??
closing.hh	??
cmap.hh	??
cmp.hh	??
color/color.hh	??
color.hh	??
color_morpher.hh	??
colorconv.hh	??
compare.hh	??
compose.hh	??
connected.hh	??
integre/ntg/core/contract.hh	??
metallic/mlc/contract.hh	??
abstract/conversion.hh	??
conversion.hh	??
conversion_ng_se.hh	??
convolution.hh	??
coord.hh	??
copy.hh	??
cplx.hh	??
vect/cplx.hh	??
cplx_representation.hh	??
cycle.hh	??
real/cycle.hh	??
debug.hh	??
dilation.hh	??
dir-iter.hh	??
dir.hh	??
dmap.hh	??
dmap.hxx	??
dpoint.hh	??
dpoint1d.hh	??
dpoint1d.hxx	??
dpoint2d.hh	??
dpoint2d.hxx	??
dpoint3d.hh	??
dpoint3d.hxx	??

dwt.hh	??
energies.hh	??
energies.hxx	??
enum_value.hh	??
environments.hh	??
erosion.hh	??
extrema.hh	??
extrema.hxx	??
extrema_killer.hh	??
fake.hh	??
fast_gaussian.hh	??
fast_gaussian.hxx	??
fast_gaussian_coefficient.hh	??
fast_morpho.hh	??
fast_morpho.hxx	??
fft.hh	??
file.hh	??
fill.hh	??
flat-zone.hh	??
float.hh	??
fold.hh	??
force.hh	??
fwd-dir-iter.hh	??
fwd_iter1d.hh	??
fwd_iter2d.hh	??
fwd_iter3d.hh	??
generate.hh	??
generic_morpher.hh	??
geodesic_dilation.hh	??
geodesic_erosion.hh	??
global_ops.hh	??
global_ops_defs.hh	??
global_ops_traits.hh	??
gradient.hh	??
greedy.hh	??
greedy.hxx	??
gz.hh	??
gz_stream.hh	??
histogram.hh	??
hit_or_miss.hh	??
hsi.hh	??
hsl.hh	??
hsv.hh	??
core/abstract/image.hh	??
core/image.hh	??
io/image.hh	??
image1d.hh	??
image1d_size.hh	??
image2d.hh	??
image2d_size.hh	??
image3d.hh	??
image3d_size.hh	??
image_array.hh	??
image_array1d.hh	??

image_array2d.hh	??
image_array3d.hh	??
image_base.hh	??
image_impl.hh	??
image_read.hh	??
image_size.hh	??
image_with_dim.hh	??
image_with_impl.hh	??
image_with_type.hh	??
image_with_type_with_dim.hh	??
image_write.hh	??
int.hh	??
int_s.hh	??
int_u.hh	??
inter-pixel.hh	??
interval.hh	??
invert.hh	??
is_a.hh	??
iter.hh	??
iter1d.hh	??
iter2d.hh	??
iter3d.hh	??
iter_morpher.hh	??
key.hh	??
key.hxx	??
lambda.hh	??
laplacian.hh	??
level.hh	??
logic.hh	??
loop.hh	??
lut.hh	??
olena/oln/core/macros.hh	??
olena/oln/math/macros.hh	??
integre/ntg/core/internal/macros.hh	??
integre/ntg/core/macros.hh	??
integre/ntg/config/math.hh	??
metallic/mlc/math.hh	??
md5.hh	??
md5.hxx	??
minmax.hh	??
nagao.hh	??
nagao.hxx	??
nd.hh	??
neighborhood.hh	??
neighborhood1d.hh	??
neighborhood2d.hh	??
neighborhood3d.hh	??
neighborhoodnd.hh	??
snakes/node.hh	??
topo/inter-pixel/node.hh	??
node.hxx	??
nrgb.hh	??
nrbxyz.hh	??
array/objs.hh	??

objs.hh	??
opdecls.hh	??
opening.hh	??
ops.hh	??
optional.hh	??
optraits_builtin_int.hh	??
optraits_real.hh	??
optraits_real_defs.hh	??
pconf.hh	??
piece_morpher.hh	??
pnm_common.hh	??
pnm_read.hh	??
pnm_read_2d.hh	??
pnm_read_3d.hh	??
pnm_read_data.hh	??
pnm_write.hh	??
pnm_write_2d.hh	??
pnm_write_3d.hh	??
pnm_write_data.hh	??
point.hh	??
point1d.hh	??
point1d.hxx	??
point2d.hh	??
point2d.hxx	??
point3d.hh	??
point3d.hxx	??
pred_succ.hh	??
predecls.hh	??
range.hh	??
real/range.hh	??
readable.hh	??
real_value.hh	??
reconstruction.hh	??
rgb.hh	??
rgbhsi.hh	??
rgbhsl.hh	??
rgbhsv.hh	??
rgbnrgb.hh	??
rgbxyz.hh	??
rgbyiq.hh	??
rgbyuv.hh	??
se.hh	??
se_neighborhood.hh	??
se_stat.hh	??
se_stat.hxx	??
se_window.hh	??
segment.hh	??
segment.hxx	??
set_level.hh	??
sigma.hh	??
slicing_morpher.hh	??
snakes_base.hh	??
snakes_base.hxx	??
special_points.hh	??

splitse.hh	??
morpho/stat.hh	??
utils/stat.hh	??
stream_wrapper.hh	??
stretch.hh	??
struct_elt.hh	??
subq_morpher.hh	??
olena/oln/config/system.hh	??
integre/ntg/config/system.hh	??
metallic/mlc/config/system.hh	??
tarjan.hh	??
tarjan_with_attr.hh	??
thickening.hh	??
thinning.hh	??
threshold.hh	??
timer.hh	??
top_hat.hh	??
traits.hh	??
traverse.hh	??
integre/ntg/core/type.hh	??
metallic/mlc/type.hh	??
type_traits.hh	??
typeadj.hh	??
typetraits_builtin_int.hh	??
union.hh	??
utils.hh	??
value.hh	??
value_to_point.hh	??
vec.hh	??
vect/vect.hh	??
vect_value.hh	??
w_window.hh	??
w_window1d.hh	??
w_window2d.hh	??
w_window3d.hh	??
w_windownd.hh	??
watershed.hh	??
watershed.hxx	??
wavelet_coeffs.hh	??
window.hh	??
window1d.hh	??
window2d.hh	??
window3d.hh	??
window_base.hh	??
windownd.hh	??
winitier.hh	??
winneighb.hh	??
xyz.hh	??
yi_q.hh	??
yuv.hh	??
zeta.hh	??

Chapter 5

Olena Page Index

5.1 Olena Related Pages

Here is a list of all related documentation pages:

Todo List	??
Deprecated List	??

Chapter 6

Olena Namespace Documentation

6.1 internal Namespace Reference

abstract internal.

Functions

- `template<class DestType, class I> mute< I, DestType >::ret create_minima_image_from_bin_ (const abstract::non_vectorial_image< I > &input)`
Create extremum image from another one.
- `template<class I> mute< I, ntg::bin >::ret ima_to_bin_ (const abstract::non_vectorial_image< I > &input)`
Create binary image from another one.

6.1.1 Detailed Description

abstract internal.

6.1.2 Function Documentation

- 6.1.2.1** `template<class DestType, class I> mute<I, DestType>::ret create_minima_image_from_bin_ (const abstract::non_vectorial_image< I > &
input)`

Create extremum image from another one.

Parameters:

DestType Type of data in the wanted image.

I Exact type of the input image.

- input Input image.

Definition at line 41 of file extrema.hxx.

```

42  {
43      oln_iter_type(I) p(input);
44      typename mute<I, DestType>::ret output(input.size());
45      for_all (p)
46          // FIXME: min() and max() should be inf() and sup()
47          // once these functions exist. Otherwise it doesn't
48          // work on float.
49          output[p] = (input[p] ?
50                      ntg_min_val(DestType) :
51                      ntg_max_val(DestType));
52      return output;
53  }
```

6.1.2.2 `template<class I> mute<I, ntg::bin>::ret ima_to_bin_ (const abstract::non_vectorial_image< I > &input)`

Create binary image from another one.

Parameters:

I Exact type of the image.

- input Input image.

Definition at line 64 of file extrema.hxx.

```

65  {
66      oln_iter_type(I) p(input);
67      typename mute<I, ntg::bin>::ret output(input.size());
68      for_all (p)
69          output[p] = (input[p] ? true : false);
70      return output;
71  }
```

6.2 io Namespace Reference

io namespace.

6.2.1 Detailed Description

io namespace.

6.3 oln Namespace Reference

oln namespace.

Classes

- struct [dpoint_traits](#)< [abstract::dpoint](#)< Exact > >
- struct [image_traits](#)< [abstract::image](#)< Exact > >
- struct [image_size_traits](#)< [abstract::image_size](#)< Exact > >
- struct [image_traits](#)< [abstract::image_with_dim](#)< 1, Exact > >
- struct [image_traits](#)< [abstract::image_with_dim](#)< 2, Exact > >
- struct [image_traits](#)< [abstract::image_with_dim](#)< 3, Exact > >
- struct [image_traits](#)< [abstract::image_with_impl](#)< Impl, Exact > >
- struct [image_traits](#)< [abstract::image_with_type](#)< T, Exact > >
- struct [iter_traits](#)< [abstract::iter](#)< Exact > >
Traits for [abstract::iter](#).
- struct [iter_traits](#)< [abstract::iter1d](#)< Exact > >
Traits for [iter::iter1d](#).
- struct [iter_traits](#)< [abstract::iter2d](#)< Exact > >
Traits for [abstract::iter2d](#).
- struct [iter_traits](#)< [abstract::iter3d](#)< Exact > >
Traits for [abstract::iter3d](#).
- struct [struct_elt_traits](#)< [abstract::neighborhood](#)< Exact > >
Traits for [abstract::neighborhood](#).
- struct [struct_elt_traits](#)< [abstract::neighborhoodnd](#)< Exact > >
Traits for [abstract::neighborhoodnd](#).
- struct [point_traits](#)< [abstract::point](#)< Exact > >
- struct [struct_elt_traits](#)< [abstract::struct_elt](#)< Exact > >
Traits for [abstract::struct_elt](#).
- struct [struct_elt_traits](#)< [abstract::w_window](#)< Exact > >
Traits for [abstract::w_window](#).
- struct [struct_elt_traits](#)< [abstract::w_windownd](#)< Exact > >
Traits for [abstract::w_windownd](#).
- struct [struct_elt_traits](#)< [abstract::window](#)< Exact > >
- struct [struct_elt_traits](#)< [abstract::window_base](#)< Sup, Exact > >
Traits for [abstract::window_base](#).
- struct [window_base_friend_traits](#)< [abstract::neighborhood](#)< Exact > >
- struct [window_base_friend_traits](#)< [abstract::window](#)< Exact > >
- struct [window_base_friend_traits](#)< [abstract::w_window](#)< Exact > >

- struct [struct_elt_traits< abstract::windownd< Exact > >](#)
Traits for [abstract::windownd](#).
- class [mirror_behavior](#)
- class [value_behavior](#)
- class [replicate_behavior](#)
[replicate_behavior](#)
- struct [iter_traits< bkd_iter1d< Exact > >](#)
Traits for [bkd_iter1d](#).
- class [bkd_iter1d](#)
Backward Iterator of 1 dimension.
- struct [iter_traits< bkd_iter2d< Exact > >](#)
- class [bkd_iter2d](#)
- struct [iter_traits< bkd_iter3d< Exact > >](#)
- class [bkd_iter3d](#)
- class [box](#)
- struct [f_identity](#)
- struct [dpoint_traits< dpoint1d >](#)
- class [dpoint1d](#)
- struct [dpoint_traits< dpoint2d >](#)
- class [dpoint2d](#)
- struct [dpoint_traits< dpoint3d >](#)
- class [dpoint3d](#)
- struct [iter_traits< fwd_iter1d< Exact > >](#)
- class [fwd_iter1d](#)
Forward Iterator on image 1 dimension.
- struct [iter_traits< fwd_iter2d< Exact > >](#)
- class [fwd_iter2d](#)
Backward Iterator on image 2 dimension.
- struct [iter_traits< fwd_iter3d< Exact > >](#)
- class [fwd_iter3d](#)
Backward Iterator on image 3 dimension.
- struct [image_id< image< Dim, T, Impl, Exact > >](#)
- struct [image_traits< image< Dim, T, Impl, Exact > >](#)
- class [image](#)
- struct [mute](#)
- struct [dim_traits](#)
- struct [image_id< image1d< T, Exact > >](#)
- struct [image_traits< image1d< T, Exact > >](#)
- class [image1d](#)
- struct [image1d::mute](#)
Define ret equal to [image1d<U>](#).
- struct [dim_traits< 1, T, Exact >](#)

- struct [image_size_traits< image1d_size >](#)
- struct [image1d_size](#)
- struct [image_id< image2d< T, Exact > >](#)
- struct [image_traits< image2d< T, Exact > >](#)
- class [image2d](#)
- struct [image2d::mute](#)

Define ret equal to image2d<U>.

- struct [dim_traits< 2, T, Exact >](#)
- struct [image_size_traits< image2d_size >](#)
- struct [image2d_size](#)
- struct [image_id< image3d< T, Exact > >](#)
- struct [image_traits< image3d< T, Exact > >](#)
- class [image3d](#)
- struct [image3d::mute](#)

Define ret equal to image3d<U>.

- struct [dim_traits< 3, T, Exact >](#)
- struct [image_size_traits< image3d_size >](#)
- struct [image3d_size](#)
- struct [impl_traits< impl::image_array< T, Exact > >](#)
- struct [impl_traits< impl::image_array1d< T > >](#)
- struct [impl_traits< impl::image_array2d< T > >](#)
- struct [impl_traits< impl::image_array3d< T > >](#)
- struct [impl_traits< impl::image_impl< Exact > >](#)
- struct [struct_elt_traits< neighborhood1d >](#)

Traits for neighborhood1d.

- class [neighborhood1d](#)

Neighborhood 1 dimension.

- struct [struct_elt_traits< neighborhood2d >](#)

Traits for neighborhood2d.

- class [neighborhood2d](#)

Neighborhood 2 dimensions.

- struct [struct_elt_traits< neighborhood3d >](#)

Traits for neighborhood3d.

- class [neighborhood3d](#)

Neighborhood 3 dimensions.

- struct [point_traits< point1d >](#)
- class [point1d](#)
- struct [point_traits< point2d >](#)
- class [point2d](#)
- struct [point_traits< point3d >](#)
- class [point3d](#)
- struct [struct_elt_traits< w_window1d< T > >](#)

Traits for w_windownd1d.

- class [w_window1d](#)
Window 1 dimension with weight.
- struct [struct_elt_traits< w_window2d< T > >](#)
Traits for w_windownd2d.
- class [w_window2d](#)
Window 2 dimensions with weight.
- struct [struct_elt_traits< w_window3d< T > >](#)
Traits for w_windownd3d.
- class [w_window3d](#)
Window 3 dimensions with weight.
- struct [struct_elt_traits< window1d >](#)
Traits for [window1d](#).
- class [window1d](#)
Window 1 dimension.
- struct [struct_elt_traits< window2d >](#)
Traits for [window2d](#).
- class [window2d](#)
Window 2 dimensions.
- struct [struct_elt_traits< window3d >](#)
Traits for [window3d](#).
- class [window3d](#)
Window 3 dimensions.
- struct **winit**
- struct **winneighb**
- struct [image_id< morpher::color_morpher< I, Exact > >](#)
Retrieve types and dimension of the color_morpher.
- struct [image_traits< morpher::color_morpher< SrcType, Exact > >](#)
Specialized version for color_morpher.
- struct **image_traits< morpher::abstract::generic_morpher< SrcType, Exact > >**
- struct [image_id< morpher::iter_morpher< SrcType, IterType, Exact > >](#)
Informations about the iter morpher.
- struct [image_traits< morpher::iter_morpher< SrcType, IterType, Exact > >](#)
Traits for iter morpher.

- struct [image_id< morpher::super_piece_morpher< SrcType, Exact > >](#)
Informations about the super piece morpher.
- struct [image_id< morpher::piece_morpher< SrcType, Exact > >](#)
Informations about the piece morpher.
- struct [image_traits< morpher::piece_morpher< SrcType, Exact > >](#)
Traits for piece morpher.
- struct [image_id< morpher::super_slicing_morpher< SrcType, Exact > >](#)
Informations about the super slicing morpher.
- struct [image_id< morpher::super_slicing_morpher< const SrcType, Exact > >](#)
Informations about the const super slicing morpher.
- struct [image_id< morpher::slicing_morpher< SrcType, Exact > >](#)
Informations about the slicing morpher.
- struct [image_id< morpher::slicing_morpher< const SrcType, Exact > >](#)
Informations about the const slicing morpher.
- struct [image_traits< morpher::slicing_morpher< SrcType, Exact > >](#)
Traits for slicing morpher.
- struct [image_id< oln::morpher::subq_morpher< SrcType, N, Exact > >](#)
- struct [image_traits< oln::morpher::subq_morpher< SrcType, N, Exact > >](#)
- struct [iter_traits< topo::inter_pixel::bkd_dir_iter< Dim, Exact > >](#)
- struct [iter_traits< topo::inter_pixel::fwd_dir_iter< Dim, Exact > >](#)
- struct [iter_traits< topo::inter_pixel::internal::dir_iter_< 1, Exact > >](#)
Traits for iterator for 1D directions.
- struct [iter_traits< topo::inter_pixel::internal::dir_iter_< 2, Exact > >](#)
Traits for iterator for 2D directions.
- struct [iter_traits< topo::inter_pixel::internal::dir_iter_< 3, Exact > >](#)
Traits for iterator for 3D directions.
- class [image_traits](#)

Typedefs

- typedef int [coord](#)
coord == int

Functions

- `template<class E> E inter (const abstract::neighborhood< E > &lhs, const abstract::neighborhood< E > &rhs)`
Compute intersection between two neighborhood.
- `template<class E> E uni (const abstract::neighborhood< E > &lhs, const abstract::neighborhood< E > &rhs)`
Compute union between two neighborhood.
- `template<class E> struct_elt_traits< E >::win_type mk_win_from_neighb (const abstract::neighborhood< E > &n)`
Construct a window from a neighborhood.
- `template<class E> E inter (const abstract::w_window< E > &lhs, const abstract::w_window< E > &rhs)`
Compute intersection between two w_windows.
- `template<class E> E uni (const abstract::w_window< E > &lhs, const abstract::w_window< E > &rhs)`
Compute union between two w_windows.
- `template<class E> E inter (const abstract::window< E > &lhs, const abstract::window< E > &rhs)`
Compute intersection between two windows.
- `template<class E> E uni (const abstract::window< E > &lhs, const abstract::window< E > &rhs)`
Compute union between two windows.
- `template<class AdaptableUnaryFun, class I> mute< I, typename AdaptableUnaryFun::result_type >::ret apply (AdaptableUnaryFun f, const abstract::image< I > &input)`
Standard unary apply procedure. Apply a function f to each element of input.
- `template<class AdaptableUnaryFun, class I> mute< I, typename AdaptableUnaryFun::result_type >::ret apply (const abstract::image< I > &input)`
Standard unary apply procedure. Apply function f to each element of input, the function is passed as a type and is instantiated. For template functions passed as template-id, one need to instantiate the function for the type of the [abstract::image](#).
- `template<class AdaptableBinaryFun, class I1, class I2> mute< I1, typename AdaptableBinaryFun::result_type >::ret apply2 (AdaptableBinaryFun f, const abstract::image< I1 > &input1, const abstract::image< I2 > &input2)`
Standard binary apply procedure. Apply function f to each element of input1 and input2.
- `template<class AdaptableBinaryFun, class I1, class I2> mute< I1, typename AdaptableBinaryFun::result_type >::ret apply2 (const abstract::image< I1 > &input1, const abstract::image< I2 > &input2)`
Standard binary apply procedure. Apply function f to each element of input1 and input2. The function is passed as a type and is instantiated. For template functions passed as template-id, one need to instantiate the function for the type of the [abstract::images](#).

- `template<template< class > class AdaptableBinaryFun, class I> mute< I, typename AdaptableBinaryFun< typename mlc::exact< I >::ret::value_type >::result_type >::ret apply2 (const abstract::image< I > &input1, const abstract::image< I > &input2)`
Standard binary apply procedure. Apply function f to each element of input1 and input2. The function is passed as a type and is instantiated. For template functions passed as template-id, one need to instantiate the function for the type of the [abstract::image](#)s.
- `template<class UnaryFun, class I> abstract::image< I > & apply_self (UnaryFun f, abstract::image< I > &input)`
Main [apply_self\(\)](#) function. Note only a UnaryFun is required, not a AdaptableUnaryFunc, because as an [abstract::image](#) is overwritten, the output type is already known.
- `template<class UnaryFun, class I> abstract::image< I > & apply_self (abstract::image< I > &input)`
Only a UnaryFun is required, not a AdaptableUnaryFunc, because as an [abstract::image](#) is overwritten, the output type is already know. The function is instantiated.
- `template<class UnaryFun, class I1, class I2> abstract::image< I1 > & apply2_self (UnaryFun f, abstract::image< I1 > &input1, const abstract::image< I2 > &input2)`
Main [apply2_exact\(\)](#) function.
- `template<class UnaryFun, class I1, class I2> abstract::image< I1 > & apply2_self (abstract::image< I1 > &input1, const abstract::image< I1 > &input2)`
The function is instantiated.
- `template<template< class, class > class UnaryFun, class I1, class I2> abstract::image< I1 > & apply2_self (abstract::image< I1 > &input1, const abstract::image< I2 > &input2)`
If the function is passed as a template-id, it is instantiated for the type of the input elements.
- `template<template< class > class UnaryFun, class I> abstract::image< I > & apply2_self (abstract::image< I > &input1, const abstract::image< I > &input2)`
If I1 == I2 and the UnaryFun has only one parameter.
- `mirror_behavior mirror_bhv ()`
To call Ctors with type inference.
- `template<class T> value_behavior< T > value_bhv (const T &value)`
To call Ctors with type inference.
- `replicate_behavior replicate_bhv ()`
To call Ctors with type inference.
- `template<class UF1, class UF2> internal::compose_uu_< UF1, UF2 > compose_uu (const UF1 &f1, const UF2 &f2)`
Compose two unary functors F1 & F2.
- `template<class UF1, class BF2> internal::compose_ub_< UF1, BF2 > compose_ub (const UF1 &f1, const BF2 &f2)`
Compose a unary functor F1 with a binary functor F2.
- `template<class BF1, class UF2> internal::compose_bu_< BF1, UF2 > compose_bu (const BF1 &f1, const UF2 &f2)`

Compose a binary functor F1 and an unary functor F2.

- `template<class AdaptableBinaryFun, class I> AdaptableBinaryFun::result_type fold (AdaptableBinaryFun f, typename mlc::typeadj< typename AdaptableBinaryFun::result_type >::mutable_val val, const abstract::image< I > &input)`

Compute $f(\dots f(f(val, i_0), i_1) \dots, i_n)$, where $i_0 \dots i_n$ are the value associated to each [abstract::image](#) point. f could return a reference or a const. Make sure VAL is assignable.

- `template<class AdaptableBinaryFun, class I> AdaptableBinaryFun::result_type fold (AdaptableBinaryFun f, const abstract::image< I > &input)`

Compute $f(\dots f(f(i_0, i_1), i_2) \dots, i_n)$, where $i_0 \dots i_n$ are the value associated to each [abstract::image](#) point. f could return a reference or a const, so make sure VAL is assignable.

- `template<class T> void allocate_data (T *&buffer, size_t s)`

Allocate an array of s elements of T type.

- `template<class T> void desallocate_data (T *&buffer)`

Free memory pointed to by buffer, then set buffer to 0.

- `template<class T> void pretreat_1d_data (T *&buffer, T *&buffer_, const image1d_size &s)`

Build an image data array with the real data and the border.

- `template<class T> void pretreat_2d_data (T *&buffer, T **&array, const image2d_size &s)`

Build an image data array with the real data and the border.

- `template<class T> void desallocate_2d_data (T **&array, const image2d_size &s)`

Free the [image2d](#) data array.

- `template<class T> void pretreat_3d_data (T *&buffer, T **&array2, T ***&array, const image3d_size &s)`

Build an image data array with the real data and the border.

- `template<class T> void desallocate_3d_data (T **&array2, T ***&array, const image3d_size &s)`

Free the [image3d](#) data array.

- `const neighborhood1d & neighb_c2 ()`

Create a neighborhood (1 dimension) with 1 element : 1.

- `neighborhood1d mk_neighb_segment (unsigned width)`

Create a neighborhood (1 dimension).

– width The width.

- `window1d mk_win_from_neighb (const neighborhood1d &n)`

Convert a window (1 dimension) to a neighborhood (1 dimension).

– n The neighborhood to convert.

- `const neighborhood2d & neighb_c4 ()`

Create a neighborhood (2 dimension) with 0,1, 1,0.

- `const neighborhood2d & neighb_c8 ()`
Create a neighborhood (2 dimension) with 4 coordinates: 0,1, 1,1, 1,0, 1,-1.
- `neighborhood2d mk_neighb_rectangle` (unsigned nrows, unsigned ncols)
Create a rectangular neighborhood (2 dimensions).
 - *nrows* Number of row.
 - *ncols* Number of column.
- `neighborhood2d mk_neighb_square` (unsigned width)
Create a square neighborhood (2 dimensions).
 - *width* Number of column and row.
- `window2d mk_win_from_neighb` (const `neighborhood2d` &n)
Convert a window (2 dimensions) to a neighborhood (2 dimensions).
 - *n* The neighborhood to convert.
- `const neighborhood3d & neighb_c6 ()`
Create a neighborhood (3 dimension) with 3 coordinates.
- `const neighborhood3d & neighb_c18 ()`
Create a neighborhood (3 dimension) with 9 coordinates.
- `const neighborhood3d & neighb_c26 ()`
Create a neighborhood (3 dimension) with 13 coordinates.
- `neighborhood3d mk_neighb_block` (unsigned nslices, unsigned nrows, unsigned ncols)
Create a block neighborhood (3 dimension).
 - *nslices* Number of slice.
 - *nrows* Number of row.
 - *ncols* Number of column.
- `neighborhood3d mk_neighb_cube` (unsigned width)
Create a cube neighborhood (3 dimension).
 - *width* Number of slice, column and row.
- `window3d mk_win_from_neighb` (const `neighborhood3d` &n)
Convert a window (3 dimensions) to a neighborhood (3 dimensions).
 - *n* The neighborhood to convert.
- `template<class I, class F> const F & traverse` (F &f, const `abstract::image< I >` &input)
Call the functor f on each point of the input image.
- `template<class F, class I> const F traverse` (const `abstract::image< I >` &input)
Create a functor f whose type is F<oln_value_type(I), F2<oln_value_type(I)> >, then call it on each point of the input image.
- `template<class I1, class I2, class F> const F & traverse2` (F &f, const `abstract::image< I1 >` &input1, const `abstract::image< I2 >` &input2)
Call functor f whose type is F on each point of the two input images.

- `template<template< class > class F, class I> const F< typename mlc::exact< I >::ret::value_type > traverse2 (const abstract::image< I > &input1, const abstract::image< I > &input2)`
Create a functor f whose type is $F<oln_value_type(I)>$, then call it on each point of the two input images.
- `template<template< class, class > class F, class I1, class I2> const F< typename mlc::exact< I1 >::ret::value_type, typename mlc::exact< I2 >::ret::value_type > traverse2 (const abstract::image< I1 > &input1, const abstract::image< I2 > &input2)`
Create a functor f whose type is $F<oln_value_type(I1), oln_value_type(I2)>$, then call it on each point of the two input images.
- `template<class T> w_window1d< T > mk_w_win_from_win (T weight, const window1d &win)`
Convert a window (1 dimension) to a w_window (1 dimension).
 - *weight* The weight to set for each element of the window.
 - *win* The window to convert.
- `template<class T> w_window2d< T > mk_w_win_from_win (T weight, const window2d &win)`
Convert a window (2 dimension) to a w_window (2 dimension).
 - *weight* The weight to set for each element of the window.
 - *win* The window to convert.
- `template<class T> w_window3d< T > mk_w_win_from_win (T weight, const window3d &win)`
Convert a window (3 dimension) to a w_window (3 dimension).
 - *weight* The weight to set for each element of the window.
 - *win* The window to convert.
- `const window1d & win_c2_only ()`
Create a window (1 dimension) of 2 elements (-1, 1).
- `const window1d & win_c2p ()`
Create a window (1 dimension) of 3 elements (-1, 0, 1).
- `window1d mk_win_segment (unsigned width)`
Create a window (1 dimension) with width elements : -width / 2, ..., 1, 2, ..., width / 2
 - *width* The width.
- `const window2d & win_c4_only ()`
Create a window (2 dimensions) of 4 elements.
- `const window2d & win_c4p ()`
Create a window (2 dimensions) of 5 elements.
- `const window2d & win_c8_only ()`
Create a window (2 dimensions) of 8 elements.
- `const window2d & win_c8p ()`
Create a window (2 dimensions) of 9 elements.
- `window2d mk_win_rectangle (unsigned nrows, unsigned ncols)`
Create a rectangular window (2 dimensions).

- *nrows* Number of row.
 - *ncols* Number of column.
- [window2d mk_win_ellipse](#) (float yradius, float xradius)
Create an ellipse window (2 dimensions).
- [window2d mk_win_square](#) (unsigned width)
Create a square window (2 dimensions).
 - *width* Number of column and row.
- [window2d mk_win_disc](#) (float radius)
Create a disc window (2 dimensions).
 - *radius* Radius of the disc.
- `const window3d & win_c6_only ()`
Create a window (3 dimensions) of 6 elements.
- `const window3d & win_c6p ()`
Create a window (3 dimensions) of 7 elements.
- `const window3d & win_c18_only ()`
Create a window (3 dimensions) of 18 elements.
- `const window3d & win_c18p ()`
Create a window (3 dimensions) of 19 elements.
- `const window3d & win_c26_only ()`
Create a window (3 dimensions) of 26 elements.
- `const window3d & win_c26p ()`
Create a window (3 dimensions) of 27 elements.
- [window3d mk_win_block](#) (unsigned nslices, unsigned nrows, unsigned ncols)
Create a block window (3 dimension).
 - *nslices* Number of slice.
 - *nrows* Number of row.
 - *ncols* Number of column.
- [window3d mk_win_ellipsoid](#) (float zradius, float yradius, float xradius)
Create an ellipsoid window (3 dimension).
 - *zradius* radius Z.
 - *yradius* radius Y.
 - *xradius* radius X.
- [window3d mk_win_cube](#) (unsigned width)
Create a cube neighborhood (3 dimension).
 - *width* Number of slice, column and row.
- [window3d mk_win_ball](#) (float radius)
Create a ball neighborhood (3 dimension).
 - *radius* The radius.

6.3.1 Detailed Description

oln namespace.

6.3.2 Function Documentation

6.3.2.1 `template<class T> void allocate_data_ (T *& buffer, size_t s)`

Allocate an array of s elements of T type.

Precondition:

$s > 0$

Definition at line 48 of file `image_array.hh`.

Referenced by `oln::impl::image_array3d< T >::border_reallocate_and_copy_()`, `oln::impl::image_array2d< T >::border_reallocate_and_copy_()`, `oln::impl::image_array1d< T >::border_reallocate_and_copy_()`, and `oln::impl::image_array< T, image_array2d< T > >::image_array()`.

```

49  {
50      precondition(s > 0);
51      buffer = new T[s];
52  }
```

6.3.2.2 `template<class AdaptableUnaryFun, class I> mute<I, typename AdaptableUnaryFun::result_type>::ret apply (AdaptableUnaryFun f, const abstract::image< I > & input) [inline]`

Standard unary *apply* procedure. Apply a function f to each element of *input*.

Sample of code : Threshold the value of the image.

```

#include <oln/basics2d.hh>
#include <oln/level/threshold.hh>
#include <ntg/all.hh>
using namespace ntg;
int main()
{
    oln::image2d<int_u8> in = oln::load(IMG_IN "lena256.pgm");
    int_u8 th      = 127;
    rgb_8 low     = rgb_8(100, 0, 0);
    rgb_8 height  = rgb_8(0, 200, 255);

    oln::image2d<rgb_8> out
        = apply(oln::level::threshold<int_u8, rgb_8 >(th, low, height),
               in);
    save(out, IMG_OUT "oln_level_threshold.ppm");
}
```



=>



Definition at line 75 of file `apply.hh`.

References `oln::abstract::image< Exact >::size()`.

Referenced by `apply()`, and `oln::level::invert()`.

```

76  {
77      typename mute<I, typename AdaptableUnaryFun::result_type>::ret
78          output(input.size());
79      oln_iter_type(I) p(input);
80      for_all(p) output[p] = f(input[p]);
81      return output;
82  }
```

6.3.2.3 `template<template< class > class AdaptableBinaryFun, class I> mute<I, typename AdaptableBinaryFun<typename mlc::exact< I >::ret::value_type>::result_type>::ret apply2 (const abstract::image< I > &input1, const abstract::image< I > &input2)`
`[inline]`

Standard binary *apply* procedure. Apply function *f* to each element of *input1* and *input2*. The function is passed as a type and is instantiated. For template functions passed as template-id, one need to instantiate the function for the type of the `abstract::images`.

Todo

FIXME: Don't we want to name these functions 'apply()' too?

Definition at line 185 of file `apply.hh`.

References `apply2()`.

```

186  {
187      // Workaround for g++-2.95 bug.
188      AdaptableBinaryFun<oln_value_type(I)> tmp;
189      return apply2(tmp, input1, input2);
190  }
```

6.3.2.4 `template<class AdaptableBinaryFun, class I1, class I2> mute< I1, typename AdaptableBinaryFun< typename mlc::exact< I1 >::ret::value_type, typename mlc::exact< I2 >::ret::value_type >::result_type >::ret oln::apply2 (const abstract::image< I1 > & input1, const abstract::image< I2 > & input2)` [inline]

Standard binary *apply* procedure. Apply function *f* to each element of *input1* and *input2*. The function is passed as a type and is instantiated. For template functions passed as template-id, one need to instantiate the function for the type of the [abstract::images](#).

Todo

FIXME: Don't we want to name these functions 'apply()' too?

Definition at line 148 of file `apply.hh`.

References `apply2()`.

```
149  {
150      return apply2(AdaptableBinaryFun(), input1, input2);
151  }
```

6.3.2.5 `template<template< class > class UnaryFun, class I> abstract::image<I>& apply2_self (abstract::image< I > & input1, const abstract::image< I > & input2)` [inline]

If *I1* == *I2* and the `UnaryFun` has only one parameter.

See also:

[apply_self\(\)](#)

Definition at line 283 of file `apply.hh`.

References `apply2_self()`.

```
284  {
285      // Workaround for g++-2.95 bug.
286      UnaryFun<oln_value_type(I)> tmp;
287      return apply2_self(tmp, input1, input2);
288  }
```

6.3.2.6 `template<template< class, class > class UnaryFun, class I1, class I2> abstract::image<I1>& apply2_self (abstract::image< I1 > & input1, const abstract::image< I2 > & input2)` [inline]

If the function is passed as a template-id, it is instantiated for the type of the input elements.

See also:

[apply_self\(\)](#)

Definition at line 270 of file `apply.hh`.

References `apply2_self()`.

```
271  {
272      // Workaround for g++-2.95 bug.
273      UnaryFun<oln_value_type(I1),oln_value_type(I2)> tmp;
274      return apply2_self(tmp, input1, input2);
275  }
```

6.3.2.7 `template<class UnaryFun, class I1, class I2> abstract::image<I1>& apply2_self
(abstract::image< I1 > & input1, const abstract::image< I1 > & input2)` [`inline`]

The function is instantiated.

See also:

[apply_self\(\)](#)

Definition at line 258 of file `apply.hh`.

References `apply_self()`.

```
259  {
260      return apply_self(UnaryFun(), input1, input2);
261  }
```

6.3.2.8 `template<class UnaryFun, class I1, class I2> abstract::image<I1>& apply2_self
(UnaryFun f, abstract::image< I1 > & input1, const abstract::image< I2 > & input2)`

Main `apply2_exact()` function.

See also:

[apply_self\(\)](#)

Definition at line 243 of file `apply.hh`.

References `oln::abstract::image< Exact >::size()`.

Referenced by `apply2_self()`.

```
245  {
246      precondition(input1.size() == input2.size());
247      oln_iter_type(I1) p(input1);
248      for_all(p) input1[p] = f(input1[p], input2[p]);
249      return input1;
250  }
```

6.3.2.9 `template<class T> void desallocate_data_ (T *& buffer)`

Free memory pointed to by `buffer`, then set `buffer` to 0.

Precondition:

`buffer != 0`

Definition at line 61 of file `image_array.hh`.

Referenced by `oln::impl::image_array3d< T >::border_reallocate_and_copy_()`, `oln::impl::image_array2d< T >::border_reallocate_and_copy_()`, and `oln::impl::image_array1d< T >::border_reallocate_and_copy_()`.

```
62  {
63      precondition(buffer != 0);
64      delete[] buffer;
65      buffer = 0; // security
66  }
```

6.3.2.10 `template<class AdaptableBinaryFun, class I> AdaptableBinaryFun::result_type fold (AdaptableBinaryFun f, const abstract::image< I > & input) [inline]`

Compute $f(\dots f(f(i_0, i_1), i_2) \dots, i_n)$, where $i_0 \dots i_n$ are the value associated to each `abstract::image` point. f could return a reference or a const, so make sure VAL is assignable.

Todo

FIXME: Ensure that `first_argument_type == result_type`.

Definition at line 69 of file `fold.hh`.

```

70  {
71      oln_iter_type(I) p(input);
72      p = begin;
73      typename mlc::typeadj<
74          typename AdaptableBinaryFun::result_type>::mutable_val val
75          = input[p];
76      for_all_remaining(p)
77          val = f(val, input[p]);
78      return val;
79  }
```

6.3.2.11 `template<class AdaptableBinaryFun, class I> AdaptableBinaryFun::result_type fold (AdaptableBinaryFun f, typename mlc::typeadj< typename AdaptableBinaryFun::result_type >::mutable_val val, const abstract::image< I > & input) [inline]`

Compute $f(\dots f(f(val, i_0), i_1) \dots, i_n)$, where $i_0 \dots i_n$ are the value associated to each `abstract::image` point. f could return a reference or a const. Make sure VAL is assignable.

Todo

FIXME: Ensure that `first_argument_type == result_type`.

Definition at line 49 of file `fold.hh`.

```

53  {
54      oln_iter_type(I) p(input);
55      for_all(p)
56          val = f(val, input[p]);
57      return val;
58  }
```

6.3.2.12 `neighborhood3d mk_neighb_block (unsigned nslices, unsigned nrows, unsigned ncols) [inline]`

Create a block neighborhood (3 dimension).

- `nslices` Number of slice.
- `nrows` Number of row.
- `ncols` Number of column.

Returns:

The new neighborhood (3d).

Precondition:

nslices ≥ 3 .
 nslices $2 \equiv 1$.
 nrows ≥ 3 .
 nrows $2 \equiv 1$.
 ncols ≥ 3 .
 ncols $2 \equiv 1$.

Definition at line 247 of file neighborhood3d.hh.

References `oln::neighborhood3d::add()`, and `coord`.

Referenced by `mk_neighb_cube()`.

```

248  {
249      precondition(nslices >= 3 && (nslices % 2) == 1);
250      precondition(nrows >= 3 && (nrows % 2) == 1);
251      precondition(ncols >= 3 && (ncols % 2) == 1);
252      neighborhood3d neighb(nrows * ncols);
253      int half_nslices = nslices / 2;
254      int half_nrows = nrows / 2;
255      int half_ncols = ncols / 2;
256      for (coord slice = - half_nslices; slice <= half_nslices; ++slice)
257          for (coord row = - half_nrows; row <= half_nrows; ++row)
258              for (coord col = (slice > 0) ? 0 : 1 ; col <= half_ncols; ++col)
259                  neighb.add(slice, row, col);
260      return neighb;
261  }
```

6.3.2.13 `neighborhood3d` `mk_neighb_cube` (unsigned *width*) [inline]

Create a cube neighborhood (3 dimension).

- width Number of slice, column and row.

Returns:

The new neighborhood (3d).

Definition at line 269 of file neighborhood3d.hh.

References `mk_neighb_block()`.

```

270  {
271      return mk_neighb_block(width, width, width);
272  }
```

6.3.2.14 `neighborhood2d` `mk_neighb_rectangle` (unsigned *nrows*, unsigned *ncols*) [inline]

Create a rectangular neighborhood (2 dimensions).

- nrows Number of row.

- `ncols` Number of column.

Returns:

The new neighborhood (2d).

Precondition:

`nrows` ≥ 3 .
`nrows` $\% 2 == 1$.
`ncols` ≥ 3 .
`ncols` $\% 2 == 1$.

Definition at line 218 of file `neighborhood2d.hh`.

References `oln::neighborhood2d::add()`, and `coord`.

Referenced by `mk_neighb_square()`.

```

219  {
220      precondition(nrows >= 3 && (nrows % 2) == 1);
221      precondition(ncols >= 3 && (ncols % 2) == 1);
222      neighborhood2d neighb(nrows * ncols);
223      int half_nrows = nrows / 2, half_ncols = ncols / 2;
224      for (coord row = - half_nrows; row <= half_nrows; ++row)
225          for (coord col = (row <= 0)? 1 : 0; col <= half_ncols; ++col)
226              neighb.add(row, col);
227      return neighb;
228  }
```

6.3.2.15 `neighborhood1d mk_neighb_segment (unsigned width)` [inline]

Create a neighborhood (1 dimension).

- `width` The width.

Returns:

The new neighborhood.

Precondition:

`width` ≥ 3 .
`width` $\% 2 == 1$.

Add elements of coordinates 1, ..., `width` / 2.

Definition at line 193 of file `neighborhood1d.hh`.

References `oln::neighborhood1d::add()`, and `coord`.

```

194  {
195      precondition(width >= 3 && (width % 2) == 1);
196      neighborhood1d neighb(width);
197      int half_ncols = width / 2;
198      for (coord col = 1; col <= half_ncols; ++col)
199          neighb.add(col);
200      return neighb;
201  }
```

6.3.2.16 [neighborhood2d](#) **mk_neighb_square** (unsigned *width*) [inline]

Create a square neighborhood (2 dimensions).

- width Number of column and row.

Returns:

The new neighborhood (2d).

Definition at line 236 of file neighborhood2d.hh.

References [mk_neighb_rectangle\(\)](#).

```

237  {
238      return mk_neighb_rectangle(width, width);
239  }
```

6.3.2.17 [template<class T> w_window3d](#)<T> **mk_w_win_from_win** (T *weight*, const window3d & *win*)

Convert a window (3 dimension) to a w_window (3 dimension).

- weight The weight to set for each element of the window.
- win The window to convert.

Returns:

The new w_window.

Definition at line 208 of file w_window3d.hh.

References [oln::w_window3d< T >::add\(\)](#), [oln::abstract::struct_elt< window3d >::card\(\)](#), and [oln::abstract::struct_elt< window3d >::dp\(\)](#).

```

209  {
210      w_window3d<T> w_win(win.card());
211      for (unsigned i = 0; i < win.card(); ++i)
212          w_win.add(win.dp(i), weight);
213      return w_win;
214  }
```

6.3.2.18 [template<class T> w_window2d](#)<T> **mk_w_win_from_win** (T *weight*, const window2d & *win*)

Convert a window (2 dimension) to a w_window (2 dimension).

- weight The weight to set for each element of the window.
- win The window to convert.

Returns:

The new w_window.

Definition at line 221 of file w_window2d.hh.

References `oln::w_window2d< T >::add()`, `oln::abstract::struct_elt< window2d >::card()`, and `oln::abstract::struct_elt< window2d >::dp()`.

```

222  {
223      w_window2d<T> w_win(win.card());
224      for (unsigned i = 0; i < win.card(); ++i)
225          w_win.add(win.dp(i), weight);
226      return w_win;
227  }
```

6.3.2.19 `template<class T> w_window1d<T> mk_w_win_from_win (T weight, const window1d & win)`

Convert a window (1 dimension) to a w_window (1 dimension).

- *weight* The weight to set for each element of the window.
- *win* The window to convert.

Returns:

The new w_window.

Definition at line 200 of file w_window1d.hh.

References `oln::w_window1d< T >::add()`, `oln::abstract::struct_elt< window1d >::card()`, and `oln::abstract::struct_elt< window1d >::dp()`.

```

201  {
202      w_window1d<T> w_win(win.card());
203      for (unsigned i = 0; i < win.card(); ++i)
204          w_win.add(win.dp(i), weight);
205      return w_win;
206  }
```

6.3.2.20 `window3d mk_win_ball (float radius) [inline]`

Create a ball neighborhood (3 dimension).

- *radius* The radius.

Returns:

The new neighborhood (3d).

Definition at line 443 of file window3d.hh.

References `mk_win_ellipsoid()`.

```

444  {
445      return mk_win_ellipsoid(radius, radius, radius);
446  }
```

6.3.2.21 window3d mk_win_block (unsigned *nslices*, unsigned *nrows*, unsigned *ncols*) [inline]

Create a block window (3 dimension).

- *nslices* Number of slice.
- *nrows* Number of row.
- *ncols* Number of column.

Returns:

The new window (3d).

Precondition:

nslices >= 3.
nslices 2 == 1.
nrows >= 3.
nrows 2 == 1.
ncols >= 3.
ncols 2 == 1.

Definition at line 363 of file window3d.hh.

References `oln::window3d::add()`, and `coord`.

Referenced by `mk_win_cube()`.

```

364  {
365      precondition(nslices >= 3 && (nslices % 2) == 1);
366      precondition(nrows >= 3 && (nrows % 2) == 1);
367      precondition(ncols >= 3 && (ncols % 2) == 1);
368      window3d win(nrows * ncols);
369      int half_nslices = nslices / 2;
370      int half_nrows = nrows / 2;
371      int half_ncols = ncols / 2;
372      for (coord slice = - half_nslices; slice <= half_nslices; ++slice)
373          for (coord row = - half_nrows; row <= half_nrows; ++row)
374              for (coord col = - half_ncols; col <= half_ncols; ++col)
375                  win.add(slice, row, col);
376      return win;
377  }
```

6.3.2.22 window3d mk_win_cube (unsigned *width*) [inline]

Create a cube neighborhood (3 dimension).

- *width* Number of slice, column and row.

Returns:

The new neighborhood (3d).

Definition at line 432 of file window3d.hh.

References `mk_win_block()`.

```

433  {
434      return mk_win_block(width, width, width);
435  }
```

6.3.2.23 window2d mk_win_disc (float *radius*) [inline]

Create a disc window (2 dimensions).

- *radius* Radius of the disc.

Returns:

The new window (2d).

Definition at line 308 of file window2d.hh.

References `mk_win_ellipse()`.

```

309  {
310      return mk_win_ellipse(radius, radius);
311  }
```

6.3.2.24 window2d mk_win_ellipse (float *yradius*, float *xradius*) [inline]

Create an ellipse window (2 dimensions).

Returns:

The new window.

The ellipse formula is :

$$\frac{x^2}{xradius^2} + \frac{y^2}{yradius^2} = 1$$

Definition at line 266 of file window2d.hh.

References `oln::window2d::add()`, and `coord`.

Referenced by `mk_win_disc()`.

```

267  {
268      precondition(yradius > 0);
269      precondition(xradius > 0);
270
271      window2d win;
272      coord ymax = (coord)roundf(yradius);
273      float yr2 = yradius * yradius;
274      for (coord y = -ymax; y <= ymax; ++y)
275      {
276          /*
277              x^2      y^2
278              ----- + ----- = 1
279              xradius^2  yradius^2
280
281          */
282          float v = 1 - y * y / yr2;
283          if (v < 0) v = 0; // Can happen because ymax has been rounded.
284          coord xmax = (coord)roundf(xradius * sqrtf(v));
285          for (coord x = -xmax; x <= xmax; ++x)
286              win.add(y, x);
287      }
288      return win;
289  }
```

6.3.2.25 [window3d](#) `mk_win_ellipsoid (float zradius, float yradius, float xradius)` [inline]

Create an ellipsoid window (3 dimension).

- *zradius* radius Z.
- *yradius* radius Y.
- *xradius* radius X.

Precondition:

zradius > 0
yradius > 0
xradius > 0

The ellipsoid formula is :

$$\frac{x^2}{xradius^2} + \frac{y^2}{yradius^2} + \frac{z^2}{zradius^2} = 1$$

Definition at line 392 of file `window3d.hh`.

References `oln::window3d::add()`, and `coord`.

Referenced by `mk_win_ball()`.

```

393 {
394     precondition(zradius > 0);
395     precondition(yradius > 0);
396     precondition(xradius > 0);
397
398     window3d win;
399     coord zmax = (coord)roundf(zradius);
400     float zr2 = zradius * zradius;
401     float yr2 = yradius * yradius;
402     for (coord z = -zmax; z <= zmax; ++z)
403     {
404         /*
405             x^2      y^2      z^2
406             ----- + ----- + ----- = 1
407             xradius^2  yradius^2  zradius^2
408         */
409
410         /* Set x to 0 in the above formula to find ymax. */
411         float v = 1 - z * z / zr2;
412         if (v < 0) v = 0; // Can happen because zmax has been rounded.
413         coord ymax = (coord)roundf(yradius * sqrtf(v));
414         for (coord y = -ymax; y <= ymax; ++y)
415         {
416             float w = v - y * y / yr2;
417             if (w < 0) w = 0; // Can happen because ymax has been rounded.
418             coord xmax = (coord)roundf(xradius * sqrtf(w));
419             for (coord x = -xmax; x <= xmax; ++x)
420                 win.add(z, y, x);
421         }
422     }
423     return win;
424 }
```

6.3.2.26 [window3d](#) `mk_win_from_neighb (const neighborhood3d & n)` [inline]

Convert a window (3 dimensions) to a neighborhood (3 dimensions).

- `n` The neighborhood to convert.

Returns:

The new window.

Definition at line 280 of file `neighborhood3d.hh`.

References `oln::window3d::add()`, `oln::abstract::neighborhood< neighborhood3d >::card()`, and `oln::abstract::neighborhood< neighborhood3d >::dp()`.

```

281  {
282      window3d win(n.card());
283      for (unsigned i = 0; i < n.card(); ++i)
284          win.add(n.dp(i));
285      return win;
286  }
```

6.3.2.27 `window2d mk_win_from_neighb (const neighborhood2d & n)` [inline]

Convert a window (2 dimensions) to a neighborhood (2 dimensions).

- `n` The neighborhood to convert.

Returns:

The new window.

Definition at line 247 of file `neighborhood2d.hh`.

References `oln::window2d::add()`, `oln::abstract::neighborhood< neighborhood2d >::card()`, and `oln::abstract::neighborhood< neighborhood2d >::dp()`.

```

248  {
249      window2d win(n.card());
250      for (unsigned i = 0; i < n.card(); ++i)
251          win.add(n.dp(i));
252      return win;
253  }
```

6.3.2.28 `window1d mk_win_from_neighb (const neighborhood1d & n)` [inline]

Convert a window (1 dimension) to a neighborhood (1 dimension).

- `n` The neighborhood to convert.

Returns:

The new window.

Definition at line 209 of file `neighborhood1d.hh`.

References `oln::window1d::add()`, `oln::abstract::neighborhood< neighborhood1d >::card()`, and `oln::abstract::neighborhood< neighborhood1d >::dp()`.

```

210  {
211      window1d win(n.card());
212      for (unsigned i = 0; i < n.card(); ++i)
213          win.add(n.dp(i));
214      return win;
215  }
```

6.3.2.29 [window2d](#) **mk_win_rectangle** (unsigned *nrows*, unsigned *ncols*) [inline]

Create a rectangular window (2 dimensions).

- *nrows* Number of row.
- *ncols* Number of column.

Returns:

The new window (2d).

Precondition:

nrows >= 3.
nrows 2 == 1.
ncols >= 3.
ncols 2 == 1.

Definition at line 246 of file window2d.hh.

References `oln::window2d::add()`, and `coord`.

Referenced by `mk_win_square()`.

```

247  {
248      precondition(nrows >= 3 && (nrows % 2) == 1);
249      precondition(ncols >= 3 && (ncols % 2) == 1);
250      window2d win(nrows * ncols);
251      int half_nrows = nrows / 2, half_ncols = ncols / 2;
252      for (coord row = - half_nrows; row <= half_nrows; ++row)
253          for (coord col = - half_ncols; col <= half_ncols; ++col)
254              win.add(row, col);
255      return win;
256  }
```

6.3.2.30 [window1d](#) **mk_win_segment** (unsigned *width*) [inline]

Create a window (1 dimension) with width elements : -width / 2, ..., 1, 2, ..., width / 2

- *width* The width.

Returns:

The new neighborhood.

Precondition:

width >= 3.
width 2 == 1.

Definition at line 203 of file window1d.hh.

References `oln::window1d::add()`, and `coord`.

```

204  {
205      precondition(width >= 3 && (width % 2) == 1);
206      window1d win(width);
207      int half_ncols = width / 2;
208      for (coord col = - half_ncols; col <= half_ncols; ++col)
209          win.add(col);
210      return win;
211  }
```

6.3.2.31 `window2d mk_win_square (unsigned width)` `[inline]`

Create a square window (2 dimensions).

- *width* Number of column and row.

Returns:

The new window (2d).

Definition at line 297 of file window2d.hh.

References `mk_win_rectangle()`.

```
298  {
299      return mk_win_rectangle(width, width);
300  }
```

6.3.2.32 `const neighborhood3d& neighb_c18 ()` `[inline]`

Create a neighborhood (3 dimension) with 9 coordinates.

Returns:

The new neighborhood.

Definition at line 192 of file neighborhood3d.hh.

References `coord`.

```
193  {
194      static const coord crd[] = { 0, 0, 1,
195                                  0, 1, -1,
196                                  0, 1, 0,
197                                  0, 1, 1,
198                                  1, -1, 0,
199                                  1, 0, -1,
200                                  1, 0, 0,
201                                  1, 0, 1,
202                                  1, 1, 0 };
203      static neighborhood3d neighb(9, crd);
204      return neighb;
205  }
```

6.3.2.33 `const neighborhood1d& neighb_c2 ()` `[inline]`

Create a neighborhood (1 dimension) with 1 element : 1.

Returns:

The new neighborhood.

Definition at line 176 of file neighborhood1d.hh.

References `coord`.

```
177  {
178      static const coord crd[] = { 1 };
179      static const neighborhood1d neighb(1, crd);
180      return neighb;
181  }
```

6.3.2.34 `const neighborhood3d& neighb_c26 () [inline]`

Create a neighborhood (3 dimension) with 13 coordinates.

Returns:

The new neighborhood.

Definition at line 212 of file neighborhood3d.hh.

References coord.

```

213 {
214     static const coord crd[] = { 0, 0, 1,
215                                 0, 1, -1,
216                                 0, 1, 0,
217                                 0, 1, 1,
218                                 1, -1, -1,
219                                 1, -1, 0,
220                                 1, -1, 1,
221                                 1, 0, -1,
222                                 1, 0, 0,
223                                 1, 0, 1,
224                                 1, 1, -1,
225                                 1, 1, 0,
226                                 1, 1, 1 };
227     static neighborhood3d neighb(13, crd);
228     return neighb;
229 }
```

6.3.2.35 `const neighborhood2d& neighb_c4 () [inline]`

Create a neighborhood (2 dimension) with 0,1, 1,0.

Returns:

The new neighborhood.

Definition at line 184 of file neighborhood2d.hh.

References coord.

```

185 {
186     static const coord crd[] = { 0,1, 1,0 };
187     static const neighborhood2d neighb(2, crd);
188     return neighb;
189 }
```

6.3.2.36 `const neighborhood3d& neighb_c6 () [inline]`

Create a neighborhood (3 dimension) with 3 coordinates.

Returns:

The new neighborhood.

Definition at line 178 of file neighborhood3d.hh.

References coord.


```

179  {
180      static const coord crd[] = { 0, 0, 1,
181                                  0, 1, 0,
182                                  1, 0, 0};
183      static const neighborhood3d neighb(3, crd);
184      return neighb;
185  }

```

6.3.2.37 const neighborhood2d& neighb_c8 () [inline]

Create a neighborhood (2 dimension) with 4 coordinates: 0,1, 1,1, 1,0, 1,-1.

Returns:

The new neighborhood.

Definition at line 198 of file neighborhood2d.hh.

References coord.

```

199  {
200      static const coord crd[] = { 0,1, 1,1, 1,0, 1,-1 };
201      static const neighborhood2d neighb(4, crd);
202      return neighb;
203  }

```

6.3.2.38 template<template< class, class > class F, class I1, class I2> const F<typename mlc::exact< I1 >::ret::value_type, typename mlc::exact< I2 >::ret::value_type> traverse2 (const abstract::image< I1 > & input1, const abstract::image< I2 > & input2) [inline]

Create a functor f whose type is $F<oln_value_type(I1), oln_value_type(I2)>$, then call it on each point of the two input images.

Precondition:

$input1.size() == input2.size()$

Definition at line 145 of file traverse.hh.

References traverse2().

```

147  {
148      F<oln_value_type(I1), oln_value_type(I2)> f;
149      return traverse2(f, input1, input2);
150  }

```

6.3.2.39 template<template< class > class F, class I> const F<typename mlc::exact< I >::ret::value_type> traverse2 (const abstract::image< I > & input1, const abstract::image< I > & input2) [inline]

Create a functor f whose type is $F<oln_value_type(I)>$, then call it on each point of the two input images.

Precondition:

$input1.size() == input2.size()$

Definition at line 130 of file `traverse.hh`.

References `traverse2()`.

```

131  {
132      F<oln_value_type(I)> f;
133      return traverse2(f, input1, input2);
134  }
```

6.3.2.40 `template<class I1, class I2, class F> const F& traverse2 (F &f, const abstract::image< I1 > &input1, const abstract::image< I2 > &input2)`

Call functor *f* whose type is F on each point of the two input images.

Precondition:

`input1.size() == input2.size()`

Definition at line 112 of file `traverse.hh`.

References `oln::abstract::image< Exact >::size()`.

Referenced by `traverse2()`.

```

115  {
116      precondition(input1.size() == input2.size());
117      oln_iter_type(I1) p(input1);
118      for_all(p) f(input1[p], input2[p]);
119      return f;
120  }
```

6.3.2.41 `const window3d& win_c18_only () [inline]`

Create a window (3 dimensions) of 18 elements.

Returns:

The new window.

Definition at line 214 of file `window3d.hh`.

References `coord`.

```

215  {
216      static const coord crd[] = { -1, -1,  0,
217                                   -1,  0, -1,
218                                   -1,  0,  0,
219                                   -1,  0,  1,
220                                   -1,  1,  0,
221                                   0, -1, -1,
222                                   0, -1,  0,
223                                   0, -1,  1,
224                                   0,  0, -1,
225                                   0,  0,  1,
226                                   0,  1, -1,
227                                   0,  1,  0,
228                                   0,  1,  1,
229                                   1, -1,  0,
230                                   1,  0, -1,
```

```

231             1,  0,  0,
232             1,  0,  1,
233             1,  1,  0 };
234     static window3d win(18, crd);
235     return win;
236 }

```

6.3.2.42 const [window3d](#)& win_c18p () [inline]

Create a window (3 dimensions) of 19 elements.

Returns:

The new window.

It's the same than [win_c18_only\(\)](#) plus the 0,0,0 point.

Definition at line 245 of file window3d.hh.

References [coord](#).

```

246 {
247     static const coord crd[] = { -1, -1,  0,
248                                   -1,  0, -1,
249                                   -1,  0,  0,
250                                   -1,  0,  1,
251                                   -1,  1,  0,
252                                   0, -1, -1,
253                                   0, -1,  0,
254                                   0, -1,  1,
255                                   0,  0, -1,
256                                   0,  0,  0,
257                                   0,  0,  1,
258                                   0,  1, -1,
259                                   0,  1,  0,
260                                   0,  1,  1,
261                                   1, -1,  0,
262                                   1,  0, -1,
263                                   1,  0,  0,
264                                   1,  0,  1,
265                                   1,  1,  0 };
266     static window3d win(19, crd);
267     return win;
268 }

```

6.3.2.43 const [window3d](#)& win_c26_only () [inline]

Create a window (3 dimensions) of 26 elements.

Returns:

The new window.

Definition at line 275 of file window3d.hh.

References [coord](#).

```

276 {
277     static const coord crd[] = { -1, -1, -1,
278                                   -1, -1,  0,

```

```

279         -1, -1, 1,
280         -1, 0, -1,
281         -1, 0, 0,
282         -1, 0, 1,
283         -1, 1, -1,
284         -1, 1, 0,
285         -1, 1, 1,
286         0, -1, -1,
287         0, -1, 0,
288         0, -1, 1,
289         0, 0, -1,
290         0, 0, 1,
291         0, 1, -1,
292         0, 1, 0,
293         0, 1, 1,
294         1, -1, -1,
295         1, -1, 0,
296         1, -1, 1,
297         1, 0, -1,
298         1, 0, 0,
299         1, 0, 1,
300         1, 1, -1,
301         1, 1, 0,
302         1, 1, 1 };
303     static window3d win(26, crd);
304     return win;
305 }

```

6.3.2.44 const [window3d](#)& win_c26p () [inline]

Create a window (3 dimensions) of 27 elements.

Returns:

The new window.

It's the same than [win_c26_only\(\)](#) plus the 0,0,0 point.

Definition at line 314 of file window3d.hh.

References [coord](#).

```

315     {
316         static const coord crd[] = { -1, -1, -1,
317                                     -1, -1, 0,
318                                     -1, -1, 1,
319                                     -1, 0, -1,
320                                     -1, 0, 0,
321                                     -1, 0, 1,
322                                     -1, 1, -1,
323                                     -1, 1, 0,
324                                     -1, 1, 1,
325                                     0, -1, -1,
326                                     0, -1, 0,
327                                     0, -1, 1,
328                                     0, 0, -1,
329                                     0, 0, 0,
330                                     0, 0, 1,
331                                     0, 1, -1,
332                                     0, 1, 0,
333                                     0, 1, 1,
334                                     1, -1, -1,
335                                     1, -1, 0,
336                                     1, -1, 1,

```

```

337             1,  0, -1,
338             1,  0,  0,
339             1,  0,  1,
340             1,  1, -1,
341             1,  1,  0,
342             1,  1,  1 };
343     static window3d win(27, crd);
344     return win;
345 }
```

6.3.2.45 const [window1d](#)& win_c2_only () [inline]

Create a window (1 dimension) of 2 elements (-1, 1).

Returns:

The new neighborhood.

Definition at line 175 of file window1d.hh.

References [coord](#).

```

176 {
177     static const coord crd[] = { -1, 1 };
178     static const window1d win(2, crd);
179     return win;
180 }
```

6.3.2.46 const [window1d](#)& win_c2p () [inline]

Create a window (1 dimension) of 3 elements (-1, 0, 1).

Returns:

The new neighborhood.

Definition at line 187 of file window1d.hh.

References [coord](#).

```

188 {
189     static const coord crd[] = { -1, 0, 1 };
190     static const window1d win(3, crd);
191     return win;
192 }
```

6.3.2.47 const [window2d](#)& win_c4_only () [inline]

Create a window (2 dimensions) of 4 elements.

Returns:

The new window.

Definition at line 186 of file window2d.hh.

References [coord](#).

```
187 {
188     static const coord crd[] = { -1,0,  0,-1, 0,1,  1,0 };
189     static const window2d win(4, crd);
190     return win;
191 }
```

6.3.2.48 const [window2d](#)& win_c4p () [inline]

Create a window (2 dimensions) of 5 elements.

Returns:

The new window.

It's the same than [win_c4_only\(\)](#) plus the 0,0 point.

Definition at line 200 of file window2d.hh.

References [coord](#).

```
201 {
202     static const coord crd[] = { -1,0,  0,-1, 0,0, 0,1,  1,0 };
203     static const window2d win(5, crd);
204     return win;
205 }
```

6.3.2.49 const [window3d](#)& win_c6_only () [inline]

Create a window (3 dimensions) of 6 elements.

Returns:

The new window.

Definition at line 177 of file window3d.hh.

References [coord](#).

```
178 {
179     static const coord crd[] = { -1,  0,  0,
180                                0, -1,  0,
181                                0,  0, -1,
182                                0,  0,  1,
183                                0,  1,  0,
184                                1,  0,  0};
185     static const window3d win(6, crd);
186     return win;
187 }
```

6.3.2.50 const [window3d](#)& win_c6p () [inline]

Create a window (3 dimensions) of 7 elements.

Returns:

The new window.

It's the same than [win_c6_only\(\)](#) plus the 0,0,0 point.

Definition at line 196 of file window3d.hh.

References [coord](#).

```

197  {
198      static const coord crd[] = { -1,  0,  0,
199                                  0, -1,  0,
200                                  0,  0, -1,
201                                  0,  0,  0,
202                                  0,  0,  1,
203                                  0,  1,  0,
204                                  1,  0,  0};
205      static const window3d win(7, crd);
206      return win;
207  }
```

6.3.2.51 const [window2d](#)& win_c8_only () [inline]

Create a window (2 dimensions) of 8 elements.

Returns:

The new window.

Definition at line 212 of file window2d.hh.

References [coord](#).

```

213  {
214      static const coord crd[] = { -1,-1, -1,0, -1,1,  0,-1, 0,1,  1,-1,  1,0,  1,1 };
215      static const window2d win(8, crd);
216      return win;
217  }
```

6.3.2.52 const [window2d](#)& win_c8p () [inline]

Create a window (2 dimensions) of 9 elements.

Returns:

The new window.

It's the same than [win_c8_only](#) more the 0,0 point.

Definition at line 226 of file window2d.hh.

References [coord](#).

```

227  {
228      static const coord crd[] = { -1,-1, -1,0, -1,1,  0,-1, 0,0, 0,1,  1,-1,  1,0,  1,1 };
229      static const window2d win(9, crd);
230      return win;
231  }
```

6.4 oln::abstract Namespace Reference

oln::abstract namespace.

Classes

- class [behavior](#)
- struct [dpoint](#)
- class [image](#)
- struct [image_size](#)
- class [image_with_dim< 1, Exact >](#)
The specialized version for [image1d](#).
- class [image_with_dim< 2, Exact >](#)
The specialized version for [image2d](#).
- class [image_with_dim< 3, Exact >](#)
The specialized version for [image3d](#).
- class [image_with_impl](#)
- class [image_with_type](#)
- struct [data_type_image](#)
- struct [data_type_image_with_dim](#)
- class [vectorial_image](#)
- class [vectorial_image_with_dim](#)
- class [non_vectorial_image](#)
- class [non_vectorial_image_with_dim](#)
- class [binary_image](#)
- class [binary_image_with_dim](#)
- class [integer_image](#)
- class [integer_image_with_dim](#)
- class [decimal_image](#)
- class [decimal_image_with_dim](#)
- struct [image_with_type_with_dim_switch](#)
- struct [iter](#)
Iterator.
- class [iter1d](#)
Iterator on image of 1 dimension.
- class [iter2d](#)
Iterator on image of 2 dimensions.
- class [iter3d](#)
- struct [neighborhood](#)
Neighborhood.
- struct [neighborhoodnd](#)
Neighborhood N dimensions.

- struct [point](#)
- struct [struct_elt](#)
- struct [w_window](#)
Weight Window.
- struct [w_windownd](#)
Weight Window N dimensions.
- struct [window](#)
Window.
- struct [window_base](#)
Window Base.
- struct [windownd](#)
Window N dimensions.

6.4.1 Detailed Description

oln::abstract namespace.

6.5 oln::arith Namespace Reference

Arithmetic implementation.

Classes

- struct [f_logic_and](#)
Functor AND operators.
- struct **f_logic_and_cst**
- struct [f_logic_or](#)
Functor OR operators.
- struct **f_logic_or_cst**
- struct [f_logic_and_not](#)
Functor AND NOT operators.
- struct **f_logic_and_not_cst**
- struct [f_logic_not](#)
Functor NOT operator.
- struct **f_plus**
- struct **default_f_plus**
- struct **f_plus_cst**
- struct **f_minus**
- struct **default_f_minus**
- struct **f_minus_cst**
- struct **f_times**
- struct **default_f_times**
- struct **f_times_cst**
- struct **f_div**
- struct **default_f_div**
- struct **f_div_cst**
- struct **f_min**
- struct **default_f_min**
- struct **f_min_cst**
- struct **f_max**
- struct **default_f_max**
- struct **f_max_cst**
- struct **arith_return_type_proxy_plus_**
- struct **arith_return_type_proxy_cst_plus_**
- struct **arith_return_type_proxy_minus_**
- struct **arith_return_type_proxy_cst_minus_**
- struct **arith_return_type_proxy_times_**
- struct **arith_return_type_proxy_cst_times_**
- struct **arith_return_type_proxy_div_**
- struct **arith_return_type_proxy_cst_div_**
- struct **arith_return_type_proxy_min_**
- struct **arith_return_type_proxy_cst_min_**
- struct **arith_return_type_proxy_max_**
- struct **arith_return_type_proxy_cst_max_**

Functions

- `template<class I1, class I2> mute< I1, typename f_logic_and::result_type >::ret logic_and (const abstract::image< I1 > &input1, const abstract::image< I2 > &input2)`

AND NOT operators.

- `template<class C, class B, class I1, class I2> mute< I1, typename convoutput< C, B, typename f_logic_and::result_type >::ret >::ret logic_and (const convert::abstract::conversion< C, B > &conv, const abstract::image< I1 > &input1, const abstract::image< I2 > &input2)`
- `template<class I, class T> mute< I, typename f_logic_and_cst::result_type >::ret logic_and_cst (const abstract::image< I > &input, T val)`
- `template<class C, class B, class I, class T> mute< I, typename convoutput< C, B, typename f_logic_and_cst::result_type >::ret >::ret logic_and_cst (const convert::abstract::conversion< C, B > &conv, const abstract::image< I > &input, T val)`
- `template<class I1, class I2> mute< I1, typename f_logic_or::result_type >::ret logic_or (const abstract::image< I1 > &input1, const abstract::image< I2 > &input2)`

OR operators.

- `template<class C, class B, class I1, class I2> mute< I1, typename convoutput< C, B, typename f_logic_or::result_type >::ret >::ret logic_or (const convert::abstract::conversion< C, B > &conv, const abstract::image< I1 > &input1, const abstract::image< I2 > &input2)`
- `template<class I, class T> mute< I, typename f_logic_or_cst::result_type >::ret logic_or_cst (const abstract::image< I > &input, T val)`
- `template<class C, class B, class I, class T> mute< I, typename convoutput< C, B, typename f_logic_or_cst::result_type >::ret >::ret logic_or_cst (const convert::abstract::conversion< C, B > &conv, const abstract::image< I > &input, T val)`
- `template<class I1, class I2> mute< I1, typename f_logic_and_not::result_type >::ret logic_and_not (const abstract::image< I1 > &input1, const abstract::image< I2 > &input2)`

AND NOT operators.

- `template<class C, class B, class I1, class I2> mute< I1, typename convoutput< C, B, typename f_logic_and_not::result_type >::ret >::ret logic_and_not (const convert::abstract::conversion< C, B > &conv, const abstract::image< I1 > &input1, const abstract::image< I2 > &input2)`
- `template<class I, class T> mute< I, typename f_logic_and_not_cst::result_type >::ret logic_and_not_cst (const abstract::image< I > &input, T val)`
- `template<class C, class B, class I, class T> mute< I, typename convoutput< C, B, typename f_logic_and_not_cst::result_type >::ret >::ret logic_and_not_cst (const convert::abstract::conversion< C, B > &conv, const abstract::image< I > &input, T val)`
- `template<class I> mute< I, typename f_logic_not::result_type >::ret logic_not (const abstract::image< I > &input1)`

NOT operator.

- `template<class C, class B, class I> mute< I, typename convoutput< C, B, typename f_logic_not::result_type >::ret >::ret logic_not (const convert::abstract::conversion< C, B > &conv, const abstract::image< I > &input1)`
- `template<class I1, class I2> arith_return_type_proxy_plus< I1, I2 >::ret plus (const abstract::image< I1 > &input1, const abstract::image< I2 > &input2)`
- `template<class C, class B, class I1, class I2> mute< I1, typename convoutput< C, B, typename f_plus< typename mlc::exact< I1 >::ret::value_type, typename mlc::exact< I2 >::ret::value_type, typename ntg::internal::deduce_from_traits< ntg::internal::operator_plus, typename mlc::exact< I1 >::ret::value_type, typename mlc::exact< I2 >::ret::value_type >::ret >::ret plus (const convert::abstract::conversion< C, B > &conv, const abstract::image< I1 > &input1, const abstract::image< I2 > &input2)`

- `template<class I, class T> arith_return_type_proxy_cst_plus_< I, T >::ret plus_cst` (const `abstract::image< I >` &input, T val)
- `template<class C, class B, class I, class T> mute< I, typename convoutput< C, B, typename f_plus_cst< typename mlc::exact< I >::ret::value_type, T, typename ntg::internal::deduce_from_traits< ntg::internal::operator_plus, typename mlc::exact< I >::ret::value_type, T >::ret >::result_type >::ret >::ret plus_cst` (const `convert::abstract::conversion< C, B >` &conv, const `abstract::image< I >` &input, T val)
- `template<class I1, class I2> arith_return_type_proxy_minus_< I1, I2 >::ret minus` (const `abstract::image< I1 >` &input1, const `abstract::image< I2 >` &input2)
- `template<class C, class B, class I1, class I2> mute< I1, typename convoutput< C, B, typename f_minus< typename mlc::exact< I1 >::ret::value_type, typename mlc::exact< I2 >::ret::value_type, typename ntg::internal::deduce_from_traits< ntg::internal::operator_minus, typename mlc::exact< I1 >::ret::value_type, typename mlc::exact< I2 >::ret::value_type >::ret >::result_type >::ret >::ret minus` (const `convert::abstract::conversion< C, B >` &conv, const `abstract::image< I1 >` &input1, const `abstract::image< I2 >` &input2)
- `template<class I, class T> arith_return_type_proxy_cst_minus_< I, T >::ret minus_cst` (const `abstract::image< I >` &input, T val)
- `template<class C, class B, class I, class T> mute< I, typename convoutput< C, B, typename f_minus_cst< typename mlc::exact< I >::ret::value_type, T, typename ntg::internal::deduce_from_traits< ntg::internal::operator_minus, typename mlc::exact< I >::ret::value_type, T >::ret >::result_type >::ret >::ret minus_cst` (const `convert::abstract::conversion< C, B >` &conv, const `abstract::image< I >` &input, T val)
- `template<class I1, class I2> arith_return_type_proxy_times_< I1, I2 >::ret times` (const `abstract::image< I1 >` &input1, const `abstract::image< I2 >` &input2)
- `template<class C, class B, class I1, class I2> mute< I1, typename convoutput< C, B, typename f_times< typename mlc::exact< I1 >::ret::value_type, typename mlc::exact< I2 >::ret::value_type, typename ntg::internal::deduce_from_traits< ntg::internal::operator_times, typename mlc::exact< I1 >::ret::value_type, typename mlc::exact< I2 >::ret::value_type >::ret >::result_type >::ret >::ret times` (const `convert::abstract::conversion< C, B >` &conv, const `abstract::image< I1 >` &input1, const `abstract::image< I2 >` &input2)
- `template<class I, class T> arith_return_type_proxy_cst_times_< I, T >::ret times_cst` (const `abstract::image< I >` &input, T val)
- `template<class C, class B, class I, class T> mute< I, typename convoutput< C, B, typename f_times_cst< typename mlc::exact< I >::ret::value_type, T, typename ntg::internal::deduce_from_traits< ntg::internal::operator_times, typename mlc::exact< I >::ret::value_type, T >::ret >::result_type >::ret >::ret times_cst` (const `convert::abstract::conversion< C, B >` &conv, const `abstract::image< I >` &input, T val)
- `template<class I1, class I2> arith_return_type_proxy_div_< I1, I2 >::ret div` (const `abstract::image< I1 >` &input1, const `abstract::image< I2 >` &input2)
- `template<class C, class B, class I1, class I2> mute< I1, typename convoutput< C, B, typename f_div< typename mlc::exact< I1 >::ret::value_type, typename mlc::exact< I2 >::ret::value_type, typename ntg::internal::deduce_from_traits< ntg::internal::operator_div, typename mlc::exact< I1 >::ret::value_type, typename mlc::exact< I2 >::ret::value_type >::ret >::result_type >::ret >::ret div` (const `convert::abstract::conversion< C, B >` &conv, const `abstract::image< I1 >` &input1, const `abstract::image< I2 >` &input2)
- `template<class I, class T> arith_return_type_proxy_cst_div_< I, T >::ret div_cst` (const `abstract::image< I >` &input, T val)
- `template<class C, class B, class I, class T> mute< I, typename convoutput< C, B, typename f_div_cst< typename mlc::exact< I >::ret::value_type, T, typename ntg::internal::deduce_from_traits< ntg::internal::operator_div, typename mlc::exact< I >::ret::value_type, T >::ret >::result_type >::ret >::ret div_cst` (const `convert::abstract::conversion< C, B >` &conv, const `abstract::image< I >` &input, T val)

- `template<class I1, class I2> arith_return_type_proxy_min_< I1, I2 >::ret min` (const `abstract::image< I1 > &input1`, const `abstract::image< I2 > &input2`)
- `template<class C, class B, class I1, class I2> mute< I1, typename convoutput< C, B, typename f_min< typename mlc::exact< I1 >::ret::value_type, typename mlc::exact< I2 >::ret::value_type, typename ntg::internal::deduce_from_traits< ntg::internal::operator_min, typename mlc::exact< I1 >::ret::value_type, typename mlc::exact< I2 >::ret::value_type >::ret >::ret >::ret min` (const `convert::abstract::conversion< C, B > &conv`, const `abstract::image< I1 > &input1`, const `abstract::image< I2 > &input2`)
- `template<class I, class T> arith_return_type_proxy_cst_min_< I, T >::ret min_cst` (const `abstract::image< I > &input`, T val)
- `template<class C, class B, class I, class T> mute< I, typename convoutput< C, B, typename f_min_cst< typename mlc::exact< I >::ret::value_type, T, typename ntg::internal::deduce_from_traits< ntg::internal::operator_min, typename mlc::exact< I >::ret::value_type, T >::ret >::result_type >::ret >::ret min_cst` (const `convert::abstract::conversion< C, B > &conv`, const `abstract::image< I > &input`, T val)
- `template<class I1, class I2> arith_return_type_proxy_max_< I1, I2 >::ret max` (const `abstract::image< I1 > &input1`, const `abstract::image< I2 > &input2`)
- `template<class C, class B, class I1, class I2> mute< I1, typename convoutput< C, B, typename f_max< typename mlc::exact< I1 >::ret::value_type, typename mlc::exact< I2 >::ret::value_type, typename ntg::internal::deduce_from_traits< ntg::internal::operator_max, typename mlc::exact< I1 >::ret::value_type, typename mlc::exact< I2 >::ret::value_type >::ret >::result_type >::ret >::ret max` (const `convert::abstract::conversion< C, B > &conv`, const `abstract::image< I1 > &input1`, const `abstract::image< I2 > &input2`)
- `template<class I, class T> arith_return_type_proxy_cst_max_< I, T >::ret max_cst` (const `abstract::image< I > &input`, T val)
- `template<class C, class B, class I, class T> mute< I, typename convoutput< C, B, typename f_max_cst< typename mlc::exact< I >::ret::value_type, T, typename ntg::internal::deduce_from_traits< ntg::internal::operator_max, typename mlc::exact< I >::ret::value_type, T >::ret >::result_type >::ret >::ret max_cst` (const `convert::abstract::conversion< C, B > &conv`, const `abstract::image< I > &input`, T val)

6.5.1 Detailed Description

Arithmetic implementation.

6.6 oln::convert Namespace Reference

Conversion implementation (for example cast, color, or neighborhood to window).

Classes

- struct [bound](#)
- struct [cast](#)
- struct [convoutput](#)
- struct [force](#)
- struct [f_nrgb_to_xyz](#)
- struct [f_xyz_to_nrgb](#)
- struct [f_rgb_to_hsi](#)
- struct [f_hsi_to_rgb](#)
- struct [f_rgb_to_hsl](#)
- struct [f_hsl_to_rgb](#)
- struct [f_rgb_to_hsv](#)
- struct [f_hsv_to_rgb](#)
- struct [f_rgb_to_nrgb](#)
- struct [f_nrgb_to_rgb](#)
- struct [f_rgb_to_xyz](#)
- struct [f_xyz_to_rgb](#)
- struct [f_rgb_to_yiq](#)
- struct [f_yiq_to_rgb](#)
- struct [f_rgb_to_yuv](#)
- struct [f_yuv_to_rgb](#)
- struct [stretch](#)
- struct [value_to_point](#)
- struct [value_to_point::doit_binary](#)
Convert a binary to a point.
- struct [value_to_point::doit_not_binary](#)
Convert a non vectorial to a point.
- struct [value_to_point< ntg::color< 3, Qbits, S >, Exact >](#)

Functions

- template<class C, class B, class UF> [internal::compconv1_< C, UF >](#) [compconv1](#) (const [abstract::conversion](#)< C, B > &conv, const UF &func)
- template<class C, class B, class BF> [internal::compconv2_< C, BF >](#) [compconv2](#) (const [abstract::conversion](#)< C, B > &conv, const BF &func)
- template<class C, class B, class I> [mute< I, typename convoutput< C, B, typename mlc::exact< I >::ret::value_type >::ret >::ret apply](#) (const [abstract::conversion](#)< C, B > &conv, const [oln::abstract::image](#)< I > &input)
- template<class N> [oln::abstract::neighborhood< N >::win_type](#) [ng_to_se](#) (const [oln::abstract::neighborhood](#)< N > &Ng)
- template<class N> [oln::abstract::neighborhood< N >::win_type](#) [ng_to_cse](#) (const [oln::abstract::neighborhood](#)< N > &Ng)

- `template<unsigned inbits, unsigned outbits> color< 3, outbits, xyz_traits > nrgb_to_xyz (const color< 3, inbits, nrgb_traits > &v)`
- `template<unsigned inbits, unsigned outbits> color< 3, outbits, nrgb_traits > xyz_to_nrgb (const color< 3, inbits, xyz_traits > &v)`
- `template<unsigned inbits, unsigned outbits> color< 3, outbits, hsi_traits > rgb_to_hsi (const color< 3, inbits, rgb_traits > &v)`
- `template<unsigned inbits, unsigned outbits> color< 3, outbits, rgb_traits > hsi_to_rgb (const color< 3, inbits, hsi_traits > &v)`
- `template<unsigned inbits, unsigned outbits> color< 3, inbits, hsl_traits > rgb_to_hsl (const color< 3, outbits, rgb_traits > &v)`
- `template<unsigned inbits, unsigned outbits> color< 3, outbits, rgb_traits > hsl_to_rgb (const color< 3, inbits, hsl_traits > &v)`
- `template<unsigned inbits, unsigned outbits> color< 3, outbits, hsv_traits > rgb_to_hsv (const color< 3, inbits, rgb_traits > &v)`
- `template<unsigned inbits, unsigned outbits> color< 3, outbits, rgb_traits > hsv_to_rgb (const color< 3, inbits, hsv_traits > &v)`
- `template<unsigned inbits, unsigned outbits> color< 3, outbits, nrgb_traits > rgb_to_nrgb (const color< 3, inbits, rgb_traits > &v)`
- `template<unsigned inbits, unsigned outbits> color< 3, outbits, rgb_traits > nrgb_to_rgb (const color< 3, inbits, nrgb_traits > &v)`
- `template<unsigned inbits, unsigned outbits> color< 3, outbits, xyz_traits > rgb_to_xyz (const color< 3, inbits, rgb_traits > &v)`
- `template<unsigned inbits, unsigned outbits> color< 3, outbits, rgb_traits > xyz_to_rgb (const color< 3, outbits, xyz_traits > &v)`
- `template<unsigned inbits, unsigned outbits> color< 3, outbits, yiq_traits > rgb_to_yiq (const color< 3, inbits, rgb_traits > &v)`
- `template<unsigned inbits, unsigned outbits> color< 3, outbits, rgb_traits > yiq_to_rgb (const color< 3, inbits, yiq_traits > &v)`
- `template<unsigned inbits, unsigned outbits> color< 3, outbits, yuv_traits > rgb_to_yuv (const color< 3, inbits, rgb_traits > &v)`
- `template<unsigned inbits, unsigned outbits> color< 3, outbits, rgb_traits > yuv_to_rgb (const color< 3, inbits, yuv_traits > &v)`
- `template<class DestValue, class I> mute< I, DestValue >::ret stretch_balance (const oln::abstract::non_vectorial_image< I > &in, const typename mlc::exact< I >::ret::value_type &min_in=ntg::type_traits< typename mlc::exact< I >::ret::value_type >::min(), const typename mlc::exact< I >::ret::value_type &max_in=ntg::type_traits< typename mlc::exact< I >::ret::value_type >::max(), const DestValue &min_out=ntg::type_traits< DestValue >::min(), const DestValue &max_out=ntg::type_traits< DestValue >::max())`

Variables

- `const float sqrt3_3 = sqrt(3) / 3`
- `const float inv_sqrt6 = 1 / sqrt(6)`
- `const float inv_sqrt2 = 1 / sqrt(2)`

6.6.1 Detailed Description

Conversion implementation (for example cast, color, or neighborhood to window).

6.6.2 Function Documentation

6.6.2.1 `template<class C, class B, class I> mute<I, typename convoutput<C, B, typename mlc::exact< I >::ret::value_type>::ret>::ret apply (const abstract::conversion< C, B > & conv, const oln::abstract::image< I > & input) [inline]`

Apply function that retrieve the result type within the conversion class.

The core [oln::apply](#) function, cannot apply all conversion function, because they do not all define result_type. So we define another apply function here, to apply conversions.

Definition at line 141 of file conversion.hh.

References [compconv1\(\)](#).

```
142     {
143         /* CONV can now be wrapped as an Adaptable Unary Function
144            because we know the input type. Composing CONV with the
145            identity for the input type will cause such wrapping to
146            happen. */
147         return apply(compconv1(conv, f_identity<oln_value_type(I)>()), input);
148     }
```

6.6.2.2 `template<class C, class B, class UF> internal::compconv1_<C, UF> compconv1 (const abstract::conversion< C, B > & conv, const UF & func)`

Friendly procedure that build an [internal::compconv1_](#) with type deduction.

Definition at line 119 of file conversion.hh.

Referenced by [apply\(\)](#).

```
120     {
121         return internal::compconv1_<C, UF>(conv.exact(), func);
122     }
```

6.6.2.3 `template<class C, class B, class BF> internal::compconv2_<C, BF> compconv2 (const abstract::conversion< C, B > & conv, const BF & func)`

Likewise for [compconv2_](#).

Definition at line 127 of file conversion.hh.

```
128     {
129         return internal::compconv2_<C, BF>(conv.exact(), func);
130     }
```

6.6.2.4 `template<unsigned inbits, unsigned outbits> color<3, outbits, rgb_traits> hsi_to_rgb (const color< 3, inbits, hsi_traits > & v)`

Conversion from HSI to RGB color space.

See also:

[f_rgb_to_hsl](#)

Definition at line 143 of file rgbhsi.hh.

```

144     {
145         f_hsi_to_rgb<inbits, outbits> f;
146
147         return f(v);
148     }

```

6.6.2.5 template<unsigned inbits, unsigned outbits> [color](#)<3, outbits, rgb_traits> hsl_to_rgb (const [color](#)< 3, inbits, hsl_traits > & v)

Conversion from HSL to RGB color space.

See also:

[f_rgb_to_hsl](#)

Definition at line 217 of file rgbhsl.hh.

```

218     {
219         f_hsl_to_rgb<inbits, outbits> f;
220         return f(v);
221     }

```

6.6.2.6 template<unsigned inbits, unsigned outbits> [color](#)<3, outbits, rgb_traits> hsv_to_rgb (const [color](#)< 3, inbits, hsv_traits > & v)

Conversion from HSV to RGB.

See also:

[f_rgb_to_hsl](#)

Definition at line 195 of file rgbhsv.hh.

```

196     {
197         f_hsv_to_rgb<inbits, outbits> f;
198         return f(v);
199     }

```

6.6.2.7 template<class N> [oln::abstract::neighborhood](#)<N>::win_type ng_to_cse (const [oln::abstract::neighborhood](#)< N > & Ng)

Convert a neighborhood to a window and add the center.

See also:

[ng_to_cs](#)

Definition at line 64 of file conversion_ng_se.hh.

References [oln::abstract::neighborhood](#)< Exact >::add().

```

65     {
66         typename oln::abstract::neighborhood<N>::win_type output;
67         oln_iter_type(N) p(Ng);
68         for_all(p)
69             output.add(p);
70         oln_dpoint_type(N) zero;
71         for (unsigned size = 0; size < N::dim; ++size)
72             zero.nth(size) = 0;
73         output.add(zero);
74         return output;
75     }

```

6.6.2.8 `template<class N> oln::abstract::neighborhood<N>::win_type ng_to_se (const oln::abstract::neighborhood< N > & Ng)`

Convert a neighborhood to a window.

See also:

[ng_to_cse](#)

Definition at line 49 of file `conversion_ng_se.hh`.

References `oln::abstract::neighborhood< Exact >::add()`.

```

50     {
51         typename oln::abstract::neighborhood<N>::win_type output;
52         oln_iter_type(N) p(Ng);
53         for_all(p)
54             output.add(p);
55         return output;
56     }

```

6.6.2.9 `template<unsigned inbits, unsigned outbits> color<3, outbits, rgb_traits> nrgb_to_rgb (const color< 3, inbits, nrgb_traits > & v)`

Conversion from N-RGB to RGB.

See also:

[f_rgb_to_hsl](#)

Definition at line 135 of file `rgbnrgb.hh`.

```

136     {
137         f_nrgb_to_rgb<inbits, outbits> f;
138
139         return f(v);
140     }

```

6.6.2.10 `template<unsigned inbits, unsigned outbits> color<3, outbits, xyz_traits> nrgb_to_xyz (const color< 3, inbits, nrgb_traits > & v)`

Conversion from N-RGB to XYZ color space.

Deprecated

A composition should be performed with `nrgb->rgb` and `rgb->xyz`.

See also:

[f_nrgb_to_xyz](#) for more information.

Definition at line 121 of file `nrgbxyz.hh`.

```
122     {  
123         f_nrgb_to_xyz<inbits, outbits> f;  
124  
125         return f(v);  
126     }
```

6.6.2.11 `template<unsigned inbits, unsigned outbits> color<3, outbits, hsi_traits> rgb_to_hsi(const color< 3, inbits, rgb_traits > & v)`

Conversion from RGB to HSI color space.

See also:

[f_rgb_to_hsi](#)

Definition at line 98 of file `rgbhsi.hh`.

```
99     {  
100         f_rgb_to_hsi<inbits, outbits> f;  
101  
102         return f(v);  
103     }
```

6.6.2.12 `template<unsigned inbits, unsigned outbits> color<3, inbits, hsl_traits> rgb_to_hsl(const color< 3, outbits, rgb_traits > & v)`

Conversion from RGB to HSL color space.

See also:

[f_rgb_to_hsl](#)

Definition at line 142 of file `rgbhsl.hh`.

```
143     {  
144         f_rgb_to_hsl<inbits, outbits> f;  
145         return f(v);  
146     }
```

6.6.2.13 `template<unsigned inbits, unsigned outbits> color<3, outbits, hsv_traits> rgb_to_hsv(const color< 3, inbits, rgb_traits > & v)`

Conversion from RGB to HSV.

See also:

[f_rgb_to_hsl](#)

Definition at line 111 of file `rgbhsv.hh`.

```

112     {
113         f_rgb_to_hsv<inbits, outbits> f;
114         return f(v);
115     }

```

6.6.2.14 `template<unsigned inbits, unsigned outbits> color<3, outbits, nrgb_traits> rgb_to_nrgb(const color< 3, inbits, rgb_traits > & v)`

Conversion from RGB to N-RGB.

See also:

[f_rgb_to_hsl](#)

Definition at line 90 of file `rgbnrgb.hh`.

```

91     {
92         f_rgb_to_nrgb<inbits, outbits> f;
93
94         return f(v);
95     }

```

6.6.2.15 `template<class DestValue, class I> mute<I, DestValue>::ret stretch_balance(const oln::abstract::non_vectorial_image< I > & in, const typename mlc::exact< I >::ret::value_type & min_in = ntg::type_traits< typename mlc::exact< I >::ret::value_type >::min(), const typename mlc::exact< I >::ret::value_type & max_in = ntg::type_traits< typename mlc::exact< I >::ret::value_type >::max(), const DestValue & min_out = ntg::type_traits< DestValue >::min(), const DestValue & max_out = ntg::type_traits< DestValue >::max())`

Stretch the value of an image.

This function stretches values between *min_in* and *max_in* of an image *in*, to a range that goes from *min_out* to *max_out*.

- *in* Input image, must be have scalar values
- *min_in* Lower bound of the range in the input. All values smaller than *min_in* are converted to *min_out*.
- *max_in* Upper bound of the range in the input. All values greater than *max_in* are converted to *max_out*.
- *min_out* Low bound of the range in the output.
- *max_out* Upper bound of the range in the output.

Difference between "`apply(stretch<T>(), im)`" and "`stretch_balance<T>(im)`": the first one stretches all the range of `oln_value_type(T)`, the second stretches only values that appear in the image:

```

#include <oln/basics1d.hh>
#include <oln/convert/stretch.hh>
#include <ntg/all.hh>
#include <iostream>

int main()
{
    oln::image1d<ntg::int_u<17> > im(4);
    im(0) = 0; im(1) = 2; im(2) = 5000; im(3) = 4000;

    // print "0 170 511 340"
    //(values of the *image* are dispatched on the range [0..2^9-1])
    std::cout << oln::convert::stretch_balance<ntg::int_u<9> >(im) << std::endl;

    //print " 0 0 19 16"    (19 is the result of 5000 * 2^9 / 2^17)
    //(values of *int_u<17>* are dispatched on the range [0..2^9-1])
    std::cout << apply(oln::convert::stretch<ntg::int_u<9> >(), im) << std::endl;
}

```

This function is useful to stretch images of label:

```

#include <oln/basics2d.hh>
#include <oln/convert/stretch.hh>
#include <oln/level/cc.hh>
#include <ntg/all.hh>
#include <iostream>

int main()
{
    oln::image2d<ntg::bin> light = oln::load(IMG_IN "face_se.pbm");

    //Extraction of the connected components:
    unsigned card;
    oln::image2d<ntg::int_u8> cc
        = oln::level::frontp_connected_component<ntg::int_u8>(light,
                                                                oln::neighb_c8(),
                                                                card);

    oln::io::save(cc, IMG_OUT "oln_convert_stretch_dark.pgm");
    oln::io::save(oln::convert::stretch_balance<ntg::int_u8>(cc),
                  IMG_OUT "oln_convert_stretch_balance.pgm");
}

```



=> Without stretch_balance:



=> With stretch_-



balance:

Definition at line 166 of file stretch.hh.

References `oln::abstract::image< Exact >::size()`.

```

173     {
174         typedef typename
175             ntg_is_a(DestValue, ntg::non_vectorial)::ensure_type ensure_type;
176
177         typename mute<I, DestValue>::ret out(in.size());
178
179         //FIXME: I would like to remove the static_cast.
180         std::vector<ntg_cumul_type(DestValue)>
181             tab(static_cast<int>(max_in - min_in + 1));
182         typedef typename std::set<oln_value_type(I)> set_type;
183         set_type s;
184         oln_iter_type(I) it(in);
185
186         for_all(it)
187             if (in[it] <= max_in && in[it] >= min_in)
188                 s.insert(in[it]);
189         if (s.size() == 1)
190         {
191             for_all(it)
192                 out[it] = ntg_zero_val(DestValue);
193             return out;
194         }
195         {
196             unsigned cpt = 0;
197             for (typename set_type::const_iterator it(s.begin());
198                  it != s.end(); ++it, ++cpt)
199                 tab[*it - min_in] = cpt * (max_out - min_out) / (s.size() - 1);
200         }
201         for_all(it)
202             if (min_in <= in[it])
203             {
204                 if (in[it] <= max_in)
205                     out[it] = tab[in[it] - min_in] + min_out;
206                 else
207                     out[it] = max_out;
208             }
209         else
210             out[it] = min_out;
211         return out;
212     }

```

6.6.2.16 `template<unsigned inbits, unsigned outbits> color<3, outbits, nrgb_traits> xyz_to_nrgb
(const color< 3, inbits, xyz_traits > & v)`

Conversion from XYZ to N-RGB color space.

Deprecated

a composition should be performed with `xyz->rgb` and `rgb->nrgb`.

See also:

[f_nrgb_to_xyz](#) for more information.

Definition at line 171 of file `nrgbxyz.hh`.

```
172     {  
173         f_xyz_to_nrgb<inbits, outbits> f;  
174  
175         return f(v);  
176     }
```

6.7 oln::convert::abstract Namespace Reference

Base classes for conversion.

Classes

- struct [color_conversion](#)
- struct [conversion](#)
- struct **conversion::output**
- struct [conversion_to_type](#)

Base class for the conversion to a specific type.

- struct [conversion_from_type_to_type](#)

Base class if both input and output types of the conversion are fixed.

6.7.1 Detailed Description

Base classes for conversion.

6.8 oln::convert::abstract::internal Namespace Reference

Internal purpose only.

Classes

- struct `output`
Retrieve the result type of a conversion.
- struct `output< conversion_from_type_to_type< Argument_Type, Result_Type, Exact, Base >, Argument_Type >`
- struct `output< conversion_to_type< Result_Type, Exact, Base >, T >`

6.8.1 Detailed Description

Internal purpose only.

6.9 oln::convert::internal Namespace Reference

Internal purpose only.

Classes

- struct [compconv1_](#)
- struct [compconv2_](#)

Functions

- float **RGB** (float q1, float q2, float hue)

6.9.1 Detailed Description

Internal purpose only.

6.10 oln::convol Namespace Reference

Algorithms related to convolution.

Functions

- template<class I, unsigned Size, typename SE, typename Sum> [oln::mute< I >::ret nagao_generalized](#) (const [oln::abstract::image< I >](#) &in, const [internal::se_array< Size, SE >](#) &sa)
- template<class I> [oln::mute< I >::ret nagao](#) (const [oln::abstract::non_vectorial_image_with_dim< 2, I >](#) &in)
- template<class I> [oln::mute< I >::ret nagao](#) (const [oln::abstract::vectorial_image_with_dim< 2, I >](#) &in)

6.10.1 Detailed Description

Algorithms related to convolution.

6.10.2 Function Documentation

6.10.2.1 template<class I> [oln::mute< I >::ret oln::convol::nagao](#) (const [oln::abstract::vectorial_image_with_dim< 2, I >](#) &in) [inline]

Standard Nagao filter 5x5.

Apply the 5*5 nagao filter on a vectorial image.

```
#include <oln/convol/nagao.hh>
#include <oln/basics2d.hh>
#include <ntg/all.hh>
int main()
{
    oln::image2d<ntg::rgb_8> imc = oln::load(IMG_IN "lena.ppm");
    oln::save(oln::convol::nagao(imc), IMG_OUT "oln_convол_nagao.ppm");
}
```





=>

Definition at line 196 of file nagao.hxx.

```
197     {  
198         typedef ntg::rgb_8::float_vec_type float_vec_type;  
199         return nagao_generalized<I, 9, window2d, float_vec_type>  
200             (in, internal::mk_nagao_windows_5x5());  
201     }
```


6.10.2.2 `template<class I> oln::mute< I >::ret oln::convol::nagao (const oln::abstract::non_vectorial_image_with_dim< 2, I > &in) [inline]`

Standard Nagao filter 5x5.

Apply the 5*5 nagao filter on a non vectorial image.

```
#include <oln/convol/nagao.hh>
#include <oln/basics2d.hh>
#include <ntg/all.hh>
int main()
{
    oln::image2d<ntg::int_u8> imc = oln::load(IMG_IN "lena256.pgm");
    oln::save(oln::convol::nagao(imc),
              IMG_OUT "oln_convol_nagao_256.pgm");
}
```



=>



Definition at line 168 of file nagao.hxx.

```
169     {
170         return nagao_generalized<I, 9, window2d, ntg::float_s>
171             (in, internal::mk_nagao_windows_5x5());
172     }
```

6.10.2.3 `template<class I, unsigned Size, typename SE, typename Sum> oln::mute< I >::ret oln::convol::nagao_generalized (const oln::abstract::image< I > &in, const internal::se_array< Size, SE > &sa) [inline]`

A Nagao filter generalized.

Each point in the input corresponds to the mean of the window in which has the smallest variance.

Parameters:

in Input image.

sa array of structuring elements.

See also:

[oln::convol::nagao](#)

Definition at line 133 of file nagao.hxx.

References `oln::abstract::image< Exact >::border_adapt_width()`, `oln::convol::internal::se_array< Size, S >::delta()`, and `oln::abstract::image< Exact >::size()`.

```
135     {
136         ntg_compare_nb_comp(Sum, oln_value_type(I))::ensure();
137         in.border_adapt_width(sa.delta());
138         oln_concrete_type(I) out(in.size());
139         oln_iter_type(I) it(out);
140         for_all(it)
141             out[it] = internal::mean_of_smaller_variance<I, SE, Size, Sum>
142                 (in, it, sa);
143         return out;
144     }
```

6.11 oln::convol::fast Namespace Reference

Implementation of algorithms for large structuring elements.

Functions

- `template<class C, class B, class I, class BE> mute< I, typename convoutput< C, B, typename mlc::exact< I >::ret::value_type >::ret >::ret gaussian (const convert::abstract::conversion< C, B > &input_conv, const abstract::image< I > &in, ntg::float_s sigma, const abstract::behavior< BE > &behavior)`

Gaussian filter.

- `template<class C, class B, class I, class BE> mute< I, typename convoutput< C, B, typename mlc::exact< I >::ret::value_type >::ret >::ret gaussian_derivative (const convert::abstract::conversion< C, B > &input_conv, const abstract::image< I > &in, ntg::float_s sigma, const abstract::behavior< BE > &behavior)`

Derivative gaussian filter.

- `template<class C, class B, class I, class BE> mute< I, typename convoutput< C, B, typename mlc::exact< I >::ret::value_type >::ret >::ret gaussian_second_derivative (const convert::abstract::conversion< C, B > &input_conv, const abstract::image< I > &in, ntg::float_s sigma, const abstract::behavior< BE > &behavior)`

Second derivative gaussian filter.

- `template<class I, class BE> oln::mute< I >::ret gaussian (const abstract::image< I > &in, ntg::float_s sigma, const abstract::behavior< BE > &behavior)`

Gaussian filter with a default conversion.

- `template<class I, class BE> oln::mute< I >::ret gaussian_derivative (const abstract::image< I > &in, ntg::float_s sigma, const abstract::behavior< BE > &behavior)`

Derivative gaussian filter with a default conversion.

- `template<class I, class BE> oln::mute< I >::ret gaussian_second_derivative (const abstract::image< I > &in, ntg::float_s sigma, const abstract::behavior< BE > &behavior=mirror_behavior<>())`

Second derivative gaussian filter with a default conversion.

- `template<class I> oln::mute< I >::ret gaussian (const abstract::image< I > &in, ntg::float_s sigma)`

Gaussian filter with a default conversion and a default behavior.

- `template<class I> oln::mute< I >::ret gaussian_derivative (const abstract::image< I > &in, ntg::float_s sigma)`

Derivative gaussian filter with a default conversion and a default behavior.

- `template<class I> oln::mute< I >::ret gaussian_second_derivative (const abstract::image< I > &in, ntg::float_s sigma)`

Second derivative gaussian filter with a default conversion and a default behavior.

Variables

- `mute< I, typename convoutput< C, B, oln_value_type(I)>::ret >::re gaussian)(const convert::abstract::conversion< C, B > &c, const abstract::image< I > &in, ntg::float_s sigma, const abstract::behavior< BE > &behavior)`
- `mute< I, typename convoutput< C, B, oln_value_type(I)>::ret >::re gaussian_derivative)(const convert::abstract::conversion< C, B > &c, const abstract::image< I > &in, ntg::float_s sigma, const abstract::behavior< BE > &behavior)`
- `mute< I, typename convoutput< C, B, oln_value_type(I)>::ret >::re gaussian_second_derivative)(const convert::abstract::conversion< C, B > &c, const abstract::image< I > &in, ntg::float_s sigma, const abstract::behavior< BE > &behavior)`

6.11.1 Detailed Description

Implementation of algorithms for large structuring elements.

The algorithms you can find here are fast. This mean these ones are evolved versions for large structuring elements.

6.11.2 Function Documentation

6.11.2.1 `template<class I> oln::mute< I >::ret gaussian (const abstract::image< I > &in, ntg::float_s sigma) [inline]`

Gaussian filter with a default conversion and a default behavior.

Gaussian filter implementation from "Recursively implementing the gaussian and its derivatives" Deriche 93 INRIA REPORT (num RR-1893).

Warning:

The content of the border is the mirror of the image.

Parameters:

I Exact type of the image.

- *in* Input image.
- *sigma* Value of sigma when computing the gaussian.

```
#include <oln/basics2d.hh>
#include <oln/convol/fast_gaussian.hh>
#include <oln/core/behavior.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::convol::fast::gaussian(im1, 2),
              IMG_OUT "oln_convol_fast_gaussian_default.pbm");
    return 0;
}
```



=>

**Warning:**

If sigma is big enough, the result may differ a little bit with the -O3 flag of g++.

Definition at line 326 of file fast_gaussian.hh.

References gaussian(), and oln::mirror_bhv().

```
327         { return gaussian(convert::force<oln_value_type(I)>(), in, sigma,
328                           mirror_bhv()); }
```

6.11.2.2 `template<class I, class BE> oln::mute< I >::ret gaussian (const abstract::image< I > & in, ntg::float_s sigma, const abstract::behavior< BE > & behavior) [inline]`

Gaussian filter with a default conversion.

Gaussian filter implementation from "Recursively implementing the gaussian and its derivatives" Deriche 93 INRIA REPORT (num RR-1893).

Parameters:

I Exact type of the image.

BE Exact type of the behavior.

- in Input image.
- sigma Value of sigma when computing the gaussian.
- behavior Object to know how to work on borders.

```
#include <oln/basics2d.hh>
#include <oln/convol/fast_gaussian.hh>
```

```

#include <oln/core/behavior.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::convol::fast::gaussian(im1, 2,
                                           oln::value_behavior<oln_value_type_(im_type)>(0)),
              IMG_OUT "oln_convool_fast_gaussian.pbm");
    return 0;
}

```



=>

**Warning:**

If sigma is big enough, the result may differ a little bit with the -O3 flag of g++.

Definition at line 180 of file fast_gaussian.hh.

References gaussian().

```

182    { return gaussian(convert::force<oln_value_type(I)>(), in, sigma,
183                      behavior); }

```

6.11.2.3 `template<class C, class B, class I, class BE> mute<I, typename convoutput<C, B, typename mlc::exact< I >::ret::value_type>::ret>::ret gaussian (const convert::abstract::conversion< C, B > &input_conv, const abstract::image< I > &in, ntg::float_s sigma, const abstract::behavior< BE > &behavior)`

Gaussian filter.

Gaussian filter implementation from "Recursively implementing the gaussian and its derivatives" Deriche 93 INRIA REPORT (num RR-1893).

Parameters:*C* Exact type of the conversion object.*B* Base type of the conversion object.*I* Exact type of the image.*BE* Exact type of the behavior.

- input_conv Converter object.
- in Input image.
- sigma Value of sigma when computing the gaussian.
- behavior Object to know how to work on borders.

Warning:

If sigma is big enough, the result may differ a little bit with the -O3 flag of g++.

Referenced by gaussian().

6.11.2.4 `template<class I> oln::mute< I >::ret gaussian_derivative (const abstract::image< I > & in, ntg::float_s sigma) [inline]`

Derivative gaussian filter with a default conversion and a default behavior.

Gaussian filter implementation from "Recursively implementing the gaussian and its derivatives" Deriche 93 INRIA REPORT (num RR-1893).

Warning:

The content of the border is the mirror of the image.

Parameters:*I* Exact type of the image.*BE* Exact type of the behavior.

- in Input image.
- sigma Value of sigma when computing the gaussian.
- behavior Object to know how to work on borders.

```
#include <oln/basics2d.hh>
#include <oln/convol/fast_gaussian.hh>
#include <oln/core/behavior.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::convol::fast::gaussian_derivative(im1, 2),
              IMG_OUT "oln_convol_fast_gaussian_derivative_default.pbm");
    return 0;
}
```

**Warning:**

If sigma is big enough, the result may differ a little bit with the -O3 flag of g++.

Definition at line 376 of file fast_gaussian.hh.

References gaussian_derivative(), and oln::mirror_bhv().

```

377     { return gaussian_derivative(convert::force<oln_value_type(I)>(), in, sigma,
378                                mirror_bhv()); }

```

6.11.2.5 `template<class I, class BE> oln::mute< I >::ret gaussian_derivative (const abstract::image< I > & in, ntg::float_s sigma, const abstract::behavior< BE > & behavior) [inline]`

Derivative gaussian filter with a default conversion.

Gaussian filter implementation from "Recursively implementing the gaussian and its derivatives" Deriche 93 INRIA REPORT (num RR-1893).

Parameters:

I Exact type of the image.

BE Exact type of the behavior.

- in Input image.
- sigma Value of sigma when computing the gaussian.
- behavior Object to know how to work on borders.

```

#include <oln/basics2d.hh>
#include <oln/convol/fast_gaussian.hh>
#include <oln/core/behavior.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::convol::fast::gaussian_derivative(im1, 2,
                                                    oln::value_behavior<oln_value_type_(im_type)>(0)),
              IMG_OUT "oln_convol_fast_gaussian_derivative.pbm");
    return 0;
}

```



=>

**Warning:**

If sigma is big enough, the result may differ a little bit with the -O3 flag of g++.

Definition at line 228 of file fast_gaussian.hh.

References gaussian_derivative().

```

230     { return gaussian_derivative(convert::force<oln_value_type(I)>(), in, sigma,
231                               behavior); }

```

6.11.2.6 `template<class C, class B, class I, class BE> mute<I, typename convoutput<C, B, typename mlc::exact< I >::ret::value_type>::ret>::ret gaussian_derivative (const convert::abstract::conversion< C, B > &input_conv, const abstract::image< I > &in, ntg::float_s sigma, const abstract::behavior< BE > &behavior)`

Derivative gaussian filter.

Gaussian filter implementation from "Recursively implementing the gaussian and its derivatives" Deriche 93 INRIA REPORT (num RR-1893).

Parameters:

C Exact type of the conversion object.

B Base type of the conversion object.

I Exact type of the image.

BE Exact type of the behavior.

- input_conv Converter object.
- in Input image.
- sigma Value of sigma when computing the gaussian.
- behavior Object to know how to work on borders.

Warning:

If sigma is big enough, the result may differ a little bit with the -O3 flag of g++.

Referenced by gaussian_derivative().

6.11.2.7 `template<class I> oln::mute< I >::ret gaussian_second_derivative (const abstract::image< I > & in, ntg::float_s sigma) [inline]`

Second derivative gaussian filter with a default conversion and a default behavior.

Gaussian filter implementation from "Recursively implementing the gaussian and its derivatives" Deriche 93 INRIA REPORT (num RR-1893).

Warning:

The content of the border is the mirror of the image.

Parameters:

I Exact type of the image.

BE Exact type of the behavior.

- in Input image.
- sigma Value of sigma when computing the gaussian.
- behavior Object to know how to work on borders.

```
#include <oln/basics2d.hh>
#include <oln/convol/fast_gaussian.hh>
#include <oln/core/behavior.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::convol::fast::gaussian_second_derivative(im1, 2),
              IMG_OUT "oln_convol_fast_gaussian_second_derivative_default.pbm");
    return 0;
}
```




=>

**Warning:**

If sigma is big enough, the result may differ a little bit with the -O3 flag of g++.

Definition at line 426 of file fast_gaussian.hh.

References gaussian_second_derivative(), and oln::mirror_bhv().

```
427     { return gaussian_second_derivative(convert::force<oln_value_type(I)>(), in, sigma,
428                                       mirror_bhv()); }
```

6.11.2.8 `template<class I, class BE> oln::mute< I >::ret gaussian_second_derivative (const abstract::image< I > & in, ntg::float_s sigma, const abstract::behavior< BE > & behavior = mirror_behavior<>()) [inline]`

Second derivative gaussian filter with a default conversion.

Gaussian filter implementation from "Recursively implementing the gaussian and its derivatives" Deriche 93 INRIA REPORT (num RR-1893).

Parameters:

I Exact type of the image.

BE Exact type of the behavior.

- in Input image.
- sigma Value of sigma when computing the gaussian.
- behavior Object to know how to work on borders.

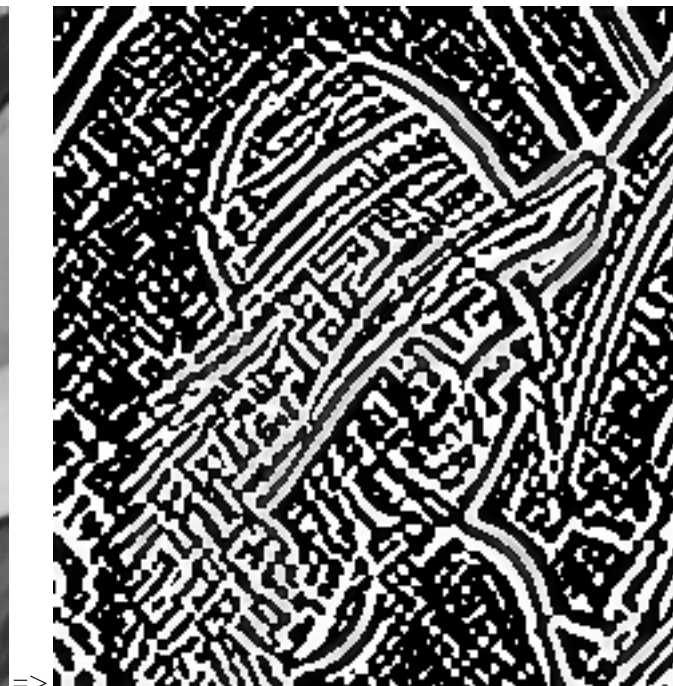

```

#include <oln/basics2d.hh>
#include <oln/convol/fast_gaussian.hh>
#include <oln/core/behavior.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::convol::fast::gaussian_second_derivative(im1, 2,
                                                            oln::value_behavior<oln_value_type_(im_type)>(0)),
              IMG_OUT "oln_convol_fast_gaussian_second_derivative.pbm");
    return 0;
}

```

**Warning:**

If sigma is big enough, the result may differ a little bit with the -O3 flag of g++.

Definition at line 275 of file fast_gaussian.hh.

References gaussian_second_derivative().

```

278    { return gaussian_second_derivative(convert::force<oln_value_type(I)>(),
279                                     in, sigma, behavior); }

```

6.11.2.9 `template<class C, class B, class I, class BE> mute<I, typename convoutput<C, B, typename mlc::exact< I >::ret::value_type>::ret>::ret gaussian_second_derivative(const convert::abstract::conversion< C, B > & input_conv, const abstract::image< I > & in, ntg::float_s sigma, const abstract::behavior< BE > & behavior)`

Second derivative gaussian filter.

Gaussian filter implementation from "Recursively implementing the gaussian and its derivatives" Deriche 93 INRIA REPORT (num RR-1893).

Parameters:

C Exact type of the conversion object.

B Base type of the conversion object.

I Exact type of the image.

BE Exact type of the behavior.

- input_conv Converter object.
- in Input image.
- sigma Value of sigma when computing the gaussian.
- behavior Object to know how to work on borders.

Warning:

If sigma is big enough, the result may differ a little bit with the -O3 flag of g++.

Referenced by gaussian_second_derivative().

6.12 oln::convol::fast::internal Namespace Reference

Internal purpose only.

Classes

- struct [gaussian_](#)
Compute the gaussian filter.
- struct [gaussian_< 1 >](#)
- struct [gaussian_< 2 >](#)
- struct [gaussian_< 3 >](#)
- struct [recursivefilter_coef_](#)
Data structure for coefficients used for a recursive filter call.

Functions

- template<class WorkType, class FloatType, class I> void [recursivefilter_](#) (I &[image](#), const [recursivefilter_coef_](#)< FloatType > &c, const oln_point_type(I)&start, const oln_point_type(I)&finish, [coord](#) len, const oln_dpoint_type(I)&d)
Recursive filter.

Variables

- [mute](#)< I, typename [convoutput](#)< C, B, oln_value_type(I)>::ret >::re [gaussian_common_](#))(const [convert::abstract::conversion](#)< C, B > &c, const [abstract::image](#)< I > &in, const F &coef, ntg::float_s sigma, const [abstract::behavior](#)< BE > &behavior)
Common code for filter (derivative, second derivative, etc.).

6.12.1 Detailed Description

Internal purpose only.

6.12.2 Function Documentation

- 6.12.2.1** template<class WorkType, class FloatType, class I> void recursivefilter_ (I & *image*, const recursivefilter_coef_< FloatType > & *c*, const oln_point_type(I)& *start*, const oln_point_type(I)& *finish*, coord *len*, const oln_dpoint_type(I)& *d*)

Recursive filter.

Recursive filter, works on a line.

Todo

FIXME: Until something clever is designed, the line is defined by two points (START and FINISH) and a displacement dpoint (D).

Parameters:

Worktype Type the algorithm work on.

FloatType Type of coefficients.

I Type of image.

- image Image to process.
- c Coefficients.
- start Start of the line.
- finish End point of the line.
- len Length of data to process.
- d Displacement dpoint.

Definition at line 67 of file fast_gaussian.hxx.

References `oln::coord,` `oln::convol::fast::internal::recursivefilter_coef_-`
`< FloatT >::d,` `oln::convol::fast::internal::recursivefilter_coef_< Float-`
`T >::dm,` `oln::convol::fast::internal::recursivefilter_coef_< FloatT >::n,` and
`oln::convol::fast::internal::recursivefilter_coef_< FloatT >::nm.`

```

73     {
74         std::vector<WorkType> tmp1(len);
75         std::vector<WorkType> tmp2(len);
76
77         // The fourth degree approximation implies to have a special
78         // look on the four first points we consider that there is
79         // no signal before 0 (to be discussed)
80
81         // --
82         // Causal part
83
84         tmp1[0] =
85             c.n[0]*image[start];
86
87         tmp1[1] =
88             c.n[0]*image[start + d]
89             + c.n[1]*image[start]
90             - c.d[1]*tmp1[0];
91
92         tmp1[2] =
93             c.n[0]*image[start + d + d]
94             + c.n[1]*image[start + d]
95             + c.n[2]*image[start]
96             - c.d[1]*tmp1[1]
97             - c.d[2]*tmp1[0];
98
99         tmp1[3] =
100             c.n[0]*image[start + d + d + d]
101             + c.n[1]*image[start + d + d]
102             + c.n[2]*image[start + d]
103             + c.n[3]*image[start]
104             - c.d[1]*tmp1[2] - c.d[2]*tmp1[1]
105             - c.d[3]*tmp1[0];
106
107         oln_point_type(I) current(start + d + d + d + d);
108         for (coord i = 4; i < len; ++i)
109             {
110                 tmp1[i] =
111                     c.n[0]*image[current]

```

```

112         + c.n[1]*image[current - d]
113         + c.n[2]*image[current - d - d]
114         + c.n[3]*image[current - d - d - d]
115         - c.d[1]*tmp1[i - 1] - c.d[2]*tmp1[i - 2]
116         - c.d[3]*tmp1[i - 3] - c.d[4]*tmp1[i - 4];
117     current += d;
118 }
119
120 // Non causal part
121
122 tmp2[len - 1] = 0;
123
124 tmp2[len - 2] =
125     c.nm[1]*image[finish];
126
127 tmp2[len - 3] =
128     c.nm[1]*image[finish - d]
129     + c.nm[2]*image[finish]
130     - c.dm[1]*tmp2[len-2];
131
132 tmp2[len - 4] =
133     c.nm[1]*image[finish - d - d]
134     + c.nm[2]*image[finish - d]
135     + c.nm[3]*image[finish]
136     - c.dm[1]*tmp2[len-3]
137     - c.dm[2]*tmp2[len-2];
138
139 current = finish - d - d - d ;
140
141 for (coord i = len - 5; i >= 0; --i)
142 {
143     tmp2[i] =
144         c.nm[1]*image[current]
145         + c.nm[2]*image[current + d]
146         + c.nm[3]*image[current + d + d]
147         + c.nm[4]*image[current + d + d + d]
148         - c.dm[1]*tmp2[i+1] - c.dm[2]*tmp2[i+2]
149         - c.dm[3]*tmp2[i+3] - c.dm[4]*tmp2[i+4];
150     current -= d;
151 }
152
153 // Combine results from causal and non-causal parts.
154
155 current = start;
156 for (coord i = 0; i < len; ++i)
157 {
158     image[current] = ntg::cast::force<oln_value_type(I)>(tmp1[i] + tmp2[i]);
159     current += d;
160 }
161 }

```

6.12.3 Variable Documentation

6.12.3.1 `mute<I, typename convoutput<C,B,oln_value_type(I)>::ret>::re
oln::convol::fast::internal::gaussian_common>(const con-
vert::abstract::conversion<C,B>& c, const abstract::image<I>& in, const F& coef,
ntg::float_s sigma, const abstract::behavior<BE> &behavior)`

Common code for filter (derivative, second derivative, etc.).

Parameters:

- C* Exact type of the conversion object.
- B* Base type of the conversion object.

I Exact type of the image.

BE Exact type of the behavior.

- input_conv Converter object.
- in Input image.
- sigma Value of sigma when computing the gaussian.
- behavior Object to know how to work on borders.

Definition at line 297 of file fast_gaussian.hxx.

```

302     {
303         typename mute<I, ntg::float_s>::ret work_img(in.size());
304
305         oln_iter_type(I) it(in);
306         for_all(it)
307             work_img[it] = ntg::cast::force<ntg::float_s>(in[it]);
308
309         // On tiny sigma, Derich algorithm doesn't work.
310         // It is the same thing that to convolve with a Dirac.
311         if (sigma > 0.006)
312         {
313             /* FIXME: relation between sigma and the border shouldn't
314              * be linear, so when sigma is big enough, the signal may
315              * be parasitized by the non signal values.
316              */
317             behavior.adapt_border(work_img, ntg::cast::round<coord>(5 * sigma));
318
319             gaussian<I::dim>::doit(work_img, coef);
320         }
321         /* Convert the result image to the user-requested datatype.
322          * FIXME: We are making an unnecessary copy in case the
323          * user expects a ntg::float_s image. */
324         typename mute<I, typename convoutput<C,B,oln_value_type(I)>::ret>::ret
325             out_img(in.size());
326
327         for_all(it)
328             out_img[it] = c(work_img[it]);
329
330         return out_img;
331     }

```

6.13 oln::convol::slow Namespace Reference

Convolution algorithms.

Functions

- `template<class DestValue, class I, class Win> mute< I, DestValue >::ret convolve (const abstract::image< I > &input, const abstract::w_window< Win > &win)`
Perform a convolution of an image and a window.
- `template<class DestValue, class I, class Info, class Win> mute< I, DestValue >::ret convolve (const abstract::image< I > &input, const mlc::array2d< Info, Win > &arr)`
Perform a convolution of an image and a window.

6.13.1 Detailed Description

Convolution algorithms.

6.13.2 Function Documentation

6.13.2.1 `template<class DestValue, class I, class Info, class Win> mute<I, DestValue>::ret convolve (const abstract::image< I > &input, const mlc::array2d< Info, Win > &arr)`

Perform a convolution of an image and a window.

Parameters:

DestValue Data type of the output image you want.

I Exact type of the input image.

Info Informations about the array.

Win Data type of the array.

- `input` The image to process.
- `arr` The array to convolve with.

Todo

FIXME: don't use array1d, ..., arraynd.

Definition at line 97 of file convolution.hh.

```

99      {
100          return convolve<DestValue>(input, static_cast< w_window2d<Win> >(arr));
101          // FIXME: Should be abstract::w_window<T_arr>. Adjust #include once done.
102      }
```

6.13.2.2 `template<class DestValue, class I, class Win> mute<I, DestValue>::ret convolve (const abstract::image< I > & input, const abstract::w_window< Win > & win)`

Perform a convolution of an image and a window.

Parameters:

DestValue Data type of the output image you want.

I Exact type of the input image.

Win Exact type of the window.

- input The image to process.
- win The window to convolve with.

Todo

FIXME: we must always specify DestValue.

Definition at line 62 of file convolution.hh.

References `oln::abstract::image< Exact >::border_adapt_copy()`, `oln::abstract::struct_elt< Exact >::card()`, `oln::abstract::struct_elt< Exact >::delta()`, `oln::abstract::struct_elt< Exact >::dp()`, `oln::abstract::image< Exact >::size()`, and `oln::abstract::w_window< Exact >::w()`.

```

64     {
65         mlc::eq<I::dim, Win::dim>::ensure();
66
67         typename mute<I, DestValue>::ret output(input.size());
68         input.border_adapt_copy(win.delta());
69         oln_iter_type(I) p_im(input);
70         for_all(p_im)
71         {
72             DestValue sum = ntg_zero_val(DestValue);
73             for (unsigned i = 0; i < win.card(); ++i)
74                 sum += static_cast<DestValue> (win.w(i)) *
75                     static_cast<DestValue> (input[p_im - win.dp(i)]);
76             output[p_im] = sum;
77         }
78
79         return output;
80     }

```


6.14 oln::impl Namespace Reference

Representation of the image in memory.

Classes

- class [image_array](#)
- class [image_array1d](#)
- class [image_array2d](#)
- class [image_array3d](#)
- class [image_impl](#)

6.14.1 Detailed Description

Representation of the image in memory.

6.15 oln::internal Namespace Reference

Internal purpose only.

Classes

- struct [default_less< abstract::dpoint< Exact > >](#)
- struct [default_less< abstract::point< Exact > >](#)
- struct [compose_uu_](#)
- struct [compose_ub_](#)
- struct [compose_bu_](#)
- struct [default_less< coord >](#)
[default_less<coord>](#)
- struct [default_less< dpoint1d >](#)
- struct [default_less< dpoint2d >](#)
- struct [default_less< dpoint3d >](#)
- struct [_fake](#)
- struct [default_less< point1d >](#)
- struct [default_less< point2d >](#)
- struct [default_less< point3d >](#)
- struct [wavelet_coeffs_](#)
Wavelet coefficient data structure.
- struct [dim_skip_iterate_rec_](#)
Functions used to iterate over all dimensions except one.
- struct [dim_skip_iterate_rec_< dim, skip, 0 >](#)
Specialization of [dim_skip_iterate_rec_](#) with current dimension set to 0.
- struct [dim_iterate_rec_](#)
Iterate over all dimensions except one.
- struct [dim_iterate_rec_< dim, 0 >](#)
Specialization of [dim_iterate_rec_](#) with skip dimension set to 0.

Enumerations

- enum [dwt_transform_dir_](#) { [dwt_fwd](#), [dwt_bwd](#) }
Enum to know if you go in forward or backward order.

Functions

- template<class I, class K> void [dwt_transform_step_](#) ([abstract::image< I >](#) &im, const typename [mlc::exact< I >::ret::point_type](#) &p_, const unsigned d, const unsigned n, const K &coeffs)
Step of a dwt transform.

- template<class I, class K> void [dwt_transform_inv_step_](#) (abstract::image< I > &im, const type-name mlc::exact< I >::ret::point_type &p_, const unsigned d, const unsigned n, const K &coeffs)

Step of a dwt invert transform.

- template<class I, class K> void [dwt_transform_](#) (abstract::image< I > &im, const unsigned l1, const unsigned l2, const K &coeffs, [transforms::dwt_transform_type](#) t)

Internal dwt transform function.

- template<class I, class K> void [dwt_transform_inv_](#) (abstract::image< I > &im, const unsigned l1, const unsigned l2, const K &coeffs, [transforms::dwt_transform_type](#) t)

Internal dwt invert transform function.

Variables

- const ntg::float_d [ln_2_](#) = 0.6931471805599453092

Value of $\ln(2)$.

6.15.1 Detailed Description

Internal purpose only.

6.15.2 Function Documentation

6.15.2.1 template<class I, class K> void dwt_transform_ (abstract::image< I > &im, const unsigned l1, const unsigned l2, const K &coeffs, transforms::dwt_transform_type t)

Internal dwt transform function.

Parameters:

I Exact type of the image to process.

K Type of coefficients.

Definition at line 414 of file dwt.hh.

```

419     {
420         oln_point_type(I) p;
421
422         switch (t) {
423             case transforms::dwt_std:
424                 dim_iterate_rec_<I::dim, I::dim>::doit(im, p, l1, l2, coeffs, dwt_fwd);
425                 break;
426             case transforms::dwt_non_std:
427                 for (unsigned n = l2; n >= l1; n >= 1)
428                     dim_iterate_rec_<I::dim, I::dim>::doit(im, p, n, n, coeffs, dwt_fwd);
429                 break;
430         }
431     }
```

6.15.2.2 `template<class I, class K> void dwt_transform_inv_ (abstract::image< I > & im, const unsigned l1, const unsigned l2, const K & coeffs, transforms::dwt_transform_type t)`

Internal dwt invert transform function.

Parameters:

I Exact type of the image to process.

K Type of coefficients.

Definition at line 440 of file dwt.hh.

```

445     {
446         oln_point_type(I) p;
447
448         switch (t) {
449             case transforms::dwt_std:
450                 dim_iterate_rec_<I::dim, I::dim>::doit(im, p, l1, l2, coeffs, dwt_bwd);
451                 break;
452             case transforms::dwt_non_std:
453                 for (unsigned n = l1; n <= l2; n <= 1)
454                     dim_iterate_rec_<I::dim, I::dim>::doit(im, p, n, n, coeffs, dwt_bwd);
455                 break;
456             }
457     }
```

6.15.2.3 `template<class I, class K> void dwt_transform_inv_step_ (abstract::image< I > & im, const typename mlc::exact< I >::ret::point_type & p_, const unsigned d, const unsigned n, const K & coeffs)`

Step of a dwt invert transform.

Parameters:

I Exact type of the input image.

K type of coefficients.

- *im* Image to process.
- *p_* Point to work on.
- *d* Component *d* of *p*.
- *n* Step number.
- *coeffs* Coefficients.

Definition at line 219 of file dwt.hh.

Referenced by `oln::internal::dim_skip_iterate_rec_< dim, skip, 0 >::doit()`.

```

224     {
225         assertion(n >= coeffs.size());
226
227         const unsigned half = n >> 1;
228         unsigned lim = coeffs.size() - 2;
229         ntg::float_d* tmp = new ntg::float_d[n];
230         oln_point_type(I) p(p_), q(p_);
231         unsigned i, j, k, l;
```

```

232
233     for (i = half - lim / 2, j = 0; j < lim; i++, j += 2) {
234         tmp[j] = 0;
235         tmp[j + 1] = 0;
236         for (k = 0, l = 0; l < coeffs.size(); k++, l += 2) {
237             p.nth(d) = (i + k) % half;
238             q.nth(d) = p.nth(d) + half;
239             tmp[j] += im[p] * coeffs.getInvH(l) + im[q] * coeffs.getInvH(l + 1);
240             tmp[j + 1] += im[p] * coeffs.getInvG(l) + im[q] * coeffs.getInvG(l + 1);
241         }
242     }
243     lim = n - 1;
244     for (i = 0; j < lim; i++, j += 2) {
245         tmp[j] = 0;
246         tmp[j + 1] = 0;
247         for (k = 0, l = 0; l < coeffs.size(); k++, l += 2) {
248             p.nth(d) = i + k;
249             q.nth(d) = p.nth(d) + half;
250             tmp[j] += im[p] * coeffs.getInvH(l) + im[q] * coeffs.getInvH(l + 1);
251             tmp[j + 1] += im[p] * coeffs.getInvG(l) + im[q] * coeffs.getInvG(l + 1);
252         }
253     }
254
255     for (i = 0; i < n; i++) {
256         p.nth(d) = i;
257         im[p] = tmp[i];
258     }
259
260     delete[] tmp;
261 }

```

6.15.2.4 template<class I, class K> void dwt_transform_step_ (abstract::image< I > &im, const typename mlc::exact< I >::ret::point_type &p_, const unsigned d, const unsigned n, const K &coeffs)

Step of a dwt transform.

Parameters:

I Exact type of the input image.

K type of coefficients.

- im Image to process.
- p_ Point to work on.
- d Component d of p.
- n Step number.
- coeffs Coefficients.

Definition at line 165 of file dwt.hh.

Referenced by oln::internal::dim_skip_iterate_rec_< dim, skip, 0 >::doit().

```

170     {
171         assertion(n >= coeffs.size());
172
173         const unsigned    half = n >> 1;
174         unsigned          lim = n + 1 - coeffs.size();

```

```
175     ntg::float_d*      tmp = new ntg::float_d[n];
176     oln_point_type(I) p(p_);
177     unsigned          i, j, k;
178
179     for (i = 0, j = 0; j < lim; j += 2, i++) {
180         tmp[i] = 0;
181         tmp[i + half] = 0;
182         for (k = 0; k < coeffs.size(); k++) {
183             p.nth(d) = j + k;
184             tmp[i] += im[p] * coeffs.getH(k);
185             tmp[i + half] += im[p] * coeffs.getG(k);
186         }
187     }
188     for (; i < half; j += 2, i++) {
189         tmp[i] = 0;
190         tmp[i + half] = 0;
191         for (unsigned k = 0; k < coeffs.size(); k++) {
192             p.nth(d) = (j + k) % n;
193             tmp[i] += im[p] * coeffs.getH(k);
194             tmp[i + half] += im[p] * coeffs.getG(k);
195         }
196     }
197
198     for (i = 0; i < n; i++) {
199         p.nth(d) = i;
200         im[p] = tmp[i];
201     }
202
203     delete[] tmp;
204 }
```

6.16 oln::io::gz Namespace Reference

Functions for gz files.

Classes

- class [zfilebuf](#)
Performs operation on compressed files.
- class [zfilestream_common](#)
Define an interface for compressed file stream manipulation.
- class [zifstream](#)
Read only zstream.
- class [zofstream](#)
- class [zomanip](#)
Define a pair func / val to perform manipulation on zofstream.

Functions

- `template<class T> zofstream & operator<< (zofstream &s, const zomanip< T > &m)`
Apply a function on s via the operator <<.
- `zofstream & setcompressionlevel (zofstream &s, int l)`
Set the compression level of s to l.
- `zofstream & setcompressionstrategy (zofstream &s, int l)`
Set the compression strategy of s to l.
- `zomanip< int > setcompressionlevel (int l)`
Specialized version for [zomanip](#)<int>.
- `zomanip< int > setcompressionstrategy (int l)`
Specialized version for [zomanip](#)<int>.

6.16.1 Detailed Description

Functions for gz files.

6.17 oln::level Namespace Reference

Level algorithm implementation.

Classes

- struct [f_invert](#)
Fctor to invert a value.
- class [hlut](#)
Look up table "id" version.
- class [hlut_def](#)
Look up table "default" version.
- class [threshold](#)
Threshold the value of the image.

Functions

- template<class DestType, class I, class E> [mute](#)< I, DestType >::ret [frontp_connected_component](#) (const [abstract::binary_image](#)< I > &input, const [abstract::neighborhood](#)< E > &se, unsigned &nb_label)
Processing frontp_connected_component extract the connected components.
- template<class DestType, class I, class E> [mute](#)< I, DestType >::ret [frontp_connected_component](#) (const [abstract::image](#)< I > &input, const [abstract::neighborhood](#)< E > &se)
- template<class I> [mute](#)< I, ntg::bin >::ret [extract_i_cc](#) (const [abstract::image](#)< I > &input, type-name mlc::exact< I >::ret::value_type i)
- template<class I> mlc::exact< I >::ret::value_type [get_n_cc](#) (const [abstract::image](#)< I > &input)
- template<class I1, class I2> bool [is_greater_or_equal](#) (const [abstract::image](#)< I1 > &input1, const [abstract::image](#)< I2 > &input2)
Tests if all pixels of input1 are greater or equal than input2.
- template<class I1, class I2> bool [is_greater](#) (const [abstract::image](#)< I1 > &input1, const [abstract::image](#)< I2 > &input2)
Tests if all pixels of input1 are greater than input2.
- template<class I1, class I2> bool [is_lower_or_equal](#) (const [abstract::image](#)< I1 > &input1, const [abstract::image](#)< I2 > &input2)
Tests if all pixels of input1 are lower or equal than input2.
- template<class I1, class I2> bool [is_lower](#) (const [abstract::image](#)< I1 > &input1, const [abstract::image](#)< I2 > &input2)
Tests if all pixel of input1 are lower than input2.
- template<class I1, class I2> bool [is_equal](#) (const [abstract::image](#)< I1 > &input1, const [abstract::image](#)< I2 > &input2)

Tests if input1 is equal to input2.

- template<class DestType, class I, class N> [mute](#)< I, DestType >::ret **connected_component** (const [abstract::binary_image_with_dim](#)< 2, I > &input, const [abstract::neighborhood](#)< N > &Ng)
- template<class I> [oln::mute](#)< I >::ret **fill** ([abstract::image](#)< I > &im, const typename mlc::exact< I >::ret::value_type &val)

Fill the image with a value.

- template<class I> [oln::mute](#)< I >::ret **invert** (const [abstract::image](#)< I > &input)

Return the image inverted.

- template<class I> void **invert_self** ([abstract::image](#)< I > &input)

Invert an image.

- template<class I> void **set_level** ([abstract::image_with_dim](#)< 2, I > &inout, const typename mlc::exact< I >::ret::point_type &p1, const typename mlc::exact< I >::ret::point_type &p2, typename mlc::exact< I >::ret::value_type level)

Draw a line between two points in the image.

- template<class I, class BoxType> void **set_level** ([abstract::image_with_dim](#)< 2, I > &inout, BoxType &box, const typename mlc::exact< I >::ret::value_type &level)

6.17.1 Detailed Description

Level algorithm implementation.

6.17.2 Function Documentation

6.17.2.1 template<class DestType, class I, class E> [mute](#)<I, DestType>::ret **frontp_connected_component** (const [abstract::binary_image](#)< I > &input, const [abstract::neighborhood](#)< E > &se, unsigned &nb_label)

Processing frontp_connected_component extract the connected components.

It removes the small (in area) connected components of the upper level sets of *input* using *se* as structural element.

REF: The implementation uses front propagation.

Precondition:

the input must be a binary image.

- input Image of markers.
- se Neighbourhood.
- nb_label Returns the number of label (optional).

Returns:

An image of type *DestType*.

See also:

level::connected_component

Todo

FIXME: Should probably be turned into a class.

```
#include <oln/basics2d.hh>
#include <oln/level/cc.hh>
#include <ntg/all.hh>
#include <iostream>

int main()
{
    unsigned card;
    oln::image2d<ntg::bin> light = oln::load(IMG_IN "face_se.pbm");
    save(oln::level::frontp_connected_component<ntg::int_u8>(light,
                                                             oln::neighb_c8(),
                                                             card),
        IMG_OUT "oln_level_frontp_connected_component.pgm");
    std::cout << card << " connected components" << std::endl;
}
```



Definition at line 96 of file cc.hh.

References `oln::abstract::image< Exact >::hold()`, and `oln::abstract::image< Exact >::size()`.

```
99     {
100         typename mute<I, DestType>::ret output(input.size());
101         level::fill(output, 0);
102
103         typedef std::set<oln_point_type(I),
104                        oln::internal::default_less<oln_point_type(I) > >
105                points_set;
106
107         I is_processed(input.size());
108         level::fill(is_processed, false);
109         DestType cur_label = 1;
110         oln_iter_type(I) p(input);
111         for_all(p) if ((input[p] == true)&& (is_processed[p] == false))
112             {
113                 //propagation front
114                 points_set component;
115                 component.insert(p.cur());
116                 points_set points_to_process;
117                 points_to_process.insert(p.cur());
118                 while (!points_to_process.empty())
119                     {
120                         //set label and net neighbors
121                         points_set next;
122                         for (typename points_set::const_iterator i =
123                             points_to_process.begin();
```

```

124         i != points_to_process.end();
125         ++i)
126     {
127         component.insert(*i);
128         oln_neighb_type(E) p_prime(se, *i);
129         for_all (p_prime) if(input.hold(p_prime) &&
130                        (input[p_prime] == true))
131             next.insert(p_prime.cur());
132     }
133     points_to_process.clear();
134     set_difference(next.begin(), next.end(),
135                  component.begin(), component.end(),
136                  inserter(points_to_process,
137                          points_to_process.begin()),
138                  oln::internal::default_less<oln_point_type(I) >());
139 }
140 for (typename points_set::const_iterator i = component.begin();
141      i != component.end();
142      ++i)
143 {
144     output[*i] = cur_label;
145     is_processed[*i] = true;
146 }
147 cur_label++;
148 }
149 nb_label = cur_label;
150 return output;
151 }

```

6.17.2.2 `template<class I1, class I2> bool is_equal (const abstract::image< I1 > &input1, const abstract::image< I2 > &input2) [inline]`

Tests if input1 is equal to input2.

Precondition:

`input1.size() == input2.size()`

Definition at line 112 of file compare.hh.

References `oln::abstract::image< Exact >::size()`.

```

114     {
115         precondition(input1.size() == input2.size());
116         oln_iter_type(I1) p(input1);
117         for_all (p)
118             if (!(input1[p] == input2[p]))
119                 return false;
120         return true;
121     }

```

6.17.2.3 `template<class I1, class I2> bool is_greater (const abstract::image< I1 > &input1, const abstract::image< I2 > &input2) [inline]`

Tests if all pixels of input1 are greater than input2.

Precondition:

`input1.size() == input2.size()`

Definition at line 60 of file compare.hh.

References `oln::abstract::image< Exact >::size()`.

```

62     {
63         precondition(input1.size() == input2.size());
64         oln_iter_type(I1) p(input1);
65         for_all (p)
66             if (!(input1[p] > input2[p]))
67                 return false;
68         return true;
69     }

```

6.17.2.4 `template<class I1, class I2> bool is_greater_or_equal (const abstract::image< I1 > & input1, const abstract::image< I2 > & input2) [inline]`

Tests if all pixels of input1 are greater or equal than input2.

Precondition:

`input1.size() == input2.size()`

Definition at line 43 of file compare.hh.

References `oln::abstract::image< Exact >::size()`.

```

45     {
46         precondition(input1.size() == input2.size());
47         oln_iter_type(I1) p(input1);
48         for_all (p)
49             if (!(input1[p] >= input2[p]))
50                 return false;
51         return true;
52     }

```

6.17.2.5 `template<class I1, class I2> bool is_lower (const abstract::image< I1 > & input1, const abstract::image< I2 > & input2) [inline]`

Tests if all pixel of input1 are lower than input2.

Precondition:

`input1.size() == input2.size()`

Definition at line 94 of file compare.hh.

References `oln::abstract::image< Exact >::size()`.

```

96     {
97         precondition(input1.size() == input2.size());
98         oln_iter_type(I1) p(input1);
99         for_all (p)
100             if (!(input1[p] < input2[p]))
101                 return false;
102         return true;
103     }

```

6.17.2.6 `template<class I1, class I2> bool is_lower_or_equal (const abstract::image< I1 > & input1, const abstract::image< I2 > & input2) [inline]`

Tests if all pixels of input1 are lower or equal than input2.

Precondition:

`input1.size() == input2.size()`

Definition at line 77 of file `compare.hh`.

References `oln::abstract::image< Exact >::size()`.

```

79      {
80          precondition(input1.size() == input2.size());
81          oln_iter_type(I1) p(input1);
82          for_all (p)
83              if (!(input1[p] <= input2[p]))
84                  return false;
85          return true;
86      }

```

6.17.2.7 `template<class I, class BoxType> void set_level (abstract::image_with_dim< 2, I > & inout, BoxType & box, const typename mlc::exact< I >::ret::value_type & level)`

Draw a box in the image.

See also:

[set_level](#) for an example.

Definition at line 141 of file `set_level.hh`.

References `set_level()`.

```

144      {
145          if (box.card() != 0)
146              {
147                  dpoint2d drows(box.top().row() - box.bottom().row(), 0);
148                  dpoint2d dcols(0, box.top().col() - box.bottom().col());
149                  set_level(inout, box.top(), box.top() - drows, level);
150                  set_level(inout, box.top(), box.top() - dcols, level);
151                  set_level(inout, box.bottom(), box.bottom() + drows, level);
152                  set_level(inout, box.bottom(), box.bottom() + dcols, level);
153              }
154      }

```

6.17.2.8 `template<class I> void set_level (abstract::image_with_dim< 2, I > & inout, const typename mlc::exact< I >::ret::point_type & p1, const typename mlc::exact< I >::ret::point_type & p2, typename mlc::exact< I >::ret::value_type level)`

Draw a line between two points in the image.

- `inout` a 2d image.
- `p1` A point in the image.

- p2 A point in the image (p1 and p2 can be swapped).
- level Value of the line.

```
#include <oln/basics2d.hh>
#include <oln/level/set_level.hh>
#include <ntg/all.hh>

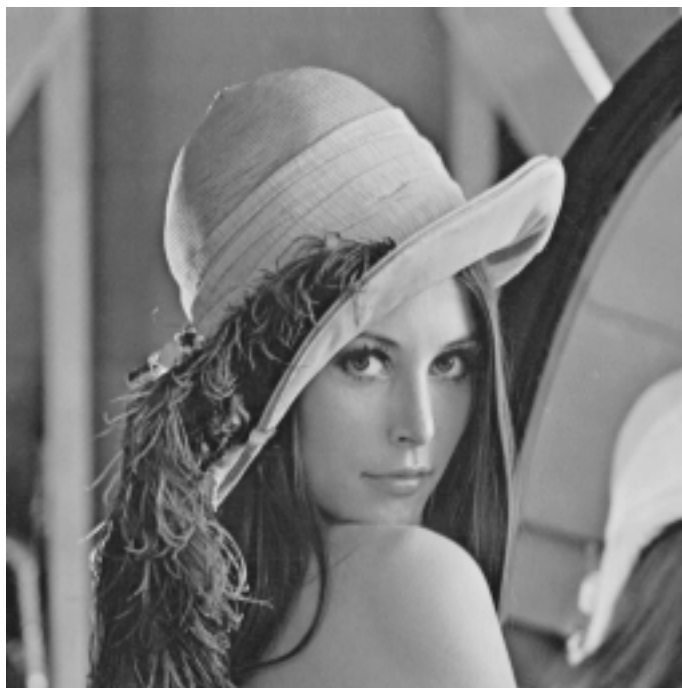
int main()
{
    oln::image2d<ntg::int_u8> in = oln::load(IMG_IN "lena256.pgm");
    oln::coord R = in.nrows();
    oln::coord C = in.ncols();

    //Draw the line
    oln::level::set_level(in,
                          oln::point2d(R * 1/4, C * 1/2),
                          oln::point2d(R * 4/5, C * 7/8),
                          0);

    oln::box<oln::point2d> b;
    b.add(oln::point2d(R * 1/6, C * 1/6));
    b.add(oln::point2d(R * 4/5, C * 1/3));

    //Draw the box
    oln::level::set_level(in, b, 255);

    save(in, IMG_OUT "oln_level_set_level.pgm");
}
```



=>



Definition at line 83 of file set_level.hh.

Referenced by set_level().

```
87     {
88         int iRow1 = p1.row();
89         int iCol1 = p1.col();
```

```
90     int iRow2 = p2.row();
91     int iCol2 = p2.col();
92     int dRow = ( iRow2 > iRow1 ? iRow2 - iRow1 : iRow1 - iRow2 );
93     int ddRow = 2 * dRow;
94     int dCol = ( iCol2 > iCol1 ? iCol2 - iCol1 : iCol1 - iCol2 );
95     int ddCol = 2 * dCol;
96     int sRow = ( iRow2 != iRow1 ? ( iRow2 > iRow1 ? 1 : -1 ) : 0 );
97     int sCol = ( iCol2 != iCol1 ? ( iCol2 > iCol1 ? 1 : -1 ) : 0 );
98     int i, e;
99     int iRow = iRow1;
100     int iCol = iCol1;
101
102     if ( dCol > dRow )
103     {
104         e = ddRow - dCol;
105         for ( i = 0; i < dCol; ++i )
106         {
107             inout(iRow, iCol) = level;
108             while ( e >= 0 )
109             {
110                 iRow += sRow;
111                 e -= ddCol;
112             }
113             iCol += sCol;
114             e += ddRow;
115         }
116     }
117     else
118     {
119         e = ddCol - dRow;
120         for ( i = 0; i < dRow; ++i )
121         {
122             inout(iRow,iCol) = level;
123             while ( e >= 0 )
124             {
125                 iCol += sCol;
126                 e -= ddRow;
127             }
128             iRow += sRow;
129             e += ddCol;
130         }
131     }
132     inout(iRow,iCol) = level;
133 }
```

6.18 oln::math Namespace Reference

Useful functions.

Classes

- struct [f_sqr](#)
Square fctor.
- struct [f_abs](#)
Absolute value fctor.

Functions

- template<class T> const T [sqr](#) (const T &val)
Square function.
- template<class T> const T [abs](#) (const T &val)
Absolute value function.
- template<typename DestValue, typename I, typename J> DestValue [dot_product](#) (const I &i, const J &j)
Dot product.

6.18.1 Detailed Description

Useful functions.

Todo

FIXME: I'm not proud of the code below think it could be better...

FIXME: this code sounds really odd. Why does the operator() take value<Self> instead of Self directly ? FIXME: Self should be renamed into Exact.

6.18.2 Function Documentation

6.18.2.1 template<typename DestValue, typename I, typename J> DestValue dot_product (const I &i, const J &j)

Dot product.

```
#include <oln/math/macros.hh>
#include <ntg/all.hh>
int main()
{
    ntg::int_u8 i(3), j(4);
    // Print "12"
    std::cout << oln::math::dot_product<ntg::float_s>(i, j) << std::endl;

    ntg::rgb_8 blue(0, 0, 200), yellow(0, 30, 10);
```



```
// Print "2000"
std::cout << oln::math::dot_product<ntg::float_s>(blue, yellow)
          << std::endl;
}
```

Definition at line 142 of file olena/oln/math/macros.hh.

```
143     {
144         typedef typename mlc::if_<ntg_is_a(I, ntg::vectorial)::ret,
145             internal::f_dot_product_v<DestValue, I, J>,
146             internal::f_dot_product_nv<DestValue, I, J> >::ret fctor;
147         return fctor::product(i, j);
148     }
```

6.19 oln::math::internal Namespace Reference

Internal purpose only.

Classes

- struct [f_dot_product_nv](#)
Dot product for non-vectorial types.
- struct [f_dot_product_v](#)
Dot product for vectorial types.

6.19.1 Detailed Description

Internal purpose only.

6.20 oln::morpher Namespace Reference

Contain all the morpher relative declarations and functions.

Classes

- class [super_color_morpher](#)
Abstract [color_morpher](#) class used for code factorization.
- struct [color_morpher](#)
- struct [color_morpher](#)< const SrcType, Exact >
The specialized version for 'const' declared images.
- struct [iter_morpher](#)
- struct [iter_morpher](#)< const SrcType, IterType, Exact >
The specialized version for 'const' declared images.
- class [super_piece_morpher](#)
Abstract piece morpher class used for code factorization.
- struct [piece_morpher](#)
The default piece morpher class.
- struct [piece_morpher](#)< const SrcType, Exact >
The specialized version for 'const' images.
- class [super_slicing_morpher](#)
Abstract slicing morpher class used for code factorization.
- struct [slicing_morpher](#)
The default slicing morpher class.
- struct [slicing_morpher](#)< const SrcType, Exact >
The specialized version for 'const' images.
- struct [color_mute](#)
- struct [color_mute](#)< ntg::color< nbcomps_, nbits_, color_system >, N >
Specialized version for [ntg::color](#).
- struct [subq_morpher](#)
Sub quantify an image.

Functions

- template<class I> const [color_morpher](#)< I > [rmorph](#) (I &ima)
Instantiate a temporary read-only [color_morpher](#).
- template<class I> const [color_morpher](#)< I > [gmorph](#) (I &ima)

Instantiate a temporary read-only [color_morpher](#).

- `template<class I> const color_morpher< I > bmorph (I &ima)`

Instantiate a temporary read-only [color_morpher](#).

- `template<class IterType, class I> const iter_morpher< I, IterType > iter_morph (I &ima)`

Instantiate a temporary read-only iter morpher.

- `template<class I, class PointType, class SizeType> const piece_morpher< I > piece_morph (I &ima, const PointType p, const SizeType s)`

Instantiate a temporary read-only piece morpher.

- *ima* The image.
- *p* The reference's point.
- *s* The size of the window.

- `oln::image1d_size image_size_dec (const oln::image2d_size &image_size)`

Return a size of N-1 dimension.

- `oln::image2d_size image_size_dec (const oln::image3d_size &image_size)`

Return a size of N-1 dimension.

- `template<class I> const slicing_morpher< I > slicing_morph (I &ima, coord slice)`

Instantiate a temporary read-only slicing morpher.

- *ima* The image.
- *slice* The slice.

- `template<unsigned S, class I> const subq_morpher< I, S > sqmorph (I &ima)`

Instantiate a temporary read-only [subq_morpher](#).

6.20.1 Detailed Description

Contain all the morpher relative declarations and functions.

6.20.2 Function Documentation

6.20.2.1 `template<class I> const color_morpher<I> bmorph (I & ima)`

Instantiate a temporary read-only [color_morpher](#).

The image will be viewed according to its blue layer.

```
#include <oln/morpher/color_morpher.hh>
#include <oln/basics2d.hh>
#include <ntg/all.hh>
int main()
{
    oln::image2d<ntg::rgb_8> imc = oln::load(IMG_IN "lena.ppm");
    oln::save(oln::morpher::bmorph(imc),
             IMG_OUT "oln_morpher_blue_morpher.pgm");
}
```





=>

Definition at line 397 of file color_morpher.hh.

```
398     {  
399         return color_morpher<I>(ima, ntg::rgb_B);  
400     }
```

6.20.2.2 `template<class I> const color_morpher<I> gmorph (I & ima)`

Instantiate a temporary read-only [color_morpher](#).

The image will be viewed according to its green layer.

```
#include <oln/morpher/color_morpher.hh>
#include <oln/basics2d.hh>
#include <ntg/all.hh>
int main()
{
    oln::image2d<ntg::rgb_8> imc = oln::load(IMG_IN "lena.ppm");
    oln::save(oln::morpher::gmorph(imc),
              IMG_OUT "oln_morpher_green_morpher.pgm");
}
```





=>

Definition at line 369 of file color_morpher.hh.

```
370     {  
371         return color_morpher<I>(ima, ntg::rgb_G);  
372     }
```

6.20.2.3 `template<class IterType, class I> const iter_morpher<I, IterType> iter_morph (I & ima)`

Instantiate a temporary read-only iter morpher.

The image will be viewed according to its iterator type. So the resulting image will be reversed.

```
#include <oln/morpher/iter_morpher.hh>
#include <oln/basics2d.hh>
#include <ntg/all.hh>
template <class E, class F>
void foo(const oln::abstract::image<E>& src,
         oln::abstract::image<F>& dst)
{
    oln_iter_type(oln::abstract::image<E>) it_src(src);
    oln_iter_type(oln::abstract::image<F>) it_dst(dst);

    it_dst = mlc::begin;
    for_all(it_src)
    {
        dst[it_dst] = src[it_src];
        ++it_dst;
    }
}
int main()
{
    const oln::image2d<ntg::rgb_8> im = oln::load(IMG_IN "lena.ppm");
    oln::image2d<ntg::rgb_8> im_out(im.size());

    foo(oln::morpher::iter_morph<
        oln_bkd_iter_type(oln::image2d<ntg::rgb_8>)>(im), im_out);
    oln::save(im_out, IMG_OUT "oln_morpher_iter.pgm");
}
```





=>

Definition at line 273 of file iter_morpher.hh.

```
274     {  
275         return iter_morpher<I, IterType>(ima);  
276     }
```

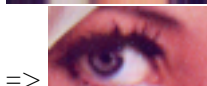
6.20.2.4 `template<class I, class PointType, class SizeType> const piece_morpher<I> piece_morph (I & ima, const PointType p, const SizeType s)`

Instantiate a temporary read-only piece morpher.

- ima The image.
- p The reference's point.
- s The size of the window.

A piece of the image will be viewed.

```
#include <oln/morpher/piece_morpher.hh>
#include <oln/basics2d.hh>
#include <ntg/all.hh>
int main()
{
    oln::image2d<ntg::rgb_8> imc = oln::load(IMG_IN "lena.ppm");
    oln::save(oln::morpher::piece_morph(imc,
                                         oln::dpoint2d(246, 244),
                                         oln::image2d_size(30, 60,
                                                             imc.border()))),
             IMG_OUT "oln_morpher_piece_morpher.pgm");
}
```

=>

Definition at line 347 of file piece_morpher.hh.

```
348     {  
349         return piece_morpher<I>(ima, p, s);  
350     }
```

6.20.2.5 `template<class I> const color_morpher<I> rmorph (I & ima)`

Instantiate a temporary read-only `color_morpher`.

The image will be viewed according to its red layer.

```
#include <oln/morpher/color_morpher.hh>
#include <oln/basics2d.hh>
#include <ntg/all.hh>
int main()
{
    oln::image2d<ntg::rgb_8> imc = oln::load(IMG_IN "lena.ppm");
    oln::save(oln::morpher::rmorph(imc),
              IMG_OUT "oln_morpher_red_morpher.pgm");
}
```






=>

Definition at line 341 of file color_morpher.hh.

```
342     {  
343         return color_morpher<I>(ima, ntg::rgb_R);  
344     }
```

6.20.2.6 `template<class I> const slicing_morpher<I> slicing_morph (I & ima, coord slice)`

Instantiate a temporary read-only slicing morpher.

- ima The image.

- slice The slice.

A slice of the image will be viewed.

```
#include <oln/morpher/slicing_morpher.hh>
#include <oln/basics2d.hh>
#include <ntg/all.hh>
int main()
{
    oln::image2d<ntg::rgb_8> imc = oln::load(IMG_IN "lena.ppm");
    oln::save(oln::morpher::slicing_morph(imc, 5),
              IMG_OUT "oln_morpher_slicing_morpher.pgm");
}
```



=>

Definition at line 405 of file slicing_morpher.hh.

References oln::coord.

```
406     {  
407         return slicing_morpher<I>(ima, slice);  
408     }
```

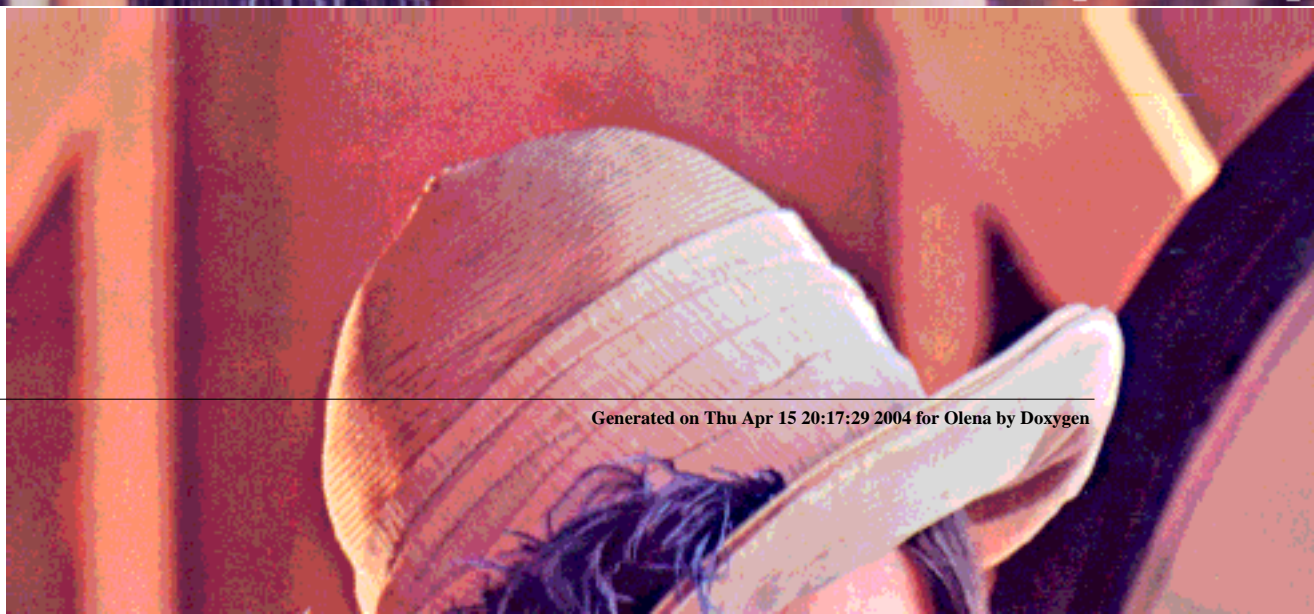
6.20.2.7 `template<unsigned S, class I> const subq_morpher<I, S> sqmorph (I & ima)`

Instantiate a temporary read-only `subq_morpher`.

Parameters:

S Indicate the color depth of the resulting image. It can't be higher than the color depth of *ima*.

```
#include <oln/morpher/subq_morpher.hh>
#include <oln/basics2d.hh>
#include <ntg/all.hh>
int main()
{
    oln::image2d<ntg::rgb_8> imc = oln::load(IMG_IN "lena.ppm");
    oln::save(oln::morpher::sqmorph<3>(imc),
              IMG_OUT "oln_morpher_subq_morpher.ppm");
}
```

Definition at line 222 of file subq_morpher.hh.

```
223     {  
224         return subq_morpher<I, S>(ima);  
225     }
```

6.21 oln::morpher::abstract Namespace Reference

Implementation of oln::abstract::generic_morpher.

Classes

- struct **dest_type**< 1, T >
- struct **dest_type**< 2, T >
- struct **dest_type**< 3, T >
- class [generic_morpher](#)

6.21.1 Detailed Description

Implementation of oln::abstract::generic_morpher.

6.22 oln::morpho Namespace Reference

Algorithm based on morphological mathematic.

Classes

- struct [sort_dimensions](#)
functor to sort dimensions.
- struct [cmp_queue_elt](#)

Functions

- template<class I, class E> [oln::mute](#)< I >::ret [closing](#) (const [abstract::non_vectorial_image](#)< I > &input, const [abstract::struct_elt](#)< E > &se)
Processing closing.
- template<class I, class E> [oln::mute](#)< I >::ret [dilation](#) (const [abstract::non_vectorial_image](#)< I > &input, const [abstract::struct_elt](#)< E > &se)
Processing dilation.
- template<class I, class E> [oln::mute](#)< I >::ret [n_dilation](#) (const [abstract::non_vectorial_image](#)< I > &input, const [abstract::struct_elt](#)< E > &se, unsigned n)
Perform morphological dilation iterated n times.
- template<class I, class E> [oln::mute](#)< I >::ret [erosion](#) (const [abstract::non_vectorial_image](#)< I > &input, const [abstract::struct_elt](#)< E > &se)
Perform a morphological erosion.
- template<class I, class E> [oln::mute](#)< I >::ret [n_erosion](#) (const [abstract::non_vectorial_image](#)< I > &input, const [abstract::struct_elt](#)< E > &se, unsigned n)
Perform morphological erosion iterated n times.
- template<class I, class N> [mute](#)< I, ntg::bin >::ret [internal_kill_cc_area](#) (const [abstract::binary_image_with_dim](#)< 2, I > &input, const unsigned int area, const [abstract::neighborhood](#)< N > &Ng)
Kill connex components smaller than a given area.
- template<class I, class N> [oln::mute](#)< I >::ret [sure_maxima_killer](#) (const [abstract::non_vectorial_image](#)< I > &input, const unsigned int area, const [abstract::neighborhood](#)< N > &Ng)
Maxima killer.
- template<class I, class N> [image2d](#)< ntg::int_u8 > [sure_minima_killer](#) (const [abstract::non_vectorial_image](#)< I > &input, const unsigned int area, const [abstract::neighborhood](#)< N > &Ng)
Minima killer.
- template<class P, class I, class N> bool [is_a_strict_minimum](#) (const [abstract::point](#)< P > &p, const [abstract::non_vectorial_image](#)< I > &input, const [abstract::neighborhood](#)< N > &Ng)
Check if a point is a strict minimum.

- template<class P, class I, class N> bool [is_a_strict_maximum](#) (const [abstract::point](#)< P > &p, const [abstract::non_vectorial_image](#)< I > &input, const [abstract::neighborhood](#)< N > &Ng)

Check if a point is a strict maximum.

- template<class I, class N> [oln::mute](#)< I >::ret [fast_minima_killer](#) (const [abstract::non_vectorial_image](#)< I > &input, const unsigned int area, const [abstract::neighborhood](#)< N > &Ng)

Minima killer.

- template<class I, class N> [oln::mute](#)< I >::ret [fast_maxima_killer](#) (const [abstract::non_vectorial_image](#)< I > &input, const unsigned int area, const [abstract::neighborhood](#)< N > &Ng)

Maxima killer.

- template<class I, class E> [oln::mute](#)< I >::ret [fast_morpho](#) (const [abstract::non_vectorial_image](#)< I > &input, const [abstract::struct_elt](#)< E > &se, typename [mlc::exact](#)< I >::ret::value_type(*func)(const [utils::histogram](#)< typename [mlc::exact](#)< I >::ret::value_type > &))

- template<class I, class E, class H> [oln::mute](#)< I >::ret [fast_morpho](#) (const [abstract::non_vectorial_image](#)< I > &input, const [abstract::struct_elt](#)< E > &se)

Fast morpho algorithm.

- template<class I1, class I2, class N> [oln::mute](#)< I1 >::ret [geodesic_dilation](#) (const [abstract::non_vectorial_image](#)< I1 > &marker, const [abstract::non_vectorial_image](#)< I2 > &mask, const [abstract::neighborhood](#)< N > &Ng)

Processing a geodesic dilation.

- template<class I1, class I2, class N> [oln::mute](#)< I1 >::ret [geodesic_erosion](#) (const [abstract::non_vectorial_image](#)< I1 > &marker, const [abstract::non_vectorial_image](#)< I2 > &mask, const [abstract::neighborhood](#)< N > &Ng)

Processing a geodesic erosion.

- template<class C, class B, class I, class E> [mute](#)< I, typename [convoutput](#)< C, B, typename [mlc::exact](#)< I >::ret::value_type >::ret >::ret [beucher_gradient](#) (const [convert::abstract::conversion](#)< C, B > &c, const [abstract::non_vectorial_image](#)< I > &input, const [abstract::struct_elt](#)< E > &se)

Process a morphological beucher gradient.

- template<class I, class E> [oln::mute](#)< I >::ret [external_gradient](#) (const [abstract::non_vectorial_image](#)< I > &input, const [abstract::struct_elt](#)< E > &se)

Process a morphological beucher gradient.

- template<class C, class B, class I, class E1, class E2> [mute](#)< I, typename [convoutput](#)< C, B, typename [mlc::exact](#)< I >::ret::value_type >::ret >::ret [hit_or_miss](#) (const [convert::abstract::conversion](#)< C, B > &c, const [abstract::non_vectorial_image](#)< I > &input, const [abstract::struct_elt](#)< E1 > &se1, const [abstract::struct_elt](#)< E2 > &se2)

Preform a 'hit or miss' transform.

- template<class I, class E1, class E2> [oln::mute](#)< I >::ret [hit_or_miss](#) (const [abstract::non_vectorial_image](#)< I > &input, const [abstract::struct_elt](#)< E1 > &se1, const [abstract::struct_elt](#)< E2 > &se2)

Preform a 'hit or miss' transform.

- `template<class I, class E1, class E2> oln::mute< I >::ret hit_or_miss_opening` (const `abstract::non_vectorial_image< I >` &input, const `abstract::struct_elt< E1 >` &se1, const `abstract::struct_elt< E2 >` &se2)
Perform an hit or miss opening.
- `template<class I, class E1, class E2> oln::mute< I >::ret hit_or_miss_opening_bg` (const `abstract::non_vectorial_image< I >` &input, const `abstract::struct_elt< E1 >` &se1, const `abstract::struct_elt< E2 >` &se2)
Perform an hit or miss opening of background.
- `template<class I, class E1, class E2> oln::mute< I >::ret hit_or_miss_closing` (const `abstract::non_vectorial_image< I >` &input, const `abstract::struct_elt< E1 >` &se1, const `abstract::struct_elt< E2 >` &se2)
Perform an hit or miss closing.
- `template<class I, class E1, class E2> oln::mute< I >::ret hit_or_miss_closing_bg` (const `abstract::non_vectorial_image< I >` &input, const `abstract::struct_elt< E1 >` &se1, const `abstract::struct_elt< E2 >` &se2)
Perform an hit or miss closing of background.
- `template<class C, class B, class I, class E> mute< I, typename convoutput< C, B, typename mlc::exact< I >::ret::value_type >::ret >::ret laplacian` (const `convert::abstract::conversion< C, B >` &c, const `abstract::non_vectorial_image< I >` &input, const `abstract::struct_elt< E >` &se)
Compute the laplacian of an image.
- `template<class DestValue, class I, class E> mute< I, DestValue >::ret laplacian` (const `abstract::non_vectorial_image< I >` &input, const `abstract::struct_elt< E >` &se)
Compute the laplacian of an image.
- `template<class I, class E> oln::mute< I >::ret opening` (const `abstract::non_vectorial_image< I >` &input, const `abstract::struct_elt< E >` &se)
Perform a morphological opening.
- `template<class E> mlc::exact< E >::ret get_plus_se_only` (const `abstract::struct_elt< E >` &se)
Get a sub part of a structuring element.
- `template<class E> mlc::exact< E >::ret get_plus_se_p` (const `abstract::struct_elt< E >` &se)
Get a sub part of a structuring element.
- `template<class E> mlc::exact< E >::ret get_minus_se_only` (const `abstract::struct_elt< E >` &se)
Get a sub part of a structuring element.
- `template<class E> mlc::exact< E >::ret get_minus_se_p` (const `abstract::struct_elt< E >` &se)
Get a sub part of a structuring element.
- `template<class I, class E> mlc::exact< I >::ret::value_type max` (const `abstract::non_vectorial_image< I >` &input, const `typename mlc::exact< I >::ret::point_type &p`, const `abstract::struct_elt< E >` &se)
Maximum of a structuring element.

- `template<class I, class E> mlc::exact< I >::ret::value_type min (const abstract::non_vectorial_image< I > &input, const typename mlc::exact< I >::ret::point_type &p, const abstract::struct_elt< E > &se)`

Minimum of a structuring element.

- `template<class I, class E1, class E2> oln::mute< I >::ret thickening (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E1 > &se1, const abstract::struct_elt< E2 > &se2)`

Thicken an image.

- `template<class I, class E1, class E2> oln::mute< I >::ret thinning (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E1 > &se1, const abstract::struct_elt< E2 > &se2)`

Thin an image.

- `template<class C, class B, class I, class E> mute< I, typename convoutput< C, B, typename mlc::exact< I >::ret::value_type >::ret >::ret white_top_hat (const convert::abstract::conversion< C, B > &c, const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Compute the white top hat of an image.

- `template<class I, class E> oln::mute< I >::ret white_top_hat (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Compute the white top hat of an image.

- `template<class C, class B, class I, class E> mute< I, typename convoutput< C, B, typename mlc::exact< I >::ret::value_type >::ret >::ret black_top_hat (const convert::abstract::conversion< C, B > &c, const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Compute the black top hat of an image.

- `template<class I, class E> oln::mute< I >::ret black_top_hat (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Compute the black top hat of an image.

- `template<class C, class B, class I, class E> mute< I, typename convoutput< C, B, typename mlc::exact< I >::ret::value_type >::ret >::ret self_complementary_top_hat (const convert::abstract::conversion< C, B > &c, const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Compute the self complementary top hat of an image.

- `template<class I, class E> oln::mute< I >::ret self_complementary_top_hat (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Compute the self complementary top hat of an image.

- `template<class C, class B, class I, class E> mute< I, typename convoutput< C, B, typename mlc::exact< I >::ret::value_type >::ret >::ret top_hat_contrast_op (const convert::abstract::conversion< C, B > &c, const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Top hat contrast operator.

- `template<class I, class E> oln::mute< I >::ret top_hat_contrast_op (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Top hat contrast operator.

- `template<class DestValue, class I, class N> mute< I, DestValue >::ret watershed_seg (const abstract::non_vectorial_image< I > &im_i, const abstract::neighborhood< N > &Ng)`
Segmented watershed.
- `template<class DestValue, class I, class N> mute< I, DestValue >::ret watershed_con (const abstract::non_vectorial_image< I > &im_i, const abstract::neighborhood< N > &Ng)`
Connected watershed.
- `template<class I1, class I2, class N> oln::mute< I2 >::ret & watershed_seg_or (const abstract::non_vectorial_image< I1 > &D, abstract::non_vectorial_image< I2 > &M, const abstract::neighborhood< N > &Ng)`
Segmented watershed with user-supplied starting points.

6.22.1 Detailed Description

Algorithm based on morphological mathematic.

6.22.2 Function Documentation

- 6.22.2.1** `template<class C, class B, class I, class E> mute<I, typename convoutput<C, B,typename mlc::exact< I >::ret::value_type>::ret>::ret beucher_gradient (const convert::abstract::conversion< C, B > &c, const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Process a morphological beucher gradient.

Compute the arithmetic difference between the diltation and the erosion of input using se as structural element. Soille, p67.

- c Conversion functor.
- input Image to process.
- se Structuring element.

Definition at line 79 of file gradient.hh.

- 6.22.2.2** `template<class I, class E> oln::mute< I >::ret black_top_hat (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Compute the black top hat of an image.

Parameters:

- I* Exact type of the image.
- E* Exact type of the structuring element.
- input Image to process.
- se Structuring element.

```

#include <oln/basics2d.hh>
#include <oln/morpho/top_hat.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

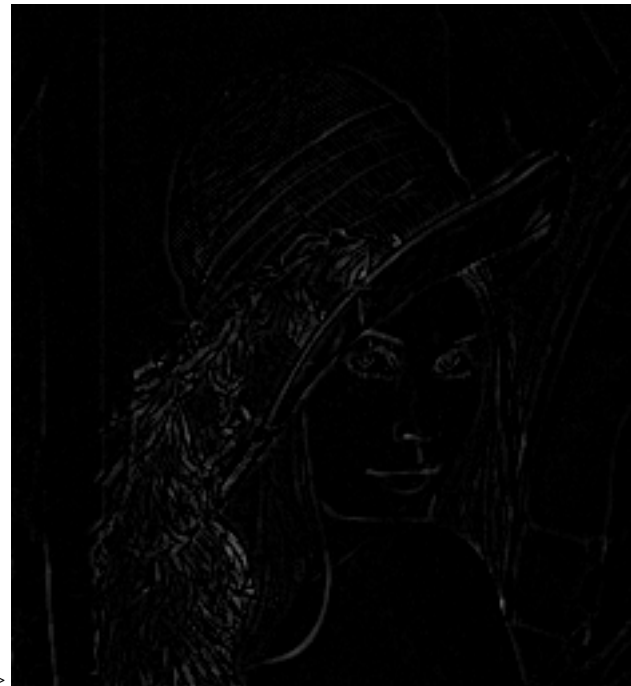
    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::morpho::fast::black_top_hat(im1,
                                                oln::win_c8p()),
              IMG_OUT "oln_morpho_fast_black_top_hat_overload.pbm");
}

```



=>



Definition at line 233 of file top_hat.hh.

6.22.2.3 `template<class C, class B, class I, class E> mute<I, typename convoutput<C, B,typename mlc::exact< I >::ret::value_type>::ret>::ret black_top_hat (const convert::abstract::conversion< C, B > & c, const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Compute the black top hat of an image.

Compute black top hat of input using *se* as structuring element. Soille p.105.

- *c* Conversion object.
- *input* Image to process.
- *se* Structuring element.

```

#include <oln/basics2d.hh>
#include <oln/morpho/top_hat.hh>
#include <oln/level/compare.hh>

```

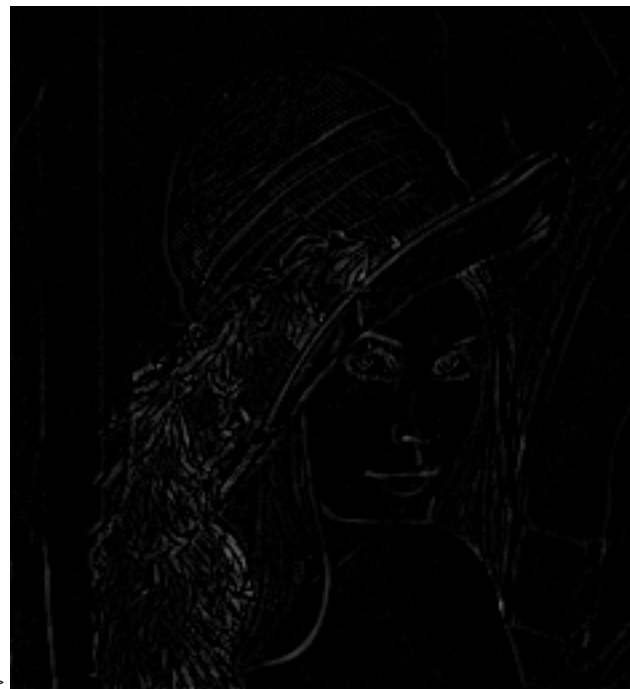
```
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::morpho::fast::black_top_hat
              (oln::convert::bound<ntg::int_u8>(),
               im1,
               oln::win_c8p()),
              IMG_OUT "oln_morpho_fast_black_top_hat.pbm");
}
```



=>



Definition at line 191 of file top_hat.hh.

6.22.2.4 template<class I, class E> [oln::mute](#)< I >::ret closing (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)

Processing closing.

Compute the morphological closing of input using se as structuring element.

Parameters:

- I* Exact type of the input image.
- E* Exact type of the structuring element.

- input Input image to close.
- se Structuring element to use.

```
#include <oln/basics2d.hh>
#include <oln/morpho/closing.hh>
#include <oln/level/compare.hh>
```

```

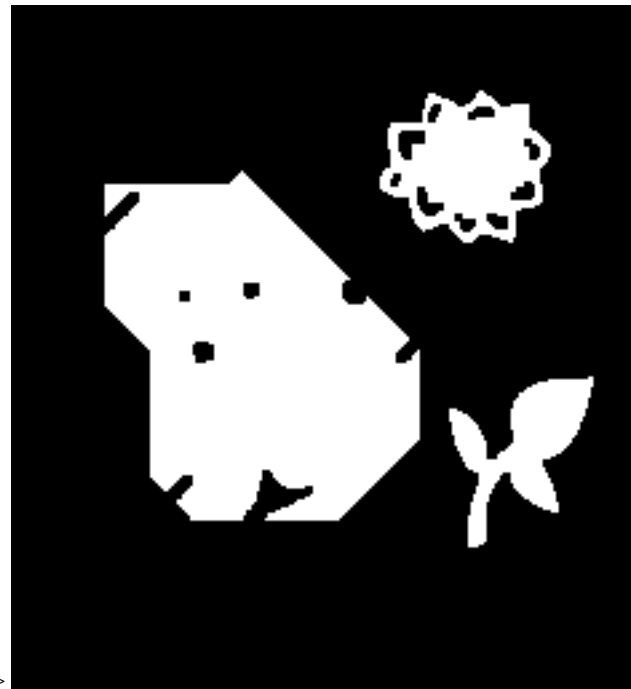
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>      im_type;

    im_type im1(oln::load(IMG_IN "object.pbm"));
    save(oln::morpho::closing(im1, oln::win_c8p()),
        IMG_OUT "oln_morpho_closing.pbm");
    return 0;
}

```



=>



Definition at line 118 of file closing.hh.

6.22.2.5 `template<class I, class E> oln::mute< I >::ret dilation (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Processing dilation.

Compute the morphological dilation of input using *se* as structural element.

On grey-scale images, each point is replaced by the maximum value of its neighbors, as indicated by *se*.
On binary images, a logical or is performed between neighbors.

The [morpho::fast](#) version of this function use a different algorithm: This algorithm is described in Implementation of morphological operations from: M. Van Droogenbroeck and H. Talbot. "Fast computation of morphological operations with arbitrary structuring elements". Pattern Recognition Letters, 17(14):1451-1460, 1996.

An histogram of the value of the neighborhood indicated by *se* is updated while iterating over all point of the image. Doing so is more efficient when the structural element is large.

Parameters:

- I* Exact type of the input image.
- E* Exact type of the neighborhood.

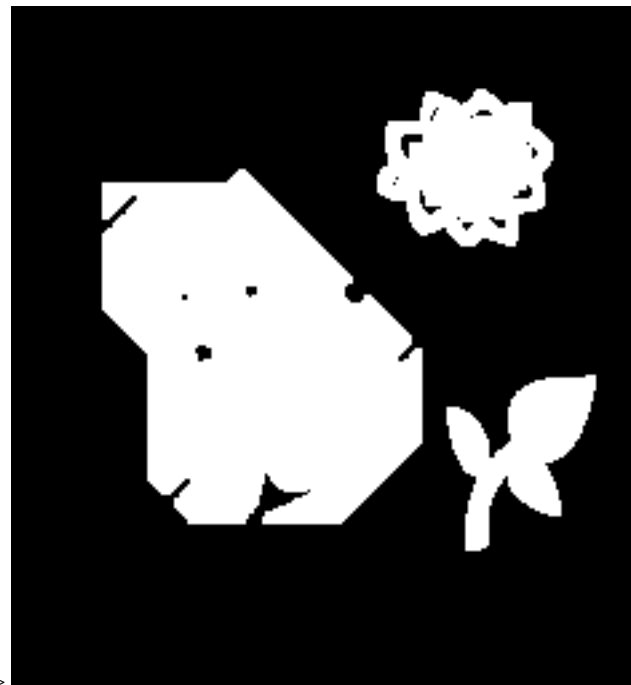
- input The input image.
- se Structuring element to use.

```
#include <oln/basics2d.hh>
#include <oln/morpho/dilation.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>      im_type;

    im_type im1(oln::load(IMG_IN "object.pbm"));
    save(oln::morpho::dilation(im1, oln::win_c8p()),
        IMG_OUT "oln_morpho_dilation.pbm");
}
```



=>



Definition at line 92 of file dilation.hh.

References `oln::abstract::image< Exact >::border_adapt_copy()`, `oln::abstract::struct_elt< Exact >::delta()`, and `oln::abstract::image< Exact >::size()`.

Referenced by `geodesic_dilation()`, and `n_dilation()`.

```
94     {
95         mlc::eq<I::dim, E::dim>::ensure();
96
97         oln_concrete_type(I) output(input.size());
98         input.border_adapt_copy(se.delta());
99         oln_iter_type(I) p(input);
100
101         for_all (p)
102             output[p] = morpho::max(input, p, se);
103         return output;
104     }
```

6.22.2.6 `template<class I, class E> oln::mute< I >::ret erosion (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Perform a morphological erosion.

Compute the morphological erosion of input using *se* as structuring element.

On grey-scale images, each point is replaced by the minimum value of its neighbors, as indicated by *se*. On binary images, a logical and is performed between neighbors. The [morpho::fast](#) version of this function use a different algorithm: an histogram of the value of the neighborhood indicated by *se* is updated while iterating over all point of the image. Doing so is more efficient when the structuring element is large.

Parameters:

- I* Exact type of the input image.
- E* Exact type of the structuring element.

- *input* Input image.
- *se* Structuring element to use.

```
#include <oln/basics2d.hh>
#include <oln/morpho/erosion.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>          im_type;

    im_type im1(oln::load(IMG_IN "object.pbm"));
    save(oln::morpho::erosion(im1, oln::win_c8p()),
        IMG_OUT "oln_morpho_erosion.pbm");
}
```



=>



See also:

[oln::morpho::fast::erosion\(\)](#)

Definition at line 86 of file erosion.hh.

References `oln::abstract::image< Exact >::border_adapt_copy()`, `oln::abstract::struct_elt< Exact >::delta()`, and `oln::abstract::image< Exact >::size()`.

Referenced by `geodesic_erosion()`, and `n_erosion()`.

```

88     {
89         mlc::eq<I::dim, E::dim>::ensure();
90         oln_concrete_type(I) output(input.size());
91         se.delta();
92         input.border_adapt_copy(se.delta());
93         oln_iter_type(I) p(input);
94         for_all (p)
95             output[p] = morpho::min(input, p, se);
96         return output;
97     }

```

6.22.2.7 `template<class I, class E> oln::mute< I >::ret external_gradient (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Process a morphological beucher gradient.

Parameters:

- I* Exact type of the input image.
- E* Exact type of the structuring element.

Returns:

The beucher gradient of the input.

- input Image to process.
- se Structuring element.

```

#include <oln/basics2d.hh>
#include <oln/morpho/gradient.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena128.pgm"));

    save(oln::morpho::beucher_gradient(im1, oln::win_c8p()),
        IMG_OUT "oln_morpho_beucher_gradient.pbm");
    return 0;
}
\endcode

\image html lena128_pgm.png
\image latex lena128_pgm.png
=>
\image html oln_morpho_beucher_gradient.png
\image latex oln_morpho_beucher_gradient.png

```

```

*/
template<class I, class E>
typename oln::mute< I >::ret
    beucher_gradient(const abstract::non_vectorial_image<I>& input,
                    const abstract::struct_elt<E>& se)
{
    return beucher_gradient(convert::force<typename mlc::exact< I >::ret::value_type>(), input, se);
}

template<class C, class B, class I, class E>
typename mute<I, typename convoutput<C, B,typename mlc::exact< I >::ret::value_type>::ret>::ret
internal_gradient(const convert::abstract::conversion<C, B>& c,
                const abstract::non_vectorial_image<I>& input,
                const abstract::struct_elt<E>& se)
{
    return arith::minus(c, input, erosion(input, se));
}

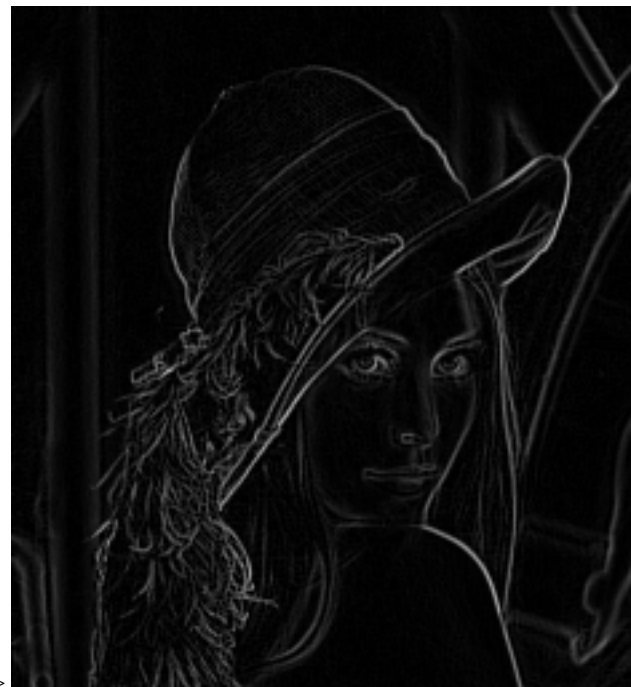
template<class I, class E>
typename oln::mute< I >::ret
    internal_gradient(const abstract::non_vectorial_image<I>& input, const
                    abstract::struct_elt<E>& se)
{
    return internal_gradient(convert::force<typename mlc::exact< I >::ret::value_type>(), input, se);
}

template<class C, class B, class I, class E>
typename mute<I, typename convoutput<C, B,typename mlc::exact< I >::ret::value_type>::ret>::ret
external_gradient(const convert::abstract::conversion<C, B>& c,
                const abstract::non_vectorial_image<I>& input,
                const abstract::struct_elt<E>& se)
{
    return arith::minus(c, dilation(input, se), input);
}

```



=>



Definition at line 252 of file gradient.hh.

6.22.2.8 `template<class I, class N> oln::mute< I >::ret fast_maxima_killer (const abstract::non_vectorial_image< I > & input, const unsigned int area, const abstract::neighborhood< N > & Ng)`

Maxima killer.

It removes the small (in area) connected components of the upper level sets of input using Ng as neighborhood. The implementation is based on *stak*. Guichard and Morel, Image iterative smoothing and PDE's. Book in preparation. p 265.

See also:

[oln::morpho::fast::card_opening](#).

Parameters:

I Image exact type.

N Neighborhood exact type.

- input The input image.
- area Threshold to use.
- Ng The neighborhood to use.

```
#include <oln/basics2d.hh>
#include <oln/morpho/extrema_killer.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type                                im(oln::load(IMG_IN "lena128.pgm"));

    oln::save(oln::morpho::fast_maxima_killer(im, 200, oln::neighb_c4()),
              IMG_OUT "oln_morpho_fast_maxima_killer.pgm");
    return 0;
}
```



Warning:

Even if it is called fast maxima killer, this algorithm is slow. Please use card closing instead.

Definition at line 486 of file `extrema_killer.hh`.

References `oln::abstract::image< Exact >::clone()`, `is_a_strict_maximum()`, and `oln::abstract::image< Exact >::size()`.

```

489 {
490     mlc::eq<I::dim, N::dim>::ensure();
491
492     std::vector<oln_point_type(I)> cur_maximum;
493     oln_concrete_type(I) working_input = input.clone();
494     typename mute<I, ntg::bin>::ret not_processed_map(input.size());
495     level::fill(not_processed_map, true);
496
497     // STEP 2: search for a local minimum
498     oln_iter_type(I) p(working_input);
499     for_all(p)
500     {
501         if (is_a_strict_maximum(p.cur(), working_input, Ng)
502             && not_processed_map[p])
503         {
504             cur_maximum.push_back(p);
505             oln_value_type(I) lambda = working_input[p];
506
507             typename mute<I, ntg::bin>::ret is_visited(input.size());
508             level::fill(is_visited, false);
509             is_visited[p] = true;
510             //STEP 3
511             bool go_on = true; // FIXME: ditto
512             while (go_on)
513             {
514                 typedef oln_value_type(I) I_type;
515                 oln_point_type(I) arg_max = p;
516                 oln_value_type(I) max = ntg_min_val(I_type);
517                 for (unsigned i = 0; i < cur_maximum.size(); ++i)
518                 {
519                     oln_neighb_type(N) p_prime(Ng, cur_maximum[i]);
520                     for_all(p_prime) if (working_input.hold(p_prime) &&
521                                         (is_visited[p_prime] == false))
522                     {
523                         if (working_input[p_prime] >= max)
524                         {
525                             max = working_input[p_prime];
526                             arg_max = p_prime;
527                         }
528                     }
529                 }
530                 // go to step 4
531                 if (working_input[arg_max] <= lambda)
532                 {
533                     // step 5
534                     if (cur_maximum.size() < area)
535                     {
536                         lambda = working_input[arg_max]; // END MODIF2
537                         cur_maximum.push_back(arg_max); //MODIF 1
538                         is_visited[arg_max] = true;
539                         // go to step 3
540                     }
541                     else
542                     {
543                         go_on = false;
544                         //go to step 6
545                     }
546                 }
547                 else
548                 {
549                     go_on = false;
550                     //go to step 6
551                 }
552             } // end while "go on"
553
554             // going to step 6
555             for (unsigned i = 0; i < cur_maximum.size(); ++i)

```

```

556         {
557             //          cout << cur_minimum[i] << " " << lambda << endl;
558             working_input[cur_maximum[i]] = lambda;
559             not_processed_map[cur_maximum[i]] = false;
560         }
561         cur_maximum.erase(cur_maximum.begin(), cur_maximum.end());
562
563         // STEP 7: look for a new minimum
564     } // end processing current minimum
565
566     } // END looking for minimum
567     return working_input;
568 }

```

6.22.2.9 `template<class I, class N> oln::mute< I >::ret fast_minima_killer (const abstract::non_vectorial_image< I > &input, const unsigned int area, const abstract::neighborhood< N > &Ng)`

Minima killer.

It removes the small (in area) connected components of the lower level sets of input using Ng as neighborhood. The implementation is based on *stak*. Guichard and Morel, Image iterative smoothing and PDE's. Book in preparation. p 265.

See also:

[oln::morpho::fast::card_closing](#)

Parameters:

I Image exact type.

N Neighborhood exact type.

- input The input image.
- area Threshold to use.
- Ng The neighborhood to use.

```

#include <oln/basics2d.hh>
#include <oln/morpho/extrema_killer.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type                                im(oln::load(IMG_IN "lena128.pgm"));

    oln::save(oln::morpho::fast_minima_killer(im, 200, oln::neighb_c4()),
              IMG_OUT "oln_morpho_fast_minima_killer.pgm");
    return 0;
}

```

**Warning:**

Even if it is called fast minima killer, this algorithm is slow. Please use card opening instead.

Definition at line 359 of file extrema_killer.hh.

References `oln::abstract::image< Exact >::clone()`, `is_a_strict_minimum()`, and `oln::abstract::image< Exact >::size()`.

```

362     {
363         mlc::eq<I::dim, N::dim>::ensure();
364
365         std::vector<oln_point_type(I)> cur_minimum;
366         cur_minimum.reserve(15000);
367         oln_concrete_type(I) working_input = input.clone();
368         typename mute<I, ntg::bin>::ret not_processed_map(input.size());
369         level::fill(not_processed_map, true);
370
371         // STEP 2: search for a local minimum
372         oln_iter_type(I) p(working_input);
373         for_all(p)
374         {
375             if (is_a_strict_minimum(p.cur(), working_input, Ng)
376                 && not_processed_map[p])
377             {
378                 cur_minimum.push_back(p);
379                 oln_value_type(I) lambda = working_input[p];
380
381                 // FIXME: This should better be moved out of the loop.
382                 typename mute<I, ntg::bin>::ret is_visited(input.size());
383                 level::fill(is_visited, false);
384                 is_visited[p] = true;
385                 //STEP 3
386                 bool go_on = true; // FIXME: Use break instead of go_on = false.
387                 while (go_on)
388                 {
389                     typedef oln_value_type(I) I_type;
390                     oln_point_type(I) arg_min = p;
391                     oln_value_type(I) min = ntg_max_val(I_type);
392                     for (unsigned i = 0; i < cur_minimum.size(); ++i)
393                     {
394                         oln_neighb_type(N) p_prime(Ng, cur_minimum[i]);
395                         for_all(p_prime) if (working_input.hold(p_prime) &&
396                             (is_visited[p_prime] == false))
397                         {
398                             if (working_input[p_prime] <= min)
399                             {
400                                 min = working_input[p_prime];
401                                 arg_min = p_prime;
402                             }
403                         }
404                     }
405                     // go to step 4

```

```

406         if (working_input[arg_min] >= lambda)
407         {
408             // step 5
409             if (cur_minimum.size() < area)
410             {
411                 lambda = working_input[arg_min]; // END MODIF2
412                 cur_minimum.push_back(arg_min); //MODIF 1
413                 is_visited[arg_min] = true;
414                 // go to step 3
415             }
416             else
417             {
418                 go_on = false;
419                 //go to step 6
420             }
421         }
422         else
423         {
424             go_on = false; //go to step 6
425         }
426     } // end while "go on"
427
428     // going to step 6
429     for (unsigned i = 0; i < cur_minimum.size(); ++i)
430     {
431         working_input[cur_minimum[i]] = lambda;
432         not_processed_map[cur_minimum[i]] = false;
433     }
434     cur_minimum.erase(cur_minimum.begin(), cur_minimum.end());
435
436     // STEP 7: look for a new minimum
437 } // end processing current minimum
438
439 } // END looking for minimum
440 return working_input;
441 }

```

6.22.2.10 template<class I, class E, class H> [oln::mute](#)< I >::ret fast_morpho (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)

Fast morpho algorithm.

Todo

FIXME: This algorithm should be moved to the namespace [oln::morpho::fast](#).

FIXME: add tests.

```

#include <oln/basics2d.hh>
#include <oln/morpho/erosion.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>          im_type;

    im_type im1(oln::load(IMG_IN "object.pbm"));
    save(oln::morpho::fast_morpho<im_type, oln::window2d,
                                     oln::utils::histogram_min<ntg::bin> >
        (im1, oln::win_c8p()),
        IMG_OUT "oln_morpho_fast_morpho.pbm");
}

```



=>



Definition at line 316 of file fast_morpho.hxx.

References `oln::abstract::image< Exact >::border_adapt_copy()`, `oln::abstract::struct_elt< Exact >::delta()`, and `oln::abstract::image< Exact >::size()`.

```

318     {
319         enum { dim = E::dim };
320
321         // prepare output
322         oln_concrete_type(I) output(input.size());
323         input.border_adapt_copy(se.delta());
324
325         // compute delta structuring elements for forward movements
326         E se_add[dim];
327         E se_rem[dim];
328         internal::find_struct_elts(se, se_add, se_rem);
329
330         // compute delta structuring elements for backward movements
331         E se_add_back[dim];
332         E se_rem_back[dim];
333         for (unsigned n = 0; n < dim; ++n)
334             for (unsigned i = 0; i < se_add[n].card(); ++i)
335             {
336                 oln_dpoint_type(I) dp = se_add[n].dp(i);
337                 dp.nth(n) += 1;
338                 se_rem_back[n].add(dp);
339
340                 dp = se_rem[n].dp(i);
341                 dp.nth(n) += 1;
342                 se_add_back[n].add(dp);
343             }
344
345         // Order dimensions
346         unsigned dims[dim];
347         for (unsigned n = 0; n < dim; ++n)
348             dims[n] = n;
349         sort_dimensions<E> s(se_add);
350         std::sort(dims, dims + dim, s);
351     }

```



```

352     const typename I::size_type size = input.size();
353
354     // Initialize the histogram with the values around the first point.
355     H hist;
356     oln_point_type(I) p;
357
358     //     oln_iter_type(E) dp(se);
359     typename E::iter_type    dp(se);
360     for_all(dp)
361         ++hist[input[p + dp]];
362
363     // Process the image.
364     // -----
365
366     output[p] = hist.res();    // First point.
367
368     internal::fast_morpho_inner<l, E::dim, const I,
369         const typename I::size_type, H,
370         const E, oln_point_type(I), oln_concrete_type(I)>::doit(input.exact(), size, hist,
371                                                                 se_add, se_rem,
372                                                                 se_add_back, se_rem_back,
373                                                                 p, output, dims);
374     return output;
375 }

```

6.22.2.11 `template<class I, class E> oln::mute< I >::ret fast_morpho (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se, typename mlc::exact< I >::ret::value_type(*func)(const utils::histogram< typename mlc::exact< I >::ret::value_type > &)) [inline]`

Do not exist !!!

Todo

FIXME: REMOVE ME.

6.22.2.12 `template<class I1, class I2, class N> oln::mute< I1 >::ret geodesic_dilation (const abstract::non_vectorial_image< I1 > &marker, const abstract::non_vectorial_image< I2 > &mask, const abstract::neighborhood< N > &Ng)`

Processing a geodesic dilation.

Parameters:

I1 Exact type of image marker.

I2 Exact type of image mask.

N Exact type of neighborhood.

- marker Image to work on.
- mask Image used for geodesic dilation.
- Ng Neighborhood to use.

Compute the geodesic dilation of marker with respect to the mask image using se as structuring element. Soille p.156.

Precondition:

Mask must be greater or equal than marker.

```
#include <oln/basics2d.hh>
#include <oln/morpho/opening.hh>
#include <oln/morpho/geodesic_dilation.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena128.pgm"));
    im_type    im2 = oln::morpho::opening(im1, oln::win_c4p());

    save(oln::morpho::geodesic_dilation(im2, im1, oln::neighb_c4()),
        IMG_OUT "oln_morpho_geodesic_dilation.pbm");

    return 0;
}
```



Definition at line 89 of file geodesic_dilation.hh.

References dilation(), and oln::abstract::image< Exact >::size().

```
92    {
93        mlc::eq<I1::dim, I2::dim>::ensure();
94        mlc::eq<I1::dim, N::dim>::ensure();
95        precondition(marker.size() == mask.size());
96        precondition(level::is_greater_or_equal(mask, marker));
97        return arith::min<oln_concrete_type(I1)>(dilation(marker,
98                                                    convert::ng_to_cse(Ng)),
99                                                    mask);
100    }
```

6.22.2.13 `template<class I1, class I2, class N> oln::mute< I1 >::ret geodesic_erosion (const abstract::non_vectorial_image< I1 > & marker, const abstract::non_vectorial_image< I2 > & mask, const abstract::neighborhood< N > & Ng)`

Processing a geodesic erosion.

Parameters:

- I1* Exact type of image marker.
- I2* Exact type of image mask.
- N* Exact type of neighborhood.

- marker Image to work on.

- mask Image used for geodesic dilation.
- Ng Neighborhood to use.

Compute the geodesic erosion of marker with respect to the mask mask image using se as structural element. Soille p.158.

Precondition:

Marker must be greater or equal than mask.

```
#include <oln/basics2d.hh>
#include <oln/morpho/opening.hh>
#include <oln/morpho/geodesic_erosion.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type      im1(oln::load(IMG_IN "lena128.pgm"));
    im_type      im2 = oln::morpho::opening(im1, oln::win_c4p());

    save(oln::morpho::geodesic_erosion(im1, im2, oln::neighb_c4()),
        IMG_OUT "oln_morpho_geodesic_erosion.pbm");
    return 0;
}
```



Definition at line 87 of file geodesic_erosion.hh.

References erosion(), and oln::abstract::image< Exact >::size().

```
90      {
91          mlc::eq<I1::dim, I2::dim>::ensure();
92          mlc::eq<I1::dim, N::dim>::ensure();
93          precondition(marker.size() == mask.size());
94          precondition(level::is_greater_or_equal(marker, mask));
95          return arith::max<oln_concrete_type(I1)>(erosion(marker, convert::ng_to_cse(Ng)), mask);
96      }
```

6.22.2.14 template<class E> mlc::exact< E >::ref get_minus_se_only (const abstract::struct_elt< E > & se)

Get a sub part of a structuring element.

Parameters:

E Exact type of the structuring element.

- se The structuring element.

A point p take part of the new structuring element if it exists a i that belongs to $[[0..\text{dim}-1]]$ like $p(i) > 0$ and for all j that belongs to $[[0..i-1]]$ $p(j) = 0$.

Definition at line 120 of file splitse.hh.

```

121     {
122         oln_iter_type(E) dp(se);
123         E out;
124
125         for_all (dp)
126         {
127             unsigned n;
128             for (n = 0; n < E::dim; ++n)
129                 if (dp.cur().nth(n) > 0) {
130                     out.add(dp);
131                     break;
132                 } else if (dp.cur().nth(n) < 0) {
133                     break;
134                 }
135         }
136         return out;
137     }
```

6.22.2.15 `template<class E> mlc::exact< E >::ret get_minus_se_p (const abstract::struct_elt< E > & se)`

Get a sub part of a structuring element.

Parameters:

E Exact type of the structuring element.

- se The structuring element.

A point p take part of the new structuring element if it exists a i that belongs to $[[0..\text{dim}-1]]$ like $p(i) > 0$ and for all j that belongs to $[[0..i-1]]$ $p(j) = 0$ or if for all i that belongs to $[[0..\text{dim}-1]]$ $p(i) = 0$.

Definition at line 154 of file splitse.hh.

Referenced by `oln::morpho::hybrid::geodesic_reconstruction_dilation()`, `oln::morpho::sequential::geodesic_reconstruction_dilation()`, `oln::morpho::hybrid::geodesic_reconstruction_erosion()`, and `oln::morpho::sequential::geodesic_reconstruction_erosion()`.

```

155     {
156         oln_iter_type(E) dp(se);
157         E out;
158
159         for_all (dp)
160         {
161             unsigned n;
162             for (n = 0; n < E::dim; ++n)
163                 if (dp.cur().nth(n) > 0) {
164                     out.add(dp);
165                     break;
166                 } else if (dp.cur().nth(n) < 0) {
167                     break;
168                 }
169             // All p.nth(n) are 0.
```

```

170         if (n == E::dim)
171             out.add(dp);
172     }
173     return out;
174 }
```

6.22.2.16 `template<class E> mlc::exact< E >::ret get_plus_se_only (const abstract::struct_elt< E > & se)`

Get a sub part of a structuring element.

Parameters:

E Exact type of the structuring element.

- *se* The structuring element.

A point *p* take part of the new structuring element if it exists a *i* that belongs to $[[0..\text{dim}-1]]$ like $p(i) < 0$ and for all *j* that belongs to $[[0..i-1]]$ $p(j) = 0$.

Definition at line 50 of file splitse.hh.

```

51     {
52         oln_iter_type(E) dp(se);
53         E out;
54
55         for_all (dp)
56         {
57             unsigned n;
58             for (n = 0; n < E::dim; ++n)
59                 if (dp.cur().nth(n) < 0) {
60                     out.add(dp);
61                     break;
62                 } else if (dp.cur().nth(n) > 0) {
63                     break;
64                 }
65         }
66         return out;
67     }
```

6.22.2.17 `template<class E> mlc::exact< E >::ret get_plus_se_p (const abstract::struct_elt< E > & se)`

Get a sub part of a structuring element.

Parameters:

E Exact type of the structuring element.

- *se* The structuring element.

A point *p* take part of the new structuring element if it exists a *i* that belongs to $[[0..\text{dim}-1]]$ like $p(i) < 0$ and for all *j* that belongs to $[[0..i-1]]$ $p(j) = 0$ or if for all *i* that belongs to $[[0..\text{dim}-1]]$ $p(i) = 0$.

Definition at line 84 of file splitse.hh.

Referenced by `oln::morpho::hybrid::geodesic_reconstruction_dilation()`, `oln::morpho::sequential::geodesic_reconstruction_dilation()`, `oln::morpho::hybrid::geodesic_reconstruction_erosion()`, and `oln::morpho::sequential::geodesic_reconstruction_erosion()`.

```

85     {
86         oln_iter_type(E) dp(se);
87         E out;
88
89         for_all (dp)
90         {
91             unsigned n;
92             for (n = 0; n < E::dim; ++n)
93                 if (dp.cur().nth(n) < 0) {
94                     out.add(dp);
95                     break;
96                 } else if (dp.cur().nth(n) > 0) {
97                     break;
98                 }
99             // All p.nth(n) are 0.
100             if (n == E::dim)
101                 out.add(dp);
102         }
103         return out;
104     }

```

6.22.2.18 `template<class I, class E1, class E2> oln::mute< I >::ret hit_or_miss (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E1 > & se1, const abstract::struct_elt< E2 > & se2)`

Preform a 'hit or miss' transform.

Parameters:

- I* Exact type of the input image.
- E1* Exact type of the first structuring element.
- E2* Exact type of the second structuring element.

- *input* Image to process.
- *se1* First structuring element.
- *se2* Second structuring element.

```

#include <oln/basics2d.hh>
#include <oln/morpho/hit_or_miss.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>          im_type;

    im_type          im1(oln::load(IMG_IN "object.pbm"));

    oln::window2d mywin;
    mywin
        .add(-3,-2).add(-3,-1).add(-3,0).add(-3,1).add(-3,2)
        .add(-2,-1).add(-2,0).add(-2,1)
        .add(-1,0);
    oln::window2d mywin2 = - mywin;

    oln::save(oln::morpho::fast::hit_or_miss(im1, mywin, mywin2),
              IMG_OUT "oln_morpho_fast_hit_or_miss_overload.pbm");
}

```



=>



Definition at line 184 of file hit_or_miss.hh.

6.22.2.19 `template<class C, class B, class I, class E1, class E2> mute<I, typename convoutput<C, B,typename mlc::exact< I >::ret::value_type>::ret>::ret hit_or_miss (const convert::abstract::conversion< C, B > & c, const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E1 > & se1, const abstract::struct_elt< E2 > & se2)`

Perform a 'hit or miss' transform.

- c Conversion object.
- input Image to process.
- se1 First structuring element.
- se2 Second structuring element.

Compute the hit_or_miss transform of input by the composite structuring element (se1, se2). Soille p.131.

By definition se1 and se2 must have the same origin, and need to be disjoint. This algorithm has been extended to every data types (although it is not increasing). Beware the result depends upon the image data type if it is not bin.

```
#include <oln/basics2d.hh>
#include <oln/morpho/hit_or_miss.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>          im_type;

    im_type      im1(oln::load(IMG_IN "object.pbm"));

    oln::window2d mywin;
```

```

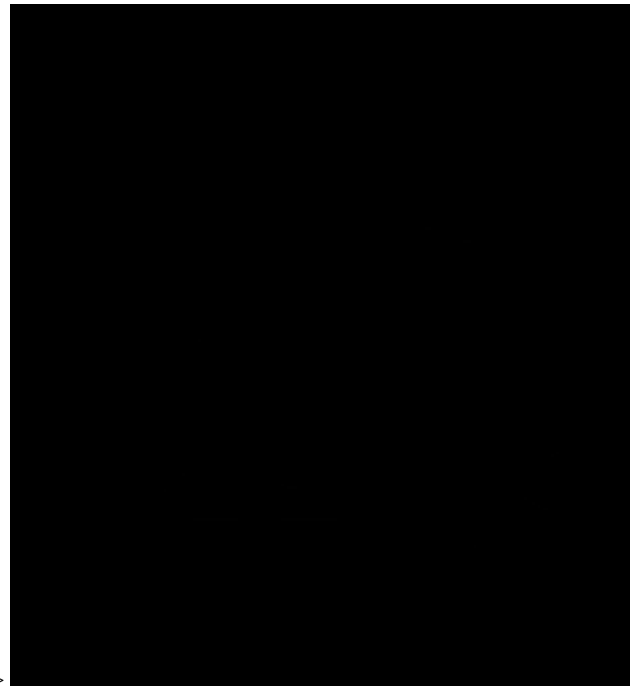
mywin
.add(-3,-2).add(-3,-1).add(-3,0).add(-3,1).add(-3,2)
.add(-2,-1).add(-2,0).add(-2,1)
.add(-1,0);
oln::window2d mywin2 = - mywin;

oln::save(oln::morpho::fast::hit_or_miss
(oln::convert::bound<ntg::int_u8>(), im1, mywin, mywin2),
IMG_OUT "oln_morpho_fast_hit_or_miss.pbm");
}

```



=>



Definition at line 122 of file hit_or_miss.hh.

6.22.2.20 `template<class I, class E1, class E2> oln::mute< I >::ret hit_or_miss_closing (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E1 > &se1, const abstract::struct_elt< E2 > &se2)`

Perform an hit or miss closing.

Compute the hit_or_miss closing of input by the composite structuring element (se1, se2). This is the dual transformation of hit-or-miss opening with respect to set complementation. Soille p.135.

By definition se1 and se2 must have the same origin, and need to be disjoint. This algorithm has been extended to every data types (althought it is not increasing). Beware the result depends upon the image data type if it is not bin.

Parameters:

- I* Exact type of the input image.
- E1* Exact type of the first structuring element.
- E2* Exact type of the second structuring element.

- input Image to process.

- se1 First structuring element.
- se2 Second structuring element.

```
#include <oln/basics2d.hh>
#include <oln/morpho/hit_or_miss.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>          im_type;

    im_type          im1(oln::load(IMG_IN "object.pbm"));

    oln::window2d mywin;
    mywin
        .add(-3,-2).add(-3,-1).add(-3,0).add(-3,1).add(-3,2)
        .add(-2,-1).add(-2,0).add(-2,1)
        .add(-1,0);
    oln::window2d mywin2 = - mywin;

    oln::save(oln::morpho::fast::hit_or_miss_closing(im1, mywin, mywin2),
              IMG_OUT "oln_morpho_fast_hit_or_miss_closing.pbm");
}
```



=>



Definition at line 374 of file hit_or_miss.hh.

6.22.2.21 `template<class I, class E1, class E2> oln::mute< I >::ret hit_or_miss_closing_bg (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E1 > &se1, const abstract::struct_elt< E2 > &se2)`

Perform an hit or miss closing of background.

Compute the hit_or_miss closing of the background of input by the composite structuring element (se1, se2). This is the dual transformation of hit-or-miss opening with respect to set complementation. Soille p.135.

By definition se1 and se2 must have the same origin, and need to be disjoint. This algorithm has been extended to every data types (although it is not increasing). Beware the result depends upon the image data type if it is not bin.

Parameters:

- I* Exact type of the input image.
- E1* Exact type of the first structuring element.
- E2* Exact type of the second structuring element.

- input Image to process.
- se1 First structuring element.
- se2 Second structuring element.

```
#include <oln/basics2d.hh>
#include <oln/morpho/hit_or_miss.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>          im_type;

    im_type      im1(oln::load(IMG_IN "object.pbm"));

    oln::window2d mywin;
    mywin
        .add(-3,-2).add(-3,-1).add(-3,0).add(-3,1).add(-3,2)
        .add(-2,-1).add(-2,0).add(-2,1)
        .add(-1,0);
    oln::window2d mywin2 = - mywin;

    oln::save(oln::morpho::fast::hit_or_miss_closing_bg(im1, mywin, mywin2),
              IMG_OUT "oln_morpho_fast_hit_or_miss_closing_bg.pbm");
}
```



=>



Definition at line 437 of file hit_or_miss.hh.

6.22.2.22 `template<class I, class E1, class E2> oln::mute< I >::ret hit_or_miss_opening (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E1 > & se1, const abstract::struct_elt< E2 > & se2)`

Perform an hit or miss opening.

Compute the hit_or_miss opening of input by the composite structuring element (se1, se2). Soille p.134.

By definition se1 and se2 must have the same origin, and need to be disjoint. This algorithm has been extended to every data types (although it is not increasing). Beware the result depends upon the image data type if it is not bin.

Parameters:

I Exact type of the input image.

E1 Exact type of the first structuring element.

E2 Exact type of the second structuring element.

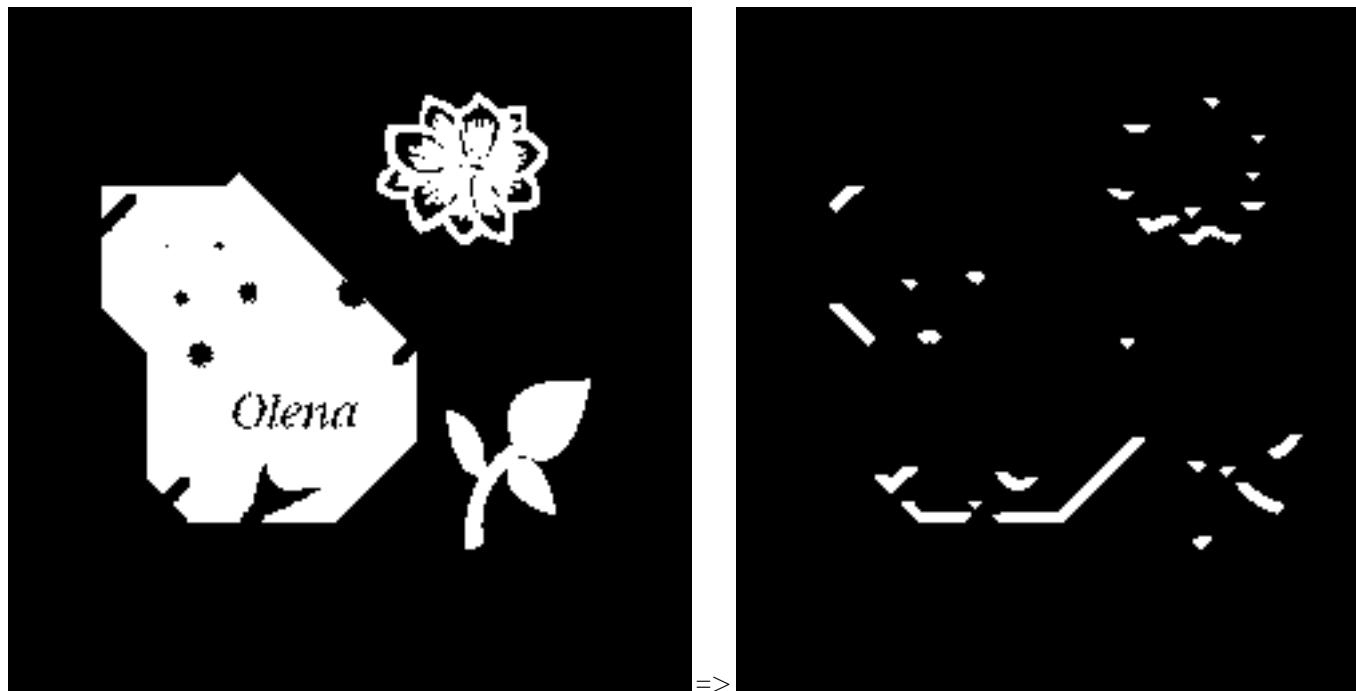
- input Image to process.
- se1 First structuring element.
- se2 Second structuring element.

```
#include <oln/basics2d.hh>
#include <oln/morpho/hit_or_miss.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>          im_type;

    im_type          im1(oln::load(IMG_IN "object.pbm"));

    oln::window2d mywin;
    mywin
        .add(-3,-2).add(-3,-1).add(-3,0).add(-3,1).add(-3,2)
        .add(-2,-1).add(-2,0).add(-2,1)
        .add(-1,0);
    oln::window2d mywin2 = - mywin;

    oln::save(oln::morpho::fast::hit_or_miss_opening(im1, mywin, mywin2),
              IMG_OUT "oln_morpho_fast_hit_or_miss_opening.pbm");
}
```



Definition at line 248 of file hit_or_miss.hh.

6.22.2.23 `template<class I, class E1, class E2> oln::mute< I >::ret hit_or_miss_opening_bg
(const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E1 > &
se1, const abstract::struct_elt< E2 > & se2)`

Perform an hit or miss opening of background.

Compute the hit_or_miss opening of the background of input by the composite structuring element (*se1*, *se2*). Soille p.135.

By definition *se1* and *se2* must have the same origin, and need to be disjoint. This algorithm has been extended to every data types (although it is not increasing). Beware the result depends upon the image data type if it is not bin.

Parameters:

- I* Exact type of the input image.
- E1* Exact type of the first structuring element.
- E2* Exact type of the second structuring element.

- input Image to process.
- *se1* First structuring element.
- *se2* Second structuring element.

```
#include <oln/basics2d.hh>
#include <oln/morpho/hit_or_miss.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>          im_type;
```

```

im_type      im1(oln::load(IMG_IN "object.pbm"));

oln::window2d mywin;
mywin
    .add(-3,-2).add(-3,-1).add(-3,0).add(-3,1).add(-3,2)
    .add(-2,-1).add(-2,0).add(-2,1)
    .add(-1,0);
oln::window2d mywin2 = - mywin;

oln::save(oln::morpho::fast::hit_or_miss_opening_bg(im1, mywin, mywin2),
          IMG_OUT "oln_morpho_fast_hit_or_miss_opening_bg.pbm");
}

```



=>



Definition at line 308 of file hit_or_miss.hh.

6.22.2.24 `template<class I, class N> mute<I, ntg::bin>::ret internal_kill_cc_area (const abstract::binary_image_with_dim< 2, I > &input, const unsigned int area, const abstract::neighborhood< N > &Ng)`

Kill connex components smaller than a given area.

Parameters:

I Exact type of the input image.
N Exact type of the neighborhood.

- input The input image.
- area The threshold to use.
- Ng The neighborhood to use.

Definition at line 63 of file extrema_killer.hh.

References `oln::level::hlut_def< T, T2 >::set()`.

Referenced by `sure_maxima_killer()`, and `sure_minima_killer()`.

```

66     {
67         typename mute<I, ntg::int_u<20> >::ret cc = level::connected_component<ntg::int_u<20> >(input, Ng);
68         // label 0 is background
69         ntg::int_u<20> max_label = 0;
70         image2d<ntg::int_u<20> >::iter_type p(cc);
71         for_all(p)
72             if (cc[p] > max_label)
73                 max_label = cc[p];
74         level::hlut_def<ntg::int_u<20> > region_area;
75         for_all(p)
76             region_area.set(cc[p], ntg::cast::force<ntg::int_u<20> >(region_area(cc[p]) + 1));
77         image2d<ntg::bin> output(input.size());
78         for_all(p)
79             if (input[p] == true)
80             {
81                 if (region_area(cc[p]) >= ntg::int_u<20> (area))
82                     output[p] = true;
83                 else
84                     output[p] = false;
85             }
86         else
87             output[p] = false;
88         return output;
89     }

```

6.22.2.25 `template<class P, class I, class N> bool is_a_strict_maximum (const abstract::point< P > & p, const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & Ng) [static]`

Check if a point is a strict maximum.

Parameters:

P Exact type of the point.

I Exact type of the image.

N Exact type of the neighborhood.

- p The point to consider.
- input The image where to get the value of the point to consider.
- Ng Type of neighborhood to use.

Definition at line 295 of file `extrema_killer.hh`.

References `oln::abstract::image< Exact >::hold()`.

Referenced by `fast_maxima_killer()`.

```

298     {
299         mlc::eq<I::dim, N::dim>::ensure();
300         mlc::eq<P::dim, N::dim>::ensure();
301
302         bool is_p_upper = true;
303         bool is_p_at_least_one_stricly_upper = false;
304         oln_neighb_type(N) p_prime(Ng, p);

```

```

305     for_all(p_prime) if (input.hold(p_prime))
306     {
307         if (input[p] > input[p_prime])
308             is_p_at_least_one_stricly_upper = true;
309         if (input[p] < input[p_prime])
310             is_p_upper = false;
311     }
312     return (is_p_upper && is_p_at_least_one_stricly_upper);
313 }

```

6.22.2.26 `template<class P, class I, class N> bool is_a_strict_minimum (const abstract::point< P > & p, const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & Ng) [static]`

Check if a point is a strict minimum.

Parameters:

P Exact type of the point.

I Exact type of the image.

N Exact type of the neighborhood.

- p The point to consider.
- input The image where to get the value of the point to consider.
- Ng Type of neighborhood to use.

Definition at line 260 of file extrema_killer.hh.

References oln::abstract::image< Exact >::hold().

Referenced by fast_minima_killer().

```

263     {
264         mlc::eq<I::dim, N::dim>::ensure();
265         mlc::eq<P::dim, N::dim>::ensure();
266
267         bool is_p_lower = true;
268         bool is_p_at_least_one_stricly_lower = false;
269         oln_neighb_type(N) p_prime(Ng, p);
270         for_all(p_prime) if (input.hold(p_prime))
271         {
272             if (input[p] < input[p_prime])
273                 is_p_at_least_one_stricly_lower = true;
274             if (input[p] > input[p_prime])
275                 is_p_lower = false;
276         }
277         return (is_p_lower && is_p_at_least_one_stricly_lower);
278     }

```

6.22.2.27 `template<class DestValue, class I, class E> mute<I, DestValue>::ret laplacian (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Compute the laplacian of an image.

Compute the laplacian of input using se as structural element.

- input Image to process.
- se Structuring element.

```
#include <oln/basics2d.hh>
#include <oln/morpho/laplacian.hh>
#include <oln/convert/stretch.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::image2d<ntg::int_s<10> > i10 =
        oln::morpho::fast::laplacian(oln::convert::bound<ntg::int_s<10> >(),
                                     im1,
                                     oln::win_c8p());

    oln::image2d<ntg::int_s<10> > f10 =
        oln::morpho::fast::laplacian<ntg::int_s<10> >(im1,
                                                         oln::win_c8p());

    oln::save(apply(oln::convert::stretch<ntg::int_u8>(),(f10)),
              IMG_OUT "oln_morpho_fast_laplacian_overload.pgm");
}
```



=>

Todo

FIXME: Not instantiated in swilena (see tools/swilena/generate_morpho_instantiations.py)

Definition at line 158 of file laplacian.hh.

6.22.2.28 `template<class C, class B, class I, class E> mute<I, typename convoutput<C, B,typename mlc::exact< I >::ret::value_type>::ret>::ret laplacian (const convert::abstract::conversion< C, B > & c, const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Compute the laplacian of an image.

Compute the laplacian of input using se as structural element.

- c Conversion object.
- input Image to process.
- se Structuring element.

```
#include <oln/basics2d.hh>
#include <oln/morpho/laplacian.hh>
#include <oln/convert/stretch.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::image2d<ntg::int_s<10> > i10 =
        oln::morpho::fast::laplacian(oln::convert::bound<ntg::int_s<10> >(),
                                    im1,
                                    oln::win_c8p());

    oln::save(apply(oln::convert::stretch<ntg::int_u8>(), i10),
              IMG_OUT "oln_morpho_fast_laplacian.pgm");
}
```



=>



Definition at line 105 of file laplacian.hh.

6.22.2.29 `template<class I, class E> mlc::exact< I >::ret::value_type max (const abstract::non_vectorial_image< I > & input, const typename mlc::exact< I >::ret::point_type & p, const abstract::struct_elt< E > & se)`

Maximum of a structuring element.

Look for the maximum in the structuring element *se* disposed on the image input, at the point *p*.

Parameters:

I Image exact type.

E Structuring element type.

- input Input image.
- p Point of the image to move the structuring element on.
- se The structuring element to use.

Definition at line 148 of file morpho/stat.hh.

```

151     {
152         mlc::eq<I::dim, E::dim>::ensure();
153         return internal::stat_<I, E>::max(input.exact(), p, se.exact());
154     }
```

6.22.2.30 `template<class I, class E> mlc::exact< I >::ret::value_type min (const abstract::non_vectorial_image< I > & input, const typename mlc::exact< I >::ret::point_type & p, const abstract::struct_elt< E > & se)`

Minimum of a structuring element.

Look for the minimum in the structuring element *se* disposed on the image input, at the point *p*.

Parameters:

I Image exact type.

E Structuring element type.

- input Input image.
- p Point of the image to move the structuring element on.
- se The structuring element to use.

Definition at line 170 of file morpho/stat.hh.

```

174     {
175         mlc::eq<I::dim, E::dim>::ensure();
176         return internal::stat_<I, E>::min(input.exact(), p, se.exact());
177     }
```

6.22.2.31 `template<class I, class E> oln::mute< I >::ret n_dilation (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se, unsigned n)`

Perform morphological dilation iterated n times.

Parameters:

- I* Exact type of the input image.
- E* Exact type of the structuring element.

- *input* Input image.
- *se* Structuring element to use.
- *n* Number of iterations.

```
#include <oln/basics2d.hh>
#include <oln/morpho/dilation.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>      im_type;

    im_type im1(oln::load(IMG_IN "object.pbm"));
    save(oln::morpho::n_dilation(im1, oln::win_c8p(), 5),
        IMG_OUT "oln_morpho_n_dilation.pbm");
}
```



=>



Definition at line 141 of file dilation.hh.

References `oln::abstract::image< Exact >::clone()`, and `dilation()`.

144 {

```

145     precondition(n > 0);
146     oln_concrete_type(I) output = input.clone();
147     for (unsigned i = 0; i < n; ++i)
148     {
149         oln_concrete_type(I) work = dilation(output, se);
150         output = work;
151     }
152     return output;
153 }

```

6.22.2.32 `template<class I, class E> oln::mute< I >::ret n_erosion (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se, unsigned n)`

Perform morphological erosion iterated *n* times.

Parameters:

- I* Exact type of the input image.
- E* Exact type of the structuring element.
- *input* Input image.
- *se* Structuring element to use.
- *n* Number of iterations.

```

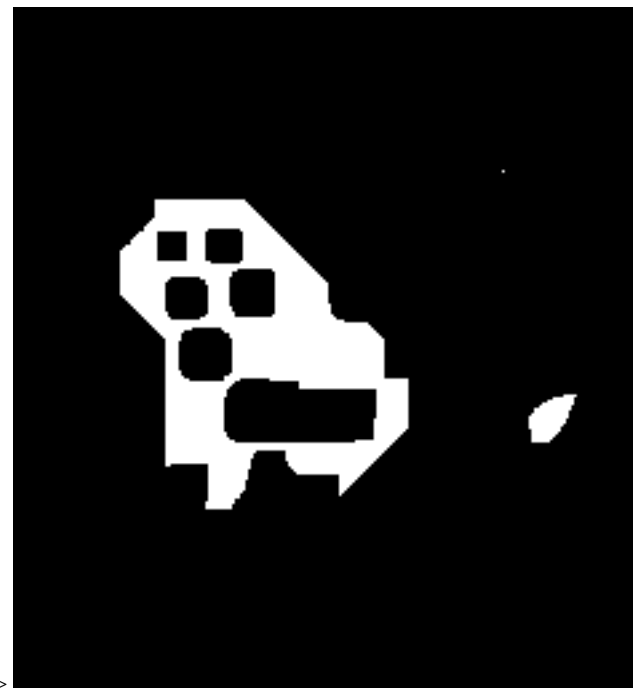
#include <oln/basics2d.hh>
#include <oln/morpho/erosion.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>      im_type;

    im_type im1(oln::load(IMG_IN "object.pbm"));
    save(oln::morpho::n_erosion(im1, oln::win_c8p(), 5),
        IMG_OUT "oln_morpho_n_erosion.pbm");
}

```



=>



Definition at line 134 of file erosion.hh.

References `oln::abstract::image< Exact >::clone()`, and `erosion()`.

```

137     {
138         //mlc::eq<I::dim, E::dim>::ensure();
139         precondition(n > 0);
140         oln_concrete_type(I) output = input.clone();
141         for (unsigned i = 0; i < n; ++i)
142             {
143                 oln_concrete_type(I) work = erosion(output, se);
144                 output = work;
145             }
146         return output;
147     }

```

6.22.2.33 `template<class I, class E> oln::mute< I >::ret opening (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Perform a morphological opening.

Compute the morphological opening of input using se as structuring element.

Parameters:

- I* Exact type of the input image.
- E* Exact type of the structuring element.

- input Image to process.
- se Structuring element.

```

#include <oln/basics2d.hh>
#include <oln/morpho/opening.hh>

```

```
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>      im_type;

    im_type      im(oln::load(IMG_IN "object.pbm"));

    oln::save(oln::morpho::fast::opening(im, oln::win_c8p()),
              IMG_OUT "oln_morpho_fast_opening.pbm");
}
```



=>



Definition at line 102 of file opening.hh.

6.22.2.34 `template<class I, class E> oln::mute< I >::ret self_complementary_top_hat (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Compute the self complementary top hat of an image.

- c Conversion object.
- input Image to process.
- se Structuring element.

```
#include <oln/basics2d.hh>
#include <oln/morpho/top_hat.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>      im_type;

    im_type      im1(oln::load(IMG_IN "lena256.pgm"));
```

```

    oln::save(oln::morpho::fast::self_complementary_top_hat(im1,
                                                            oln::win_c8p()),
              IMG_OUT "oln_morpho_fast_self_complementary_top_hat_overload.pbm");
}

```



=>



Definition at line 323 of file top_hat.hh.

6.22.2.35 `template<class C, class B, class I, class E> mute<I, typename convoutput<C, B,typename mlc::exact< I >::ret::value_type>::ret>::ret self_complementary_top_hat(const convert::abstract::conversion< C, B > &c, const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Compute the self complementary top hat of an image.

Compute self complementary top hat of input using se as structuring element. Soille p.106.

- c Conversion object.
- input Image to process.
- se Structuring element.

```

#include <oln/basics2d.hh>
#include <oln/morpho/top_hat.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::morpho::fast::self_complementary_top_hat
              (oln::convert::bound<ntg::int_u8>(),
               im1,

```

```

        oln::win_c8p()),
        IMG_OUT "oln_morpho_fast_self_complementary_top_hat.pbm");
    }

```



=>



Definition at line 283 of file top_hat.hh.

6.22.2.36 `template<class I, class N> oln::mute< I >::ret sure_maxima_killer (const abstract::non_vectorial_image< I > & input, const unsigned int area, const abstract::neighborhood< N > & Ng)`

Maxima killer.

It removes the small (in area) connected components of the upper level sets of input using se as structuring element. The implementation uses the threshold superposition principle; so it is very slow ! it works only for int_u8 images.

Parameters:

I Image exact type.

N Neighborhood exact type.

- *input* The input image.
- *area* Threshold to use.
- *Ng* The neighborhood to use.

```

#include <oln/basics2d.hh>
#include <oln/morpho/extrema_killer.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

```



```

im_type          im(oln::load(IMG_IN "lena128.pgm"));

oln::save(oln::morpho::sure_maxima_killer(im, 200,  oln::neighb_c4()),
          IMG_OUT "oln_morpho_sure_maxima_killer.pgm");
return 0;
}

```



Definition at line 132 of file extrema_killer.hh.

References `internal_kill_cc_area()`, and `oln::abstract::image< Exact >::size()`.

```

135     {
136         mlc::eq<I::dim, N::dim>::ensure();
137         typedef typename mute<I, ntg::bin >::ret ima_bin_type;
138
139         ima_bin_type* cc_level_sets = new (image2d<ntg::bin> [256]);
140         for (unsigned int i=0; i <= 255; ++i)
141             {
142                 image2d<ntg::bin> level_sets_i(input.size());
143                 image2d<ntg::int_u8>::iter_type p(input);
144                 for_all(p)
145                     if (input[p] >= oln_value_type(I) (i))
146                         level_sets_i[p] = true;
147                     else
148                         level_sets_i[p] = false;
149                 cc_level_sets[i] = internal_kill_cc_area(level_sets_i, area, Ng);
150             }
151         image2d<ntg::int_u8> output(input.size());
152         for (int i=0; i < 255 ; ++i)
153             {
154                 image2d<ntg::int_u8>::iter_type p(input);
155                 for_all(p)
156                     {
157                         if ((cc_level_sets[i])[p] == true)
158                             output[p] = i;
159                     }
160             }
161         delete[] cc_level_sets;
162
163         return output;
164     }

```

6.22.2.37 `template<class I, class N> image2d<ntg::int_u8> sure_minima_killer (const abstract::non_vectorial_image< I > & input, const unsigned int area, const abstract::neighborhood< N > & Ng)`

Minima killer.

It removes the small (in area) connected components of the lower level sets of input using se as structuring element. The implementation uses the threshold superposition principle; so it is very slow ! it works only for int_u8 images.

Parameters:

I Image exact type.

N Neighborhood exact type.

- input The input image.
- area Threshold to use.
- Ng The neighborhood to use.

```
#include <oln/basics2d.hh>
#include <oln/morpho/extrema_killer.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type                                im(oln::load(IMG_IN "lena128.pgm"));

    oln::save(oln::morpho::sure_minima_killer(im, 200, oln::neighb_c4()),
              IMG_OUT "oln_morpho_sure_minima_killer.pgm");
    return 0;
}
```



Definition at line 205 of file extrema_killer.hh.

References internal_kill_cc_area(), and oln::abstract::image< Exact >::size().

```
208     {
209         mlc::eq<I::dim, N::dim>::ensure();
210
211         typedef image2d<ntg::bin> ima_bin_type;
212
213         ima_bin_type* cc_level_sets = new (image2d<ntg::bin> [256]);
214         for (unsigned int i=0; i <= 255; ++i)
215             {
216                 image2d<ntg::bin> level_sets_i(input.size());
217                 image2d<ntg::int_u8>::iter_type p(input);
218                 for_all(p)
219                     if (input[p] <= oln_value_type(I) (i))
220                         level_sets_i[p] = true;
221                     else
222                         level_sets_i[p] = false;
223                 cc_level_sets[i] = internal_kill_cc_area(level_sets_i, area, Ng);
224             }
```

```

224     }
225     image2d<ntg::int_u8> output(input.size());
226     for (int i=255; i >= 0 ; --i)
227     {
228         image2d<ntg::int_u8>::iter_type p(input);
229         for_all(p)
230         {
231             if ((cc_level_sets[i])[p] == true)
232                 output[p] = i;
233         }
234     }
235     delete[] cc_level_sets;
236
237     return output;
238 }

```

6.22.2.38 `template<class I, class E1, class E2> oln::mute< I >::ret thickening (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E1 > & se1, const abstract::struct_elt< E2 > & se2)`

Thicken an image.

Parameters:

I Exact type of the image.

E1 Exact type of the first structuring element.

E2 Exact type of the second structuring element.

- *input* Image to process.
- *se1* First structuring element.
- *se2* Second structuring element.

```

#include <oln/basics2d.hh>
#include <oln/morpho/thickening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::morpho::fast::thickening(im1,
                                             oln::win_c8p(),
                                             oln::win_c8p(),
                                             IMG_OUT "oln_morpho_fast_thickening.pbm"));

    return 0;
}

```



=>



Definition at line 106 of file thickening.hh.

6.22.2.39 `template<class I, class E1, class E2> oln::mute< I >::ret thinning (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E1 > & se1, const abstract::struct_elt< E2 > & se2)`

Thin an image.

Parameters:

- I* Exact type of the image.
- E1* Exact type of the first structuring element.
- E2* Exact type of the second structuring element.

- input Image to process.
- se1 First structuring element.
- se2 Second structuring element.

```
#include <oln/basics2d.hh>
#include <oln/morpho/thinning.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::morpho::fast::thinning(im1,
                                         oln::win_c8p(),
                                         oln::win_c8p(),
                                         IMG_OUT "oln_morpho_fast_thinning.pbm"));

    return 0;
}
```



=>



Definition at line 106 of file thinning.hh.

6.22.2.40 `template<class I, class E> oln::mute< I >::ret top_hat_contrast_op (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Top hat contrast operator.

Enhance contrast input by adding the white top hat, then subtracting the black top hat to input. Top hats are computed using se as structuring element. Soille p.109.

- input Image to process.
- se Structuring element.

```
#include <oln/basics2d.hh>
#include <oln/morpho/top_hat.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::morpho::fast::top_hat_contrast_op(im1,
                                                    oln::win_c8p()),
              IMG_OUT "oln_morpho_fast_top_hat_contrast_op_overload.pbm");
}
```



=>



Definition at line 418 of file top_hat.hh.

6.22.2.41 `template<class C, class B, class I, class E> mute<I, typename convoutput<C, B,typename mlc::exact< I >::ret::value_type>::ret>::ret top_hat_contrast_op (const convert::abstract::conversion< C, B > &c, const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Top hat contrast operator.

Enhance contrast input by adding the white top hat, then subtracting the black top hat to input. Top hats are computed using se as structuring element. Soille p.109.

- c Conversion object.
- input Image to process.
- se Structuring element.

```
#include <oln/basics2d.hh>
#include <oln/morpho/top_hat.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::morpho::fast::top_hat_contrast_op
              (oln::convert::bound<ntg::int_u8>(),
               im1,
               oln::win_c8p()),
              IMG_OUT "oln_morpho_fast_top_hat_contrast_op.pbm");
}
```



=>



Definition at line 371 of file top_hat.hh.

6.22.2.42 `template<class DestValue, class I, class N> mute< I, DestValue >::ret
 oln::morpho::watershed_con (const abstract::non_vectorial_image< I > & im_i, const
 abstract::neighborhood< N > & Ng)`

Connected watershed.

Compute the connected watershed for image *im* using neighborhood *ng*.

`watershed_con` creates an output image whose values have type `DestValue` (which should be discrete). In this output all basins are labeled using values from `DestValue::min()` to `DestValue::max() - 4` (the remaining values are used internally by the algorithm).

When there are more basins than `DestValue` can hold, wrapping occurs (i.e., the same label is used for several basin). This is potentially harmful, because if two connected basins are labeled with the same value they will appear as one basin.

This is based on the original algorithm presented by Vincent and Soille, but modified to not output watersheds.

Parameters:

DestValue Type of the data in output image.

I Exact type of the image.

N Exact type of the neighborhood.

- input Image of levels.
- Ng Neighborhood to consider.

Precondition:

`DestValue` should be large enough.

```

#include <oln/basics2d.hh>
#include <oln/morpho/watershed.hh>
#include <ntg/all.hh>

#include <oln/morpho/attribute_closing_opening.hh>
#include <oln/morpho/gradient.hh>
#include <oln/convert/stretch.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena256.pgm"));

    // Gradient of the image
    im1 = oln::morpho::beucher_gradient(im1, oln::win_c8p());

    // Remove local minima smaller than 200 pixels
    im1 = oln::morpho::fast::card_closing(im1, oln::neighb_c8(),
                                          200);

    oln::save(im1, IMG_OUT "oln_morpho_watershed_con_tmp.pgm");

    // Use the watershed to comment the image
    im_type w = oln::morpho::watershed_con<ntg::int_u8>(im1,
                                                         oln::neighb_c8());

    oln::save(oln::convert::stretch_balance<ntg::int_u8>(w),
              IMG_OUT "oln_morpho_watershed_con.pgm");
}

```



=>





Definition at line 286 of file watershed.hxx.

```

287     {
288         return internal::soille_watershed_<
289             internal::watershed_con_point_handler_, DestValue> (im_i, Ng);
290     }
291 
```

6.22.2.43 `template<class DestValue, class I, class N> mute< I, DestValue >::ret
oln::morpho::watershed_seg (const abstract::non_vectorial_image< I > & im_i, const
abstract::neighborhood< N > & Ng)`

Segmented watershed.

Compute the segmented watershed for image *im* using neighborhood *ng*.

`watershed_seg` creates an output image whose values have type `DestValue` (which should be discrete). In this output image, `DestValue::max()` indicates a watershed, and all basins are labeled using values from `DestValue::min()` to `DestValue::max() - 4` (the remaining values are used internally by the algorithm).

When there are more basins than `DestValue` can hold, wrapping occurs (i.e., the same label is used for several basin).

This is based on the original algorithm presented by Vincent and Soille. (FIXME: ref?)

Parameters:

DestValue Type of the data in output image.

I Exact type of the image.

N Exact type of the neighborhood.

- *im_i* Image of levels.
- *Ng* Neighborhood to consider.

Precondition:

DestValue should be large enough.

```
#include <oln/basics2d.hh>
#include <oln/morpho/watershed.hh>
#include <ntg/all.hh>

#include <oln/morpho/attribute_closing_opening.hh>
#include <oln/morpho/gradient.hh>
#include <oln/convert/stretch.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena256.pgm"));

    // Gradient of the image
    im1 = oln::morpho::beucher_gradient(im1, oln::win_c8p());

    // Remove local minima smaller than 200 pixels
    im1 = oln::morpho::fast::card_closing(im1, oln::neighb_c8(),
                                          200);

    oln::save(im1, IMG_OUT "oln_morpho_watershed_seg_tmp.pgm");

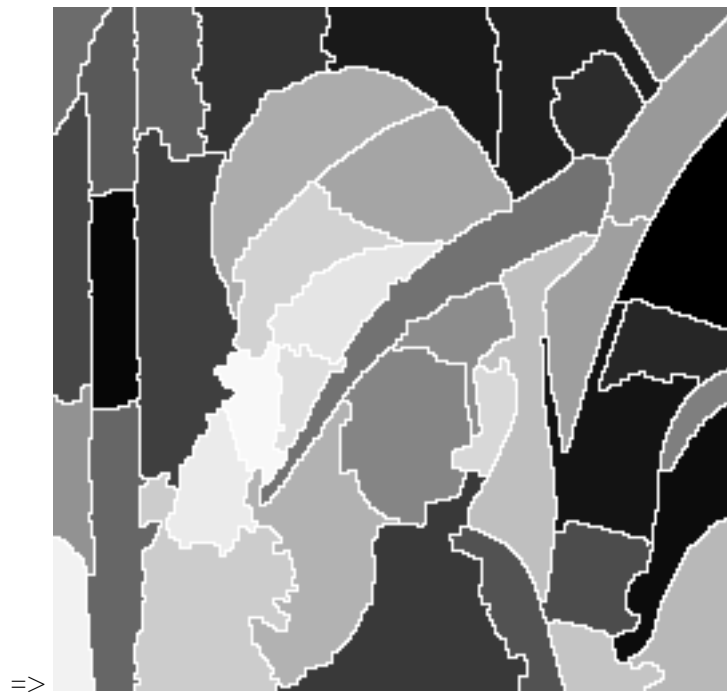
    // Use the watershed to segment the image
    im_type w = oln::morpho::watershed_seg<ntg::int_u8>(im1,
                                                         oln::neighb_c8());

    oln::save(oln::convert::stretch_balance<ntg::int_u8>(w),
              IMG_OUT "oln_morpho_watershed_seg.pgm");
}
```



=>





=> Definition at line 277 of file watershed.hxx.

```

278     {
279         return internal::soille_watershed_<
280             internal::watershed_seg_point_handler_, DestValue> (im_i, Ng);
281     }
282 
```

6.22.2.44 `template<class I1, class I2, class N> oln::mute< I2 >::ret & oln::morpho::watershed_seg_or (const abstract::non_vectorial_image< I1 > & D, abstract::non_vectorial_image< I2 > & M, const abstract::neighborhood< N > & Ng) [inline]`

Segmented watershed with user-supplied starting points.

Compute a segmented watershed for image levels using neighborhood ng, and markers as starting point for the flooding algorithm.

markers is an image of the same size as levels and containing discrete values indicating label associated to each basin. On input, fill markers with `oln_value_type(I2)min()` (this is the unknown label) and mark the starting points or regions (usually these are minima in levels) using a value between `oln_value_type(I2)min() + 1` and `oln_value_type(I2)max() - 1`.

`watershed_seg_or` will flood levels from these non-unknown starting points, labeling basins using the value you assigned to them, and marking watershed lines with `oln_value_type(I2)max()`. markers should not contains any `oln_value_type(I2)min()` value on output.

This is based on the original algorithm presented by D'Ornellas et al. (FIXME: ref?)

Parameters:

I1 Exact type of the D image.

I2 Exact type of the M image.

N Exact type of the neighborhood.

- D Input image.
- M Image of labels.
- Ng Neighborhood to consider.

Todo

FIXME: Not instantiated in swilena (see tools/swilena/generate_morpho_instantiations.py)

Definition at line 314 of file watershed.hxx.

References `oln::abstract::image< Exact >::hold()`.

```

316     {
317
318         typedef oln_value_type(I2) value_type;
319         const oln_value_type(I2) wshed = ntg_max_val(value_type);
320         const oln_value_type(I2) unknown = ntg_min_val(value_type);
321
322         typedef std::pair<oln_point_type(I1), oln_value_type(I1)> queue_elt_type;
323         std::priority_queue< queue_elt_type, std::vector< queue_elt_type >,
324             cmp_queue_elt<I1> > PQ;
325
326         // Initialization
327         // Enqueue all labeled points which have a unlabeled neighbor.
328         {
329             oln_iter_type(I2) p(Labels);
330             oln_neighb_type(N) p_prime(Ng, p);
331             for_all(p)
332                 if (Labels[p] != unknown)
333                     for_all (p_prime)
334                         if (Labels.hold(p_prime) && Labels[p_prime] == unknown)
335                             {
336                                 PQ.push(queue_elt_type(p, In[p]));
337                                 break;
338                             }
339         }
340
341         // Propagation
342
343         oln_point_type(I1) p;
344         oln_neighb_type(N) p_prime(Ng, p);
345         oln_neighb_type(N) p_second(Ng, p_prime);
346         while (! PQ.empty())
347         {
348             // Get the lowest point in the queue.
349             p = PQ.top().first;
350             PQ.pop();
351             // Consider all neighbors of p.
352             for_all (p_prime) if (Labels.hold(p_prime))
353                 {
354                     // Focus on unlabeled neighbors of p.
355                     if (Labels[p_prime] == unknown)
356                     {
357                         // Since p_prime is a neighbor of p, it should
358                         // be either labeled using the same label as p,
359                         // or marked as watershed.
360
361                         // It's a watershed if, among the neighbors of
362                         // p_prime (which itself is a neighbor of p), there
363                         // exists a point with a label different from the
364                         // label of pt. EXISTS is set to true in this case.
365                         bool exists = false;
366                         for_all (p_second)
367                             // FIXME: We should not need the iterate over

```

```

368         // the neighbors of p_prime which are also
369         // neighbors of p, since none of these can have
370         // have a different label. It should be possible
371         // to precompute an array of the resulting windows
372         // for each neighbor (of p).
373         if (Labels.hold(p_second)
374             && Labels[p_second] != unknown
375             && Labels[p_second] != wshed
376             && Labels[p_second] != Labels[p])
377         {
378             exists = true;
379             break;
380         }
381         if (exists)
382             Labels[p_prime] = wshed;
383         else
384         {
385             Labels[p_prime] = Labels[p];
386             PQ.push(queue_elt_type(p_prime, In[p_prime]));
387         }
388     }
389 }
390 }
391 return Labels.exact();
392 }
393

```

6.22.2.45 `template<class I, class E> oln::mute< I >::ret white_top_hat (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Compute the white top hat of an image.

Parameters:

I Exact type of the image.

E Exact type of the structuring element.

- input Image to process.
- se Structuring element.

```

#include <oln/basics2d.hh>
#include <oln/morpho/top_hat.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::morpho::fast::white_top_hat(im1,
                                                oln::win_c8p()),
              IMG_OUT "oln_morpho_fast_white_top_hat_overload.pbm");
}

```



=>



Definition at line 145 of file top_hat.hh.

6.22.2.46 `template<class C, class B, class I, class E> mute<I, typename convoutput<C, B,typename mlc::exact< I >::ret::value_type>::ret>::ret white_top_hat (const convert::abstract::conversion< C, B > & c, const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Compute the white top hat of an image.

Compute white top hat of input using se as structuring element. Soille p.105.

- c Conversion object.
- input Image to process.
- se Structuring element.

```
#include <oln/basics2d.hh>
#include <oln/morpho/top_hat.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::morpho::fast::white_top_hat
              (oln::convert::bound<ntg::int_u8>(),
               im1,
               oln::win_c8p()),
              IMG_OUT "oln_morpho_fast_white_top_hat.pbm");
}
```



=>



Definition at line 104 of file top_hat.hh.

6.23 oln::morpho::attr Namespace Reference

Implementation of attributes.

Classes

- class [attribute](#)
Attribute abstract class.
- class [card_type](#)
Cardinal attribute.
- class **card_full_type**
- class [integral_type](#)
Integral attribute.
- class [other_image](#)
Metaclass used to change attribute behavior.
- class **ball_parent_change**
- class [height_type](#)
Attribute working on height between components.
- class [maxvalue_type](#)
Max value attribute.
- class [minvalue_type](#)
Min value attribute.
- class [ball_type](#)
Ball attribute.
- class [dist_type](#)
Dist attribute.
- class [cube_type](#)
Cube attribute.
- class [box_type](#)
Box attribute.
- struct [attr_traits< integral_type< T, Exact > >](#)
Trait specialization for the integral attribute.
- struct [attr_traits< height_type< T, Exact > >](#)
Trait specialization for the height attribute.
- struct [attr_traits< card_type< T, Exact > >](#)
Trait specialization for card attribute.

- struct `attr_traits< card_full_type< I, T, Exact > >`
Trait specialization for card_full attribute.
- struct `attr_traits< maxvalue_type< T, Exact > >`
Trait specialization for the maxvalue attribute.
- struct `attr_traits< minvalue_type< T, Exact > >`
Trait specialization for the minvalue attribute.
- struct `attr_traits< ball_type< I, Exact > >`
Trait specialization for the ball attribute.
- struct `attr_traits< dist_type< I, Exact > >`
Trait specialization for the dist attribute.
- struct `attr_traits< cube_type< I, Exact > >`
Trait specialization for the cube attribute.
- struct `attr_traits< box_type< I, Exact > >`
Trait specialization for the box attribute.
- struct `attr_traits< other_image< Dad, I, Exact > >`
*Trait specialization for the *other_image* attribute.*
- struct `attr_traits< ball_parent_change< I, Exact > >`
Trait specialization for the ball_parent_change attribute.
- struct `change_exact< integral_type< T, OldExact >, NewExact >`
Change the exact type of an attribute.
- class `other_point`
Metaclass used to change attribute behavior.

6.23.1 Detailed Description

Implementation of attributes.

6.24 oln::morpho::attr::tools Namespace Reference

Useful tools for morphological math.

Functions

- `template<class T> T diffabs (const T &v1, const T &v2)`
Absolute value of difference between v1 and v2.

6.24.1 Detailed Description

Useful tools for morphological math.

6.24.2 Function Documentation

6.24.2.1 `template<class T> T diffabs (const T & v1, const T & v2)`

Absolute value of difference between v1 and v2.

Warning:

Should be moved elsewhere.

Definition at line 65 of file attributes.hh.

```
66      {  
67      return v1 > v2 ? v1 - v2 : v2 - v1;  
68      }
```

6.25 oln::morpho::env Namespace Reference

Implementation of environments used by attributes.

Classes

- struct [NullEnv](#)
Useless environment.
- struct [OtherImageEnv](#)
Environment containing image.
- struct [ParentEnv](#)
Environment containing point.

6.25.1 Detailed Description

Implementation of environments used by attributes.

6.26 oln::morpho::env::abstract Namespace Reference

Classes

- struct [env](#)

Top of environment hierarchy.

6.26.1 Detailed Description

brief Abstract stuff for environments.

6.27 oln::morpho::fast Namespace Reference

Algorithm enhanced for large structuring elements.

Functions

- template<class I, class N> [oln::mute< I >::ret card_closing](#) (const [abstract::non_vectorial_image< I >](#) &input, const [abstract::neighborhood< N >](#) &Ng, const typename oln::morpho::attr::attr_traits< [attr::card_type< unsigned >](#) >::lambda_type &lambda)
Perform a cardinal closing.
- template<class I, class N> [oln::mute< I >::ret card_opening](#) (const [abstract::non_vectorial_image< I >](#) &input, const [abstract::neighborhood< N >](#) &Ng, const typename oln::morpho::attr::attr_traits< [attr::card_type< unsigned >](#) >::lambda_type &lambda)
Perform a cardinal opening.
- template<class I, class N> [oln::mute< I >::ret integral_closing](#) (const [abstract::non_vectorial_image< I >](#) &input, const [abstract::neighborhood< N >](#) &Ng, const typename oln::morpho::attr::attr_traits< [attr::integral_type< unsigned >](#) >::lambda_type &lambda)
Perform an integral closing.
- template<class I, class N> [oln::mute< I >::ret integral_opening](#) (const [abstract::non_vectorial_image< I >](#) &input, const [abstract::neighborhood< N >](#) &Ng, const typename oln::morpho::attr::attr_traits< [attr::integral_type< unsigned >](#) >::lambda_type &lambda)
Perform an integral opening.
- template<class I, class N> [oln::mute< I >::ret height_opening](#) (const [abstract::non_vectorial_image< I >](#) &input, const [abstract::neighborhood< N >](#) &Ng, const typename oln::morpho::attr::attr_traits< [attr::height_type< unsigned >](#) >::lambda_type &lambda)
Perform a height closing.
- template<class I, class N> [oln::mute< I >::ret height_closing](#) (const [abstract::non_vectorial_image< I >](#) &input, const [abstract::neighborhood< N >](#) &Ng, const typename oln::morpho::attr::attr_traits< [attr::height_type< unsigned >](#) >::lambda_type &lambda)
Perform a height closing.
- template<class I, class N> [oln::mute< I >::ret maxvalue_closing](#) (const [abstract::non_vectorial_image< I >](#) &input, const [abstract::neighborhood< N >](#) &Ng, const typename oln::morpho::attr::attr_traits< [attr::maxvalue_type< unsigned >](#) >::lambda_type &lambda)
Perform a maxvalue closing.
- template<class I, class N> [oln::mute< I >::ret maxvalue_opening](#) (const [abstract::non_vectorial_image< I >](#) &input, const [abstract::neighborhood< N >](#) &Ng, const typename oln::morpho::attr::attr_traits< [attr::maxvalue_type< unsigned >](#) >::lambda_type &lambda)
Perform a maxvalue opening.
- template<class I, class N> [oln::mute< I >::ret minvalue_opening](#) (const [abstract::non_vectorial_image< I >](#) &input, const [abstract::neighborhood< N >](#) &Ng, const typename oln::morpho::attr::attr_traits< [attr::minvalue_type< unsigned >](#) >::lambda_type &lambda)
Perform a minvalue opening.

- `template<class I, class N> oln::mute< I >::ret minvalue_closing` (const `abstract::non_vectorial_image< I >` &input, const `abstract::neighborhood< N >` &Ng, const typename `oln::morpho::attr::attr_traits< attr::minvalue_type< unsigned > >::lambda_type` &lambda)
Perform a minvalue closing.
- `template<class I, class N> oln::mute< I >::ret ball_opening` (const `abstract::non_vectorial_image< I >` &input, const `abstract::neighborhood< N >` &Ng, const typename `oln::morpho::attr::attr_traits< attr::ball_type< I > >::lambda_type` &lambda)
Perform a ball opening.
- `template<class I, class N> oln::mute< I >::ret ball_closing` (const `abstract::non_vectorial_image< I >` &input, const `abstract::neighborhood< N >` &Ng, const typename `oln::morpho::attr::attr_traits< attr::ball_type< I > >::lambda_type` &lambda)
Perform a ball closing.
- `template<class I, class N> oln::mute< I >::ret dist_opening` (const `abstract::non_vectorial_image< I >` &input, const `abstract::neighborhood< N >` &Ng, const typename `oln::morpho::attr::attr_traits< attr::dist_type< I > >::lambda_type` &lambda)
Perform a dist opening.
- `template<class I, class N> oln::mute< I >::ret dist_closing` (const `abstract::non_vectorial_image< I >` &input, const `abstract::neighborhood< N >` &Ng, const typename `oln::morpho::attr::attr_traits< attr::dist_type< I > >::lambda_type` &lambda)
Perform a dist closing.
- `template<class I, class N> oln::mute< I >::ret cube_closing` (const `abstract::non_vectorial_image< I >` &input, const `abstract::neighborhood< N >` &Ng, const typename `oln::morpho::attr::attr_traits< attr::cube_type< I > >::lambda_type` &lambda)
Perform a cube closing.
- `template<class I, class N> oln::mute< I >::ret cube_opening` (const `abstract::non_vectorial_image< I >` &input, const `abstract::neighborhood< N >` &Ng, const typename `oln::morpho::attr::attr_traits< attr::cube_type< I > >::lambda_type` &lambda)
Perform a cube opening.
- `template<class I, class N> oln::mute< I >::ret box_closing` (const `abstract::non_vectorial_image< I >` &input, const `abstract::neighborhood< N >` &Ng, const typename `oln::morpho::attr::attr_traits< attr::box_type< I > >::lambda_type` &lambda)
Perform a box closing.
- `template<class I, class N> oln::mute< I >::ret box_opening` (const `abstract::non_vectorial_image< I >` &input, const `abstract::neighborhood< N >` &Ng, const typename `oln::morpho::attr::attr_traits< attr::box_type< I > >::lambda_type` &lambda)
Perform a box opening.
- `template<class I, class E> oln::mute< I >::ret closing` (const `abstract::non_vectorial_image< I >` &input, const `abstract::struct_elt< E >` &se)
Processing closing.
- `template<class I, class E> oln::mute< I >::ret dilation` (const `abstract::non_vectorial_image< I >` &input, const `abstract::struct_elt< E >` &se)

- `template<class I, class E> oln::mute< I >::ret erosion` (const `abstract::non_vectorial_image< I >` &input, const `abstract::struct_elt< E >` &se)
Perform a morphological erosion.
- `template<class C, class B, class I, class E> mute< I, typename convoutput< C, B, typename mlc::exact< I >::ret::value_type >::ret >::ret beucher_gradient` (const `convert::abstract::conversion< C, B >` &c, const `abstract::non_vectorial_image< I >` &input, const `abstract::struct_elt< E >` &se)
Process a morphological beucher gradient.
- `template<class I, class E> oln::mute< I >::ret external_gradient` (const `abstract::non_vectorial_image< I >` &input, const `abstract::struct_elt< E >` &se)
Process a morphological beucher gradient.
- `template<class C, class B, class I, class E1, class E2> mute< I, typename convoutput< C, B, typename mlc::exact< I >::ret::value_type >::ret >::ret hit_or_miss` (const `convert::abstract::conversion< C, B >` &c, const `abstract::non_vectorial_image< I >` &input, const `abstract::struct_elt< E1 >` &se1, const `abstract::struct_elt< E2 >` &se2)
Perform a 'hit or miss' transform.
- `template<class I, class E1, class E2> oln::mute< I >::ret hit_or_miss` (const `abstract::non_vectorial_image< I >` &input, const `abstract::struct_elt< E1 >` &se1, const `abstract::struct_elt< E2 >` &se2)
Perform a 'hit or miss' transform.
- `template<class I, class E1, class E2> oln::mute< I >::ret hit_or_miss_opening` (const `abstract::non_vectorial_image< I >` &input, const `abstract::struct_elt< E1 >` &se1, const `abstract::struct_elt< E2 >` &se2)
Perform an hit or miss opening.
- `template<class I, class E1, class E2> oln::mute< I >::ret hit_or_miss_opening_bg` (const `abstract::non_vectorial_image< I >` &input, const `abstract::struct_elt< E1 >` &se1, const `abstract::struct_elt< E2 >` &se2)
Perform an hit or miss opening of background.
- `template<class I, class E1, class E2> oln::mute< I >::ret hit_or_miss_closing` (const `abstract::non_vectorial_image< I >` &input, const `abstract::struct_elt< E1 >` &se1, const `abstract::struct_elt< E2 >` &se2)
Perform an hit or miss closing.
- `template<class I, class E1, class E2> oln::mute< I >::ret hit_or_miss_closing_bg` (const `abstract::non_vectorial_image< I >` &input, const `abstract::struct_elt< E1 >` &se1, const `abstract::struct_elt< E2 >` &se2)
Perform an hit or miss closing of background.
- `template<class C, class B, class I, class E> mute< I, typename convoutput< C, B, typename mlc::exact< I >::ret::value_type >::ret >::ret laplacian` (const `convert::abstract::conversion< C, B >` &c, const `abstract::non_vectorial_image< I >` &input, const `abstract::struct_elt< E >` &se)
Compute the laplacian of an image.
- `template<class DestValue, class I, class E> mute< I, DestValue >::ret laplacian` (const `abstract::non_vectorial_image< I >` &input, const `abstract::struct_elt< E >` &se)

Compute the laplacian of an image.

- `template<class I, class E> oln::mute< I >::ret opening (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Perform a morphological opening.

- `template<class I, class E1, class E2> oln::mute< I >::ret thickening (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E1 > &se1, const abstract::struct_elt< E2 > &se2)`

Thicken an image.

- `template<class I, class E1, class E2> oln::mute< I >::ret thinning (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E1 > &se1, const abstract::struct_elt< E2 > &se2)`

Thin an image.

- `template<class C, class B, class I, class E> mute< I, typename convoutput< C, B, typename mlc::exact< I >::ret::value_type >::ret >::ret white_top_hat (const convert::abstract::conversion< C, B > &c, const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Compute the white top hat of an image.

- `template<class I, class E> oln::mute< I >::ret white_top_hat (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Compute the white top hat of an image.

- `template<class C, class B, class I, class E> mute< I, typename convoutput< C, B, typename mlc::exact< I >::ret::value_type >::ret >::ret black_top_hat (const convert::abstract::conversion< C, B > &c, const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Compute the black top hat of an image.

- `template<class I, class E> oln::mute< I >::ret black_top_hat (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Compute the black top hat of an image.

- `template<class C, class B, class I, class E> mute< I, typename convoutput< C, B, typename mlc::exact< I >::ret::value_type >::ret >::ret self_complementary_top_hat (const convert::abstract::conversion< C, B > &c, const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Compute the self complementary top hat of an image.

- `template<class I, class E> oln::mute< I >::ret self_complementary_top_hat (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Compute the self complementary top hat of an image.

- `template<class C, class B, class I, class E> mute< I, typename convoutput< C, B, typename mlc::exact< I >::ret::value_type >::ret >::ret top_hat_contrast_op (const convert::abstract::conversion< C, B > &c, const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Top hat contrast operator.

- `template<class I, class E> oln::mute< I >::ret top_hat_contrast_op (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Top hat contrast operator.

6.27.1 Detailed Description

Algorithm enhanced for large structuring elements.

6.27.2 Function Documentation

6.27.2.1 `template<class I, class N> oln::mute< I >::ret ball_closing (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & Ng, const typename oln::morpho::attr::attr_traits< attr::ball_type< I > >::lambda_type & lambda)`

Perform a ball closing.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::fast::ball_closing(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_fast_ball_closing.ppm");
    return 0;
}
```



Definition at line 471 of file attribute_closing_opening.hh.

```
483 {
```

6.27.2.2 `template<class I, class N> oln::mute< I >::ret ball_opening (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & Ng, const typename oln::morpho::attr::attr_traits< attr::ball_type< I > >::lambda_type & lambda)`

Perform a ball opening.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
```

```

#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::fast::ball_opening(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_fast_ball_opening.ppm");
    return 0;
}

```



Definition at line 443 of file attribute_closing_opening.hh.

455 {

6.27.2.3 `template<class C, class B, class I, class E> mute<I, typename convoutput<C, B,typename mlc::exact< I >::ret::value_type>::ret>::ret beucher_gradient (const convert::abstract::conversion< C, B > & c, const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Process a morphological beucher gradient.

Compute the arithmetic difference between the diltation and the erosion of input using *se* as structural element. Soille, p67.

- *c* Conversion functor.
- *input* Image to process.
- *se* Structuring element.

Definition at line 298 of file gradient.hh.

6.27.2.4 `template<class I, class E> oln::mute< I >::ret black_top_hat (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Compute the black top hat of an image.

Parameters:

- I* Exact type of the image.
- E* Exact type of the structuring element.

- input Image to process.
- se Structuring element.

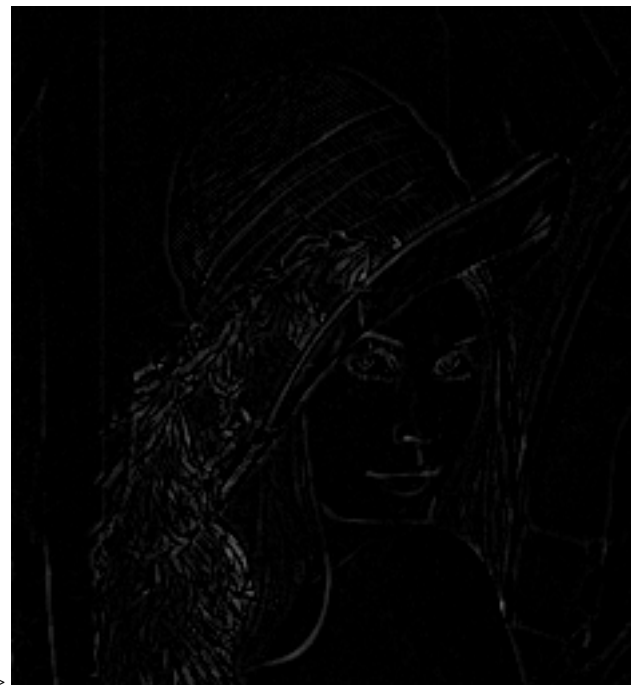
```
#include <oln/basics2d.hh>
#include <oln/morpho/top_hat.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::morpho::fast::black_top_hat(im1,
                                                oln::win_c8p()),
              IMG_OUT "oln_morpho_fast_black_top_hat_overload.pbm");
}
```



=>



Definition at line 619 of file top_hat.hh.

6.27.2.5 `template<class C, class B, class I, class E> mute<I, typename convoutput<C, B,typename mlc::exact< I >::ret::value_type>::ret>::ret black_top_hat (const convert::abstract::conversion< C, B > & c, const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Compute the black top hat of an image.

Compute black top hat of input using se as structuring element. Soille p.105.

- c Conversion object.
- input Image to process.
- se Structuring element.

```

#include <oln/basics2d.hh>
#include <oln/morpho/top_hat.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

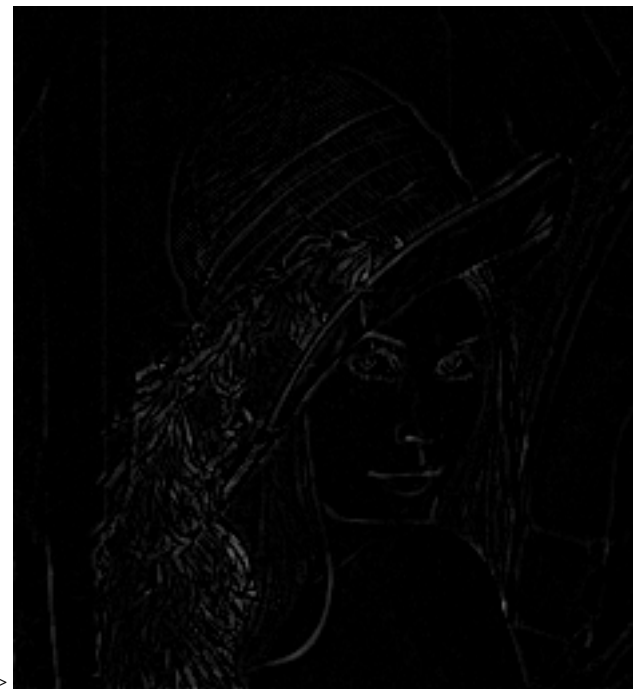
    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::morpho::fast::black_top_hat
              (oln::convert::bound<ntg::int_u8>(),
               im1,
               oln::win_c8p()),
              IMG_OUT "oln_morpho_fast_black_top_hat.pbm");
}

```



=>



Definition at line 577 of file top_hat.hh.

6.27.2.6 `template<class I, class N> oln::mute< I >::ret box_closing (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & Ng, const typename oln::morpho::attr::attr_traits< attr::box_type< I > >::lambda_type & lambda)`

Perform a box closing.

```

#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;
    ntg::vec<2, unsigned, mlc::final>    lambda;
    lambda[0] = lambda[1] = 50;
}

```

```

im_type im1(oln::load(IMG_IN "lena128.pgm"));
im1 = oln::morpho::fast::box_closing(im1, oln::neighb_c4(), lambda);
oln::save(im1, IMG_OUT "oln_morpho_fast_box_closing.ppm");
return 0;
}

```



Definition at line 612 of file attribute_closing_opening.hh.

```
624 {
```

6.27.2.7 `template<class I, class N> oln::mute< I >::ret box_opening (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & Ng, const typename oln::morpho::attr::attr_traits< attr::box_type< I > >::lambda_type & lambda)`

Perform a box opening.

```

#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;
    ntg::vec<2, unsigned, mlc::final>    lambda;
    lambda[0] = lambda[1] = 50;
    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::fast::box_opening(im1, oln::neighb_c4(), lambda);
    oln::save(im1, IMG_OUT "oln_morpho_fast_box_opening.ppm");
    return 0;
}

```



Definition at line 641 of file attribute_closing_opening.hh.

6.27.2.8 `template<class I, class N> oln::mute< I >::ret card_closing (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & Ng, const typename oln::morpho::attr::attr_traits< attr::card_type< unsigned > >::lambda_type & lambda)`

Perform a cardinal closing.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::fast::card_closing(im1, oln::neighb_c4(), 200);
    oln::save(im1, IMG_OUT "oln_morpho_fast_card_closing.ppm");
    return 0;
}
```



Definition at line 166 of file attribute_closing_opening.hh.

```
177 {
```

6.27.2.9 `template<class I, class N> oln::mute< I >::ret card_opening (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & Ng, const typename oln::morpho::attr::attr_traits< attr::card_type< unsigned > >::lambda_type & lambda)`

Perform a cardinal opening.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::fast::card_opening(im1, oln::neighb_c4(), 200);
    oln::save(im1, IMG_OUT "oln_morpho_fast_card_opening.ppm");
    return 0;
}
```



Definition at line 193 of file attribute_closing_opening.hh.

204 {

6.27.2.10 `template<class I, class E> oln::mute< I >::ret closing (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Processing closing.

Compute the morphological closing of input using se as structuring element.

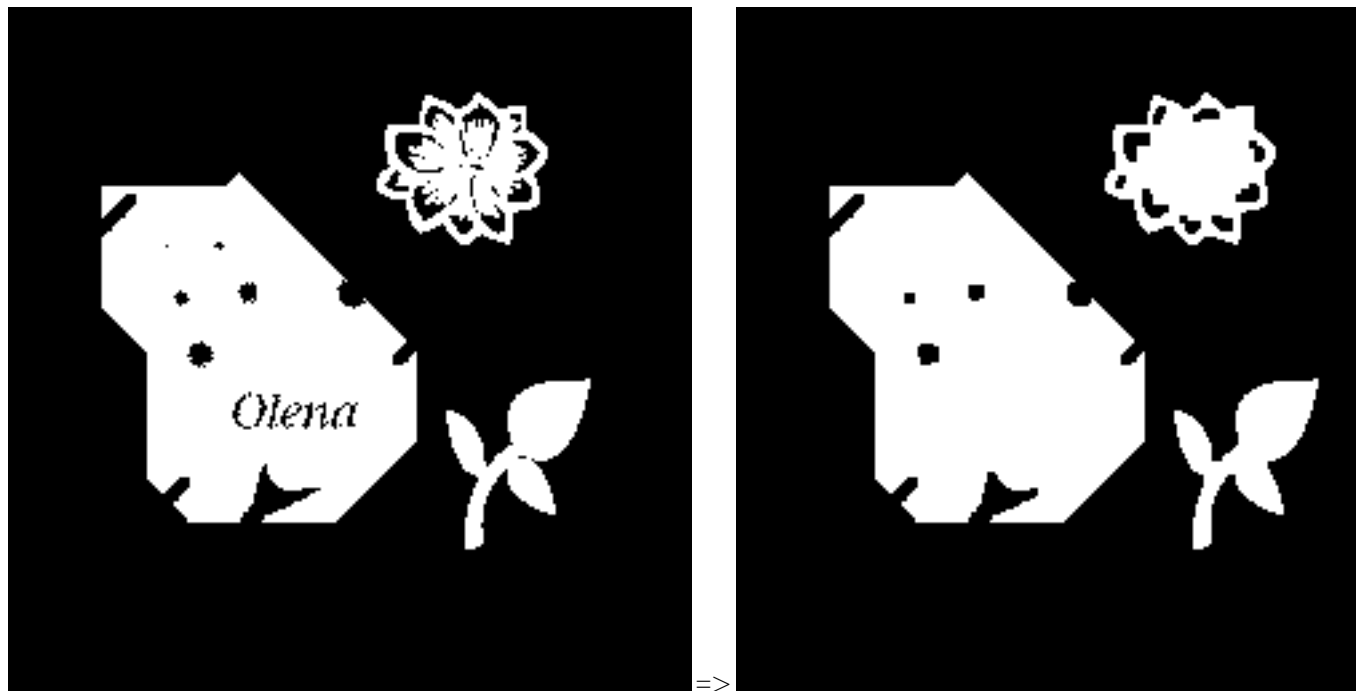
Parameters:

- I* Exact type of the input image.
- E* Exact type of the structuring element.

- input Input image to close.
- se Structuring element to use.

```
#include <oln/basics2d.hh>
#include <oln/morpho/closing.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>      im_type;

    im_type im1(oln::load(IMG_IN "object.pbm"));
    save(oln::morpho::closing(im1, oln::win_c8p()),
        IMG_OUT "oln_morpho_closing.pbm");
    return 0;
}
```



Definition at line 205 of file closing.hh.

6.27.2.11 `template<class I, class N> oln::mute< I >::ret cube_closing (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & Ng, const typename oln::morpho::attr::attr_traits< attr::cube_type< I > >::lambda_type & lambda)`

Perform a cube closing.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::fast::cube_closing(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_fast_cube_closing.ppm");
    return 0;
}
```




Definition at line 555 of file attribute_closing_opening.hh.

```
567 {
```

6.27.2.12 `template<class I, class N> oln::mute< I >::ret cube_opening (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &Ng, const typename oln::morpho::attr::attr_traits< attr::cube_type< I >>::lambda_type &lambda)`

Perform a cube opening.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::fast::cube_opening(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_fast_cube_opening.ppm");
    return 0;
}
```



Definition at line 583 of file attribute_closing_opening.hh.

```
595 {
```

6.27.2.13 `template<class I, class N> oln::mute< I >::ret dist_closing (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &Ng, const typename oln::morpho::attr::attr_traits< attr::dist_type< I > >::lambda_type &lambda)`

Perform a dist closing.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::fast::dist_closing(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_fast_dist_closing.ppm");
    return 0;
}
```



Definition at line 527 of file attribute_closing_opening.hh.

539 {

6.27.2.14 `template<class I, class N> oln::mute< I >::ret dist_opening (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &Ng, const typename oln::morpho::attr::attr_traits< attr::dist_type< I > >::lambda_type &lambda)`

Perform a dist opening.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::fast::dist_opening(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_fast_dist_opening.ppm");
    return 0;
}
```



Definition at line 499 of file attribute_closing_opening.hh.

511 {

6.27.2.15 `template<class I, class E> oln::mute< I >::ret erosion (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Perform a morphological erosion.

Compute the morphological erosion of input using se as structuring element.

Parameters:

I Exact type of the input image.

E Exact type of the structuring element.

- input Input image.
- se Structuring element to use.

```
#include <oln/basics2d.hh>
#include <oln/morpho/erosion.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>          im_type;

    im_type im1(oln::load(IMG_IN "object.pbm"));
    save(oln::morpho::fast::erosion(im1, oln::win_c8p()),
        IMG_OUT "oln_morpho_fast_erosion.pbm");
}
```



=>



Definition at line 186 of file erosion.hh.

```

188     {
189         return fast_morpho<I, E, utils::histogram_min<oln_value_type(I)> >
190             (input, se);
191     }

```

6.27.2.16 `template<class I, class E> oln::mute< I >::ret external_gradient (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Process a morphological beucher gradient.

Parameters:

- I* Exact type of the input image.
- E* Exact type of the structuring element.

Returns:

The beucher gradient of the input.

- input Image to process.
- se Structuring element.

```

#include <oln/basics2d.hh>
#include <oln/morpho/gradient.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena128.pgm"));

```

```

    save(oln::morpho::beucher_gradient(im1, oln::win_c8p()),
        IMG_OUT "oln_morpho_beucher_gradient.pbm");
    return 0;
}
\endcode

\image html lenal28_pgm.png
\image latex lenal28_pgm.png
=>
\image html oln_morpho_beucher_gradient.png
\image latex oln_morpho_beucher_gradient.png

*/
template<class I, class E>
typename oln::mute< I >::ret
    beucher_gradient(const abstract::non_vectorial_image<I>& input,
                    const abstract::struct_elt<E>& se)
{
    return beucher_gradient(convert::force<typename mlc::exact< I >::ret::value_type>(), input, se);
}

template<class C, class B, class I, class E>
typename mute<I, typename convoutput<C, B,typename mlc::exact< I >::ret::value_type>::ret>::ret
    internal_gradient(const convert::abstract::conversion<C, B>& c,
                    const abstract::non_vectorial_image<I>& input,
                    const abstract::struct_elt<E>& se)
{
    return arith::minus(c, input, erosion(input, se));
}

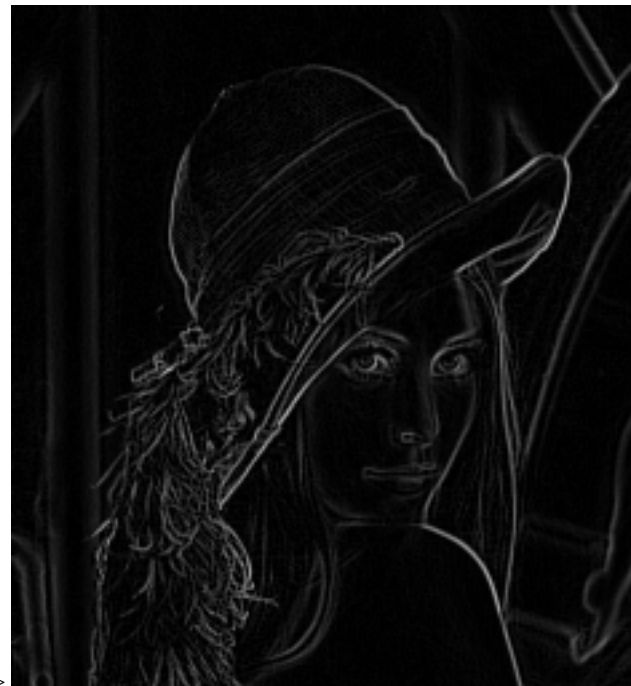
template<class I, class E>
typename oln::mute< I >::ret
    internal_gradient(const abstract::non_vectorial_image<I>& input, const
                    abstract::struct_elt<E>& se)
{
    return internal_gradient(convert::force<typename mlc::exact< I >::ret::value_type>(), input, se);
}

template<class C, class B, class I, class E>
typename mute<I, typename convoutput<C, B,typename mlc::exact< I >::ret::value_type>::ret>::ret
    external_gradient(const convert::abstract::conversion<C, B>& c,
                    const abstract::non_vectorial_image<I>& input,
                    const abstract::struct_elt<E>& se)
{
    return arith::minus(c, dilation(input, se), input);
}

```



=>



Definition at line 471 of file gradient.hh.

6.27.2.17 `template<class I, class N> oln::mute< I >::ret height_closing (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & Ng, const typename oln::morpho::attr::attr_traits< attr::height_type< unsigned > >::lambda_type & lambda)`

Perform a height closing.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::fast::height_closing(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_fast_height_closing.ppm");
    return 0;
}
```



Definition at line 302 of file attribute_closing_opening.hh.

```
315 {
```

6.27.2.18 `template<class I, class N> oln::mute< I >::ret height_opening (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &Ng, const typename oln::morpho::attr::attr_traits< attr::height_type< unsigned >>::lambda_type &lambda)`

Perform a height closing.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::fast::height_opening(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_fast_height_opening.ppm");
    return 0;
}
```



Definition at line 274 of file attribute_closing_opening.hh.

```
286 {
```

6.27.2.19 `template<class I, class E1, class E2> oln::mute< I >::ret hit_or_miss (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E1 > & se1, const abstract::struct_elt< E2 > & se2)`

Preform a 'hit or miss' transform.

Parameters:

I Exact type of the input image.

E1 Exact type of the first structuring element.

E2 Exact type of the second structuring element.

- *input* Image to process.

- *se1* First structuring element.

- *se2* Second structuring element.

```
#include <oln/basics2d.hh>
#include <oln/morpho/hit_or_miss.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>          im_type;

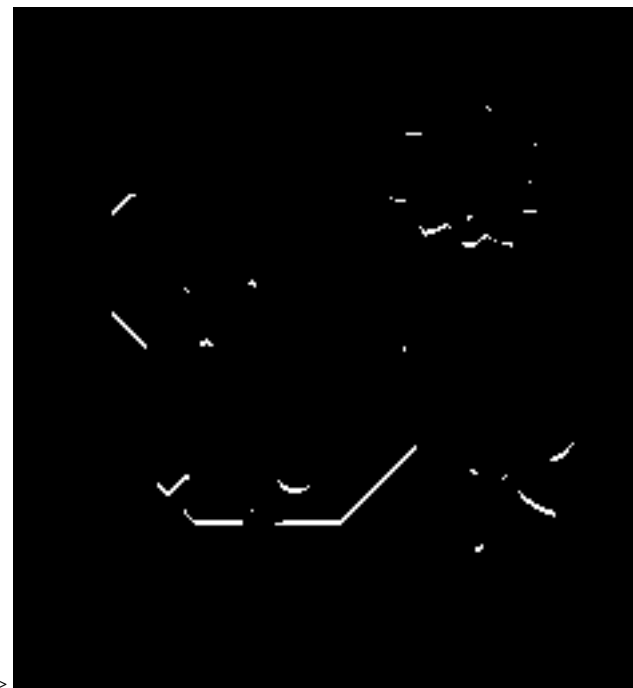
    im_type          im1(oln::load(IMG_IN "object.pbm"));

    oln::window2d mywin;
    mywin
        .add(-3,-2).add(-3,-1).add(-3,0).add(-3,1).add(-3,2)
        .add(-2,-1).add(-2,0).add(-2,1)
        .add(-1,0);
    oln::window2d mywin2 = - mywin;

    oln::save(oln::morpho::fast::hit_or_miss(im1, mywin, mywin2),
              IMG_OUT "oln_morpho_fast_hit_or_miss_overload.pbm");
}
```




=>



Definition at line 587 of file hit_or_miss.hh.

6.27.2.20 `template<class C, class B, class I, class E1, class E2> mute<I, typename convoutput<C, B, typename mlc::exact< I >::ret::value_type>::ret>::ret hit_or_miss (const convert::abstract::conversion< C, B > & c, const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E1 > & se1, const abstract::struct_elt< E2 > & se2)`

Perform a 'hit or miss' transform.

- c Conversion object.
- input Image to process.
- se1 First structuring element.
- se2 Second structuring element.

Compute the hit_or_miss transform of input by the composite structuring element (se1, se2). Soille p.131.

By definition se1 and se2 must have the same origin, and need to be disjoint. This algorithm has been extended to every data types (although it is not increasing). Beware the result depends upon the image data type if it is not bin.

```
#include <oln/basics2d.hh>
#include <oln/morpho/hit_or_miss.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>          im_type;

    im_type      im1(oln::load(IMG_IN "object.pbm"));

    oln::window2d mywin;
```

```

mywin
.add(-3,-2).add(-3,-1).add(-3,0).add(-3,1).add(-3,2)
.add(-2,-1).add(-2,0).add(-2,1)
.add(-1,0);
oln::window2d mywin2 = - mywin;

oln::save(oln::morpho::fast::hit_or_miss
(oln::convert::bound<ntg::int_u8>(), im1, mywin, mywin2),
IMG_OUT "oln_morpho_fast_hit_or_miss.pbm");
}

```



=>



Definition at line 525 of file hit_or_miss.hh.

6.27.2.21 `template<class I, class E1, class E2> oln::mute< I >::ret hit_or_miss_closing (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E1 > &se1, const abstract::struct_elt< E2 > &se2)`

Perform an hit or miss closing.

Compute the hit_or_miss closing of input by the composite structuring element (se1, se2). This is the dual transformation of hit-or-miss opening with respect to set complementation. Soille p.135.

By definition se1 and se2 must have the same origin, and need to be disjoint. This algorithm has been extended to every data types (althought it is not increasing). Beware the result depends upon the image data type if it is not bin.

Parameters:

- I** Exact type of the input image.
- E1** Exact type of the first structuring element.
- E2** Exact type of the second structuring element.

- input Image to process.

- se1 First structuring element.
- se2 Second structuring element.

```
#include <oln/basics2d.hh>
#include <oln/morpho/hit_or_miss.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>          im_type;

    im_type          im1(oln::load(IMG_IN "object.pbm"));

    oln::window2d mywin;
    mywin
        .add(-3,-2).add(-3,-1).add(-3,0).add(-3,1).add(-3,2)
        .add(-2,-1).add(-2,0).add(-2,1)
        .add(-1,0);
    oln::window2d mywin2 = - mywin;

    oln::save(oln::morpho::fast::hit_or_miss_closing(im1, mywin, mywin2),
              IMG_OUT "oln_morpho_fast_hit_or_miss_closing.pbm");
}
```



=>



Definition at line 777 of file hit_or_miss.hh.

6.27.2.22 `template<class I, class E1, class E2> oln::mute< I >::ret hit_or_miss_closing_bg (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E1 > &se1, const abstract::struct_elt< E2 > &se2)`

Perform an hit or miss closing of background.

Compute the hit_or_miss closing of the background of input by the composite structuring element (se1, se2). This is the dual transformation of hit-or-miss opening with respect to set complementation. Soille p.135.

By definition `se1` and `se2` must have the same origin, and need to be disjoint. This algorithm has been extended to every data types (although it is not increasing). Beware the result depends upon the image data type if it is not bin.

Parameters:

- I* Exact type of the input image.
- E1* Exact type of the first structuring element.
- E2* Exact type of the second structuring element.

- input Image to process.
- `se1` First structuring element.
- `se2` Second structuring element.

```
#include <oln/basics2d.hh>
#include <oln/morpho/hit_or_miss.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>          im_type;

    im_type          im1(oln::load(IMG_IN "object.pbm"));

    oln::window2d mywin;
    mywin
        .add(-3,-2).add(-3,-1).add(-3,0).add(-3,1).add(-3,2)
        .add(-2,-1).add(-2,0).add(-2,1)
        .add(-1,0);
    oln::window2d mywin2 = - mywin;

    oln::save(oln::morpho::fast::hit_or_miss_closing_bg(im1, mywin, mywin2),
              IMG_OUT "oln_morpho_fast_hit_or_miss_closing_bg.pbm");
}
```



=>



Definition at line 840 of file `hit_or_miss.hh`.

6.27.2.23 `template<class I, class E1, class E2> oln::mute< I >::ret hit_or_miss_opening (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E1 > & se1, const abstract::struct_elt< E2 > & se2)`

Perform an hit or miss opening.

Compute the hit_or_miss opening of input by the composite structuring element (se1, se2). Soille p.134.

By definition se1 and se2 must have the same origin, and need to be disjoint. This algorithm has been extended to every data types (although it is not increasing). Beware the result depends upon the image data type if it is not bin.

Parameters:

I Exact type of the input image.

E1 Exact type of the first structuring element.

E2 Exact type of the second structuring element.

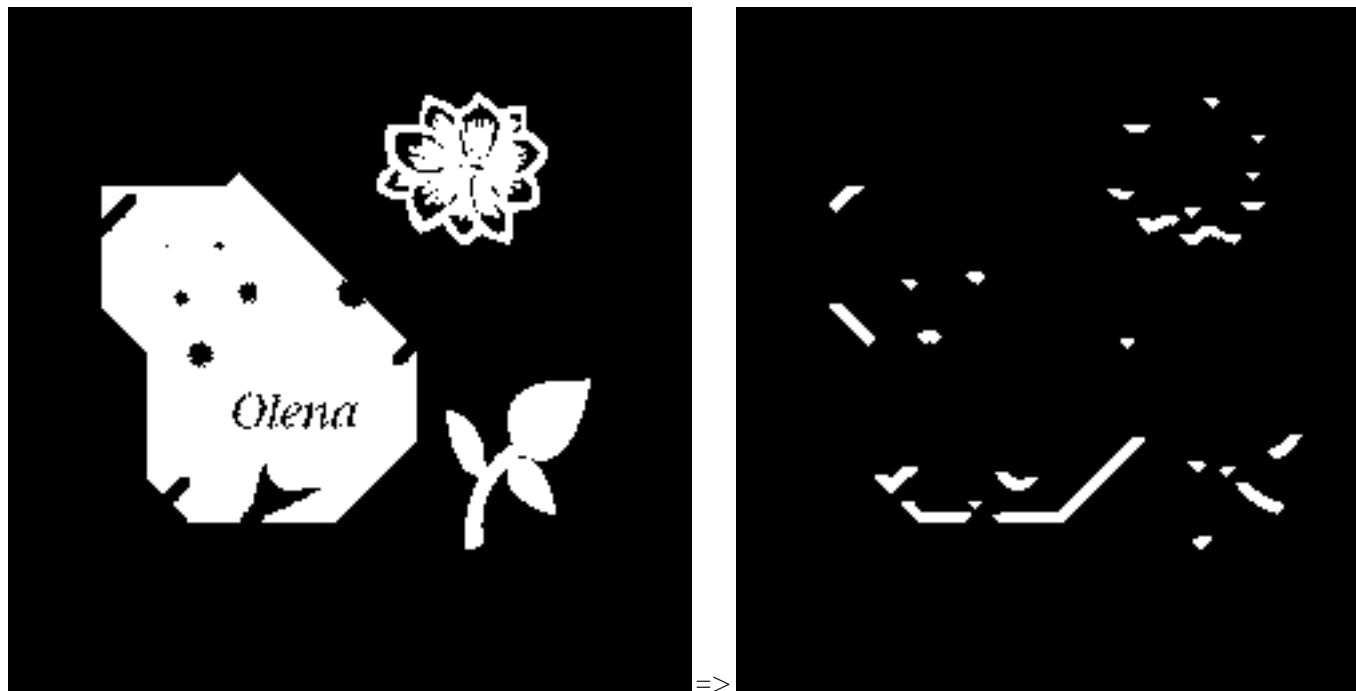
- input Image to process.
- se1 First structuring element.
- se2 Second structuring element.

```
#include <oln/basics2d.hh>
#include <oln/morpho/hit_or_miss.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>          im_type;

    im_type          im1(oln::load(IMG_IN "object.pbm"));

    oln::window2d mywin;
    mywin
        .add(-3,-2).add(-3,-1).add(-3,0).add(-3,1).add(-3,2)
        .add(-2,-1).add(-2,0).add(-2,1)
        .add(-1,0);
    oln::window2d mywin2 = - mywin;

    oln::save(oln::morpho::fast::hit_or_miss_opening(im1, mywin, mywin2),
              IMG_OUT "oln_morpho_fast_hit_or_miss_opening.pbm");
}
```



Definition at line 651 of file hit_or_miss.hh.

6.27.2.24 `template<class I, class E1, class E2> oln::mute< I >::ret hit_or_miss_opening_bg
(const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E1 > &
se1, const abstract::struct_elt< E2 > & se2)`

Perform an hit or miss opening of background.

Compute the hit_or_miss opening of the background of input by the composite structuring element (*se1*, *se2*). Soille p.135.

By definition *se1* and *se2* must have the same origin, and need to be disjoint. This algorithm has been extended to every data types (although it is not increasing). Beware the result depends upon the image data type if it is not bin.

Parameters:

- I* Exact type of the input image.
- E1* Exact type of the first structuring element.
- E2* Exact type of the second structuring element.

- input Image to process.
- *se1* First structuring element.
- *se2* Second structuring element.

```
#include <oln/basics2d.hh>
#include <oln/morpho/hit_or_miss.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>          im_type;
```

```

im_type      im1(oln::load(IMG_IN "object.pbm"));

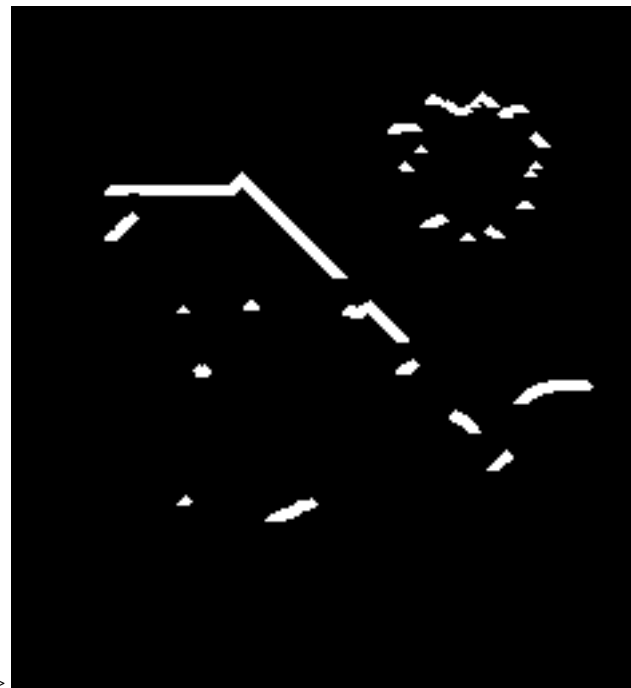
oln::window2d mywin;
mywin
    .add(-3,-2).add(-3,-1).add(-3,0).add(-3,1).add(-3,2)
    .add(-2,-1).add(-2,0).add(-2,1)
    .add(-1,0);
oln::window2d mywin2 = - mywin;

oln::save(oln::morpho::fast::hit_or_miss_opening_bg(im1, mywin, mywin2),
          IMG_OUT "oln_morpho_fast_hit_or_miss_opening_bg.pbm");
}

```



=>



Definition at line 711 of file hit_or_miss.hh.

6.27.2.25 `template<class I, class N> oln::mute< I >::ret integral_closing (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &Ng, const typename oln::morpho::attr::attr_traits< attr::integral_type< unsigned >>::lambda_type &lambda)`

Perform an integral closing.

```

#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::fast::integral_closing(im1, oln::neighb_c4(), 200);
    oln::save(im1, IMG_OUT "oln_morpho_fast_integral_closing.ppm");
    return 0;
}

```



Definition at line 220 of file attribute_closing_opening.hh.

```
231 {
```

6.27.2.26 `template<class I, class N> oln::mute< I >::ret integral_opening (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & Ng, const typename oln::morpho::attr::attr_traits< attr::integral_type< unsigned >>::lambda_type & lambda)`

Perform an integral opening.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::fast::integral_opening(im1, oln::neighb_c4(), 200);
    oln::save(im1, IMG_OUT "oln_morpho_fast_integral_opening.ppm");
    return 0;
}
```



Definition at line 247 of file attribute_closing_opening.hh.

```
258 {
```


6.27.2.27 `template<class DestValue, class I, class E> mute<I, DestValue>::ret laplacian (const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Compute the laplacian of an image.

Compute the laplacian of input using se as structural element.

- input Image to process.
- se Structuring element.

```
#include <oln/basics2d.hh>
#include <oln/morpho/laplacian.hh>
#include <oln/convert/stretch.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::image2d<ntg::int_s<10> > i10 =
        oln::morpho::fast::laplacian(oln::convert::bound<ntg::int_s<10> >(),
                                     im1,
                                     oln::win_c8p());

    oln::image2d<ntg::int_s<10> > f10 =
        oln::morpho::fast::laplacian<ntg::int_s<10> >(im1,
                                                         oln::win_c8p());

    oln::save(apply(oln::convert::stretch<ntg::int_u8>(), (f10)),
              IMG_OUT "oln_morpho_fast_laplacian_overload.pgm");
}
```



=>

Todo

FIXME: Not instantiated in swilena (see tools/swilena/generate_morpho_instantiations.py)

Definition at line 285 of file laplacian.hh.

6.27.2.28 `template<class C, class B, class I, class E> mute<I, typename convoutput<C, B,typename mlc::exact< I >::ret::value_type>::ret>::ret laplacian (const convert::abstract::conversion< C, B > & c, const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Compute the laplacian of an image.

Compute the laplacian of input using se as structural element.

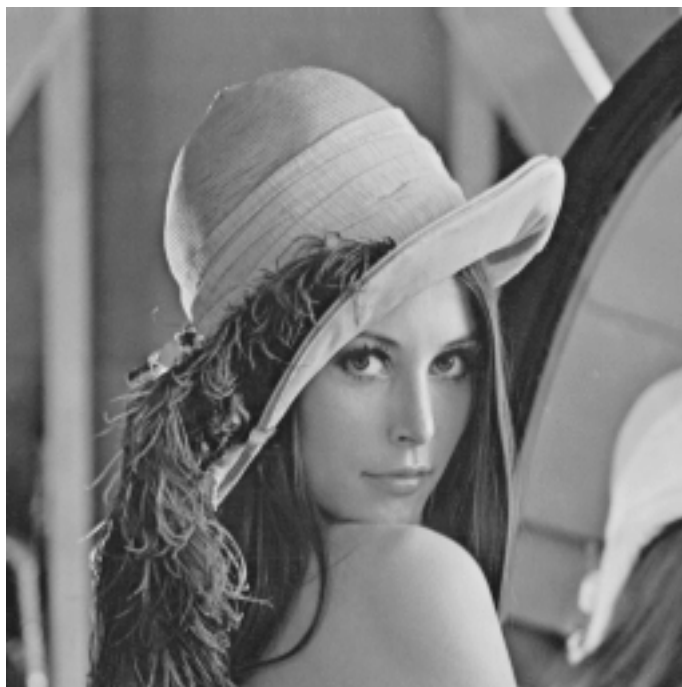
- c Conversion object.
- input Image to process.
- se Structuring element.

```
#include <oln/basics2d.hh>
#include <oln/morpho/laplacian.hh>
#include <oln/convert/stretch.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::image2d<ntg::int_s<10> > i10 =
        oln::morpho::fast::laplacian(oln::convert::bound<ntg::int_s<10> >(),
                                    im1,
                                    oln::win_c8p());

    oln::save(apply(oln::convert::stretch<ntg::int_u8>(), i10),
              IMG_OUT "oln_morpho_fast_laplacian.pgm");
}
```



=>



Definition at line 232 of file laplacian.hh.

6.27.2.29 `template<class I, class N> oln::mute< I >::ret maxvalue_closing (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &Ng, const typename oln::morpho::attr::attr_traits< attr::maxvalue_type< unsigned >>::lambda_type &lambda)`

Perform a maxvalue closing.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::fast::maxvalue_closing(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_fast_maxvalue_closing.ppm");
    return 0;
}
```



Definition at line 331 of file attribute_closing_opening.hh.

343 {

6.27.2.30 `template<class I, class N> oln::mute< I >::ret maxvalue_opening (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &Ng, const typename oln::morpho::attr::attr_traits< attr::maxvalue_type< unsigned >>::lambda_type &lambda)`

Perform a maxvalue opening.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;
```

```

im_type im1(oln::load(IMG_IN "lena128.pgm"));
im1 = oln::morpho::fast::maxvalue_opening(im1, oln::neighb_c4(), 5);
oln::save(im1, IMG_OUT "oln_morpho_fast_maxvalue_opening.ppm");
return 0;
}

```



Definition at line 359 of file attribute_closing_opening.hh.

```
371 {
```

6.27.2.31 `template<class I, class N> oln::mute< I >::ret minvalue_closing (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & Ng, const typename oln::morpho::attr::attr_traits< attr::minvalue_type< unsigned > >::lambda_type & lambda)`

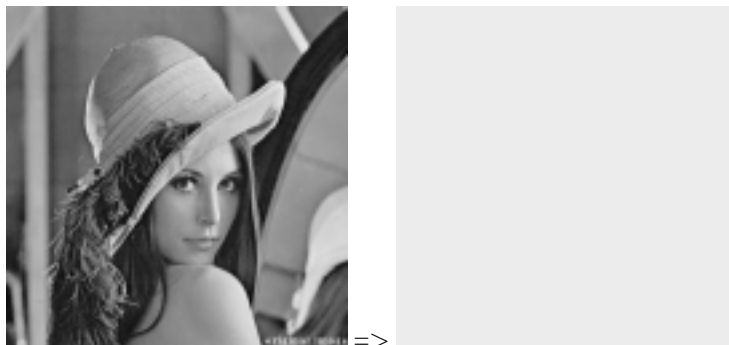
Perform a minvalue closing.

```

#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::fast::minvalue_closing(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_fast_minvalue_closing.ppm");
    return 0;
}

```



Definition at line 415 of file attribute_closing_opening.hh.

427 {

6.27.2.32 `template<class I, class N> oln::mute< I >::ret minvalue_opening (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & Ng, const typename oln::morpho::attr::attr_traits< attr::minvalue_type< unsigned >>::lambda_type & lambda)`

Perform a minvalue opening.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::fast::minvalue_opening(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_fast_minvalue_opening.ppm");
    return 0;
}
```



Definition at line 387 of file attribute_closing_opening.hh.

399 {

6.27.2.33 `template<class I, class E> oln::mute< I >::ret opening (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Perform a morphological opening.

Compute the morphological opening of input using se as structuring element.

Parameters:

- I* Exact type of the input image.
- E* Exact type of the structuring element.

- input Image to process.
- se Structuring element.

```

#include <oln/basics2d.hh>
#include <oln/morpho/opening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::bin>      im_type;

    im_type      im(oln::load(IMG_IN "object.pbm"));

    oln::save(oln::morpho::fast::opening(im, oln::win_c8p()),
              IMG_OUT "oln_morpho_fast_opening.pbm");
}

```



=>



Definition at line 174 of file opening.hh.

6.27.2.34 `template<class I, class E> oln::mute< I >::ret self_complementary_top_hat (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Compute the self complementary top hat of an image.

- *c* Conversion object.
- *input* Image to process.
- *se* Structuring element.

```

#include <oln/basics2d.hh>
#include <oln/morpho/top_hat.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>      im_type;

```

```

im_type      im1(oln::load(IMG_IN "lena256.pgm"));

oln::save(oln::morpho::fast::self_complementary_top_hat(im1,
                                                         oln::win_c8p()),
          IMG_OUT "oln_morpho_fast_self_complementary_top_hat_overload.pbm");
}

```



=>



Definition at line 709 of file top_hat.hh.

6.27.2.35 `template<class C, class B, class I, class E> mute<I, typename convoutput<C, B,typename mlc::exact< I >::ret::value_type>::ret>::ret self_complementary_top_hat (const convert::abstract::conversion< C, B > &c, const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Compute the self complementary top hat of an image.

Compute self complementary top hat of input using se as structuring element. Soille p.106.

- c Conversion object.
- input Image to process.
- se Structuring element.

```

#include <oln/basics2d.hh>
#include <oln/morpho/top_hat.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>      im_type;

    im_type      im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::morpho::fast::self_complementary_top_hat

```

```

        (oln::convert::bound<ntg::int_u8>(),
         im1,
         oln::win_c8p()),
        IMG_OUT "oln_morpho_fast_self_complementary_top_hat.pbm");
}

```



=>



Definition at line 669 of file top_hat.hh.

6.27.2.36 `template<class I, class E1, class E2> oln::mute< I >::ret thickening (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E1 > & se1, const abstract::struct_elt< E2 > & se2)`

Thicken an image.

Parameters:

- I* Exact type of the image.
- E1* Exact type of the first structuring element.
- E2* Exact type of the second structuring element.

- input Image to process.
- se1 First structuring element.
- se2 Second structuring element.

```

#include <oln/basics2d.hh>
#include <oln/morpho/thickening.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

```



```

im_type      im1(oln::load(IMG_IN "lena256.pgm"));

oln::save(oln::morpho::fast::thickening(im1,
                                         oln::win_c8p(),
                                         oln::win_c8p()),
          IMG_OUT "oln_morpho_fast_thickening.pbm");

return 0;
}

```



=>



Definition at line 194 of file thickening.hh.

6.27.2.37 `template<class I, class E1, class E2> oln::mute< I >::ret thinning (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E1 > & se1, const abstract::struct_elt< E2 > & se2)`

Thin an image.

Parameters:

- I* Exact type of the image.
- E1* Exact type of the first structuring element.
- E2* Exact type of the second structuring element.

- input Image to process.
- se1 First structuring element.
- se2 Second structuring element.

```

#include <oln/basics2d.hh>
#include <oln/morpho/thinning.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()

```

```

{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::morpho::fast::thinning(im1,
                                           oln::win_c8p(),
                                           oln::win_c8p()),
              IMG_OUT "oln_morpho_fast_thinning.pbm");

    return 0;
}

```



=>



Definition at line 194 of file thinning.hh.

6.27.2.38 `template<class I, class E> oln::mute< I >::ret top_hat_contrast_op (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Top hat contrast operator.

Enhance contrast input by adding the white top hat, then subtracting the black top hat to input. Top hats are computed using *se* as structuring element. Soille p.109.

- input Image to process.
- se Structuring element.

```

#include <oln/basics2d.hh>
#include <oln/morpho/top_hat.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

```

```

    oln::save(oln::morpho::fast::top_hat_contrast_op(im1,
                                                    oln::win_c8p()),
              IMG_OUT "oln_morpho_fast_top_hat_contrast_op_overload.pbm");
}

```



=>



Definition at line 804 of file top_hat.hh.

6.27.2.39 `template<class C, class B, class I, class E> mute<I, typename convoutput<C, B,typename mlc::exact< I >::ret::value_type>::ret>::ret top_hat_contrast_op (const convert::abstract::conversion< C, B > &c, const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)`

Top hat contrast operator.

Enhance contrast input by adding the white top hat, then subtracting the black top hat to input. Top hats are computed using se as structuring element. Soille p.109.

- c Conversion object.
- input Image to process.
- se Structuring element.

```

#include <oln/basics2d.hh>
#include <oln/morpho/top_hat.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::morpho::fast::top_hat_contrast_op

```

```

        (oln::convert::bound<ntg::int_u8>(),
         im1,
         oln::win_c8p()),
    IMG_OUT "oln_morpho_fast_top_hat_contrast_op.pbm");
}

```



=>



Definition at line 757 of file top_hat.hh.

6.27.2.40 `template<class I, class E> oln::mute< I >::ret white_top_hat (const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Compute the white top hat of an image.

Parameters:

- I* Exact type of the image.
- E* Exact type of the structuring element.

- input Image to process.
- se Structuring element.

```

#include <oln/basics2d.hh>
#include <oln/morpho/top_hat.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::morpho::fast::white_top_hat(im1,
                                                oln::win_c8p()),
              IMG_OUT "oln_morpho_fast_white_top_hat_overload.pbm");
}

```



=>



Definition at line 531 of file top_hat.hh.

6.27.2.41 `template<class C, class B, class I, class E> mute<I, typename convoutput<C, B,typename mlc::exact< I >::ret::value_type>::ret>::ret white_top_hat (const convert::abstract::conversion< C, B > & c, const abstract::non_vectorial_image< I > & input, const abstract::struct_elt< E > & se)`

Compute the white top hat of an image.

Compute white top hat of input using se as structuring element. Soille p.105.

- c Conversion object.
- input Image to process.
- se Structuring element.

```
#include <oln/basics2d.hh>
#include <oln/morpho/top_hat.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena256.pgm"));

    oln::save(oln::morpho::fast::white_top_hat
              (oln::convert::bound<ntg::int_u8>(),
               im1,
               oln::win_c8p()),
              IMG_OUT "oln_morpho_fast_white_top_hat.pbm");
}
```



=>



Definition at line 490 of file top_hat.hh.

6.28 oln::morpho::fast::tarjan Namespace Reference

oln::morpho::tarjan implementation.

Classes

- struct [tarjan_set](#)

Struct that contains everything to compute an attribute opening or closing.

6.28.1 Detailed Description

oln::morpho::tarjan implementation.

6.29 oln::morpho::fast::tarjan::internal Namespace Reference

Internal purpose only.

Functions

- `template<class I, class N, class A> oln::mute< I >::ret attr_closing_ (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &Ng, const typename oln::morpho::attr::attr_traits< A >::lambda_type &lambda, const typename oln::morpho::attr::attr_traits< A >::env_type &env=typename oln::morpho::attr::attr_traits< A >::env_type())`

Perform an attribute closing.

- `template<class I, class N, class A> oln::mute< I >::ret attr_opening_ (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &Ng, const typename oln::morpho::attr::attr_traits< A >::lambda_type &lambda, const typename oln::morpho::attr::attr_traits< A >::env_type &env=typename oln::morpho::attr::attr_traits< A >::env_type())`

Perform an attribute opening.

6.29.1 Detailed Description

Internal purpose only.

6.29.2 Function Documentation

- 6.29.2.1** `template<class I, class N, class A> oln::mute< I >::ret attr_closing_ (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &Ng, const typename oln::morpho::attr::attr_traits< A >::lambda_type &lambda, const typename oln::morpho::attr::attr_traits< A >::env_type &env = typename oln::morpho::attr::attr_traits< A >::env_type())`

Perform an attribute closing.

Parameters:

- I* Image exact type.
- N* Neighborhood exact type.
- A* Attribute exact type.
- input Input image.
- Ng Neighborhood to use.
- lambda Threshold to use.
- env Environment.

Definition at line 62 of file attribute_closing_opening.hh.

```

66         {
67             typedef tarjan::tarjan_set<oln_concrete_type(I), A > tarjan_set_type;
68             tarjan_set_type attr_closing(input.exact(), env);
69             return attr_closing.template get_comptute<true>(lambda, Ng);
70         }
```


6.29.2.2 `template<class I, class N, class A> oln::mute< I >::ret attr_opening_ (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & Ng, const typename oln::morpho::attr::attr_traits< A >::lambda_type & lambda, const typename oln::morpho::attr::attr_traits< A >::env_type & env = typename oln::morpho::attr::attr_traits< A >::env_type())`

Perform an attribute opening.

Parameters:

I Image exact type.

N Neighborhood exact type.

A Attribute exact type.

- *input* Input image.
- *Ng* Neighborhood to use.
- *lambda* Threshold to use.
- *env* Environment.

Definition at line 86 of file attribute_closing_opening.hh.

```

90      {
91          typedef tarjan::tarjan_set<oln_concrete_type(I), A > tarjan_set_type;
92          tarjan_set_type attr_opening(input.exact(), env);
93          return attr_opening.template get_comptute<false>(lambda, Ng);
94      }
```

6.30 oln::morpho::hybrid Namespace Reference

Hybrid (sure and sequential) algorithm implementation.

Functions

- `template<class I, class I2, class N> oln::mute< I >::ret minima_imposition` (const `abstract::non_vectorial_image< I >` &input, const `abstract::non_vectorial_image< I2 >` &minima_map, const `abstract::neighborhood< N >` &Ng)

Perform a minima imposition.

- `template<class I, class N> mute< I, ntg::bin >::ret regional_minima` (const `abstract::non_vectorial_image< I >` &input, const `abstract::neighborhood< N >` &Ng)

Extract regional minima.

- `template<class I1, class I2, class N> oln::mute< I1 >::ret geodesic_reconstruction_dilation` (const `abstract::non_vectorial_image< I1 >` &marker, const `abstract::non_vectorial_image< I2 >` &mask, const `abstract::neighborhood< N >` &Ng)

Perform a geodesic reconstruction dilation.

- `template<class I1, class I2, class N> oln::mute< I1 >::ret geodesic_reconstruction_erosion` (const `abstract::non_vectorial_image< I1 >` &marker, const `abstract::non_vectorial_image< I2 >` &mask, const `abstract::neighborhood< N >` &Ng)

Perform a geodesic reconstruction erosion.

6.30.1 Detailed Description

Hybrid (sure and sequential) algorithm implementation.

6.30.2 Function Documentation

- 6.30.2.1** `template<class I1, class I2, class N> oln::mute< I1 >::ret geodesic_reconstruction_dilation` (const `abstract::non_vectorial_image< I1 >` &marker, const `abstract::non_vectorial_image< I2 >` &mask, const `abstract::neighborhood< N >` &Ng)

Perform a geodesic reconstruction dilation.

Compute the reconstruction by dilation of marker with respect to the mask image using se as structuring element. Soille p.160. The algorithm used is the one defined as hybrid in Vincent(1993), Morphological grayscale reconstruction in image analysis: applications and efficient algorithms, itip, 2(2), 176–201.

Precondition:

Mask must be greater or equal than marker.

Parameters:

I1 Exact type of image marker.

I2 Exact type of image mask.

N Exact type of neighborhood.

- marker Image to work on.
- mask Image used for geodesic dilation.
- Ng Neighborhood to use.

```
#include <oln/basics2d.hh>
#include <oln/morpho/opening.hh>
#include <oln/morpho/reconstruction.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type      im1(oln::load(IMG_IN "lena128.pgm"));
    im_type      im2 = oln::morpho::opening(im1, oln::win_c4p());

    oln::save(oln::morpho::hybrid::geodesic_reconstruction_dilation(im2,
                                                                    im1,
                                                                    oln::neighb_c4()),
              IMG_OUT "oln_morpho_hybrid_geodesic_reconstruction_dilation.pbm");
    return 0;
}
```



Definition at line 284 of file reconstruction.hh.

References `oln::abstract::image< Exact >::clone()`, `oln::abstract::neighborhood< Exact >::delta()`, `oln::morpho::get_minus_se_p()`, `oln::morpho::get_plus_se_p()`, and `oln::abstract::image< Exact >::size()`.

```
287     {
288         mlc::eq<I1::dim, I2::dim>::ensure();
289         mlc::eq<I1::dim, N::dim>::ensure();
290
291         precondition(marker.size() == mask.size());
292         precondition(level::is_greater_or_equal(mask, marker));
293
294         oln_concrete_type(I1) output = marker.clone();
295         output.border_adapt_copy(Ng.delta());
296         {
297             typedef typename abstract::neighborhood<N>::win_type E;
298             E Ng_plus = get_plus_se_p(convert::ng_to_cse(Ng));
299             E Ng_minus = get_minus_se_p(convert::ng_to_cse(Ng));
300             typename I1::fwd_iter_type fwd_p(output);
301             typename I1::bkd_iter_type bkd_p(output);
302             for_all (fwd_p)
303                 output[fwd_p] = ntg::min(morpho::max(output, fwd_p, Ng_plus),
304                                           mask[fwd_p]);
305
306             std::queue<oln_point_type(I1) > fifo;
307             for_all (bkd_p)
```

```

308         {
309             output[bkd_p] = ntg::min(morpho::max(output, bkd_p, Ng_minus),
310                                     mask[bkd_p]);
311             if (internal::exist_init_dilation(bkd_p.cur(), output, mask, Ng_minus))
312                 fifo.push(bkd_p);
313         }
314         // Propagation Step
315         while (!fifo.empty())
316         {
317             oln_point_type(I1) p = fifo.front();
318             fifo.pop();
319             oln_neighb_type(N) q(Ng, p);
320             for_all (q) if (output.hold(q))
321             {
322                 if ((output[q] < output[p]) && (mask[q] != output[q]))
323                 {
324                     output[q] = ntg::min(output[p], mask[q]);
325                     fifo.push(q);
326                 }
327             }
328         }
329     }
330     return output;
331 }

```

6.30.2.2 `template<class I1, class I2, class N> oln::mute< I1 >::ret geodesic_reconstruction_erosion (const abstract::non_vectorial_image< I1 > & marker, const abstract::non_vectorial_image< I2 > & mask, const abstract::neighborhood< N > & Ng)`

Perform a geodesic reconstruction erosion.

Compute the reconstruction by erosion of marker with respect to the mask mask image using se as structuring element. Soille p.160. The algorithm used is the one defined as hybrid in Vincent(1993), Morphological grayscale reconstruction in image analysis: applications and efficient algorithms, itip, 2(2), 176–201.

Precondition:

Marker must be greater or equal than mask.

Parameters:

- I1** Exact type of image marker.
- I2** Exact type of image mask.
- N** Exact type of neighborhood.

- marker Image to work on.
- mask Image used for geodesic erosion.
- Ng Neighborhood to use.

```

#include <oln/basics2d.hh>
#include <oln/morpho/opening.hh>
#include <oln/morpho/reconstruction.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena128.pgm"));

```

```

im_type      im2 = oln::morpho::opening(im1, oln::win_c4p());

oln::save(oln::morpho::hybrid::geodesic_reconstruction_erosion(im1,
                                                                im2,
                                                                oln::neighb_c4()),
          IMG_OUT "oln_morpho_hybrid_geodesic_reconstruction_erosion.pbm");
return 0;
}

```



Definition at line 574 of file reconstruction.hh.

References oln::abstract::neighborhood< Exact >::delta(), oln::morpho::get_minus_se_p(), oln::morpho::get_plus_se_p(), and oln::abstract::image< Exact >::size().

```

577     {
578         mlc::eq<I1::dim, I2::dim>::ensure();
579         mlc::eq<I1::dim, N::dim>::ensure();
580
581         precondition(marker.size() == mask.size());
582         precondition(level::is_greater_or_equal(marker, mask));
583
584         oln_concrete_type(I1) output = marker.clone();
585         output.border_adapt_copy(Ng.delta());
586         {
587             typedef typename abstract::neighborhood<N>::win_type E;
588             E Ng_plus = get_plus_se_p(convert::ng_to_cse(Ng));
589             E Ng_minus = get_minus_se_p(convert::ng_to_cse(Ng));
590             typename I1::fwd_iter_type fwd_p(output);
591             typename I1::bkd_iter_type bkd_p(output);
592             for_all (fwd_p)
593                 output[fwd_p] = ntg::max(morpho::min(output, fwd_p, Ng_plus),
594                                         mask[fwd_p]);
595
596             std::queue<oln_point_type(I1) > fifo;
597             for_all (bkd_p)
598                 {
599                     output[bkd_p] = ntg::max(morpho::min(output, bkd_p, Ng_minus),
600                                             mask[bkd_p]);
601                     if (internal::exist_init_erosion(bkd_p.cur(), output, mask, Ng_minus))
602                         fifo.push(bkd_p);
603                 }
604             // Propagation Step
605             while (!fifo.empty())
606                 {
607                     oln_point_type(I1) p = fifo.front();
608                     fifo.pop();
609                     oln_neighb_type(N) q(Ng, p);
610                     for_all (q) if (output.hold(q))
611                         {
612                             if ((output[q] > output[p]) && (mask[q] != output[q]))
613                                 {
614                                     output[q] = ntg::max(output[p], mask[q]);

```

```

615             fifo.push(q);
616         }
617     }
618 }
619 }
620 return output;
621 }

```

6.30.2.3 `template<class I, class I2, class N> oln::mute< I >::ret minima_imposition (const abstract::non_vectorial_image< I > &input, const abstract::non_vectorial_image< I2 > &minima_map, const abstract::neighborhood< N > &Ng)`

Perform a minima imposition.

Impose minima defined by minima_map on input using Ng as neighborhood. minima_map must be a bin image (true for a minimum, false for a non minimum). Soille p.172.

Parameters:

I Exact type of the first image.

I2 Exact type of the second image.

N Exact type of the neighborhood.

- input Input image.
- minima_map Minima map image.
- Ng Neighborhood to use.

```

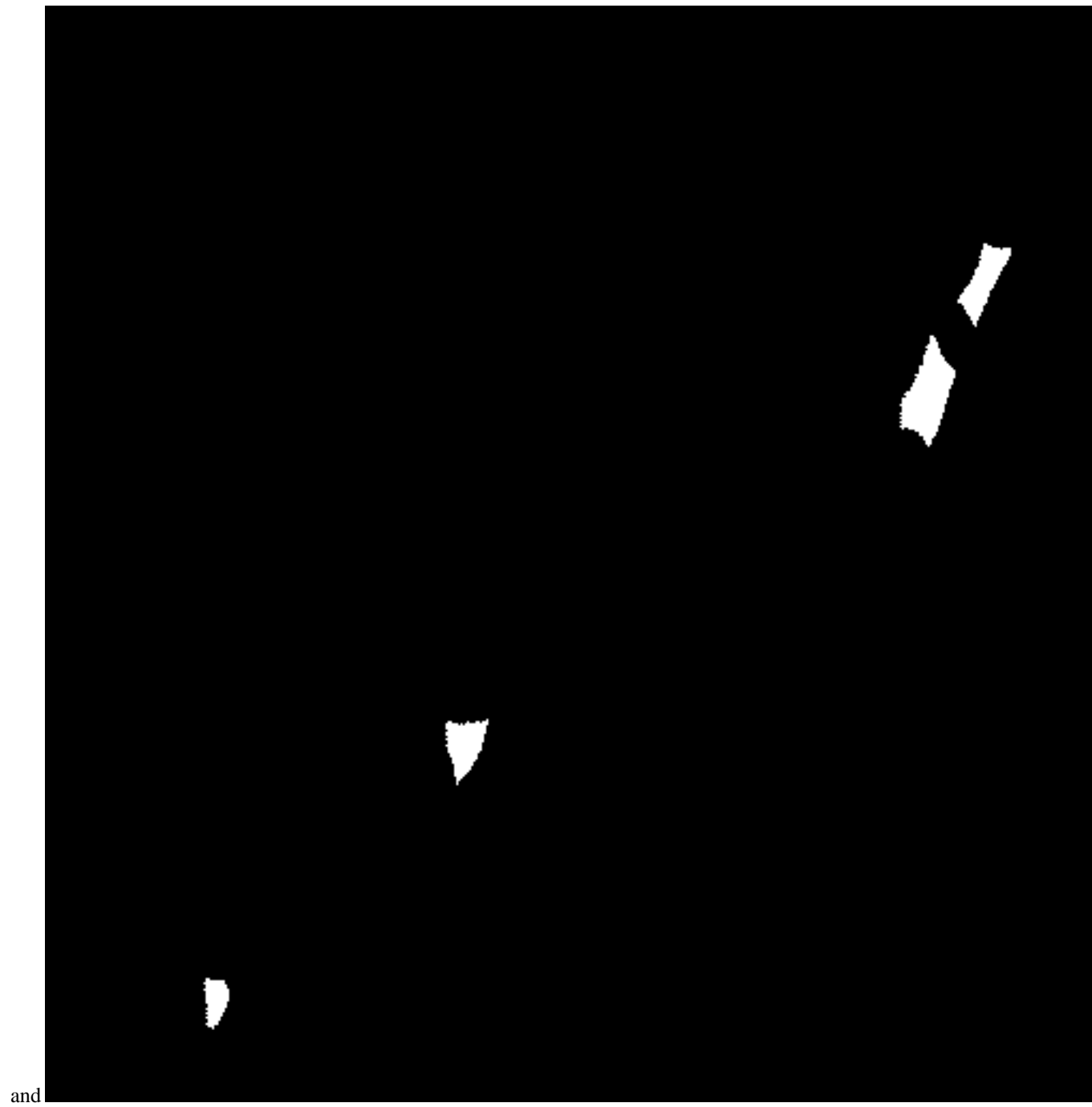
#include <oln/basics2d.hh>
#include <oln/morpho/extrema.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;
    typedef oln::image2d<ntg::bin>       bin_im_type;

    im_type          light(oln::load(IMG_IN "lena.pgm"));
    bin_im_type      minima(oln::load(IMG_IN "map.pbm"));

    oln::save(oln::morpho::sequential::minima_imposition(light,
                                                            minima,
                                                            oln::neighb_c4()),
              IMG_OUT "oln_morpho_sequential_minima_imposition.pgm");
}

```





and



=>

Definition at line 547 of file extrema.hh.

6.30.2.4 `template<class I, class N> mute<I, ntg::bin>::ret regional_minima (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & Ng)`

Extract regional minima.

Parameters:

I Exact type of input image.

Exact type of neighborhood.

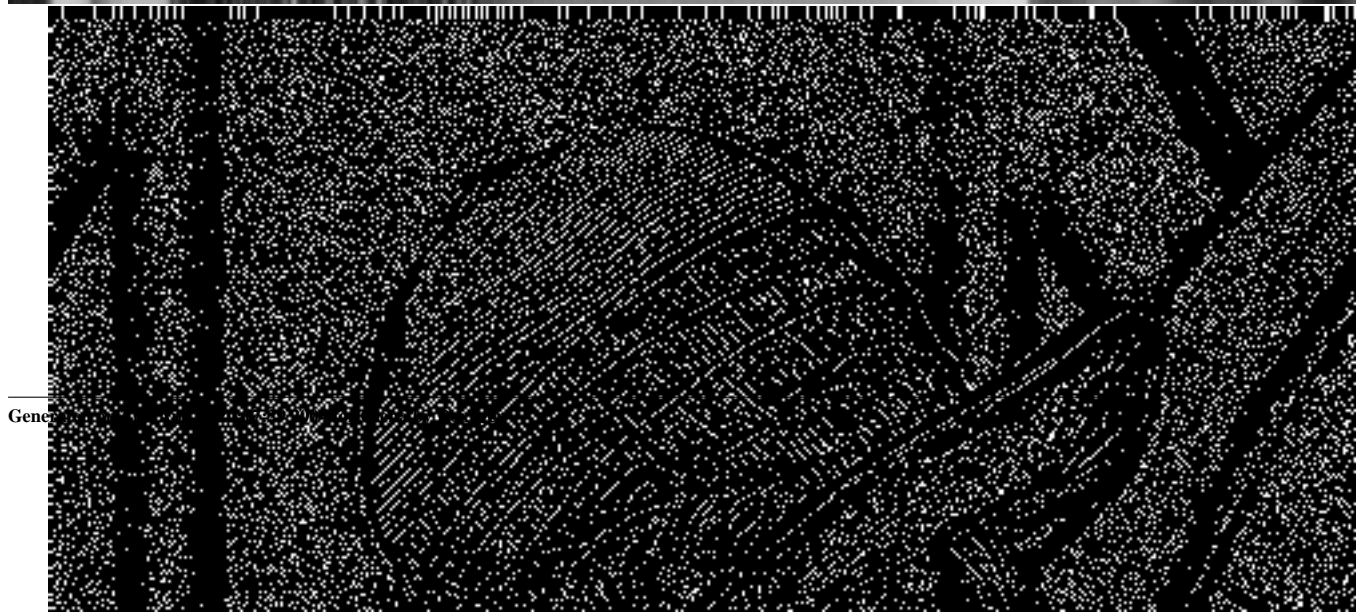
- input Input image.

- Ng Neighborhood to use.

```
#include <oln/basics2d.hh>
#include <oln/morpho/extrema.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type                                im(oln::load(IMG_IN "lena.pgm"));

    oln::save(oln::morpho::sequential::regional_minima(im, oln::neighb_c4()),
              IMG_OUT "oln_morpho_sequential_regional_minima.pgm");
}
```



Definition at line 599 of file extrema.hh.

6.31 oln::morpho::internal Namespace Reference

Internal purpose only.

Classes

- struct [fast_morpho_inner](#)
- struct **fast_morpho_inner**< Dim, Dim, I, S, H, B, P, O >
- struct [stat_](#)

Min and Max on a structuring element.

- struct **stat_**< I, E, ntg::bin >
- struct **watershed_seg_point_handler_**
- struct **watershed_con_point_handler_**

Functions

- template<class E1, class E2, class E3> void [find_struct_elts](#) (const [abstract::struct_elt](#)< E1 > &se, E2 se_add[mlc::exact< E1 >::ret::dim], E3 se_rem[mlc::exact< E1 >::ret::dim])

Find structuring elements.

- template<class I, class E1, class E2, class H> void [hist_update](#) ([utils::abstract::histogram](#)< H > &hist, const [abstract::non_vectorial_image](#)< I > &input, const typename mlc::exact< I >::ret::point_type &p, const [abstract::struct_elt](#)< E1 > &se_rem, const [abstract::struct_elt](#)< E2 > &se_add)

Update an histogram.

- template<class Point, class T> bool [watershed_seg_sort_](#) (const std::pair< Point, T > &p1, const std::pair< Point, T > &p2)

Check if the second element of p1 is lower than the second one of p2.

- template<class PointHandler, class DestValue, class I, class N> [mute](#)< I, DestValue >::ret [soille_watershed_](#) (const [abstract::non_vectorial_image](#)< I > &im_i, const [abstract::neighborhood](#)< N > &Ng)

Algorithm by Vincent and Soille.

6.31.1 Detailed Description

Internal purpose only.

6.31.2 Function Documentation

- #### 6.31.2.1
- template<class E1, class E2, class E3> void [find_struct_elts](#) (const [abstract::struct_elt](#)< E1 > &se, E2 se_add[mlc::exact< E1 >::ret::dim], E3 se_rem[mlc::exact< E1 >::ret::dim])

Find structuring elements.

Find structuring elements to be added/removed from the histogram when we move forward along each direction.

Todo

FIXME: add(dp) on w_windows associates a default weight set to 1

Definition at line 58 of file fast_morpho.hxx.

```

61     {
62         mlc_is_a(E2, abstract::struct_elt)::ensure();
63         mlc_is_a(E3, abstract::struct_elt)::ensure();
64         const unsigned dim = E1::dim;
65
66         // back[n] allows to move backward on coordinate 'n'.
67         oln_dpoint_type(E1) back[dim];
68         for (unsigned n = 0; n < dim; ++n)
69             back[n].nth(n) = -1;
70
71         oln_iter_type(E1) dp(se);
72         oln_iter_type(E1) dp_prime(se);
73
74         for_all(dp)
75         {
76             bool add[dim];          // whether to add 'dp' when moving forward
77                                     // on coordinate 'n'.
78             bool rem[dim];          // whether to remove 'dp'.
79             for (unsigned n = 0; n < dim; ++n)
80                 add[n] = rem[n] = true;
81
82             for_all(dp_prime)
83                 for (unsigned n = 0; n < dim; ++n)
84                 {
85                     if (dp_prime.cur() + back[n] == dp)
86                         // DP_PRIME is already in SE: don't add it.
87                         add[n] = false;
88
89                     if (dp.cur() + back[n] == dp_prime)
90                         // DP is still in SE: don't remove it.
91                         rem[n] = false;
92                 }
93
94             for (unsigned n = 0; n < dim; ++n)
95             {
96                 if (add[n])
97                 {
98                     se_add[n].add(dp);
99                 }
100                 if (rem[n])
101                 {
102                     se_rem[n].add(dp.cur() + back[n]);
103                 }
104             }
105         }
106
107         for (unsigned n = 0; n < dim; ++n)
108             postcondition(se_add[n].card() == se_rem[n].card());
109     }

```

6.31.2.2 `template<class I, class E1, class E2, class H> void hist_update
(utils::abstract::histogram< H > & hist, const abstract::non_vectorial_image< I > &
input, const typename mlc::exact< I >::ret::point_type & p, const abstract::struct_elt<
E1 > & se_rem, const abstract::struct_elt< E2 > & se_add)`

Update an histogram.

Update HIST by adding elements from _SE_ADD, and removing those from _SE_REM.

Definition at line 120 of file fast_morpho.hxx.

Referenced by `oln::morpho::internal::fast_morpho_inner< NP1, Dim, I, S, H, B, P, O >::doit()`.

```
125     {  
126     {  
127         oln_iter_type(E1) dp(se_rem);  
128         for_all(dp)  
129             --hist[input[p + dp]];  
130     }  
131     {  
132         oln_iter_type(E2) dp(se_add);  
133         for_all(dp)  
134             ++hist[input[p + dp]];  
135     }  
136 }
```

6.32 oln::morpho::sequential Namespace Reference

Sequential algorithm implementation.

Functions

- `template<class I, class I2, class N> oln::mute< I >::ret minima_imposition` (const `abstract::non_vectorial_image< I >` &input, const `abstract::non_vectorial_image< I2 >` &minima_map, const `abstract::neighborhood< N >` &Ng)

Perform a minima imposition.

- `template<class I, class N> mute< I, ntg::bin >::ret regional_minima` (const `abstract::non_vectorial_image< I >` &input, const `abstract::neighborhood< N >` &Ng)

Extract regional minima.

- `template<class I1, class I2, class N> oln::mute< I1 >::ret geodesic_reconstruction_dilation` (const `abstract::non_vectorial_image< I1 >` &marker, const `abstract::non_vectorial_image< I2 >` &mask, const `abstract::neighborhood< N >` &Ng)

Perform a geodesic reconstruction dilation.

- `template<class I1, class I2, class N> oln::mute< I1 >::ret geodesic_reconstruction_erosion` (const `abstract::non_vectorial_image< I1 >` &marker, const `abstract::non_vectorial_image< I2 >` &mask, const `abstract::neighborhood< N >` &Ng)

Perform a geodesic reconstruction erosion.

6.32.1 Detailed Description

Sequential algorithm implementation.

6.32.2 Function Documentation

- 6.32.2.1** `template<class I1, class I2, class N> oln::mute< I1 >::ret geodesic_reconstruction_dilation` (const `abstract::non_vectorial_image< I1 >` &marker, const `abstract::non_vectorial_image< I2 >` &mask, const `abstract::neighborhood< N >` &Ng)

Perform a geodesic reconstruction dilation.

Compute the reconstruction by dilation of marker with respect to the mask image using se as structuring element. Soille p.160. The algorithm used is the one defined as sequential in Vincent(1993), Morphological grayscale reconstruction in image analysis: applications and efficient algorithms, itip, 2(2), 176–201.

Precondition:

Mask must be greater or equal than marker.

Parameters:

I1 Exact type of image marker.

I2 Exact type of image mask.

N Exact type of neighborhood.

- marker Image to work on.
- mask Image used for geodesic dilation.
- Ng Neighborhood to use.

```
#include <oln/basics2d.hh>
#include <oln/morpho/opening.hh>
#include <oln/morpho/reconstruction.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena128.pgm"));
    im_type    im2 = oln::morpho::opening(im1, oln::win_c4p());

    oln::save(oln::morpho::sequential::geodesic_reconstruction_dilation(im2,
                                                                    im1,
                                                                    oln::neighb_c4()),
              IMG_OUT "oln_morpho_sequential_geodesic_reconstruction_dilation.pbm");
    return 0;
}
```



Definition at line 168 of file reconstruction.hh.

References oln::abstract::image< Exact >::clone(), oln::morpho::get_minus_se_p(), oln::morpho::get_plus_se_p(), and oln::abstract::image< Exact >::size().

```
171     {
172         mlc::eq<I1::dim, I2::dim>::ensure();
173         mlc::eq<I1::dim, N::dim>::ensure();
174         precondition(marker.size() == mask.size());
175         precondition(level::is_greater_or_equal(mask, marker));
176
177         // Conversion of neighborhood into a SE.
178         typedef typename abstract::neighborhood<N>::win_type E;
179         E se_plus = get_plus_se_p(convert::ng_to_cse(Ng));
180         E se_minus = get_minus_se_p(convert::ng_to_cse(Ng));
181
182         oln_concrete_type(I1) output = marker.clone();
183         bool non_stability = true;
184         typename I1::fwd_iter_type fwd_p(output);
185         typename I1::bkd_iter_type bkd_p(output);
186         while (non_stability)
187         {
188             oln_concrete_type(I1) work = output.clone();
189             work.border_adapt_copy(Ng.delta());
190             for_all (fwd_p)
191                 work[fwd_p] = ntg::min(morpho::max(work, fwd_p, se_plus), mask[fwd_p]);
```

```

192         for_all (bkd_p)
193             work[bkd_p] = ntg::min(morpho::max(work, bkd_p, se_minus), mask[bkd_p]);
194             non_stability = !(level::is_equal(work, output));
195             output = work;
196         }
197     return output;
198 }

```

6.32.2.2 `template<class I1, class I2, class N> oln::mute< I1 >::ret geodesic_reconstruction_erosion (const abstract::non_vectorial_image< I1 > & marker, const abstract::non_vectorial_image< I2 > & mask, const abstract::neighborhood< N > & Ng)`

Perform a geodesic reconstruction erosion.

Compute the reconstruction by erosion of marker with respect to the mask image using se as structuring element. Soille p.160. The algorithm used is the one defined as sequential in Vincent(1993), Morphological grayscale reconstruction in image analysis: applications and efficient algorithms, itip, 2(2), 176–201.

Precondition:

Marker must be greater or equal than mask.

Parameters:

I1 Exact type of image marker.

I2 Exact type of image mask.

N Exact type of neighborhood.

- marker Image to work on.
- mask Image used for geodesic erosion.
- Ng Neighborhood to use.

```

#include <oln/basics2d.hh>
#include <oln/morpho/opening.hh>
#include <oln/morpho/reconstruction.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena128.pgm"));
    im_type    im2 = oln::morpho::opening(im1, oln::win_c4p());

    oln::save(oln::morpho::sequential::geodesic_reconstruction_erosion(im1,
                                                                    im2,
                                                                    oln::neighb_c4()),
              IMG_OUT "oln_morpho_sequential_geodesic_reconstruction_erosion.pbm");
    return 0;
}

```



Definition at line 462 of file reconstruction.hh.

References oln::morpho::get_minus_se_p(), oln::morpho::get_plus_se_p(), and oln::abstract::image< Exact >::size().

```

465     {
466         mlc::eq<I1::dim, I2::dim>::ensure();
467         mlc::eq<I1::dim, N::dim>::ensure();
468         precondition(marker.size() == mask.size());
469         precondition(level::is_greater_or_equal(marker, mask));
470
471         typedef typename abstract::neighborhood<N>::win_type E;
472         E se_plus = get_plus_se_p(convert::ng_to_cse(Ng));
473         E se_minus = get_minus_se_p(convert::ng_to_cse(Ng));
474         oln_concrete_type(I1) output = marker.clone();
475
476         bool non_stability = true;
477         typename I1::fwd_iter_type fwd_p(output);
478         typename I1::bkd_iter_type bkd_p(output);
479         while (non_stability)
480         {
481             oln_concrete_type(I1) work = output.clone();
482             work.border_adapt_copy(Ng.delta());
483             for_all (fwd_p)
484                 work[fwd_p] = ntg::max(morpho::min(work, fwd_p, se_plus), mask[fwd_p]);
485             for_all (bkd_p)
486                 work[bkd_p] = ntg::max(morpho::min(work, bkd_p, se_minus), mask[bkd_p]);
487             non_stability = !(level::is_equal(work, output));
488             output = work;
489         }
490         return output;
491     }

```

6.32.2.3 `template<class I, class I2, class N> oln::mute< I >::ret minima_imposition (const abstract::non_vectorial_image< I > &input, const abstract::non_vectorial_image< I2 > &minima_map, const abstract::neighborhood< N > &Ng)`

Perform a minima imposition.

Impose minima defined by minima_map on input using Ng as neighborhood. minima_map must be a bin image (true for a minimum, false for a non minimum). Soille p.172.

Parameters:

I Exact type of the first image.

I2 Exact type of the second image.

N Exact type of the neighborhood.

- input Input image.
- minima_map Minima map image.
- Ng Neighborhood to use.

```
#include <oln/basics2d.hh>
#include <oln/morpho/extrema.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;
    typedef oln::image2d<ntg::bin>       bin_im_type;

    im_type          light(oln::load(IMG_IN "lena.pgm"));
    bin_im_type      minima(oln::load(IMG_IN "map.pbm"));

    oln::save(oln::morpho::sequential::minima_imposition(light,
                                                         minima,
                                                         oln::neighb_c4()),
              IMG_OUT "oln_morpho_sequential_minima_imposition.pgm");
}
```



and



=>

Definition at line 356 of file extrema.hh.

6.32.2.4 `template<class I, class N> mute<I, ntg::bin>::ret regional_minima (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & Ng)`

Extract regional minima.

Parameters:

I Exact type of input image.

Exact type of neighborhood.

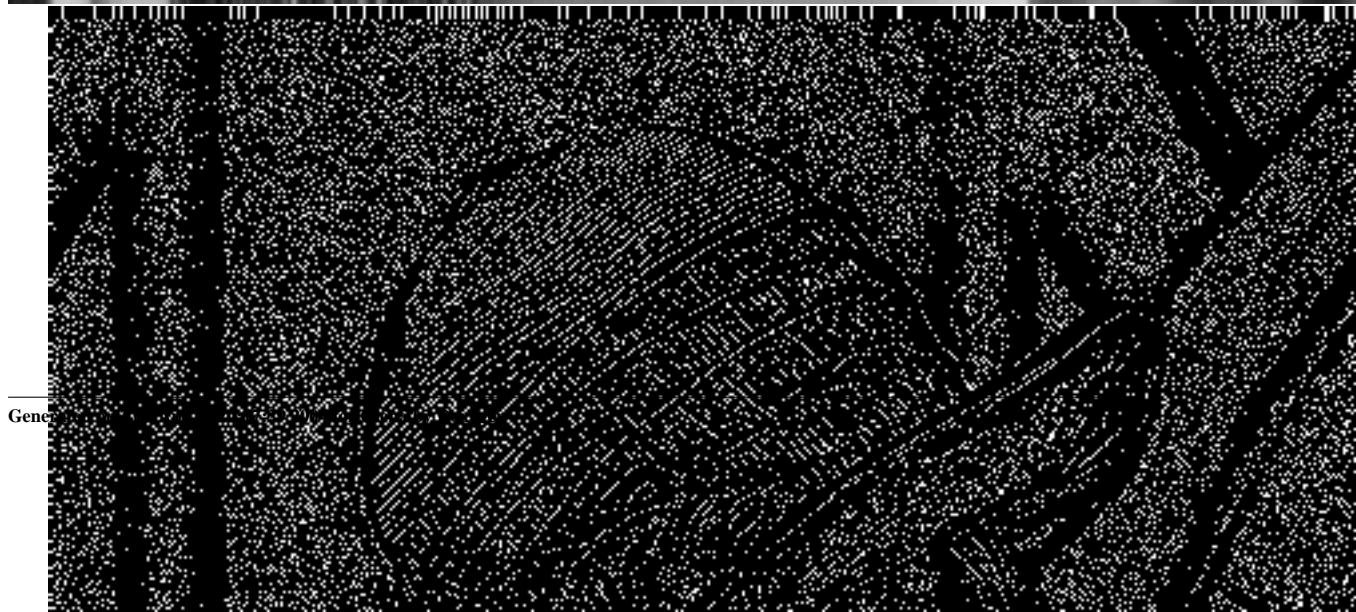
- input Input image.

- Ng Neighborhood to use.

```
#include <oln/basics2d.hh>
#include <oln/morpho/extrema.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type                                im(oln::load(IMG_IN "lena.pgm"));

    oln::save(oln::morpho::sequential::regional_minima(im, oln::neighb_c4()),
              IMG_OUT "oln_morpho_sequential_regional_minima.pgm");
}
```

Definition at line 408 of file extrema.hh.

6.33 oln::morpho::slow Namespace Reference

Algorithm that are slow (but need less memory), or that are slow if it is used with a large structuring element.

Classes

- struct [f_tarjan_map](#)

Functions

- template<class D, class I, class N> [oln::mute](#)< I >::ret [tarjan_map](#) (bool is_closing, const [abstract::non_vectorial_image](#)< I > &input, const [abstract::neighborhood](#)< N > &ng, const typename oln::morpho::attr::attr_traits< D >::lambda_type &lambda, const typename oln::morpho::attr::attr_traits< D >::env_type &env)
- template<class I, class N> [oln::mute](#)< I >::ret [card_closing](#) (const [abstract::non_vectorial_image](#)< I > &input, const [abstract::neighborhood](#)< N > &ng, const typename oln::morpho::attr::attr_traits< [attr::card_type](#)<>>::lambda_type &lambda)

Perform a cardinal closing.

- template<class I, class N> [oln::mute](#)< I >::ret [card_opening](#) (const [abstract::non_vectorial_image](#)< I > &input, const [abstract::neighborhood](#)< N > &ng, const typename oln::morpho::attr::attr_traits< [attr::card_type](#)<>>::lambda_type &lambda)

Perform a cardinal opening.

- template<class I, class N> [oln::mute](#)< I >::ret [integral_closing](#) (const [abstract::non_vectorial_image](#)< I > &input, const [abstract::neighborhood](#)< N > &ng, const typename oln::morpho::attr::attr_traits< [attr::integral_type](#)<>>::lambda_type &lambda)

Perform an integral closing.

- template<class I, class N> [oln::mute](#)< I >::ret [integral_opening](#) (const [abstract::non_vectorial_image](#)< I > &input, const [abstract::neighborhood](#)< N > &ng, const typename oln::morpho::attr::attr_traits< [attr::integral_type](#)<>>::lambda_type &lambda)

Perform an integral opening.

- template<class I, class N> [oln::mute](#)< I >::ret [height_opening](#) (const [abstract::non_vectorial_image](#)< I > &input, const [abstract::neighborhood](#)< N > &ng, const typename oln::morpho::attr::attr_traits< [attr::height_type](#)<>>::lambda_type &lambda)

Perform a height closing.

- template<class I, class N> [oln::mute](#)< I >::ret [height_closing](#) (const [abstract::non_vectorial_image](#)< I > &input, const [abstract::neighborhood](#)< N > &ng, const typename oln::morpho::attr::attr_traits< [attr::height_type](#)<>>::lambda_type &lambda)

Perform a height closing.

- template<class I, class N> [oln::mute](#)< I >::ret [maxvalue_closing](#) (const [abstract::non_vectorial_image](#)< I > &input, const [abstract::neighborhood](#)< N > &ng, const typename oln::morpho::attr::attr_traits< [attr::maxvalue_type](#)<>>::lambda_type &lambda)

Perform a maxvalue closing.

- `template<class I, class N> oln::mute< I >::ret maxvalue_opening (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &ng, const typename oln::morpho::attr::attr_traits< attr::maxvalue_type<> >::lambda_type &lambda)`

Perform a maxvalue opening.

- `template<class I, class N> oln::mute< I >::ret minvalue_opening (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &ng, const typename oln::morpho::attr::attr_traits< attr::minvalue_type<> >::lambda_type &lambda)`

Perform a minvalue opening.

- `template<class I, class N> oln::mute< I >::ret minvalue_closing (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &ng, const typename oln::morpho::attr::attr_traits< attr::minvalue_type<> >::lambda_type &lambda)`

Perform a minvalue closing.

- `template<class I, class N> oln::mute< I >::ret ball_opening (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &ng, const typename oln::morpho::attr::attr_traits< attr::ball_type< I > >::lambda_type &lambda)`

Perform a ball opening.

- `template<class I, class N> oln::mute< I >::ret ball_closing (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &ng, const typename oln::morpho::attr::attr_traits< attr::ball_type< I > >::lambda_type &lambda)`

Perform a ball closing.

- `template<class I, class N> oln::mute< I >::ret dist_opening (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &ng, const typename oln::morpho::attr::attr_traits< attr::dist_type< I > >::lambda_type &lambda)`

Perform a dist opening.

- `template<class I, class N> oln::mute< I >::ret dist_closing (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &ng, const typename oln::morpho::attr::attr_traits< attr::dist_type< I > >::lambda_type &lambda)`

Perform a dist closing.

- `template<class I, class N> oln::mute< I >::ret cube_closing (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &ng, const typename oln::morpho::attr::attr_traits< attr::cube_type< I > >::lambda_type &lambda)`

Perform a cube closing.

- `template<class I, class N> oln::mute< I >::ret cube_opening (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &ng, const typename oln::morpho::attr::attr_traits< attr::cube_type< I > >::lambda_type &lambda)`

Perform a cube opening.

- `template<class I, class N> oln::mute< I >::ret box_closing (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &ng, const typename oln::morpho::attr::attr_traits< attr::box_type< I > >::lambda_type &lambda)`

Perform a box closing.

- template<class I, class N> [oln::mute< I >::ret box_opening](#) (const [abstract::non_vectorial_image< I >](#) &input, const [abstract::neighborhood< N >](#) &ng, const typename oln::morpho::attr::attr_traits< [attr::box_type< I >](#) >::lambda_type &lambda)

Perform a box opening.

6.33.1 Detailed Description

Algorithm that are slow (but need less memory), or that are slow if it is used with a large structuring element.

6.33.2 Function Documentation

- 6.33.2.1** template<class I, class N> [oln::mute< I >::ret ball_closing](#) (const [abstract::non_vectorial_image< I >](#) &input, const [abstract::neighborhood< N >](#) &ng, const typename oln::morpho::attr::attr_traits< [attr::ball_type< I >](#) >::lambda_type &lambda)

Perform a ball closing.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening_map.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::slow::ball_closing(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_slow_ball_closing.ppm");
}
```



Definition at line 580 of file attribute_closing_opening_map.hh.

592 {

6.33.2.2 `template<class I, class N> oln::mute< I >::ret ball_opening (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & ng, const typename oln::morpho::attr::attr_traits< attr::ball_type< I > >::lambda_type & lambda)`

Perform a ball opening.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening_map.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::slow::ball_opening(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_slow_ball_opening.ppm");
}
```



Definition at line 553 of file attribute_closing_opening_map.hh.

```
565 {
```

6.33.2.3 `template<class I, class N> oln::mute< I >::ret box_closing (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & ng, const typename oln::morpho::attr::attr_traits< attr::box_type< I > >::lambda_type & lambda)`

Perform a box closing.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening_map.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;
    ntg::vec<2, unsigned, mlc::final>    lambda;
    lambda[0] = lambda[1] = 50;
    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::slow::box_closing(im1, oln::neighb_c4(), lambda);
    oln::save(im1, IMG_OUT "oln_morpho_slow_box_closing.ppm");
}
```



Definition at line 716 of file attribute_closing_opening_map.hh.

```
728 {
```

6.33.2.4 `template<class I, class N> oln::mute< I >::ret box_opening (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &ng, const typename oln::morpho::attr::attr_traits< attr::box_type< I >>::lambda_type &lambda)`

Perform a box opening.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening_map.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;
    ntg::vec<2, unsigned, mlc::final>    lambda;
    lambda[0] = lambda[1] = 50;
    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::slow::box_opening(im1, oln::neighb_c4(), lambda);
    oln::save(im1, IMG_OUT "oln_morpho_slow_box_opening.ppm");
}
```



Definition at line 744 of file attribute_closing_opening_map.hh.

6.33.2.5 `template<class I, class N> oln::mute< I >::ret card_closing (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & ng, const typename oln::morpho::attr::attr_traits< attr::card_type<> >::lambda_type & lambda)`

Perform a cardinal closing.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening_map.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::slow::card_closing(im1, oln::neighb_c4(), 200);
    oln::save(im1, IMG_OUT "oln_morpho_slow_card_closing.ppm");
}
```



Definition at line 284 of file `attribute_closing_opening_map.hh`.

```
295 {
```

6.33.2.6 `template<class I, class N> oln::mute< I >::ret card_opening (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & ng, const typename oln::morpho::attr::attr_traits< attr::card_type<> >::lambda_type & lambda)`

Perform a cardinal opening.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening_map.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::slow::card_opening(im1, oln::neighb_c4(), 200);
    oln::save(im1, IMG_OUT "oln_morpho_slow_card_opening.ppm");
}
```




Definition at line 310 of file attribute_closing_opening_map.hh.

```
321 {
```

6.33.2.7 `template<class I, class N> oln::mute< I >::ret cube_closing (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & ng, const typename oln::morpho::attr::attr_traits< attr::cube_type< I > >::lambda_type & lambda)`

Perform a cube closing.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening_map.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::slow::cube_closing(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_slow_cube_closing.ppm");
}
```



Definition at line 661 of file attribute_closing_opening_map.hh.

```
673 {
```

6.33.2.8 `template<class I, class N> oln::mute< I >::ret cube_opening (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & ng, const typename oln::morpho::attr::attr_traits< attr::cube_type< I > >::lambda_type & lambda)`

Perform a cube opening.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening_map.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::slow::cube_opening(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_slow_cube_opening.ppm");
}
```



Definition at line 688 of file attribute_closing_opening_map.hh.

```
700 {
```

6.33.2.9 `template<class I, class N> oln::mute< I >::ret dist_closing (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & ng, const typename oln::morpho::attr::attr_traits< attr::dist_type< I > >::lambda_type & lambda)`

Perform a dist closing.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening_map.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::slow::dist_closing(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_slow_dist_closing.ppm");
}
```



Definition at line 634 of file attribute_closing_opening_map.hh.

```
646 {
```

6.33.2.10 `template<class I, class N> oln::mute< I >::ret dist_opening (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &ng, const typename oln::morpho::attr::attr_traits< attr::dist_type< I > >::lambda_type &lambda)`

Perform a dist opening.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening_map.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::slow::dist_opening(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_slow_dist_opening.ppm");
}
```



Definition at line 607 of file attribute_closing_opening_map.hh.

```
619 {
```

6.33.2.11 `template<class I, class N> oln::mute< I >::ret height_closing (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &ng, const typename oln::morpho::attr::attr_traits< attr::height_type<> >::lambda_type &lambda)`

Perform a height closing.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening_map.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::slow::height_closing(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_slow_height_closing.ppm");
}
```



Definition at line 417 of file attribute_closing_opening_map.hh.

430 {

6.33.2.12 `template<class I, class N> oln::mute< I >::ret height_opening (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &ng, const typename oln::morpho::attr::attr_traits< attr::height_type<> >::lambda_type &lambda)`

Perform a height closing.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening_map.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::slow::height_opening(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_slow_height_opening.ppm");
}
```



Definition at line 390 of file attribute_closing_opening_map.hh.

```
402 {
```

6.33.2.13 `template<class I, class N> oln::mute< I >::ret integral_closing (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &ng, const typename oln::morpho::attr::attr_traits< attr::integral_type<> >::lambda_type & lambda)`

Perform an integral closing.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening_map.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::slow::integral_closing(im1, oln::neighb_c4(),
                                             200);
    oln::save(im1, IMG_OUT "oln_morpho_slow_integral_closing.ppm");
}
```



Definition at line 337 of file attribute_closing_opening_map.hh.

```
348 {
```

6.33.2.14 `template<class I, class N> oln::mute< I >::ret integral_opening (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &ng, const typename oln::morpho::attr::attr_traits< attr::integral_type<> >::lambda_type &lambda)`

Perform an integral opening.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening_map.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::slow::integral_opening(im1, oln::neighb_c4(),
                                              200);
    oln::save(im1, IMG_OUT "oln_morpho_slow_integral_opening.ppm");
}
```



Definition at line 364 of file attribute_closing_opening_map.hh.

375 {

6.33.2.15 `template<class I, class N> oln::mute< I >::ret maxvalue_closing (const abstract::non_vectorial_image< I > &input, const abstract::neighborhood< N > &ng, const typename oln::morpho::attr::attr_traits< attr::maxvalue_type<> >::lambda_type &lambda)`

Perform a maxvalue closing.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening_map.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::slow::maxvalue_closing(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_slow_maxvalue_closing.ppm");
}
```



Definition at line 445 of file attribute_closing_opening_map.hh.

```
457 {
```

6.33.2.16 `template<class I, class N> oln::mute< I >::ret maxvalue_opening (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & ng, const typename oln::morpho::attr::attr_traits< attr::maxvalue_type<> >::lambda_type & lambda)`

Perform a maxvalue opening.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening_map.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::slow::maxvalue_opening(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_slow_maxvalue_opening.ppm");
}
```



Definition at line 472 of file attribute_closing_opening_map.hh.

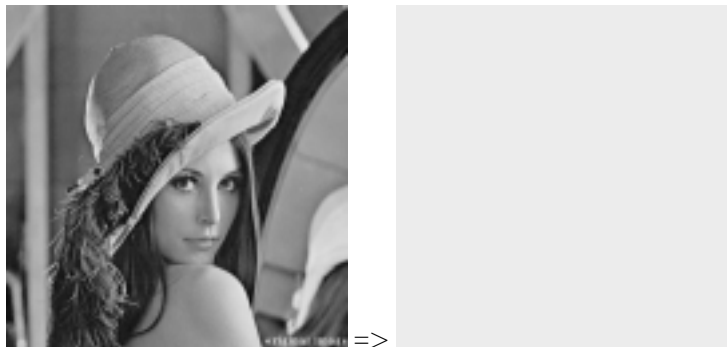
```
484 {
```

6.33.2.17 `template<class I, class N> oln::mute< I >::ret minvalue_closing (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & ng, const typename oln::morpho::attr::attr_traits< attr::minvalue_type<>>::lambda_type & lambda)`

Perform a minvalue closing.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening_map.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::slow::minvalue_closing(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_slow_minvalue_closing.ppm");
}
```



Definition at line 526 of file `attribute_closing_opening_map.hh`.

538 {

6.33.2.18 `template<class I, class N> oln::mute< I >::ret minvalue_opening (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & ng, const typename oln::morpho::attr::attr_traits< attr::minvalue_type<>>::lambda_type & lambda)`

Perform a minvalue opening.

```
#include <oln/basics2d.hh>
#include <oln/morpho/attribute_closing_opening_map.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type im1(oln::load(IMG_IN "lena128.pgm"));
    im1 = oln::morpho::slow::minvalue_opening(im1, oln::neighb_c4(), 5);
    oln::save(im1, IMG_OUT "oln_morpho_slow_minvalue_opening.ppm");
}
```




Definition at line 499 of file attribute_closing_opening_map.hh.

```
511 {
```

6.33.2.19 `template<class D, class I, class N> oln::mute< I >::ret tarjan_map (bool is_closing, const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & ng, const typename oln::morpho::attr::attr_traits< D >::lambda_type & lambda, const typename oln::morpho::attr::attr_traits< D >::env_type & env)`

Attribute closing using map. Smaller memory usage, but slower computation than the attribute_closing_opening

See "Fast morphological attribute operations using Tarjan's union-find algorithm" by Michael H. F. Wilkinson and Jos B. T. M. Roerdink

Definition at line 195 of file attribute_closing_opening_map.hh.

References `oln::morpho::slow::f_tarjan_map< I, D, Env >::res()`.

```
200     {
201         oln::morpho::slow::f_tarjan_map<I, D, attr_env_type(D) > t(is_closing,
202                                                                    input,
203                                                                    ng,
204                                                                    lambda,
205                                                                    env);
206         return t.res();
207     }
```

6.34 oln::morpho::sure Namespace Reference

Reference algorithm implementation: sure but slow.

Functions

- `template<class I, class I2, class N> oln::mute< I >::ret minima_imposition` (const `abstract::non_vectorial_image< I > &input`, const `abstract::non_vectorial_image< I2 > &minima_map`, const `abstract::neighborhood< N > &Ng`)
Perform a minima imposition.
- `template<class I, class N> mute< I, ntg::bin >::ret regional_minima` (const `abstract::non_vectorial_image< I > &input`, const `abstract::neighborhood< N > &Ng`)
Extract regional minima.
- `template<class I1, class I2, class N> oln::mute< I1 >::ret geodesic_dilation` (const `abstract::non_vectorial_image< I1 > &marker`, const `abstract::non_vectorial_image< I2 > &mask`, const `abstract::neighborhood< N > &Ng`)
Processing a geodesic dilation.
- `template<class I1, class I2, class N> oln::mute< I1 >::ret geodesic_erosion` (const `abstract::non_vectorial_image< I1 > &marker`, const `abstract::non_vectorial_image< I2 > &mask`, const `abstract::neighborhood< N > &Ng`)
Processing a geodesic erosion.
- `template<class I1, class I2, class N> oln::mute< I1 >::ret geodesic_reconstruction_dilation` (const `abstract::non_vectorial_image< I1 > &marker`, const `abstract::non_vectorial_image< I2 > &mask`, const `abstract::neighborhood< N > &Ng`)
Perform a geodesic reconstruction dilation.
- `template<class I1, class I2, class N> oln::mute< I1 >::ret geodesic_reconstruction_erosion` (const `abstract::non_vectorial_image< I1 > &marker`, const `abstract::non_vectorial_image< I2 > &mask`, const `abstract::neighborhood< N > &Ng`)
Perform a geodesic reconstruction erosion.

6.34.1 Detailed Description

Reference algorithm implementation: sure but slow.

6.34.2 Function Documentation

- 6.34.2.1** `template<class I1, class I2, class N> oln::mute< I1 >::ret geodesic_dilation` (const `abstract::non_vectorial_image< I1 > &marker`, const `abstract::non_vectorial_image< I2 > &mask`, const `abstract::neighborhood< N > &Ng`)

Processing a geodesic dilation.

Parameters:

I1 Exact type of image marker.

I2 Exact type of image mask.

N Exact type of neighborhood.

- marker Image to work on.
- mask Image used for geodesic dilation.
- Ng Neighborhood to use.

Compute the geodesic dilation of marker with respect to the mask image using se as structuring element. Soille p.156.

Precondition:

Mask must be greater or equal than marker.

Warning:

This version shouldn't be use, since it exists only to have a reference algorithm.

```
#include <oln/basics2d.hh>
#include <oln/morpho/opening.hh>
#include <oln/morpho/geodesic_dilation.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type      im1(oln::load(IMG_IN "lena128.pgm"));
    im_type      im2 = oln::morpho::opening(im1, oln::win_c4p());

    save(oln::morpho::sure::geodesic_dilation(im2, im1, oln::neighb_c4()),
        IMG_OUT "oln_morpho_sure_geodesic_dilation.pbm");

    return 0;
}
```



Definition at line 150 of file geodesic_dilation.hh.

References oln::abstract::image< Exact >::border_adapt_copy(), oln::abstract::neighborhood< Exact >::delta(), and oln::abstract::image< Exact >::size().

Referenced by geodesic_reconstruction_dilation().

```
153      {
154          mlc::eq<I1::dim, I2::dim>::ensure();
155          mlc::eq<I1::dim, N::dim>::ensure();
156          precondition(marker.size() == mask.size());
157          precondition(level::is_greater_or_equal(mask, marker));
```

```

158
159         oln_concrete_type(I1) output(marker.size());
160         marker.border_adapt_copy(Ng.delta());
161         oln_iter_type(I1) p(marker);
162         for_all (p)
163             output[p] = std::min(morpho::max(marker, p, convert::ng_to_cse(Ng)), mask[p]);
164         return output;
165     }

```

6.34.2.2 `template<class I1, class I2, class N> oln::mute< I1 >::ret geodesic_erosion (const abstract::non_vectorial_image< I1 > & marker, const abstract::non_vectorial_image< I2 > & mask, const abstract::neighborhood< N > & Ng)`

Processing a geodesic erosion.

Parameters:

I1 Exact type of image marker.

I2 Exact type of image mask.

N Exact type of neighborhood.

- marker Image to work on.
- mask Image used for geodesic dilation.
- Ng Neighborhood to use.

Compute the geodesic erosion of marker with respect to the mask mask image using se as structural element. Soille p.156. Computation is performed by hand (i.e without calling dilation).

Precondition:

Marker must be greater or equal than mask.

Warning:

This version shouldn't be use, since it exists only to have a reference algorithm.

```

#include <oln/basics2d.hh>
#include <oln/morpho/opening.hh>
#include <oln/morpho/geodesic_erosion.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena128.pgm"));
    im_type    im2 = oln::morpho::opening(im1, oln::win_c4p());

    save(oln::morpho::sure::geodesic_erosion(im1, im2, oln::neighb_c4()),
        IMG_OUT "oln_morpho_sure_geodesic_erosion.pbm");

    return 0;
}

```



Definition at line 148 of file geodesic_erosion.hh.

References `oln::abstract::neighborhood< Exact >::delta()`, and `oln::abstract::image< Exact >::size()`.

Referenced by `geodesic_reconstruction_erosion()`.

```

151     {
152         mlc::eq<I1::dim, I2::dim>::ensure();
153         mlc::eq<I1::dim, N::dim>::ensure();
154         precondition(marker.size() == mask.size());
155         precondition(level::is_greater_or_equal(marker, mask));
156
157         oln_concrete_type(I1) output(marker.size());
158         marker.border_adapt_copy(Ng.delta());
159         oln_iter_type(I1) p(marker);
160         for_all (p)
161             output[p] = ntg::max(morpho::min(marker, p, convert::ng_to_cse(Ng)), mask[p]);
162         return output;
163     }

```

6.34.2.3 `template<class I1, class I2, class N> oln::mute< I1 >::ret geodesic_reconstruction_dilation (const abstract::non_vectorial_image< I1 > & marker, const abstract::non_vectorial_image< I2 > & mask, const abstract::neighborhood< N > & Ng)`

Perform a geodesic reconstruction dilation.

Compute the reconstruction by dilation of marker with respect to the mask mask image using se as structuring element. Soille p.160. This is the simplest algorithm: iteration is performed until stability.

Warning:

This version is slow, since it is a sure one.

Precondition:

Mask must be greater or equal than marker.

Parameters:

I1 Exact type of image marker.

I2 Exact type of image mask.

N Exact type of neighborhood.

- marker Image to work on.
- mask Image used for geodesic dilation.
- Ng Neighborhood to use.

```

#include <oln/basics2d.hh>
#include <oln/morpho/opening.hh>
#include <oln/morpho/reconstruction.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type      im1(oln::load(IMG_IN "lena128.pgm"));
    im_type      im2 = oln::morpho::opening(im1, oln::win_c4p());

    oln::save(oln::morpho::sure::geodesic_reconstruction_dilation(im2,
                                                                    im1,
                                                                    oln::neighb_c4()),
              IMG_OUT "oln_morpho_sure_geodesic_reconstruction_dilation.pbm");
    return 0;
}

```



Definition at line 96 of file reconstruction.hh.

References `oln::abstract::image< Exact >::clone()`, `geodesic_dilation()`, and `oln::abstract::image< Exact >::size()`.

```

99      {
100          mlc::eq<I1::dim, I2::dim>::ensure();
101          mlc::eq<I1::dim, N::dim>::ensure();
102          precondition(marker.size() == mask.size());
103          precondition(level::is_greater_or_equal(mask, marker));
104          oln_concrete_type(I1) output = marker.clone();
105          bool non_stability = true;
106          while (non_stability)
107          {
108              oln_concrete_type(I1) work = geodesic_dilation(output, mask, Ng);
109              non_stability = !(level::is_equal(work, output));
110              output = work;
111          }
112          return output;
113      }

```

6.34.2.4 `template<class I1, class I2, class N> oln::mute< I1 >::ret geodesic_reconstruction_erosion (const abstract::non_vectorial_image< I1 > & marker, const abstract::non_vectorial_image< I2 > & mask, const abstract::neighborhood< N > & Ng)`

Perform a geodesic reconstruction erosion.

Compute the reconstruction by erosion of marker with respect to the mask image using se as structuring element. Soille p.160. This is the simplest algorithm: iteration is performed until stability.

Warning:

This version is slow, since it is a sure one.

Precondition:

Marker must be greater or equal than mask.

Parameters:

I1 Exact type of image marker.

I2 Exact type of image mask.

N Exact type of neighborhood.

- marker Image to work on.
- mask Image used for geodesic erosion.
- Ng Neighborhood to use.

```
#include <oln/basics2d.hh>
#include <oln/morpho/opening.hh>
#include <oln/morpho/reconstruction.hh>
#include <oln/level/compare.hh>
#include <ntg/all.hh>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type    im1(oln::load(IMG_IN "lena128.pgm"));
    im_type    im2 = oln::morpho::opening(im1, oln::win_c4p());

    oln::save(oln::morpho::sure::geodesic_reconstruction_erosion(im1,
                                                                    im2,
                                                                    oln::neighb_c4()),
              IMG_OUT "oln_morpho_sure_geodesic_reconstruction_erosion.pbm");
    return 0;
}
```



Definition at line 389 of file reconstruction.hh.

References `geodesic_erosion()`, and `oln::abstract::image< Exact >::size()`.

```
392    {
393        mlc::eq<I1::dim, I2::dim>::ensure();
394        mlc::eq<I1::dim, N::dim>::ensure();
395        precondition(marker.size() == mask.size());
396        precondition(level::is_greater_or_equal(marker, mask));
397        oln_concrete_type(I1) output = marker.clone();
398        bool non_stability = true;
```

```

399         while (non_stability)
400         {
401             oln_concrete_type(I1) work = geodesic_erosion(output, mask, Ng);
402             non_stability = !(level::is_equal(work, output));
403             output = work;
404         }
405         return output;
406     }

```

6.34.2.5 `template<class I, class I2, class N> oln::mute< I >::ret minima_imposition (const abstract::non_vectorial_image< I > & input, const abstract::non_vectorial_image< I2 > & minima_map, const abstract::neighborhood< N > & Ng)`

Perform a minima imposition.

Impose minima defined by minima_map on input using Ng as neighborhood. minima_map must be a bin image (true for a minimum, false for a non minimum). Soille p.172.

Parameters:

I Exact type of the first image.

I2 Exact type of the second image.

N Exact type of the neighborhood.

- *input* Input image.
- *minima_map* Minima map image.
- *Ng* Neighborhood to use.

```

#include <oln/basics2d.hh>
#include <oln/morpho/extrema.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;
    typedef oln::image2d<ntg::bin>       bin_im_type;

    im_type          light(oln::load(IMG_IN "lena.pgm"));
    bin_im_type      minima(oln::load(IMG_IN "map.pbm"));

    oln::save(oln::morpho::sequential::minima_imposition(light,
                                                         minima,
                                                         oln::neighb_c4()),
              IMG_OUT "oln_morpho_sequential_minima_imposition.pgm");
}

```




and



=>

Definition at line 165 of file extrema.hh.

6.34.2.6 `template<class I, class N> mute<I, ntg::bin>::ret regional_minima (const abstract::non_vectorial_image< I > & input, const abstract::neighborhood< N > & Ng)`

Extract regional minima.

Parameters:

I Exact type of input image.

Exact type of neighborhood.

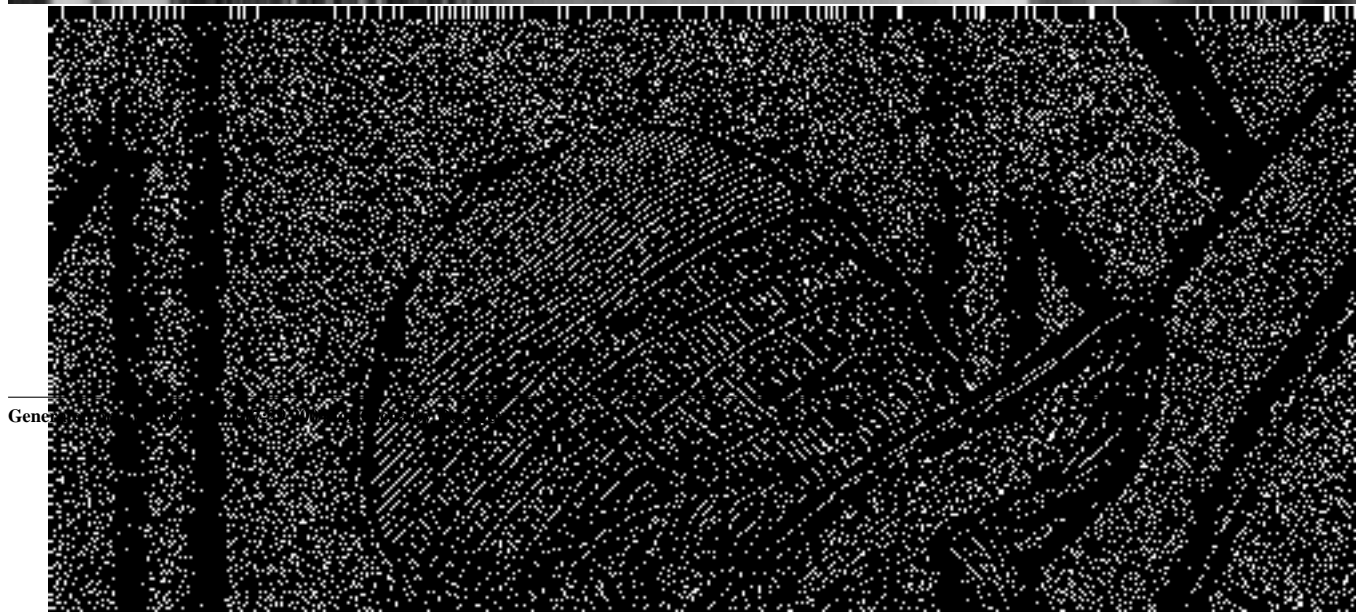
- input Input image.

- Ng Neighborhood to use.

```
#include <oln/basics2d.hh>
#include <oln/morpho/extrema.hh>
#include <ntg/all.hh>
#include <iostream>
int main()
{
    typedef oln::image2d<ntg::int_u8>    im_type;

    im_type                                im(oln::load(IMG_IN "lena.pgm"));

    oln::save(oln::morpho::sequential::regional_minima(im, oln::neighb_c4()),
              IMG_OUT "oln_morpho_sequential_regional_minima.pgm");
}
```



Definition at line 217 of file extrema.hh.

6.35 oln::snakes Namespace Reference

[oln::snakes::snake](#) implementation.

Classes

- class [energy](#)
- struct **energy::user_defined_external_energy_functor**
- class [continuity_energy](#)
- class [curvature_energy](#)
- class [image_energy](#)
- class [dummy_energy](#)

Default external energy.

- class [greedy](#)
- class [node](#)
- class [segment](#)
- class [snake](#)

6.35.1 Detailed Description

[oln::snakes::snake](#) implementation.

6.36 oln::topo Namespace Reference

Topological algorithms.

Classes

- struct [chamfer](#)
- class [dmap](#)

Functions

- template<int d10, int d11> const [chamfer](#)< int > & [mk_chamfer_3x3](#) (float coef=1.f)
- const [chamfer](#)< float > & [mk_chamfer_3x3](#) (float d10, float d11)
- template<int d10, int d11, int d21> const [chamfer](#)< int > & [mk_chamfer_5x5](#) (float coef=1.f)
- const [chamfer](#)< float > & [mk_chamfer_5x5](#) (float d10, float d11, float d21)
- const [chamfer](#)< int > & [chamfer_1_1](#) ()
- const [chamfer](#)< int > & [chamfer_1_2](#) ()
- const [chamfer](#)< int > & [chamfer_2_3](#) ()
- const [chamfer](#)< int > & [chamfer_5_7](#) ()
- const [chamfer](#)< int > & [chamfer_12_17](#) ()
- const [chamfer](#)< int > & [chessboard](#) ()
- const [chamfer](#)< int > & [cityblock](#) ()
- const [chamfer](#)< int > & [chamfer_4_6_9](#) ()
- const [chamfer](#)< int > & [chamfer_5_7_11](#) ()
- const [chamfer](#)< int > & [chamfer_9_13_20](#) ()
- const [chamfer](#)< int > & [chamfer_16_23_36](#) ()
- const [chamfer](#)< float > & [best_set_3x3](#) ()
- const [chamfer](#)< float > & [best_set_5x5](#) ()
- const [chamfer](#)< int > & [mchamfer_1_1](#) ()
- const [chamfer](#)< int > & [mchamfer_1_2](#) ()
- const [chamfer](#)< int > & [mchamfer_2_3](#) ()
- const [chamfer](#)< int > & [mchamfer_5_7](#) ()
- const [chamfer](#)< int > & [mchamfer_12_17](#) ()
- const [chamfer](#)< int > & [mchessboard](#) ()
- const [chamfer](#)< int > & [mcityblock](#) ()
- const [chamfer](#)< int > & [mchamfer_4_6_9](#) ()
- const [chamfer](#)< int > & [mchamfer_5_7_11](#) ()
- const [chamfer](#)< int > & [mchamfer_9_13_20](#) ()
- const [chamfer](#)< int > & [mchamfer_17_24_38](#) ()
- const [chamfer](#)< float > & [mbest_set_3x3](#) ()
- const [chamfer](#)< float > & [mbest_set_5x5](#) ()
- template<class I> [image2d](#)< float > [exact_dmap](#) (const [abstract::image](#)< I > &input)
Distance map using the Euclidean distance.
- float [euclidian_dist2](#) (const [point2d](#) &p1, const [point2d](#) &p2)

6.36.1 Detailed Description

Topological algorithms.

6.36.2 Function Documentation

6.36.2.1 const [chamfer](#)< float > & oln::topo::best_set_3x3 () [inline]

Best set 3x3

```

** Example of oln::topo::best_set_3x3() w;
** w.delta(): 1
** w.coef: 1
** w.fwd:
** 1.3408 0.9481 1.3408
** 0.9481 . .
** . . .
** w.bkd:
** . . .
** . . 0.9481
** 1.3408 0.9481 1.3408
**

```

Definition at line 170 of file dmap.hxx.

References [mk_chamfer_3x3\(\)](#).

```

171     { return mk_chamfer_3x3(0.9481, 1.3408); }

```

6.36.2.2 const [chamfer](#)< float > & oln::topo::best_set_5x5 () [inline]

Best set 5x5

```

** Example of oln::topo::best_set_5x5() w;
** w.delta(): 2
** w.coef: 1
** w.fwd:
** . 2.2044 . 2.2044 .
** 2.2044 1.406 0.9801 1.406 2.2044
** . 0.9801 . . .
** . . . . .
** . . . . .
** w.bkd:
** . . . . .
** . . . . .
** . . . 0.9801 .
** 2.2044 1.406 0.9801 1.406 2.2044
** . 2.2044 . 2.2044 .
**

```

Definition at line 172 of file dmap.hxx.

References [mk_chamfer_5x5\(\)](#).

```

173     { return mk_chamfer_5x5(0.9801, 1.4060, 2.2044); }

```

6.36.2.3 const [chamfer](#)< int > & oln::topo::chamfer_12_17 () [inline]

Chamfer_12_17

```

** Example of oln::topo::chamfer_12_17() w;
** w.delta(): 1
** w.coef: 12.6684
** w.fwd:
** 17 12 17
** 12 . .
** . . .
** w.bkd:
** . . .
** . . 12
** 17 12 17
**

```

Definition at line 151 of file dmap.hxx.

```
155 {
```

6.36.2.4 const [chamfer](#)< int > & oln::topo::chamfer_16_23_36() [inline]

Chamfer_16_23_36

```

** Example of oln::topo::chamfer_16_23_36() w;
** w.delta(): 2
** w.coef: 16.3351
** w.fwd:
** . 36 . 36 .
** 36 23 16 23 36
** . 16 . . .
** . . . . .
** . . . . .
** w.bkd:
** . . . . .
** . . . . .
** . . . 16 .
** 36 23 16 23 36
** . 36 . 36 .
**

```

Definition at line 168 of file dmap.hxx.

```
171 { return mk_chamfer_3x3(0.9481, 1.3408); }
```

6.36.2.5 const [chamfer](#)< int > & oln::topo::chamfer_1_1() [inline]

Chamfer_1_1

```

** Example of oln::topo::chamfer_1_1() w;
** w.delta(): 1
** w.coef: 0.9003
** w.fwd:
** 1 1 1
** 1 . .
** . . .
** w.bkd:
** . . .
** . . 1
** 1 1 1
**

```

Definition at line 147 of file dmap.hxx.

Referenced by chessboard().

```
155 {
```

6.36.2.6 const [chamfer](#)< int > & oln::topo::chamfer_1_2() [inline]

Chamfer_1_2

```

** Example of oln::topo::chamfer_1_2() w;
** w.delta(): 1
** w.coef: 1.2732
** w.fwd:
** 2 1 2
** 1 . .
** . . .
** w.bkd:
** . . .
** . . 1
** 2 1 2
**

```

Definition at line 148 of file dmap.hxx.

Referenced by cityblock().

```
155 {
```

6.36.2.7 const [chamfer](#)< int > & oln::topo::chamfer_2_3() [inline]

Chamfer_2_3

```

** Example of oln::topo::chamfer_2_3() w;
** w.delta(): 1
** w.coef: 2.1736
** w.fwd:
** 3 2 3
** 2 . .
** . . .
** w.bkd:
** . . .
** . . 2
** 3 2 3
**

```

Definition at line 149 of file dmap.hxx.

```
155 {
```

6.36.2.8 const [chamfer](#)< int > & oln::topo::chamfer_4_6_9() [inline]

Chamfer_4_6_9

```

** Example of oln::topo::chamfer_4_6_9() w;
** w.delta(): 2
** w.coef: 4.1203
** w.fwd:
**   . 9 . 9 .
**  9 6 4 6 9
**   . 4 . . .
**   . . . . .
**   . . . . .
** w.bkd:
**   . . . . .
**   . . . . .
**   . . . 4 .
**  9 6 4 6 9
**   . 9 . 9 .
**

```

Definition at line 165 of file dmap.hxx.

```

171 { return mk_chamfer_3x3(0.9481, 1.3408); }

```

6.36.2.9 const **chamfer**< int > & oln::topo::chamfer_5_7() [inline]

Chamfer_5_7

```

** Example of oln::topo::chamfer_5_7() w;
** w.delta(): 1
** w.coef: 5.2474
** w.fwd:
**  7 5 7
**  5 . .
**  . . .
** w.bkd:
**  . . .
**  . . 5
**  7 5 7
**

```

Definition at line 150 of file dmap.hxx.

```

155 {

```

6.36.2.10 const **chamfer**< int > & oln::topo::chamfer_5_7_11() [inline]

Chamfer_5_7_11

```

** Example of oln::topo::chamfer_5_7_11() w;
** w.delta(): 2
** w.coef: 5.0206
** w.fwd:
**   . 11 . 11 .
**  11 7 5 7 11
**   . 5 . . .
**   . . . . .
**   . . . . .
** w.bkd:
**   . . . . .
**   . . . . .

```

```

**      .      .      5      .
**    11  7  5  7  11
**      .  11      .  11      .
**

```

Definition at line 166 of file dmap.hxx.

```
171 { return mk_chamfer_3x3(0.9481, 1.3408); }
```

6.36.2.11 const [chamfer](#)< int > & oln::topo::chamfer_9_13_20 () [inline]

Chamfer_9_13_20

```

** Example of oln::topo::chamfer_9_13_20() w;
** w.delta(): 2
** w.coef: 9.1409
** w.fwd:
**      .      20      .      20      .
**    20  13  9  13  20
**      .      9      .      .      .
**      .      .      .      .      .
**      .      .      .      .      .
** w.bkd:
**      .      .      .      .      .
**      .      .      .      .      .
**      .      .      .      9      .
**    20  13  9  13  20
**      .      20      .      20      .
**

```

Definition at line 167 of file dmap.hxx.

```
171 { return mk_chamfer_3x3(0.9481, 1.3408); }
```

6.36.2.12 const [chamfer](#)< int > & oln::topo::chessboard () [inline]

Chessboard

```

** Example of oln::topo::chessboard() w;
** w.delta(): 1
** w.coef: 0.9003
** w.fwd:
**    1  1  1
**    1  .  .
**    .  .  .
** w.bkd:
**    .  .  .
**    .  .  1
**    1  1  1
**

```

Definition at line 154 of file dmap.hxx.

References [chamfer_1_1\(\)](#).

```

155      {
156          return chamfer_1_1();
157      }

```

6.36.2.13 `const chamfer< int > & oln::topo::cityblock ()` [inline]

Cityblock

```

** Example of oln::topo::cityblock() w;
** w.delta(): 1
** w.coef: 1.2732
** w.fwd:
** 2 1 2
** 1 . .
** . . .
** w.bkd:
** . . .
** . . 1
** 2 1 2
**

```

Definition at line 160 of file dmap.hxx.

References `chamfer_1_2()`.

```

161     {
162         return chamfer_1_2();
163     }

```

6.36.2.14 `const chamfer< float > & oln::topo::mk_chamfer_3x3 (float d10, float d11)` [inline]

Produce a chamfer mask 3x3

Todo

FIXME: This highly not thread safe !

```

** Example of oln::topo::mk_chamfer_3x3(1.5, 2.5) w;
** w.delta(): 1
** w.coef: 1
** w.fwd:
** 2.5 1.5 2.5
** 1.5 . .
** . . .
** w.bkd:
** . . .
** . . 1.5
** 2.5 1.5 2.5
**

```

Definition at line 79 of file dmap.hxx.

```

80         : add (?) , float coef = 1.f
81     {
82         static const w_window2d<float> w_win_fwd = ( mlc::floats_2d =
83                                                         d11, d10, d11, mlc::lbrk,
84                                                         d10, mlc::x(), 0.f, end );
85         static const w_window2d<float> w_win_bkd = ( mlc::floats_2d =
86                                                         0.f, mlc::x(), d10, mlc::lbrk,
87                                                         d11, d10, d11, end );
88         static const chamfer<float> ch_ =
89             chamfer<float>(w_win_fwd, w_win_bkd, 1.f);
90         return ch_;
91     }

```

6.36.2.15 `template<int d10, int d11> const chamfer< int > & oln::topo::mk_chamfer_3x3 (float coef = 1.f) [inline]`

Produce a chamfer mask 3x3

Todo

FIXME: This highly not thread safe !

```

** Example of oln::topo::mk_chamfer_3x3<1,2>(3) w;
** w.delta(): 1
** w.coef: 3
** w.fwd:
** 2 1 2
** 1 . .
** . . .
** w.bkd:
** . . .
** . . 1
** 2 1 2
**

```

Definition at line 65 of file dmap.hxx.

Referenced by `best_set_3x3()`.

```

66     {
67         static const w_window2d<int> w_win_fwd = ( mlc::ints_2d =
68             d11,      d10, d11, mlc::lbrk,
69             d10, mlc::x(), 0, end );
70         static const w_window2d<int> w_win_bkd = ( mlc::ints_2d =
71             0, mlc::x(), d10, mlc::lbrk,
72             d11,      d10, d11, end );
73         static const chamfer<int> ch_ = chamfer<int>(w_win_fwd, w_win_bkd, coef);
74         return ch_;
75     }

```

6.36.2.16 `const chamfer< float > & oln::topo::mk_chamfer_5x5 (float d10, float d11, float d21) [inline]`

Chamfer 5x5 using float

See also:

[mk_chamfer_5x5](#)

Definition at line 111 of file dmap.hxx.

```

112     {
113         const float O = 0.f;
114         static const w_window2d<float> w_win_fwd = ( mlc::floats_2d =
115             O, d21,      O, d21,  O, mlc::lbrk,
116             d21, d11,      d10, d11, d21,
117             O, d10, mlc::x(),  O,  O, end );
118         static const w_window2d<float> w_win_bkd = ( mlc::floats_2d =
119             O,      O, mlc::x(), d10,  O, mlc::lbrk,
120             d21, d11,      d10, d11, d21,
121             O, d21,      O, d21,  O, end );
122         static const chamfer<float> ch_ =
123             chamfer<float>(w_win_fwd, w_win_bkd, 1.f);
124         return ch_;
125     }

```

6.36.2.17 `template<int d10, int d11, int d21> const chamfer<int> & oln::topo::mk_chamfer_5x5 (float coef = 1.f) [inline]`

Chamfer 5x5

```

** Example of oln::topo::mk_chamfer_5x5<1, 2, 3>(4) w;
** w.delta(): 2
** w.coef: 4
** w.fwd:
**   . 3 . 3 .
**  3 2 1 2 3
**   . 1 . . .
**   . . . . .
**   . . . . .
** w.bkd:
**   . . . . .
**   . . . . .
**   . . . 1 .
**  3 2 1 2 3
**   . 3 . 3 .
**

```

Definition at line 95 of file dmap.hxx.

Referenced by `best_set_5x5()`.

```

96     {
97         static const w_window2d<int> w_win_fwd = ( mlc::ints_2d =
98             0, d21,          0, d21,    0, mlc::lbrk,
99             d21, d11,          d10, d11, d21,
100             0, d10, mlc::x(),    0,    0, end );
101         static const w_window2d<int> w_win_bkd = ( mlc::ints_2d =
102             0,    0, mlc::x(), d10,    0, mlc::lbrk,
103             d21, d11,          d10, d11, d21,
104             0, d21,          0, d21,    0, end );
105         static const chamfer<int> ch_ = chamfer<int>(w_win_fwd, w_win_bkd, coef);
106         return ch_;
107     }

```


6.37 oln::topo::combinatorial_map Namespace Reference

Namespace for combinatorial map.

Classes

- class [cmap](#)

6.37.1 Detailed Description

Namespace for combinatorial map.

6.38 oln::topo::combinatorial_map::internal Namespace Reference

oln::combinatorial_map::internal

Classes

- struct [alpha](#)
- class [any](#)
- class [anyfunc](#)
- class [beta](#)

This function must be built using assign.

- class **lambda**
- struct **node**
- class **level**
- class **sigma**
- struct **zeta**

6.38.1 Detailed Description

oln::combinatorial_map::internal

6.39 oln::topo::inter_pixel Namespace Reference

[oln::topo::inter_pixel::interpixel](#) implementation.

Classes

- class [bkd_dir_iter](#)
Backward iterator on direction.
- class [fwd_dir_iter](#)
Backward iterator on direction.
- class [interpixel](#)
- class [node](#)

6.39.1 Detailed Description

[oln::topo::inter_pixel::interpixel](#) implementation.

6.40 oln::topo::inter_pixel::internal Namespace Reference

Internal purpose only.

Classes

- class `dir_iter_`
- struct `dir_traits`
Provides the enum dir.
- struct `dir_traits< 2 >`
Provides the enum dir for 2D.

6.40.1 Detailed Description

Internal purpose only.

6.41 oln::topo::tarjan Namespace Reference

Implementation of tarjan set.

Classes

- struct [flat_zone](#)
- struct [tarjan_traits](#)< [flat_zone](#)< T, DestType, A, Exact > >
Traits specialization for [flat_zone](#).
- struct **empty_class**
- struct [tarjan_set](#)

6.41.1 Detailed Description

Implementation of tarjan set.

6.42 oln::topo::tarjan::abstract Namespace Reference

Abstract classes for tarjan based algorithms.

Classes

- struct [tarjan](#)
Top of tarjan hierarchy.
- struct [tarjan_with_attr](#)
Abstract class to perform a tarjan algorithm on an image with attribute computing .

6.42.1 Detailed Description

Abstract classes for tarjan based algorithms.

6.43 oln::topo::tarjan::obsolete Namespace Reference

Namespace containing obsolete implementation of flat zone.

Classes

- struct [flat_zone](#)

6.43.1 Detailed Description

Namespace containing obsolete implementation of flat zone.

6.44 oln::transforms Namespace Reference

Transform algorithm implementation.

Classes

- class [dwt](#)
Object to compute dwt transforms.
- struct [haar](#)
Haar wavelet coefficients.
- struct [daub4](#)
Daubechies wavelet coefficients.
- struct [daub6](#)
Daubechies wavelet coefficients.
- struct [daub8](#)
Daubechies wavelet coefficients.
- struct [daub10](#)
Daubechies wavelet coefficients.
- struct [daub12](#)
Daubechies wavelet coefficients.
- struct [daub20](#)
Daubechies wavelet coefficients.
- struct [coiflet2](#)
Coifman wavelet coefficients.
- struct [coiflet4](#)
Coifman wavelet coefficients.
- struct [coiflet6](#)
Coifman wavelet coefficients.

Enumerations

- enum [dwt_transform_type](#) { [dwt_std](#), [dwt_non_std](#) }
type of dwt to perform.

6.44.1 Detailed Description

Transform algorithm implementation.

6.45 oln::utils Namespace Reference

Utilities, such as statistics.

Classes

- class [buffer](#)
Buffer used for [MD5](#) data type abstraction.
- struct [hist_traits](#)
Traits for [oln::utils::abstract::histogram](#) hierarchy.
- struct [hist_traits](#)< [histogram](#)< T, CPT, V2P, Exact > >
Traits for [oln::utils::abstract::histogram](#) hierarchy.
- class [histogram](#)
- struct [hist_traits](#)< [histogram_minmax](#)< T, CPT, V2P, Exact > >
- class [histogram_minmax](#)
- struct [hist_traits](#)< [histogram_min](#)< T, CPT, V2P, Exact > >
- class [histogram_min](#)
- struct [hist_traits](#)< [histogram_max](#)< T, CPT, V2P, Exact > >
- class [histogram_max](#)
- struct [select_distrib_sort](#)
- struct [select_distrib_sort](#)< true >
- class [key](#)
16 bytes key
- class [MD5](#)
Class used to compute a [MD5](#) digest.
- class [se_stat](#)
Compute the variance and the mean within a window.
- struct [f_minmax](#)
Unary function that stores the min and the max.
- struct [f_moments](#)
Computes the mean, the variance and store the min, the max.
- class [timer](#)

Functions

- template<class T> [mlc::exact](#)< T >::ret::value_type [min](#) (const [abstract::histogram](#)< T > &hist)
- template<class T> [mlc::exact](#)< T >::ret::value_type [max](#) (const [abstract::histogram](#)< T > &hist)
- template<typename T, typename CPT, class V2P, class Exact> T [min](#) ([histogram_minmax](#)< T, CPT, V2P, Exact > &hist)
Minimum non-zero value of an histogram.

- `template<typename T, typename CPT, class V2P, class Exact> T min (histogram_min< T, CPT, V2P, Exact > &hist)`
Minimum non-zero value of an histogram.
- `template<typename T, typename CPT, class V2P, class Exact> T max (histogram_minmax< T, CPT, V2P, Exact > &hist)`
Maximum non-zero value of an histogram.
- `template<typename T, typename CPT, class V2P, class Exact> T max (histogram_max< T, CPT, V2P, Exact > &hist)`
Maximum non-zero value of an histogram.
- `template<class I> void distrib_sort (const oln::abstract::image< I > &im, std::vector< typename mlc::exact< I >::ret::point_type > &v)`
- `template<class I> void distrib_sort_inv (const oln::abstract::image< I > &im, std::vector< typename mlc::exact< I >::ret::point_type > &v)`
- `template<class I> key md5 (const oln::abstract::non_vectorial_image< I > &im)`
Compute The Md5 value of an image.
- `template<class I> key md5 (const oln::abstract::vectorial_image< I > &im)`
Compute The Md5 value of an image.

6.45.1 Detailed Description

Utilities, such as statistics.

6.45.2 Function Documentation

6.45.2.1 `template<class I> void distrib_sort (const oln::abstract::image< I > &im, std::vector< typename mlc::exact< I >::ret::point_type > &v)`

Sort the values of an image, and store the result in a vector

This sort is efficient.

- im Image non_vectorial.
- v sorted vector at the end of the function.

Precondition:

`precondition(v.size() == im.npoints());`
A histogram of the image has to be possible.

Definition at line 676 of file histogram.hh.

References `oln::abstract::image< Exact >::npoints()`.

```

678     {
679         typedef oln_value_type(I) val;
680
681         typedef typename ntg_is_a(val, ntg::non_vectorial)::ensure_type
682                                     ensure_type;
```

```

683
684     // check the size
685     precondition(v.size() == im.npoints());
686
687     // calculate the histogram of the image
688     utils::histogram<val> histo(im);
689
690     // Initialize the array of pointer to the point in the result.
691     // With the histogram the number of each color can be deduced and
692     // then it calculates an array of pointer for quick access to each
693     // value of the image.
694     const ntg_cumul_type(val) card = ntg_max_val(val)
695                                     - ntg_min_val(val) + 1;
696     std::vector<oln_point_type(I)* > ptr(card);
697     ptr[0] = &(v[0]);
698     for (ntg_cumul_type(val) i = 1; i < card; ++i)
699         ptr[i] = ptr[i - 1] + histo[i - 1 + ntg_min_val(val)];
700
701     // Now iterate on the image to sort point in the order of their
702     // level
703     oln_iter_type(I) p(im);
704     for_all(p)
705         *(ptr[unsigned(im[p] - ntg_min_val(val))]++) = p;
706 }

```

6.45.2.2 `template<class I> void distrib_sort_inv (const oln::abstract::image< I > & im, std::vector< typename mlc::exact< I >::ret::point_type > & v)`

Inverted sort of the values of an image, and store the result in a vector

This sort is efficient.

- im Image non_vectorial.
- v sorted vector at the end of the function.

Precondition:

precondition(v.size() == im.npoints());
A histogram of the image has to be possible.

Definition at line 719 of file histogram.hh.

References oln::abstract::image< Exact >::npoints().

```

721     {
722         typedef oln_value_type(I) val;
723
724         typedef typename ntg_is_a(val, ntg::non_vectorial)::ensure_type
725                                     ensure_type;
726
727         precondition(v.size() == im.npoints());
728
729         utils::histogram<val> histo(im);
730
731         const ntg_cumul_type(val) card = ntg_max_val(val)
732                                             - ntg_min_val(val) + 1;
733         std::vector<oln_point_type(I)* > ptr(card);
734         ptr[card - 1] = &(v[0]);
735
736         for (ntg_signed_cumul_type(val) i = card - 2; i >= 0; --i)
737             ptr[i] = ptr[i + 1] + histo[i + 1 + ntg_min_val(val)];
738
739         oln_iter_type(I) p(im);

```

```

740     for_all(p)
741         *(ptr[unsigned(im[p] - ntg_min_val(val))])++ = p;
742     }

```

6.45.2.3 `template<class T> mlc::exact< T >::ret::value_type max (const abstract::histogram< T > & hist) [inline]`

Maximum value of an histogram.

Return the higher value within the image used to build the histogram.

Note:

It can be slow when the histogram is sparse because it iterates over a large range of 0. Use [histogram_max](#) or [histogram_minmax](#) instead.

See also:

[histogram_max](#)

Definition at line 330 of file histogram.hh.

```

331     {
332         typedef typename ntg_is_a(oln_value_type(T),
333                                   ntg::non_vectorial)::ensure_type ensure_type;
334
335         oln_value_type(T) i;
336         for (i = ntg_max_val(oln_value_type(T));
337              i != ntg_min_val(oln_value_type(T));
338              i = ntg::pred(i))
339             if (hist[i] > ntg_zero_val(oln_cpt_type(T)))
340                 break;
341         return i;
342     }

```

6.45.2.4 `template<class I> key oln::utils::md5 (const oln::abstract::vectorial_image< I > & im) [inline]`

Compute The Md5 value of an image.

Parameters:

I Exact type of the image.

- im Image to process.

Vectorial image version.

Definition at line 507 of file md5.hh.

6.45.2.5 `template<class I> key oln::utils::md5 (const oln::abstract::non_vectorial_image< I > & im) [inline]`

Compute The Md5 value of an image.

Parameters:

I Exact type of the image.

- im Image to process.

Non vectorial image version.

Definition at line 490 of file md5.hh.

6.45.2.6 `template<class T> mlc::exact< T >::ret::value_type min (const abstract::histogram< T > & hist) [inline]`

Minimum value of an histogram.

Return the smaller value within the image used to build the histogram.

Note:

It can be slow when the histogram is sparse because it iterates over a large range of 0. Use [histogram_min](#) or [histogram_minmax](#) instead.

See also:

[histogram_min](#)

Definition at line 305 of file histogram.hh.

```
306     {
307         typedef typename ntg_is_a(oln_value_type(T),
308                                   ntg::non_vectorial)::ensure_type ensure_type;
309
310         oln_value_type(T) i;
311         for (i = ntg_min_val(oln_value_type(T));
312              i != ntg_max_val(oln_value_type(T));
313              i = ntg::succ(i))
314             if (hist[i] > ntg_zero_val(oln_cpt_type(T)))
315                 break;
316         return i;
317     }
```

6.46 oln::utils::abstract Namespace Reference

Abstract utilities.

Classes

- class [histogram](#)

6.46.1 Detailed Description

Abstract utilities.

6.47 oln::utils::internal Namespace Reference

Internal purpose only.

Classes

- struct [img_max_size](#)
Return the size of the space needed to explore the type T.
- struct **img_max_size**< **ntg::color**< 3, Qbits, S > >
- struct [f_to_float_](#)
- struct [f_to_float_](#)< typename **ntg::color**< ncomps, qbits, color_system >::float_vec_type, **ntg::color**< ncomps, qbits, color_system > >
Specialization of the f_to_float struct for the colors.

Functions

- template<class P> P [mk_point_looking_like](#) (coord value)

6.47.1 Detailed Description

Internal purpose only.

6.47.2 Function Documentation

6.47.2.1 template<class P> P [mk_point_looking_like](#) (coord value)

Creates a point that is used as a state.

This function should be called with a value unreachable for example a negative one.

Definition at line 45 of file `special_points.hh`.

References `oln::coord`.

```

46     {
47         P p;
48         p.nth(0) = value;
49         return p;
50     }
```


Chapter 7

Olena Class Documentation

7.1 `ntg::internal::_from_float< n, ncomps, qbits, color_system >` Struct Template Reference

```
#include <color.hh>
```

Public Types

- `typedef int_u< qbits > T`
- `typedef vec< ncomps, float > in_type`
- `typedef vec< ncomps, T > out_type`

Static Public Member Functions

- `void doit (const in_type &in, out_type &out)`

7.1.1 Detailed Description

```
template<unsigned n, unsigned ncomps, unsigned qbits, template< unsigned > class color_system>  
struct ntg::internal::_from_float< n, ncomps, qbits, color_system >
```

Helper struct to convert `vec<N,float>` to `vec<N,T>`, taking `color_system` into account.

Definition at line 124 of file `color/color.hh`.

The documentation for this struct was generated from the following file:

- `color/color.hh`

7.2 ntg::internal::_to_float< n, ncomps, qbits, color_system > Struct Template Reference

```
#include <color.hh>
```

Public Types

- typedef int_u< qbits > **T**
- typedef vec< ncomps, T > **in_type**
- typedef vec< ncomps, float > **out_type**

Static Public Member Functions

- void **doit** (const in_type &in, out_type &out)

7.2.1 Detailed Description

template<unsigned n, unsigned ncomps, unsigned qbits, template< unsigned > class color_system>
struct ntg::internal::_to_float< n, ncomps, qbits, color_system >

Helper struct to convert vec<N,T> to vec<N,float>, taking color_system into account.

Definition at line 81 of file color/color.hh.

The documentation for this struct was generated from the following file:

- color/color.hh

7.3 oln::topo::combinatorial_map::internal::alpha< U > Struct Template Reference

```
#include <alpha.hh>
```

Static Public Member Functions

- U [result](#) (const U &d)
*Returns $(d + ((d \ 2) * 2) - 1)$.*

7.3.1 Detailed Description

template<class U> struct oln::topo::combinatorial_map::internal::alpha< U >

Alpha function.

Returns $(d + ((d \ 2) * 2) - 1)$.

Definition at line 43 of file alpha.hh.

The documentation for this struct was generated from the following file:

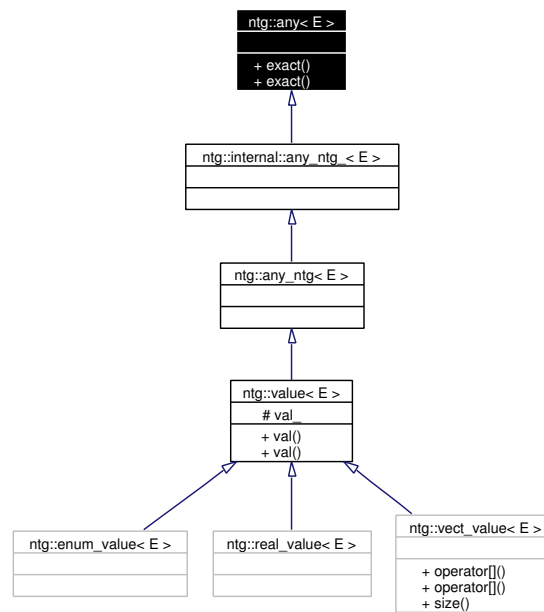
- alpha.hh

7.4 ntg::any< E > Class Template Reference

Top of static hierarchy.

```
#include <type.hh>
```

Inheritance diagram for ntg::any< E >:



Public Types

- typedef E **exact_type**

Public Member Functions

- E & **exact** ()
- const E & **exact** () const

7.4.1 Detailed Description

```
template<class E> class ntg::any< E >
```

Top of static hierarchy.

Definition at line 47 of file `integre/ntg/core/type.hh`.

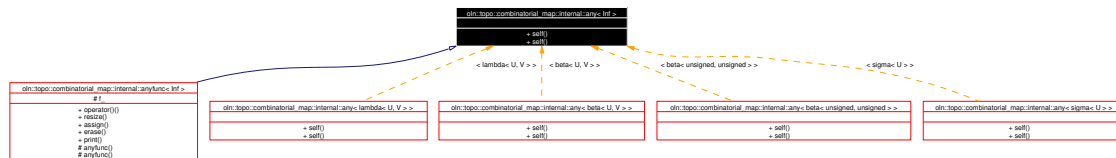
The documentation for this class was generated from the following file:

- `integre/ntg/core/type.hh`

7.5 oln::topo::combinatorial_map::internal::any< Inf > Class Template Reference

```
#include <anyfunc.hh>
```

Inheritance diagram for oln::topo::combinatorial_map::internal::any< Inf >:



Public Member Functions

- `const Inf & self () const`
- `Inf & self ()`

7.5.1 Detailed Description

`template<class Inf> class oln::topo::combinatorial_map::internal::any< Inf >`

any

Todo

FIXME: totally obsolete.

Definition at line 48 of file `anyfunc.hh`.

The documentation for this class was generated from the following file:

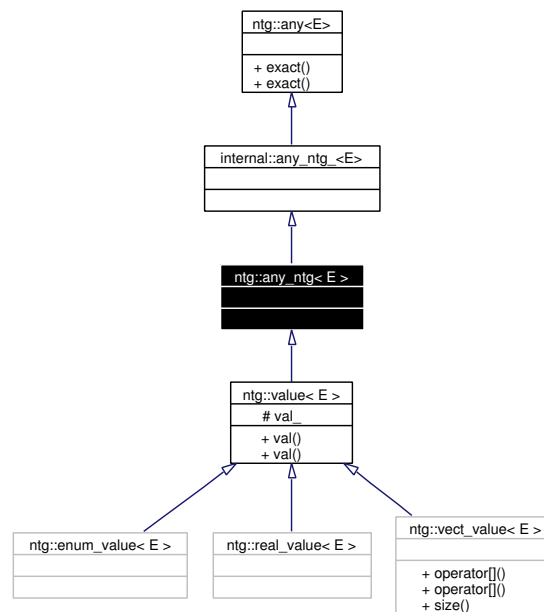
- `anyfunc.hh`

7.6 ntg::any_ntg< E > Class Template Reference

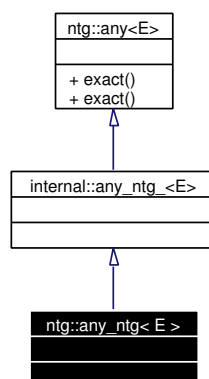
Top of the ntg types hierarchy.

```
#include <value.hh>
```

Inheritance diagram for ntg::any_ntg< E >:



Collaboration diagram for ntg::any_ntg< E >:



7.6.1 Detailed Description

```
template<class E> class ntg::any_ntg< E >
```

Top of the ntg types hierarchy.

Just an abstraction.

Definition at line 76 of file value.hh.

The documentation for this class was generated from the following file:

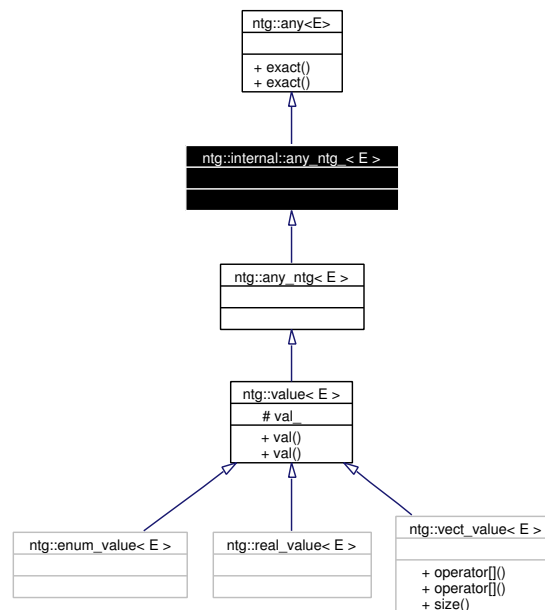
- value.hh

7.7 ntg::internal::any_ntg_< E > Class Template Reference

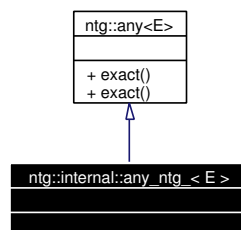
Bridge to internal for ntg type hierarchy.

```
#include <value.hh>
```

Inheritance diagram for ntg::internal::any_ntg_< E >:



Collaboration diagram for ntg::internal::any_ntg_< E >:



7.7.1 Detailed Description

```
template<class E> class ntg::internal::any_ntg_< E >
```

Bridge to internal for ntg type hierarchy.

This class is a bridge with the internal namespace. Indeed, global operators (such as `<T,U> operator+(T, U)`) are defined in the internal namespace to allow a "using namespace ntg" directive without breaking external classes interactions.

However, ntg types need to see those operators. This is done by inheriting from `internal::any_ntg_`, thanks to Koenig lookup.

See comments in <ntg/config/system.hh>.

Definition at line 62 of file value.hh.

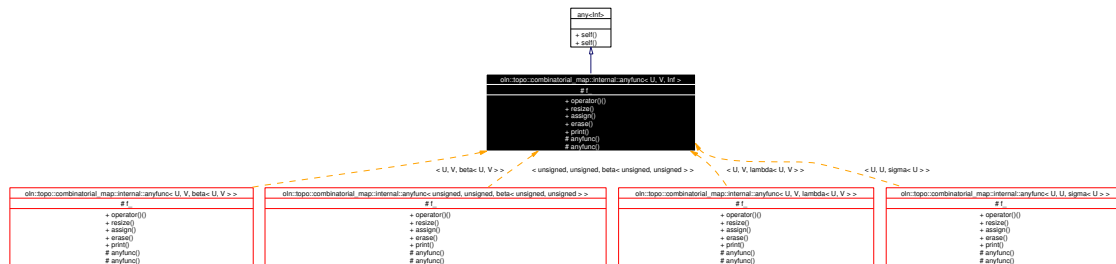
The documentation for this class was generated from the following file:

- value.hh

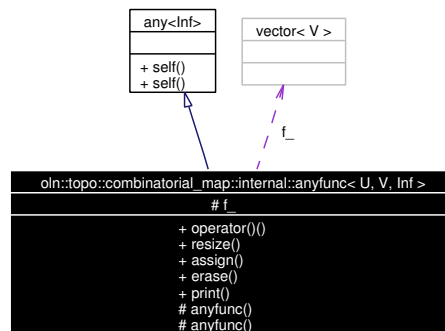
7.8 oln::topo::combinatorial_map::internal::anyfunc< U, V, Inf > Class Template Reference

```
#include <anyfunc.hh>
```

Inheritance diagram for oln::topo::combinatorial_map::internal::anyfunc< U, V, Inf >:



Collaboration diagram for oln::topo::combinatorial_map::internal::anyfunc< U, V, Inf >:



Public Member Functions

- `V operator()` (const U &e) const
Retrieve the value $f(e)$.
- void `resize` (unsigned n)
Resize the domain of f .
- void `assign` (const U &i, const V &e)
Assign a value to $f(i)$.
- void `erase` (const U &i)
 $f(i) = 0$.
- std::ostream & `print` (std::ostream &ostr) const
Print the function.

Protected Member Functions

- [anyfunc](#) (unsigned n)
Construct a function on [0..n].

Protected Attributes

- std::vector< V > f_

7.8.1 Detailed Description

`template<class U, class V, class Inf> class oln::topo::combinatorial_map::internal::anyfunc< U, V, Inf >`

Function stored in a vector.

[Todo](#)

FIXME: It has nothing to do there.

Definition at line 61 of file anyfunc.hh.

The documentation for this class was generated from the following file:

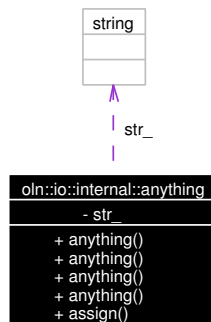
- anyfunc.hh

7.9 oln::io::internal::anything Class Reference

Anything.

```
#include <readable.hh>
```

Collaboration diagram for oln::io::internal::anything:



Public Member Functions

- [anything](#) ()

Constructor.

- [anything](#) (const [anything](#) &rhs)

Constructor.

- [anything](#) (const std::string &str)

Constructor.

- [anything](#) (const char *c)

Constructor.

- template<typename T> T & [assign](#) (T &output) const

This function will be called when applied to an operator = and load the file (str_ is the filename).

7.9.1 Detailed Description

Anything.

This class is called by oln::load and just keep the filename to load. As soon as you will use the operator = on it, assign will be called and it will read the file. If you would like some examples to know how to use that, go to [oln::abstract::iter](#)

Definition at line 55 of file readable.hh.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 oln::io::internal::anything::anything () [inline]

Constructor.

Todo

FIXME: these constructors are required by swig

Definition at line 62 of file readable.hh.

```
62 : str_() {}
```

7.9.3 Member Function Documentation

7.9.3.1 template<typename T> T& oln::io::internal::anything::assign (T & *output*) const [inline]

This function will be called when applied to an operator = and load the file (str_ is the filename).

<

Todo

FIXME: call output.clear()?

Definition at line 85 of file readable.hh.

Referenced by oln::image1d< T, Exact >::image1d(), oln::image2d< std::vector< oln::point2d > >::image2d(), oln::image3d< T, Exact >::image3d(), oln::image3d< T, Exact >::operator=(), oln::image2d< std::vector< oln::point2d > >::operator=(), and oln::image1d< T, Exact >::operator=().

```
86         {
87             read_any(output, str_);
89             return output;
90         }
```

The documentation for this class was generated from the following file:

- readable.hh

7.10 oln::morpho::attr::attr_traits< ball_parent_change< I, Exact > > Struct Template Reference

Trait specialization for the ball_parent_change attribute.

```
#include <attributes.hh>
```

Public Types

- typedef unsigned [value_type](#)
Type of data.
- typedef [value_type](#) [lambda_type](#)
Type of lambda.
- typedef [env::ParentEnv](#)< I > [env_type](#)
Type of environment.

7.10.1 Detailed Description

```
template<class I, class Exact> struct oln::morpho::attr::attr_traits< ball_parent_change< I, Exact > >
```

Trait specialization for the ball_parent_change attribute.

Definition at line 1890 of file attributes.hh.

The documentation for this struct was generated from the following file:

- attributes.hh

7.11 oln::morpho::attr::attr_traits< ball_type< I, Exact > > Struct Template Reference

Trait specialization for the ball attribute.

```
#include <attributes.hh>
```

Public Types

- typedef unsigned [value_type](#)
Type of data.
- typedef [value_type](#) [lambda_type](#)
Type of lambda.
- typedef [oln::morpho::env::NullEnv](#) [env_type](#)
Type of environment.

7.11.1 Detailed Description

```
template<class I, class Exact> struct oln::morpho::attr::attr_traits< ball_type< I, Exact > >
```

Trait specialization for the ball attribute.

Definition at line 1831 of file attributes.hh.

The documentation for this struct was generated from the following file:

- attributes.hh

7.12 oln::morpho::attr::attr_traits< box_type< I, Exact > > Struct Template Reference

Trait specialization for the box attribute.

```
#include <attributes.hh>
```

Public Types

- typedef unsigned [value_type](#)
Type of data.
- typedef ntg::vec< I::dim, [value_type](#), mlc::final > [lambda_type](#)
Type of lambda.
- typedef [oln::morpho::env::NullEnv](#) [env_type](#)
Type of environment.

7.12.1 Detailed Description

```
template<class I, class Exact> struct oln::morpho::attr::attr_traits< box_type< I, Exact > >
```

Trait specialization for the box attribute.

Definition at line 1864 of file attributes.hh.

The documentation for this struct was generated from the following file:

- attributes.hh

7.13 oln::morpho::attr::attr_traits< card_full_type< I, T, Exact > > Struct Template Reference

Trait specialization for card_full attribute.

```
#include <attributes.hh>
```

Public Types

- typedef T [value_type](#)
Type of data.
- typedef [value_type](#) [lambda_type](#)
Type of lambda.
- typedef [env::OtherImageEnv](#)< I > [env_type](#)
Type of environment.

7.13.1 Detailed Description

```
template<class I, class T, class Exact> struct oln::morpho::attr::attr_traits< card_full_type< I, T, Exact > >
```

Trait specialization for card_full attribute.

Definition at line 1798 of file attributes.hh.

The documentation for this struct was generated from the following file:

- attributes.hh

7.14 oln::morpho::attr::attr_traits< card_type< T, Exact > > Struct Template Reference

Trait specialization for card attribute.

```
#include <attributes.hh>
```

Public Types

- typedef T [value_type](#)
Type of data.
- typedef [value_type](#) [lambda_type](#)
Type of lambda.
- typedef [env::NullEnv](#) [env_type](#)
Type of environment.

7.14.1 Detailed Description

```
template<class T, class Exact> struct oln::morpho::attr::attr_traits< card_type< T, Exact > >
```

Trait specialization for card attribute.

Definition at line 1787 of file attributes.hh.

The documentation for this struct was generated from the following file:

- attributes.hh

7.15 oln::morpho::attr::attr_traits< cube_type< I, Exact > > Struct Template Reference

Trait specialization for the cube attribute.

```
#include <attributes.hh>
```

Public Types

- typedef unsigned [value_type](#)
Type of data.
- typedef [value_type](#) [lambda_type](#)
Type of lambda.
- typedef [oln::morpho::env::NullEnv](#) [env_type](#)
Type of environment.

7.15.1 Detailed Description

```
template<class I, class Exact> struct oln::morpho::attr::attr_traits< cube_type< I, Exact > >
```

Trait specialization for the cube attribute.

Definition at line 1853 of file attributes.hh.

The documentation for this struct was generated from the following file:

- attributes.hh

7.16 oln::morpho::attr::attr_traits< dist_type< I, Exact > > Struct Template Reference

Trait specialization for the dist attribute.

```
#include <attributes.hh>
```

Public Types

- typedef float [value_type](#)
Type of data.
- typedef [value_type](#) [lambda_type](#)
Type of lambda.
- typedef [oln::morpho::env::NullEnv](#) [env_type](#)
Type of environment.

7.16.1 Detailed Description

```
template<class I, class Exact> struct oln::morpho::attr::attr_traits< dist_type< I, Exact > >
```

Trait specialization for the dist attribute.

Definition at line 1842 of file attributes.hh.

The documentation for this struct was generated from the following file:

- attributes.hh

7.17 oln::morpho::attr::attr_traits< height_type< T, Exact > > Struct Template Reference

Trait specialization for the height attribute.

```
#include <attributes.hh>
```

Public Types

- typedef T [value_type](#)
Type of data.
- typedef [value_type](#) [lambda_type](#)
Type of lambda.
- typedef [env::NullEnv](#) [env_type](#)
Type of environment.

7.17.1 Detailed Description

```
template<class T, class Exact> struct oln::morpho::attr::attr_traits< height_type< T, Exact > >
```

Trait specialization for the height attribute.

Definition at line 1776 of file attributes.hh.

The documentation for this struct was generated from the following file:

- attributes.hh

7.18 oln::morpho::attr::attr_traits< integral_type< T, Exact > > Struct Template Reference

Trait specialization for the integral attribute.

```
#include <attributes.hh>
```

Public Types

- typedef T [value_type](#)
Type of data.
- typedef [value_type](#) [lambda_type](#)
Type of lambda.
- typedef [env::NullEnv](#) [env_type](#)
Type of environment.

7.18.1 Detailed Description

```
template<class T, class Exact> struct oln::morpho::attr::attr_traits< integral_type< T, Exact > >
```

Trait specialization for the integral attribute.

Definition at line 1765 of file attributes.hh.

The documentation for this struct was generated from the following file:

- attributes.hh

7.19 oln::morpho::attr::attr_traits< maxvalue_type< T, Exact > > Struct Template Reference

Trait specialization for the maxvalue attribute.

```
#include <attributes.hh>
```

Public Types

- typedef T [value_type](#)
Type of data.
- typedef [value_type](#) [lambda_type](#)
Type of lambda.
- typedef [env::NullEnv](#) [env_type](#)
Type of environment.

7.19.1 Detailed Description

```
template<class T, class Exact> struct oln::morpho::attr::attr_traits< maxvalue_type< T, Exact >  
>
```

Trait specialization for the maxvalue attribute.

Definition at line 1809 of file attributes.hh.

The documentation for this struct was generated from the following file:

- attributes.hh

7.20 oln::morpho::attr::attr_traits< minvalue_type< T, Exact > > Struct Template Reference

Trait specialization for the minvalue attribute.

```
#include <attributes.hh>
```

Public Types

- typedef T [value_type](#)
Type of data.
- typedef [value_type](#) [lambda_type](#)
Type of lambda.
- typedef [env::NullEnv](#) [env_type](#)
Type of environment.

7.20.1 Detailed Description

```
template<class T, class Exact> struct oln::morpho::attr::attr_traits< minvalue_type< T, Exact >  
>
```

Trait specialization for the minvalue attribute.

Definition at line 1820 of file attributes.hh.

The documentation for this struct was generated from the following file:

- attributes.hh

7.21 oln::morpho::attr::attr_traits< other_image< Dad, I, Exact > > Struct Template Reference

Trait specialization for the [other_image](#) attribute.

```
#include <attributes.hh>
```

Public Types

- typedef change_exact< Dad, typename mlc::exact_vt< [other_image](#)< Dad, I, Exact >, Exact >::ret >::ret [super_type](#)
Parent class type.
- typedef oln::morpho::attr::attr_traits< [super_type](#) >::value_type [value_type](#)
Type of data.
- typedef oln::morpho::attr::attr_traits< [super_type](#) >::lambda_type [lambda_type](#)
Type of lambda.
- typedef [oln::morpho::env::OtherImageEnv](#)< I > [env_type](#)
Type of environment.

7.21.1 Detailed Description

```
template<class Dad, class I, class Exact> struct oln::morpho::attr::attr_traits< other_image< Dad, I, Exact > >
```

Trait specialization for the [other_image](#) attribute.

Definition at line 1875 of file attributes.hh.

The documentation for this struct was generated from the following file:

- attributes.hh

7.22 oln::morpho::attr::attribute< Exact > Class Template Reference

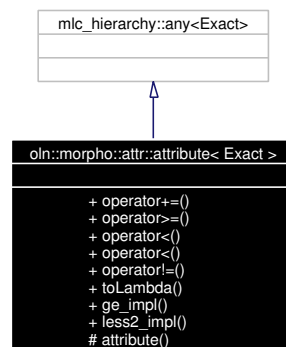
Attribute abstract class.

```
#include <attributes.hh>
```

Inheritance diagram for oln::morpho::attr::attribute< Exact >:



Collaboration diagram for oln::morpho::attr::attribute< Exact >:



Public Types

- typedef [attribute](#)< Exact > **self_type**
- typedef mlc::exact_vt< self_type, Exact >::ret **exact_type**
- typedef oln::morpho::attr::attr_traits< exact_type >::value_type **value_type**
- typedef oln::morpho::attr::attr_traits< exact_type >::env_type **env_type**
- typedef oln::morpho::attr::attr_traits< exact_type >::lambda_type **lambda_type**

Public Member Functions

- void [operator+=](#) (const exact_type &rhs)
+= *operator*
- bool [operator>=](#) (const lambda_type &lambda) const
>= *operator*
- bool [operator<](#) (const lambda_type &lambda) const
< *operator*
- bool [operator<](#) (const exact_type &x) const
< *operator*

- bool `operator!=` (const lambda_type &lambda) const
!= operator
- const lambda_type & `toLambda` () const
conversion to lambda type.
- bool `ge_impl` (const lambda_type &lambda) const
>= operator implementation.
- bool `less2_impl` (const exact_type &x) const
"<" operator implementation.

7.22.1 Detailed Description

`template<class Exact> class oln::morpho::attr::attribute< Exact >`

Attribute abstract class.

Top of the attribute hierarchy.

Definition at line 102 of file attributes.hh.

7.22.2 Member Function Documentation

7.22.2.1 `template<class Exact> bool oln::morpho::attr::attribute< Exact >::ge_impl (const lambda_type & lambda) const` [inline]

>= operator implementation.

This is an implementation of the *>=* operator. Override this method to provide a new implementation of this operator.

Warning:

This method *SHOULDN'T* be called.

Definition at line 179 of file attributes.hh.

```
180         {
181             return !(*this < lambda);
182         };
```

7.22.2.2 `template<class Exact> bool oln::morpho::attr::attribute< Exact >::less2_impl (const exact_type &x) const` [inline]

"<" operator implementation.

This is an implementation of the *<* operator. Override this method to provide a new implementation of this operator.

Warning:

This method *SHOULDN'T* be called.

Definition at line 191 of file attributes.hh.

```
192     {
193         return *this < x.toLambda();
194     };
```

7.22.2.3 `template<class Exact> bool oln::morpho::attr::attribute< Exact >::operator!= (const lambda_type & lambda) const` [inline]

!= operator

This is a static dispatcher for the != operator. This method is abstract.

Definition at line 157 of file attributes.hh.

```
158     {
159         mlc_dispatch(ne)(lambda);
160     };
```

7.22.2.4 `template<class Exact> void oln::morpho::attr::attribute< Exact >::operator+= (const exact_type & rhs)` [inline]

+= operator

This is a static dispatcher for the += operator. This method is abstract.

Definition at line 114 of file attributes.hh.

```
115     {
116         mlc_dispatch(pe)(rhs);
117     };
```

7.22.2.5 `template<class Exact> bool oln::morpho::attr::attribute< Exact >::operator< (const exact_type & x) const` [inline]

"<" operator

This is a static dispatcher for the "<" operator. This method is abstract.

Definition at line 146 of file attributes.hh.

```
147     {
148         mlc_dispatch(less2)(x);
149     };
```

7.22.2.6 `template<class Exact> bool oln::morpho::attr::attribute< Exact >::operator< (const lambda_type & lambda) const` [inline]

"<" operator

This is a static dispatcher for the "<" operator. This method is abstract.

Definition at line 135 of file attributes.hh.

```
136     {
137         mlc_dispatch(less)(lambda);
138     };
```

7.22.2.7 `template<class Exact> bool oln::morpho::attr::attribute< Exact >::operator>= (const lambda_type & lambda) const` `[inline]`

>= operator

This is a static dispatcher for the >= operator.

Definition at line 124 of file attributes.hh.

```
125         {  
126             mlc_dispatch(ge)(lambda);  
127         };
```

7.22.2.8 `template<class Exact> const lambda_type& oln::morpho::attr::attribute< Exact >::toLambda () const` `[inline]`

conversion to lambda type.

Warning:

Virtual method.

Definition at line 167 of file attributes.hh.

```
168         {  
169             mlc_dispatch(toLambda)();  
170         };
```

The documentation for this class was generated from the following file:

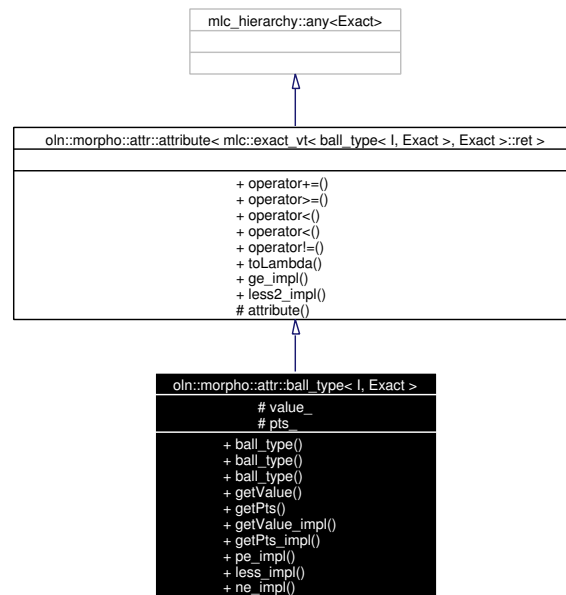
- attributes.hh

7.23 oln::morpho::attr::ball_type< I, Exact > Class Template Reference

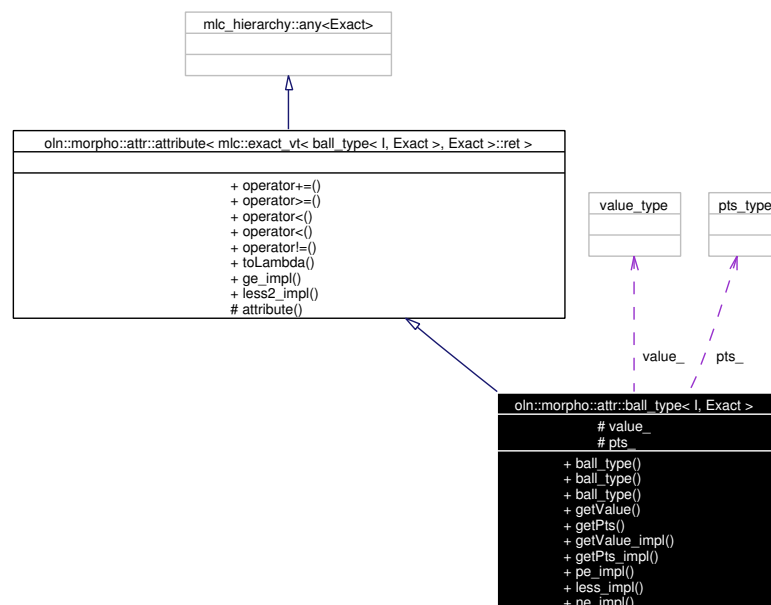
Ball attribute.

```
#include <attributes.hh>
```

Inheritance diagram for oln::morpho::attr::ball_type< I, Exact >:



Collaboration diagram for oln::morpho::attr::ball_type< I, Exact >:



Public Types

- typedef [ball_type](#)< I, Exact > [self_type](#)
Self type of the class.
- typedef mlc::exact_vt< [self_type](#), Exact >::ret **exact_type**
- typedef oln::morpho::attr::attr_traits< exact_type >::value_type **value_type**
- typedef oln::morpho::attr::attr_traits< exact_type >::env_type **env_type**
- typedef oln::morpho::attr::attr_traits< exact_type >::lambda_type **lambda_type**
- typedef [abstract::image](#)< typename mlc::exact< I >::ret > [im_type](#)
Image type.
- typedef mlc::exact< [im_type](#) >::ret::point_type [point_type](#)
Point type associated to im_type.
- typedef mlc::exact< [im_type](#) >::ret::dpoint_type [dpoint_type](#)
Dpoint type associated to im_type.
- typedef std::vector< [point_type](#) > [pts_type](#)
Point vector type.
- typedef pts_type::const_iterator [cst_iter_type](#)
const iterator on Point vector.

Public Member Functions

- [ball_type](#) ()
Basic Ctor.
- [ball_type](#) (const lambda_type &lambda)
Ctor from a lambda_type value.
- [ball_type](#) (const [im_type](#) &, const [point_type](#) &p, const env_type &)
Ctor from a point and an image.
- const value_type & [getValue](#) () const
Accessor to value_.
- const [pts_type](#) & [getPts](#) () const
Accessor to pts_.
- const value_type & [getValue_impl](#) () const
Implementation of [getValue\(\)](#).
- const [pts_type](#) & [getPts_impl](#) () const
Implementation of [getValue\(\)](#).
- void [pe_impl](#) (const [ball_type](#) &rhs)
+= operator implementation.

- bool [less_impl](#) (const lambda_type &lambda) const
"<" operator implementation.
- bool [ne_impl](#) (const lambda_type &lambda) const
!= operator implementation.

Protected Attributes

- value_type [value_](#)
Value of the attribute.
- pts_type [pts_](#)
List of point in the ball.

7.23.1 Detailed Description

template<class I, class Exact = mlc::final> class oln::morpho::attr::ball_type< I, Exact >

Ball attribute.

Parameters:

- I* Exact type of images to process.
Exact The exact type.

Definition at line 1119 of file attributes.hh.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 template<class I, class Exact = mlc::final> [oln::morpho::attr::ball_type](#)< I, Exact >::[ball_type](#) () [inline]

Basic Ctor.

Warning:

After this call, the object is only instantiated (not initialized).

Definition at line 1137 of file attributes.hh.

```
1138         {
1139         };
```


7.23.2.2 `template<class I, class Exact = mlc::final> oln::morpho::attr::ball_type< I, Exact >::ball_type (const lambda_type & lambda) [inline]`

Ctor from a lambda_type value.

- lambda Value of the attribute.

Definition at line 1146 of file attributes.hh.

References oln::morpho::attr::ball_type< I, Exact >::pts_.

```

1146                                     : value_(lambda), pts_()
1147     {
1148     };

```

7.23.2.3 `template<class I, class Exact = mlc::final> oln::morpho::attr::ball_type< I, Exact >::ball_type (const im_type &, const point_type & p, const env_type &) [inline]`

Ctor from a point and an image.

- p Point to consider in the image.

Definition at line 1156 of file attributes.hh.

References oln::morpho::attr::ball_type< I, Exact >::pts_.

```

1156                                     :
1157     value_(ntg_zero_val(value_type)), pts_()
1158
1159     {
1160         pts_.push_back(p);
1161     };

```

7.23.3 Member Function Documentation

7.23.3.1 `template<class I, class Exact = mlc::final> const pts_type& oln::morpho::attr::ball_type< I, Exact >::getPts () const [inline]`

Accessor to pts_.

Virtual method.

See also:

[getPts_impl\(\)](#)

Definition at line 1181 of file attributes.hh.

Referenced by oln::morpho::attr::ball_type< I, Exact >::pe_impl().

```

1182     {
1183         mlc_dispatch(getPts)();
1184     };

```

7.23.3.2 `template<class I, class Exact = mlc::final> const pts_type&
 oln::morpho::attr::ball_type< I, Exact >::getPts_impl () const [inline]`

Implementation of [getValue\(\)](#).

Override this method in order to provide a new version of [getPts\(\)](#).

Warning:

Do not call this method, use [getPts\(\)](#) instead.

Definition at line 1208 of file attributes.hh.

References `oln::morpho::attr::ball_type< I, Exact >::pts_`.

```
1209         {
1210             return pts_;
1211         };
```

7.23.3.3 `template<class I, class Exact = mlc::final> const value_type&
 oln::morpho::attr::ball_type< I, Exact >::getValue () const [inline]`

Accessor to `value_`.

Virtual method.

See also:

[getValue_impl\(\)](#)

Definition at line 1169 of file attributes.hh.

Referenced by `oln::morpho::attr::ball_type< I, Exact >::pe_impl()`.

```
1170         {
1171             mlc_dispatch(getValue)();
1172         };
```

7.23.3.4 `template<class I, class Exact = mlc::final> const value_type&
 oln::morpho::attr::ball_type< I, Exact >::getValue_impl () const [inline]`

Implementation of [getValue\(\)](#).

Override this method in order to provide a new version of [getValue\(\)](#).

Warning:

Do not call this method, use [getValue\(\)](#) instead.

Definition at line 1195 of file attributes.hh.

```
1196         {
1197             return value_;
1198         };
```

7.23.3.5 `template<class I, class Exact = mlc::final> bool oln::morpho::attr::ball_type< I, Exact >::less_impl (const lambda_type & lambda) const` `[inline]`

"<" operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 1249 of file attributes.hh.

```
1250         {
1251             return value_ < lambda;
1252         };
```

7.23.3.6 `template<class I, class Exact = mlc::final> bool oln::morpho::attr::ball_type< I, Exact >::ne_impl (const lambda_type & lambda) const` `[inline]`

!= operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 1261 of file attributes.hh.

```
1262         {
1263             return lambda != value_;
1264         };
```

7.23.3.7 `template<class I, class Exact = mlc::final> void oln::morpho::attr::ball_type< I, Exact >::pe_impl (const ball_type< I, Exact > & rhs)` `[inline]`

+= operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 1220 of file attributes.hh.

References `oln::morpho::attr::ball_type< I, Exact >::getPts()`, `oln::morpho::attr::ball_type< I, Exact >::getValue()`, and `oln::morpho::attr::ball_type< I, Exact >::pts_`.

```
1221         {
1222             value_type last = value_;
1223             std::copy(rhs.getPts().begin(),
1224                     rhs.getPts().end(),
```

```
1225         std::back_inserter(pts_));
1226     value_ = ntg_zero_val(value_type);
1227     for (cst_iter_type p1 = pts_.begin(); p1 != pts_.end(); ++p1)
1228         for (cst_iter_type p2 = pts_.begin(); p2 != pts_.end(); ++p2)
1229             {
1230                 unsigned d = 0;
1231                 dpoint_type p = *p1 - *p2;
1232                 for (int i = 0; i < point_traits<point_type>::dim; ++i)
1233                     d += p.nth(i) * p.nth(i);
1234                 if (d > value_)
1235                     value_ = d;
1236             }
1237     value_ /= 2;
1238     value_ = ntg::max(value_, last);
1239     value_ = ntg::max(value_, rhs.getValue());
1240 };
```

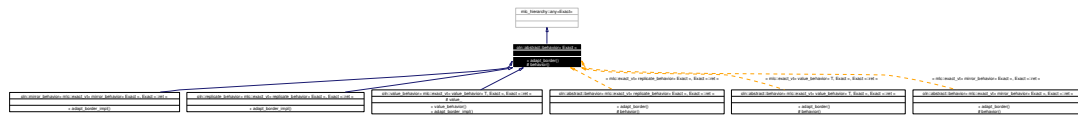
The documentation for this class was generated from the following file:

- attributes.hh

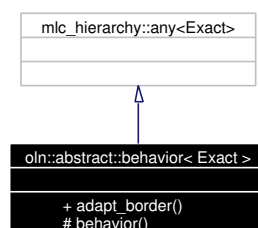
7.24 oln::abstract::behavior< Exact > Class Template Reference

```
#include <behavior.hh>
```

Inheritance diagram for oln::abstract::behavior< Exact >:



Collaboration diagram for oln::abstract::behavior< Exact >:



Public Types

- typedef [behavior< Exact >](#) [self_type](#)
- typedef [mlc::exact_vt< self_type, Exact >::ret](#) [exact_type](#)

Public Member Functions

- [template<class I> void adapt_border \(oln::abstract::image< I > &im, coord border_size\) const](#)
Adapt the border of an image.

Protected Member Functions

- [behavior \(\)](#)
CTOR.

7.24.1 Detailed Description

```
template<class Exact> class oln::abstract::behavior< Exact >
```

Behavior hierarchy.

The aim of this one is to describe how an algorithm should work on borders.

Definition at line 47 of file olena/oln/core/abstract/behavior.hh.

7.24.2 Member Typedef Documentation

7.24.2.1 `template<class Exact> typedef mlc::exact_vt< self_type , Exact >::ret oln::abstract::behavior< Exact >::exact_type`

The exact type.

Reimplemented in `oln::mirror_behavior< Exact >`, `oln::value_behavior< T, Exact >`, and `oln::replicate_behavior< Exact >`.

Definition at line 53 of file `olena/oln/core/abstract/behavior.hh`.

7.24.2.2 `template<class Exact> typedef behavior<Exact> oln::abstract::behavior< Exact >::self_type`

The self type.

Reimplemented in `oln::mirror_behavior< Exact >`, `oln::value_behavior< T, Exact >`, and `oln::replicate_behavior< Exact >`.

Definition at line 51 of file `olena/oln/core/abstract/behavior.hh`.

7.24.3 Constructor & Destructor Documentation

7.24.3.1 `template<class Exact> oln::abstract::behavior< Exact >::behavior () [inline, protected]`

CTor.

Do nothing, used only by sub-classes.

Definition at line 71 of file `olena/oln/core/abstract/behavior.hh`.

```
71 {};
```

7.24.4 Member Function Documentation

7.24.4.1 `template<class Exact> template<class I> void oln::abstract::behavior< Exact >::adapt_border (oln::abstract::image< I > & im, coord border_size) const [inline]`

Adapt the border of an image.

Adapt the border of an image regarding the kind of behavior wanted.

Definition at line 61 of file `olena/oln/core/abstract/behavior.hh`.

```
62     {
63         mlc_dispatch(adapt_border)(im, border_size);
64     };
```

The documentation for this class was generated from the following file:

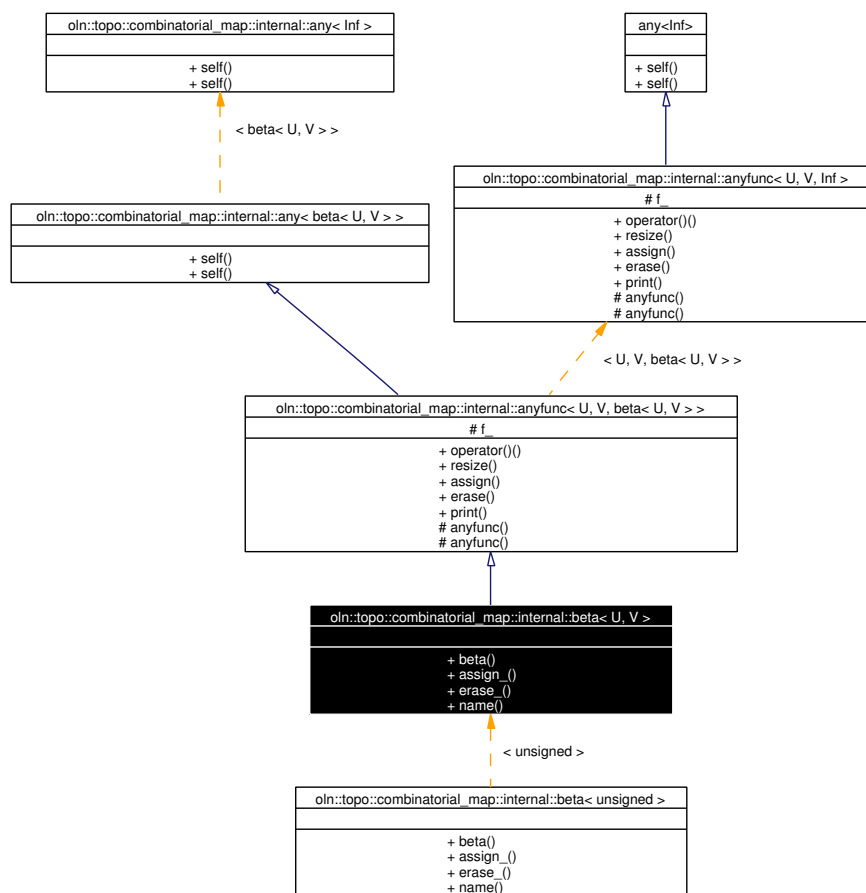
- `olena/oln/core/abstract/behavior.hh`

7.25 oln::topo::combinatorial_map::internal::beta< U, V > Class Template Reference

This function must be built using assign.

```
#include <beta.hh>
```

Inheritance diagram for oln::topo::combinatorial_map::internal::beta< U, V >:



Collaboration diagram for oln::topo::combinatorial_map::internal::beta< U, V >:



- ## Static Public Member Functions

- ### 7.25.1 Detailed Description

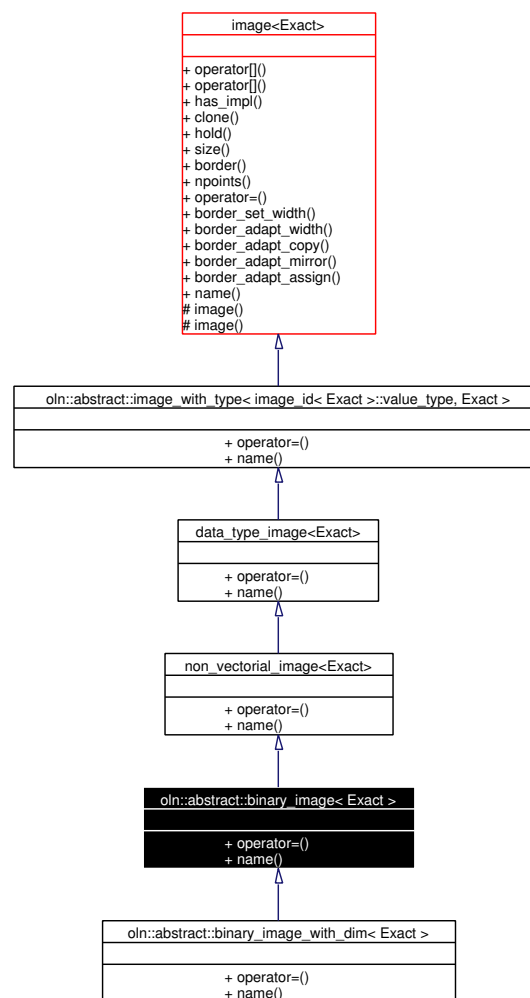
The documentation for this class was generated from the following file:

- Generated on Thu Apr 15 20:17:29 2004 for Olena by Doxygen

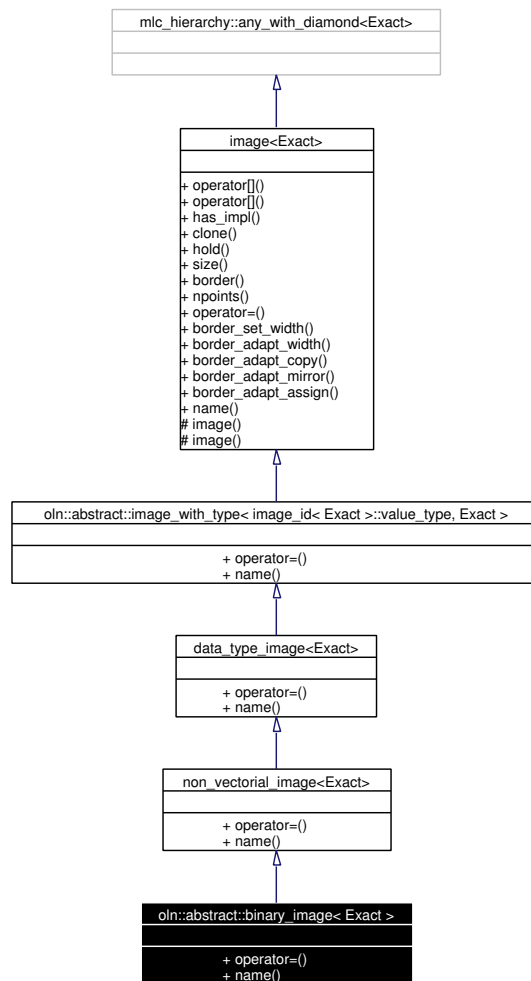
7.26 oln::abstract::binary_image< Exact > Class Template Reference

```
#include <image_with_type_with_dim.hh>
```

Inheritance diagram for oln::abstract::binary_image< Exact >:



Collaboration diagram for oln::abstract::binary_image< Exact >:



Public Types

- typedef `binary_image< Exact >` `self_type`
- typedef `Exact` `exact_type`

Public Member Functions

- `exact_type & operator= (self_type rhs)`

Perform a shallow copy between rhs and the current point, the points will not be duplicated but shared between the two images.

Static Public Member Functions

- `std::string name ()`

7.26.1 Detailed Description

`template<class Exact> class oln::abstract::binary_image< Exact >`

This class, when used in a function declaration, forces the function to only accept binary images.

Definition at line 221 of file `image_with_type_with_dim.hh`.

7.26.2 Member Function Documentation

7.26.2.1 `template<class Exact> exact_type& oln::abstract::binary_image< Exact >::operator=(self_type rhs) [inline]`

Perform a shallow copy between *rhs* and the current point, the points will not be duplicated but shared between the two images.

See also:

[`image::clone\(\)`](#)

Reimplemented from [`oln::abstract::non_vectorial_image< Exact >`](#).

Reimplemented in [`oln::abstract::binary_image_with_dim< Dim, Exact >`](#).

Definition at line 221 of file `image_with_type_with_dim.hh`.

273 {

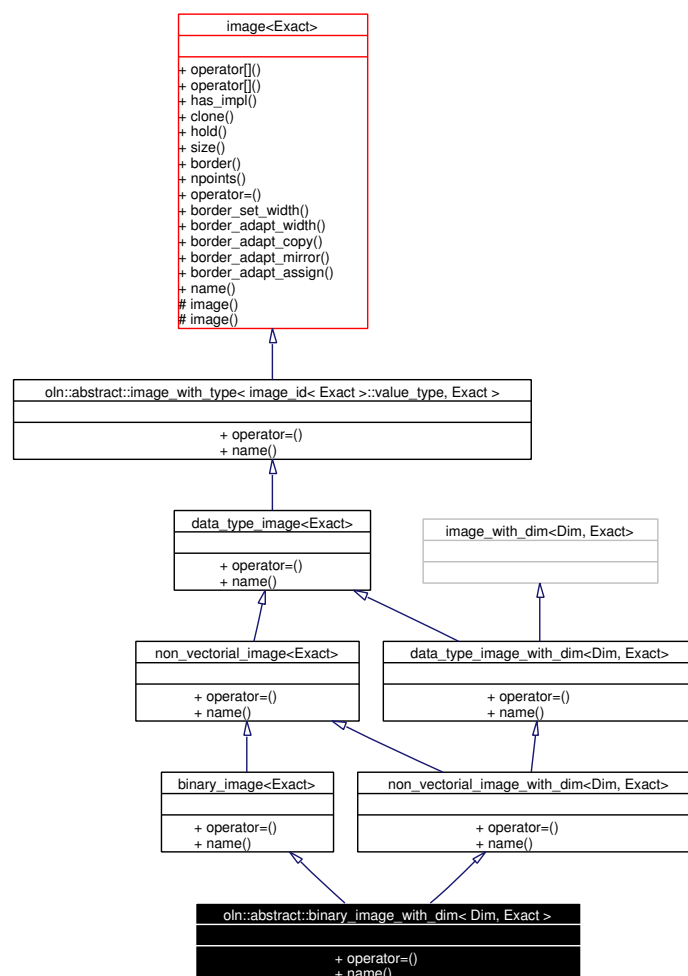
The documentation for this class was generated from the following file:

- `image_with_type_with_dim.hh`

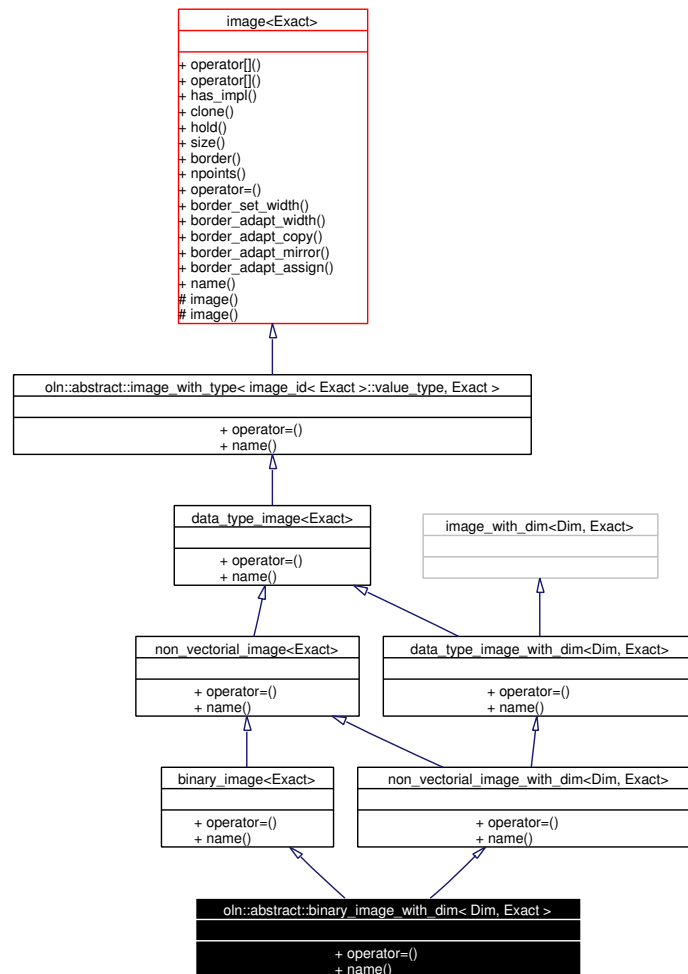
7.27 oln::abstract::binary_image_with_dim< Dim, Exact > Class Template Reference

```
#include <image_with_type_with_dim.hh>
```

Inheritance diagram for oln::abstract::binary_image_with_dim< Dim, Exact >:



Collaboration diagram for oln::abstract::binary_image_with_dim< Dim, Exact >:



Public Types

- typedef [binary_image_with_dim< Dim, Exact >](#) **self_type**
- typedef Exact **exact_type**

Public Member Functions

- exact_type & [operator=](#) (self_type rhs)

Perform a shallow copy between rhs and the current point, the points will not be duplicated but shared between the two images.

Static Public Member Functions

- std::string **name** ()

7.27.1 Detailed Description

template<unsigned Dim, class Exact> class oln::abstract::binary_image_with_dim< Dim, Exact >

This class, when used in a function declaration, forces the function to only accept binary images with a dimension equal to 'Dim'.

Definition at line 221 of file image_with_type_with_dim.hh.

7.27.2 Member Function Documentation

7.27.2.1 **template<unsigned Dim, class Exact> exact_type& oln::abstract::binary_image_with_dim< Dim, Exact >::operator= (self_type rhs)** [inline]

Perform a shallow copy between *rhs* and the current point, the points will not be duplicated but shared between the two images.

See also:

[image::clone\(\)](#)

Reimplemented from [oln::abstract::non_vectorial_image_with_dim< Dim, Exact >](#).

Definition at line 221 of file image_with_type_with_dim.hh.

273 {

The documentation for this class was generated from the following file:

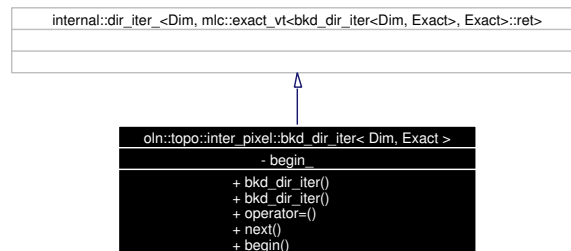
- image_with_type_with_dim.hh

7.28 oln::topo::inter_pixel::bkd_dir_iter< Dim, Exact > Class Template Reference

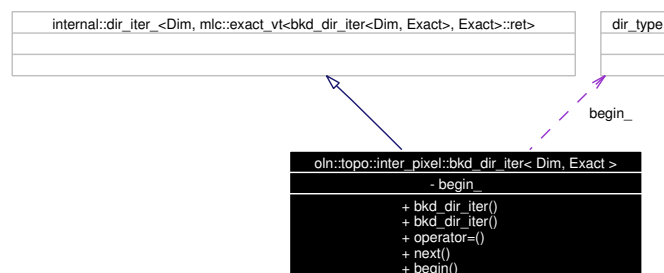
Backward iterator on direction.

```
#include <bkd-dir-iter.hh>
```

Inheritance diagram for oln::topo::inter_pixel::bkd_dir_iter< Dim, Exact >:



Collaboration diagram for oln::topo::inter_pixel::bkd_dir_iter< Dim, Exact >:



Public Member Functions

- **bkd_dir_iter** (dir_type i)
- template<class U> U **operator=** (U u)
- dir_type **next** ()
Next direction.
- dir_type **begin** ()
First direction.

7.28.1 Detailed Description

```
template<unsigned Dim, class Exact> class oln::topo::inter_pixel::bkd_dir_iter< Dim, Exact >
```

Backward iterator on direction.

Definition at line 57 of file bkd-dir-iter.hh.

7.28.2 Member Function Documentation

7.28.2.1 `template<unsigned Dim, class Exact> template<class U> U
 oln::topo::inter_pixel::bkd_dir_iter< Dim, Exact >::operator= (U u)` `[inline]`

Assignment.

Todo

FIXME: I am not sure that this respect the new paradigm.

Definition at line 74 of file bkd-dir-iter.hh.

```
74 { return super::operator=(u); }
```

The documentation for this class was generated from the following file:

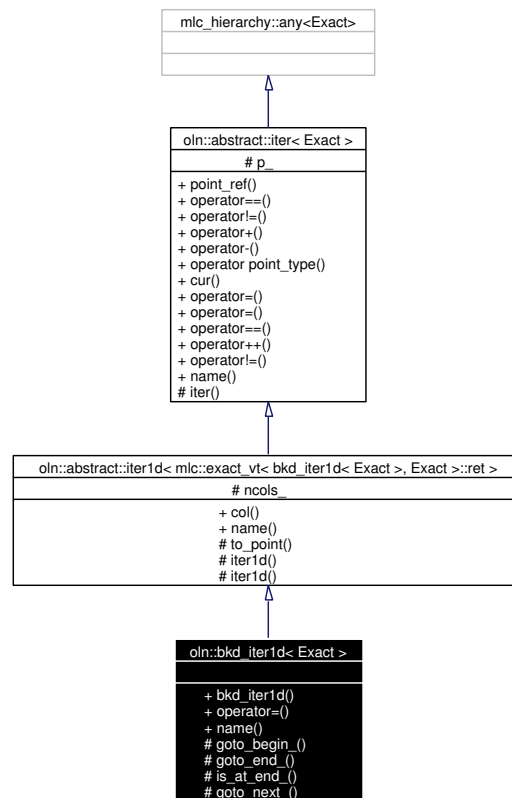
- bkd-dir-iter.hh

7.29 oln::bkd_iter1d< Exact > Class Template Reference

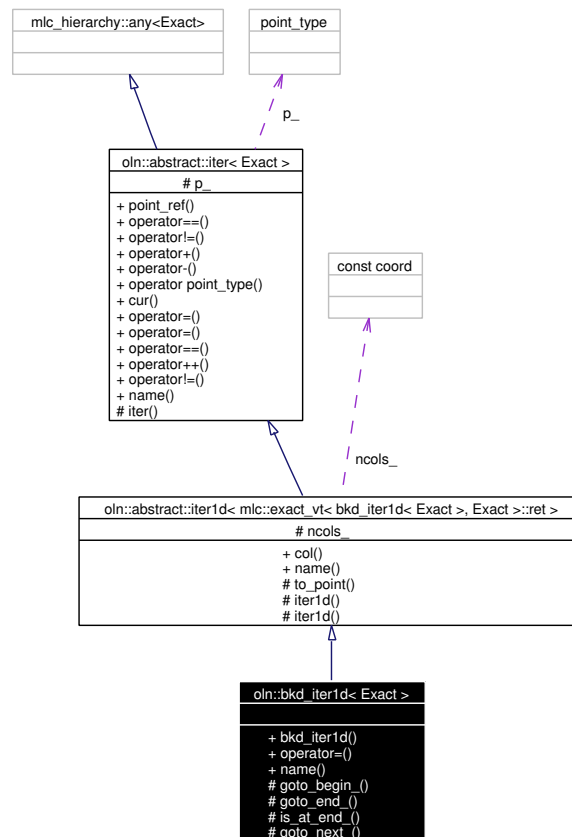
Backward Iterator of 1 dimension.

```
#include <bkd_iter1d.hh>
```

Inheritance diagram for oln::bkd_iter1d< Exact >:



Collaboration diagram for oln::bkd_iter1d< Exact >:



Public Types

- `typedef mlc::exact_vt< bkd_iter1d< Exact >, Exact >::ret exact_type`
The exact type.
- `typedef abstract::iter1d< exact_type > super_type`
The super type.
- `typedef abstract::iter< exact_type > super_iter_type`
The super iterator type.
- `typedef iter_traits< exact_type >::point_type point_type`
The associate image's type of point.
- `enum { dim = iter_traits<exact_type>::dim }`

Public Member Functions

- `template<class Image> bkd_iter1d (const Image &ima)`
Construct a backward iterator (1 dimension).
– *ima* The image to iterate.

- `template<class U> U operator= (U u)`
Set current iterator's point.

Static Public Member Functions

- `std::string name ()`
Return the name of the type.

Protected Member Functions

- `void goto_begin_ ()`
Set current point to the first iterator's point.
- `void goto_end_ ()`
Set current point to the last iterator's point.
- `bool is_at_end_ () const`
Test if iterator's current point is the last one.
- `void goto_next_ ()`
Go to the next iterator's point.

Friends

- `class abstract::iter< exact_type >`
- `class abstract::iter1d< exact_type >`

7.29.1 Detailed Description

`template<class Exact> class oln::bkd_iter1d< Exact >`

Backward Iterator of 1 dimension.

Allow iterable object (like image, window, ...) of 1 dimension backward traversing.

See also:

[iter](#)

Definition at line 58 of file `bkd_iter1d.hh`.

7.29.2 Member Typedef Documentation

7.29.2.1 `template<class Exact> typedef iter_traits<exact_type>::point_type oln::bkd_iter1d< Exact >::point_type`

The associate image's type of point.

Warning:

Prefer the macros `oln_point_type(Pointable)` and `oln_point_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented from [oln::abstract::iter< Exact >](#).

Definition at line 76 of file `bkd_iter1d.hh`.

7.29.3 Member Function Documentation

7.29.3.1 `template<class Exact> void oln::bkd_iter1d< Exact >::goto_begin_()` `[inline, protected]`

Set current point to the first iterator's point.

Set current point of iterator to the first iterator's point.

Definition at line 117 of file `bkd_iter1d.hh`.

```
118     {
119         this->p_.col() = this->ncols_ - 1;
120     }
```

7.29.3.2 `template<class Exact> void oln::bkd_iter1d< Exact >::goto_end_()` `[inline, protected]`

Set current point to the last iterator's point.

Set current point of iterator to the last iterator's point.

Definition at line 128 of file `bkd_iter1d.hh`.

```
129     {
130         this->p_.col() = -1;
131     }
```

7.29.3.3 `template<class Exact> bool oln::bkd_iter1d< Exact >::is_at_end_() const` `[inline, protected]`

Test if iterator's current point is the last one.

Returns:

True if current point is the last one.

Definition at line 138 of file `bkd_iter1d.hh`.

```
139     {
140         return this->p_.col() == -1;
141     }
```

7.29.3.4 `template<class Exact> template<class U> U oln::bkd_iter1d< Exact >::operator=(U u)` [inline]

Set current iterator's point.

Set current point of iterator to the first iterator's point.

Definition at line 97 of file bkd_iter1d.hh.

```
98     {  
99         return super_iter_type::operator=(u);  
100    }
```

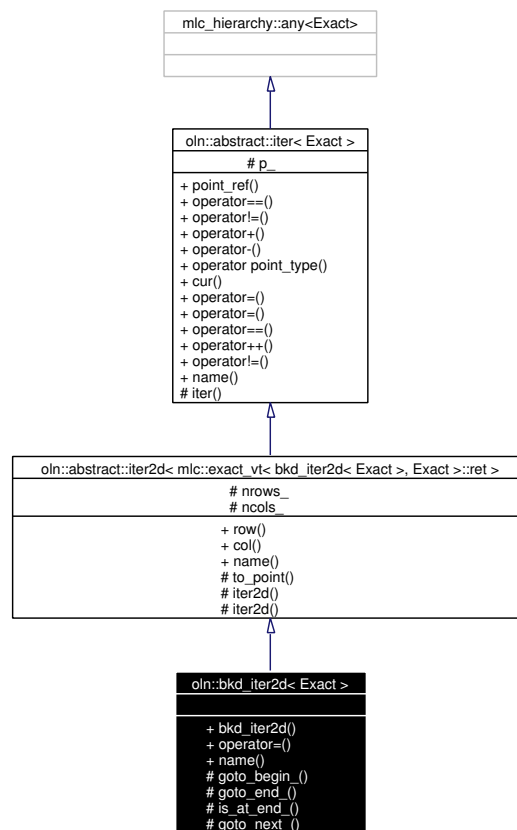
The documentation for this class was generated from the following file:

- bkd_iter1d.hh

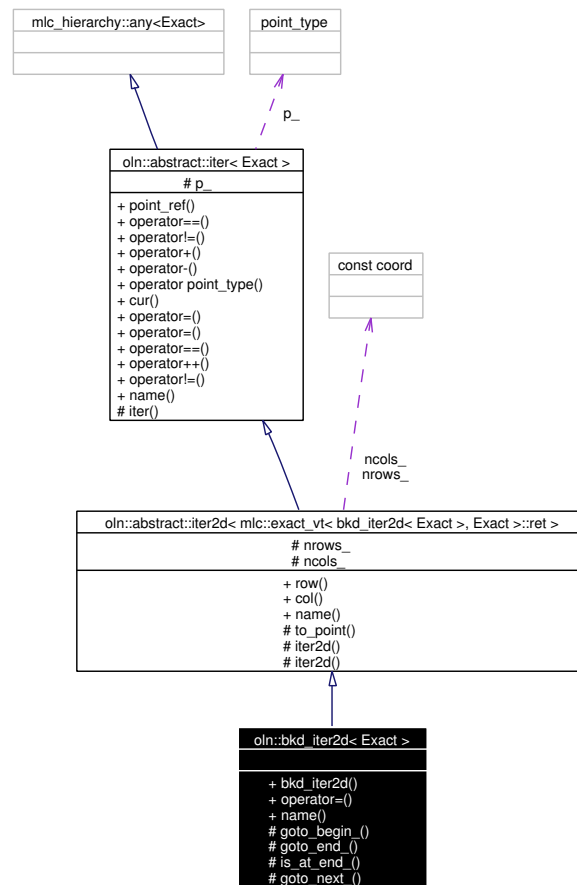
7.30 oln::bkd_iter2d< Exact > Class Template Reference

```
#include <bkd_iter2d.hh>
```

Inheritance diagram for oln::bkd_iter2d< Exact >:



Collaboration diagram for oln::bkd_iter2d< Exact >:



Public Types

- typedef `mhc::exact_vt< bkd_iter2d< Exact >, Exact >::ret` `exact_type`
The exact type.
- typedef `abstract::iter2d< exact_type >` `super_type`
The super type.
- typedef `abstract::iter< exact_type >` `super_iter_type`
The super iterator type.
- typedef `iter_traits< exact_type >::point_type` `point_type`
The associate image's type of point.
- enum { **dim** = `iter_traits<exact_type>::dim` }

Public Member Functions

- template<class Image> `bkd_iter2d` (const Image &ima)
Construct a backward iterator (2 dimension).
– *ima* The image to iterate.

- `template<class U> U operator= (U u)`

Set current iterator's point.

– *u* New current point.

Static Public Member Functions

- `std::string name ()`

Return the name of the type.

Protected Member Functions

- `void goto_begin_ ()`

Set current point to the first iterator's point.

- `void goto_end_ ()`

Set current point to the last iterator's point.

- `bool is_at_end_ () const`

Test if iterator's current point is the last one.

- `void goto_next_ ()`

Go to the next iterator's point.

Friends

- `class abstract::iter< exact_type >`
- `class abstract::iter2d< exact_type >`

7.30.1 Detailed Description

`template<class Exact> class oln::bkd_iter2d< Exact >`

Backward Iterator on image 2 dimension

Allow iterable object (like image, window, ...) of 2 dimensions backward traversing.

See also:

[iter](#)

Definition at line 58 of file `bkd_iter2d.hh`.

7.30.2 Member Typedef Documentation

7.30.2.1 `template<class Exact> typedef iter_traits<exact_type>::point_type oln::bkd_iter2d<Exact>::point_type`

The associate image's type of point.

Warning:

Prefer the macros `oln_point_type(Pointable)` and `oln_point_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented from `oln::abstract::iter< Exact >`.

Definition at line 76 of file `bkd_iter2d.hh`.

7.30.3 Member Function Documentation

7.30.3.1 `template<class Exact> void oln::bkd_iter2d< Exact >::goto_begin_() [inline, protected]`

Set current point to the first iterator's point.

Set current point of iterator to the first iterator's point.

Definition at line 116 of file `bkd_iter2d.hh`.

```

117     {
118         this->p_.row() = this->nrows_ - 1;
119         this->p_.col() = this->ncols_ - 1;
120     }
```

7.30.3.2 `template<class Exact> void oln::bkd_iter2d< Exact >::goto_end_() [inline, protected]`

Set current point to the last iterator's point.

Set current point of iterator to the last iterator's point.

Definition at line 128 of file `bkd_iter2d.hh`.

```

129     {
130         this->p_.row() = -1;
131     }
```

7.30.3.3 `template<class Exact> bool oln::bkd_iter2d< Exact >::is_at_end_() const [inline, protected]`

Test if iterator's current point is the last one.

Returns:

True if current point is the last one.

Definition at line 138 of file `bkd_iter2d.hh`.

```
139     {  
140         return this->p_.row() == -1;  
141     }
```

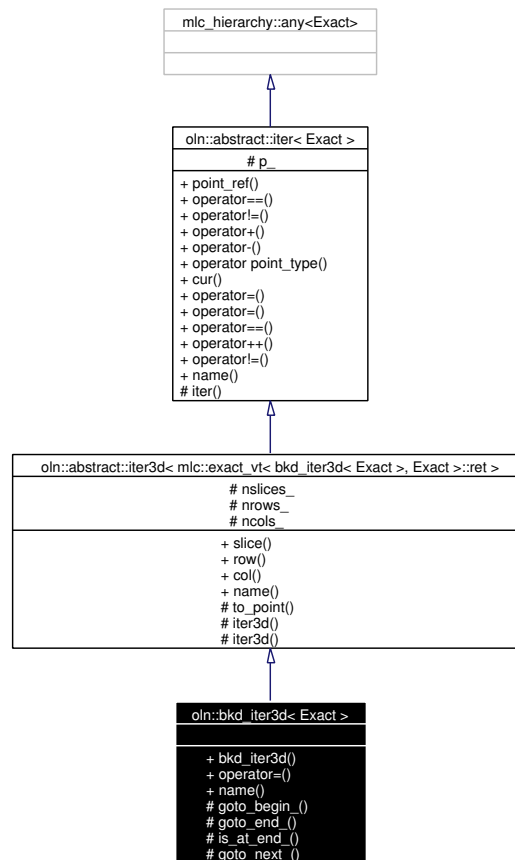
The documentation for this class was generated from the following file:

- bkd_iter2d.hh

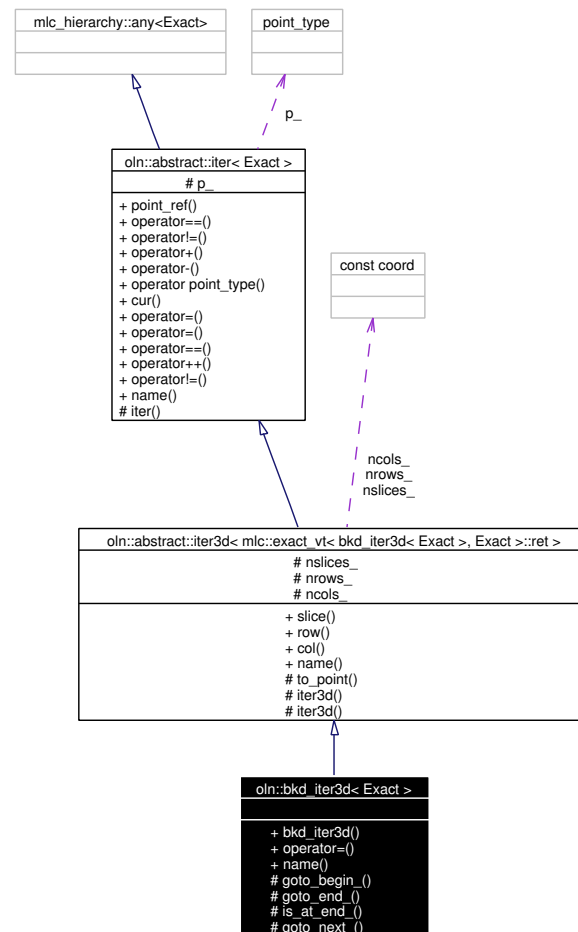
7.31 oln::bkd_iter3d< Exact > Class Template Reference

```
#include <bkd_iter3d.hh>
```

Inheritance diagram for oln::bkd_iter3d< Exact >:



Collaboration diagram for oln::bkd_iter3d< Exact >:



Public Types

- typedef `mlc::exact_vt< bkd_iter3d< Exact >, Exact >::ret exact_type`
The exact type.
- typedef `abstract::iter3d< exact_type > super_type`
The super type.
- typedef `abstract::iter< exact_type > super_iter_type`
The super iterator type.
- typedef `iter_traits< exact_type >::point_type point_type`
The associate image's type of point.
- enum { **dim** = iter_traits<exact_type>::dim }

Public Member Functions

- template<class Image> `bkd_iter3d` (const Image &ima)

Construct a backward iterator (3 dimension).

- *ima* The image to iterate.

- `template<class U> U operator= (U u)`

Set current iterator's point.

- *u* New current point.

Static Public Member Functions

- `std::string name ()`

Return the name of the type.

Protected Member Functions

- `void goto_begin_ ()`

Set current point to the first iterator's point.

- `void goto_end_ ()`

Set current point to the last iterator's point.

- `bool is_at_end_ () const`

Test if iterator's current point is the last one.

- `void goto_next_ ()`

Go to the next iterator's point.

Friends

- `class abstract::iter< exact_type >`
- `class abstract::iter3d< exact_type >`

7.31.1 Detailed Description

`template<class Exact> class oln::bkd_iter3d< Exact >`

Backward Iterator on image 3 dimension.

Allow iterable object (like image, window, ...) 3 dimensions backward traversing.

See also:

[iter](#)

Definition at line 55 of file `bkd_iter3d.hh`.

7.31.2 Member Typedef Documentation

7.31.2.1 `template<class Exact> typedef iter_traits<exact_type>::point_type oln::bkd_iter3d<Exact >::point_type`

The associate image's type of point.

Warning:

Prefer the macros `oln_point_type(Pointable)` and `oln_point_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented from `oln::abstract::iter< Exact >`.

Definition at line 73 of file `bkd_iter3d.hh`.

7.31.3 Member Function Documentation

7.31.3.1 `template<class Exact> void oln::bkd_iter3d< Exact >::goto_begin_() [inline, protected]`

Set current point to the first iterator's point.

Set current point of iterator to the first iterator's point.

Definition at line 112 of file `bkd_iter3d.hh`.

```

113     {
114         this->p_.slice() = this->nslices_ - 1;
115         this->p_.row() = this->nrows_ - 1;
116         this->p_.col() = this->ncols_ - 1;
117     }
```

7.31.3.2 `template<class Exact> void oln::bkd_iter3d< Exact >::goto_end_() [inline, protected]`

Set current point to the last iterator's point.

Set current point of iterator to the last iterator's point.

Definition at line 125 of file `bkd_iter3d.hh`.

```

126     {
127         this->p_.slice() = -1;
128     }
```

7.31.3.3 `template<class Exact> bool oln::bkd_iter3d< Exact >::is_at_end_() const [inline, protected]`

Test if iterator's current point is the last one.

Returns:

True if current point is the last one.

Definition at line 135 of file bkd_iter3d.hh.

```
136     {  
137         return this->p_.slice() == -1;  
138     }
```

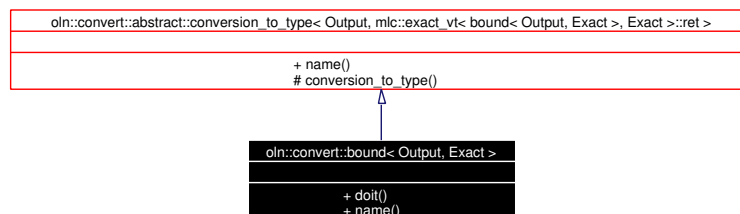
The documentation for this class was generated from the following file:

- bkd_iter3d.hh

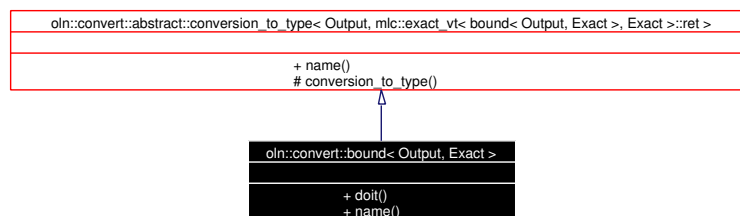
7.32 oln::convert::bound< Output, Exact > Struct Template Reference

```
#include <bound.hh>
```

Inheritance diagram for oln::convert::bound< Output, Exact >:



Collaboration diagram for oln::convert::bound< Output, Exact >:



Public Member Functions

- `template<class Input> Output doit (const Input &v) const`

Static Public Member Functions

- `std::string name ()`

7.32.1 Detailed Description

`template<class Output, class Exact = mlc::final> struct oln::convert::bound< Output, Exact >`

Like [convert::force](#), but with saturation.

Todo

- FIXME: is this really useful with new types ?

Definition at line 43 of file `bound.hh`.

The documentation for this struct was generated from the following file:

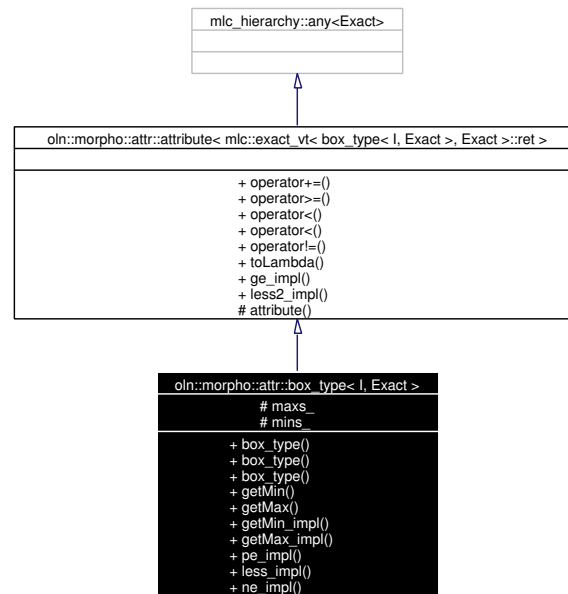
- `bound.hh`

7.33 oln::morpho::attr::box_type< I, Exact > Class Template Reference

Box attribute.

```
#include <attributes.hh>
```

Inheritance diagram for oln::morpho::attr::box_type< I, Exact >:



Collaboration diagram for oln::morpho::attr::box_type< I, Exact >:



Public Types

- typedef [box_type](#)< I, Exact > [self_type](#)
Self type of the class.
- typedef mlc::exact_vt< [self_type](#), Exact >::ret [exact_type](#)
- typedef oln::morpho::attr::attr_traits< exact_type >::value_type [value_type](#)
- typedef oln::morpho::attr::attr_traits< exact_type >::env_type [env_type](#)
- typedef oln::morpho::attr::attr_traits< exact_type >::lambda_type [lambda_type](#)
- typedef [abstract::image](#)< typename mlc::exact< I >::ret > [im_type](#)
Image type.
- typedef mlc::exact< [im_type](#) >::ret::point_type [point_type](#)
Point type associated to im_type.
- typedef mlc::exact< [im_type](#) >::ret::dpoint_type [dpoint_type](#)
Dpoint type associated to im_type.
- enum { **dim** = point_traits<point_type>::dim }

Public Member Functions

- [box_type](#) (const [lambda_type](#) &lambda)
Ctor from a lambda_type value.
- [box_type](#) ()
Basic Ctor.
- [box_type](#) (const [im_type](#) &, const [point_type](#) &p, const [env_type](#) &)
Ctor from a point and an image.
- [value_type](#) [getMin](#) (int i) const
Accessor to minimums.
- [value_type](#) [getMax](#) (int i) const
Accessor to maximums.
- [value_type](#) [getMin_impl](#) (int i) const
Accessor to minimums.
- [value_type](#) [getMax_impl](#) (int i) const
Accessor to maximums.
- void [pe_impl](#) (const [box_type](#) &rhs)
+= operator implementation.
- bool [less_impl](#) (const [lambda_type](#) &lambda) const
"<" operator implementation.
- bool [ne_impl](#) (const [lambda_type](#) &lambda) const
!= operator implementation.

Protected Attributes

- std::vector< value_type > [maxs_](#)
List of minimums.
- std::vector< value_type > [mins_](#)
List of maximums.

7.33.1 Detailed Description

template<class I, class Exact = mlc::final> class oln::morpho::attr::box_type< I, Exact >

Box attribute.

Parameters:

I Exact type of images to process.

Exact The exact type.

Definition at line 1603 of file attributes.hh.

7.33.2 Constructor & Destructor Documentation

7.33.2.1 template<class I, class Exact = mlc::final> [oln::morpho::attr::box_type](#)< I, Exact >::[box_type](#) (const lambda_type & *lambda*) [inline]

Ctor from a lambda_type value.

- lambda Value of the attribute.

Definition at line 1619 of file attributes.hh.

```

1619                                     : maxs_(dim), mins_(dim)
1620     {
1621         for (int i = 0; i < dim; ++i)
1622             {
1623                 mins_[i] = ntg_zero_val(value_type);
1624                 maxs_[i] = lambda[i];
1625             }
1626     };

```

7.33.2.2 template<class I, class Exact = mlc::final> [oln::morpho::attr::box_type](#)< I, Exact >::[box_type](#) () [inline]

Basic Ctor.

Warning:

After this call, the object is only instantiated (not initialized).

Definition at line 1634 of file attributes.hh.

```

1635     {
1636 };

```

7.33.2.3 `template<class I, class Exact = mlc::final> oln::morpho::attr::box_type< I, Exact >::box_type (const im_type &, const point_type & p, const env_type &) [inline]`

Ctor from a point and an image.

- *p* Point to consider in the image.

Definition at line 1643 of file `attributes.hh`.

```

1643                                     : maxs_(dim), mins_(dim)
1644     {
1645         for (int i = 0; i < dim; ++i)
1646             mins_[i] = maxs_[i] = p.nth(i);
1647     };

```

7.33.3 Member Function Documentation

7.33.3.1 `template<class I, class Exact = mlc::final> value_type oln::morpho::attr::box_type< I, Exact >::getMax (int i) const [inline]`

Accessor to maximums.

Virtual method.

- *i* Index of the minimum wanted.

Returns:

the *i* th maximum.

See also:

[getMax_impl\(\)](#)

Definition at line 1671 of file `attributes.hh`.

Referenced by `oln::morpho::attr::box_type< I, Exact >::pe_impl()`.

```

1672     {
1673         mlc_dispatch(getMax)(i);
1674     };

```

7.33.3.2 `template<class I, class Exact = mlc::final> value_type oln::morpho::attr::box_type< I, Exact >::getMax_impl (int i) const [inline]`

Accessor to maximums.

Virtual method.

- *i* Index of the minimum wanted.

Returns:

the *i* th maximum.

See also:

[getMax_impl\(\)](#)

Definition at line 1699 of file `attributes.hh`.

```

1700         {
1701             precondition(i < point_traits<point_type>::dim);
1702             return maxs_[i];
1703         };

```

7.33.3.3 `template<class I, class Exact = mlc::final> value_type oln::morpho::attr::box_type< I, Exact >::getMin (int i) const` [inline]

Accessor to minimums.

Virtual method.

- *i* Index of the minimum wanted.

Returns:

the *i* th minimum.

See also:

[getMin_impl\(\)](#)

Definition at line 1657 of file attributes.hh.

Referenced by `oln::morpho::attr::box_type< I, Exact >::pe_impl()`.

```

1658         {
1659             mlc_dispatch(getMin)(i);
1660         };

```

7.33.3.4 `template<class I, class Exact = mlc::final> value_type oln::morpho::attr::box_type< I, Exact >::getMin_impl (int i) const` [inline]

Accessor to minimums.

Virtual method.

- *i* Index of the minimum wanted.

Returns:

the *i* th minimum.

See also:

[getMin_impl\(\)](#)

Definition at line 1685 of file attributes.hh.

```

1686         {
1687             precondition(i < point_traits<point_type>::dim);
1688             return mins_[i];
1689         };

```

7.33.3.5 `template<class I, class Exact = mlc::final> bool oln::morpho::attr::box_type< I, Exact >::less_impl (const lambda_type & lambda) const` [inline]

"<" operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 1728 of file attributes.hh.

```

1729         {
1730             for (int i = 0; i < dim; ++i)
1731                 if ((maxs_[i] - mins_[i]) >= lambda[i])
1732                     return false;
1733             return true;
1734         }

```

7.33.3.6 `template<class I, class Exact = mlc::final> bool oln::morpho::attr::box_type< I, Exact >::ne_impl (const lambda_type & lambda) const` [inline]

!= operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 1743 of file attributes.hh.

```

1744         {
1745             for (int i = 0; i < dim; ++i)
1746                 if ((maxs_[i] - mins_[i]) == lambda[i])
1747                     return false;
1748             return true;
1749         };

```

7.33.3.7 `template<class I, class Exact = mlc::final> void oln::morpho::attr::box_type< I, Exact >::pe_impl (const box_type< I, Exact > & rhs)` [inline]

+= operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 1712 of file attributes.hh.

References oln::morpho::attr::box_type< I, Exact >::getMax(), and oln::morpho::attr::box_type< I, Exact >::getMin().

```
1713         {
1714             for (int i = 0; i < dim; ++i)
1715             {
1716                 mins_[i] = ntg::min(mins_[i], rhs.getMin(i));
1717                 maxs_[i] = ntg::max(maxs_[i], rhs.getMax(i));
1718             }
1719         }
```

The documentation for this class was generated from the following file:

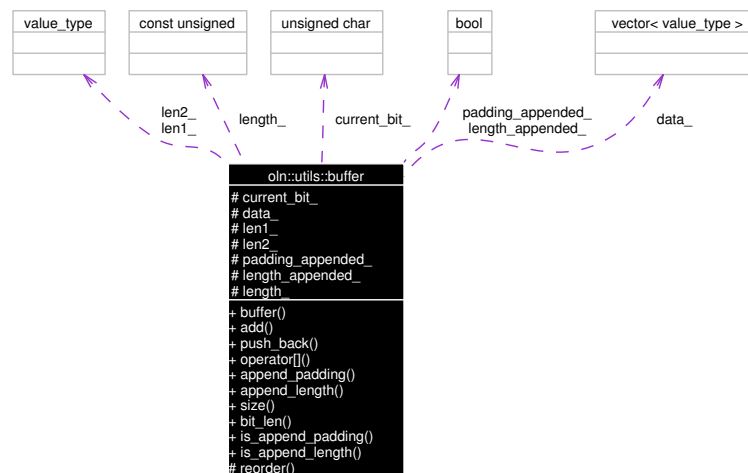
- attributes.hh

7.34 oln::utils::buffer Class Reference

Buffer used for MD5 data type abstraction.

```
#include <buffer.hh>
```

Collaboration diagram for oln::utils::buffer:



Public Types

- typedef ntg::int_u32 [value_type](#)
Used data type.

Public Member Functions

- [buffer](#) ()
Initialization of data.
- template<class E> void [add](#) (const E &e, bool count=true)
Add every bits of e in data.
- void [push_back](#) (bool bit, bool is_padding=false)
Push back a bit at the end of the buffer.
- ntg::int_u32 [operator\[\]](#) (unsigned n) const
Return the nth word.
- void [append_padding](#) ()
Append padding bits.
- void [append_length](#) ()
Append length.

- unsigned [size](#) () const
Return the number of words.
- unsigned [bit_len](#) () const
Return the length in bits.
- bool [is_append_padding](#) () const
Tell if the buffer has already been padded.
- bool [is_append_length](#) () const
Tell if the length has already been appended.

Protected Member Functions

- [value_type reorder](#) ([value_type](#) x) const
Change order of data in a word.

Protected Attributes

- unsigned char [current_bit_](#)
Number of the current bit in the current word.
- [std::vector< value_type > data_](#)
The buffer data.
- [value_type len1_](#)
first word length
- [value_type len2_](#)
second word length
- bool [padding_appended_](#)
Status of padding.
- bool [length_appended_](#)
Status of length appending.

Static Protected Attributes

- const unsigned [length_](#) = 100

7.34.1 Detailed Description

Buffer used for [MD5](#) data type abstraction.

Definition at line 38 of file `buffer.hh`.

7.34.2 Member Function Documentation

7.34.2.1 `template<class E> void oln::utils::buffer::add (const E & e, bool count = true)` `[inline]`

Add every bits of *e* in data.

- *e* Element to work on.
- *count* Tell if you want to count *e* size in the buffer size.

Definition at line 167 of file `buffer.hh`.

7.34.2.2 `void oln::utils::buffer::push_back (bool bit, bool is_padding = false)` `[inline]`

Push back a bit at the end of the buffer.

true -> push back a 1

false -> push back a 0

- *bit* Bit to push.
- *is_padding* Are you adding padding bytes ?

Definition at line 181 of file `buffer.hh`.

7.34.2.3 `buffer::value_type oln::utils::buffer::reorder (value_type x) const` `[inline, protected]`

Change order of data in a word.

- *x* Data to reorder.

Definition at line 211 of file `buffer.hh`.

7.34.3 Member Data Documentation

7.34.3.1 `const unsigned oln::utils::buffer::length_ = 100` `[static, protected]`

Capacity chunk.

Definition at line 111 of file `buffer.hh`.

The documentation for this class was generated from the following file:

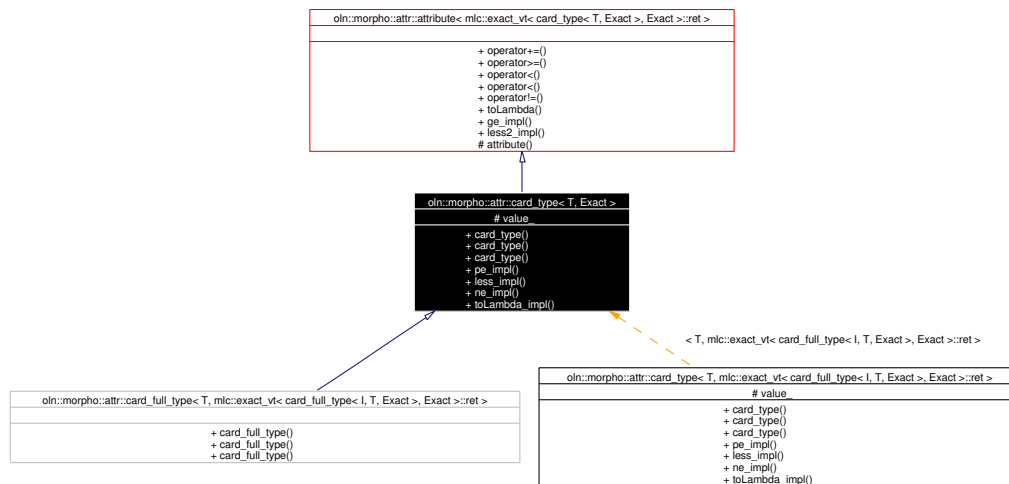
- `buffer.hh`

7.35 oln::morpho::attr::card_type< T, Exact > Class Template Reference

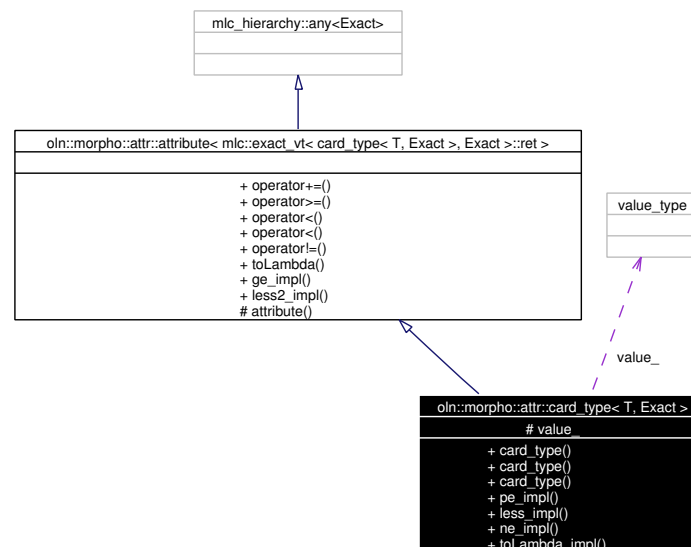
Cardinal attribute.

```
#include <attributes.hh>
```

Inheritance diagram for oln::morpho::attr::card_type< T, Exact >:



Collaboration diagram for oln::morpho::attr::card_type< T, Exact >:



Public Types

- typedef `card_type< T, Exact >` `self_type`
- typedef `mlc::exact_vt< self_type, Exact >::ret` `exact_type`

- typedef oln::morpho::attr::attr_traits< exact_type >::value_type **value_type**
- typedef oln::morpho::attr::attr_traits< exact_type >::env_type **env_type**
- typedef oln::morpho::attr::attr_traits< exact_type >::lambda_type **lambda_type**

Public Member Functions

- [card_type](#) ()
Basic Ctor.
- [card_type](#) (const lambda_type &lambda)
Ctor from a lambda_type value.
- template<class I> [card_type](#) (const [abstract::image](#)< I > &, const typename mlc::exact< I >::ret::point_type &, const env_type &)
Ctor from a point and an image.
- void [pe_impl](#) (const [self_type](#) &rhs)
+= operator implementation.
- bool [less_impl](#) (const lambda_type &lambda) const
"<" operator implementation.
- bool [ne_impl](#) (const lambda_type &lambda) const
!= operator implementation.
- const lambda_type & [toLambda_impl](#) () const
conversion to lambda type implementation.

Protected Attributes

- value_type [value_](#)

7.35.1 Detailed Description

template<class T = unsigned, class Exact = mlc::final> class oln::morpho::attr::card_type< T, Exact >

Cardinal attribute.

It is equivalent to an area in 2d, and a volume in 3D.

Definition at line 213 of file attributes.hh.

7.35.2 Member Typedef Documentation

7.35.2.1 template<class T = unsigned, class Exact = mlc::final> typedef [card_type](#)<T, Exact> [oln::morpho::attr::card_type](#)< T, Exact >::[self_type](#)

Self type of the class.

Reimplemented from [oln::morpho::attr::attribute< Exact >](#).

Definition at line 217 of file attributes.hh.

7.35.3 Constructor & Destructor Documentation

7.35.3.1 `template<class T = unsigned, class Exact = mlc::final> oln::morpho::attr::card_type< T, Exact >::card_type() [inline]`

Basic Ctor.

Warning:

After this call, the object is only instantiated (not initialized).

Definition at line 226 of file attributes.hh.

```
227         {
228         };
```

7.35.3.2 `template<class T = unsigned, class Exact = mlc::final> template<class I> oln::morpho::attr::card_type< T, Exact >::card_type (const abstract::image< I > &, const typename mlc::exact< I >::ret::point_type &, const env_type &) [inline]`

Ctor from a point and an image.

Every parameters are useless.

Definition at line 243 of file attributes.hh.

```
245                                     :
246         value_(ntg_unit_val(value_type))
247         {
248         };
```

7.35.4 Member Function Documentation

7.35.4.1 `template<class T = unsigned, class Exact = mlc::final> bool oln::morpho::attr::card_type< T, Exact >::less_impl (const lambda_type & lambda) const [inline]`

"<" operator implementation.

This is an implementation of the "<" operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T be called.

Definition at line 270 of file attributes.hh.

```
271         {
272             return value_ < lambda;
273         };
```

7.35.4.2 `template<class T = unsigned, class Exact = mlc::final> bool
oln::morpho::attr::card_type< T, Exact >::ne_impl (const lambda_type & lambda) const
 [inline]`

!= operator implementation.

This is an implementation of the != operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T be called.

Definition at line 282 of file attributes.hh.

```
283         {
284             return lambda != value_;
285         };
```

7.35.4.3 `template<class T = unsigned, class Exact = mlc::final> void
oln::morpho::attr::card_type< T, Exact >::pe_impl (const self_type & rhs) [inline]`

+= operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T be called.

Definition at line 258 of file attributes.hh.

```
259         {
260             value_ += rhs.value_;
261         };
```

7.35.4.4 `template<class T = unsigned, class Exact = mlc::final> const lambda_type&
oln::morpho::attr::card_type< T, Exact >::toLambda_impl () const [inline]`

conversion to lambda type implementation.

This is an implementation of the [toLambda\(\)](#) method. Override this method to provide a new implementation.

Warning:

This method SHOULDN'T be called .

Definition at line 294 of file attributes.hh.

```
295         {
296             return value_;
297         };
```

7.35.5 Member Data Documentation

7.35.5.1 `template<class T = unsigned, class Exact = mlc::final> value_type
 oln::morpho::attr::card_type< T, Exact >::value_` [protected]

Value used inside the class.

Definition at line 300 of file attributes.hh.

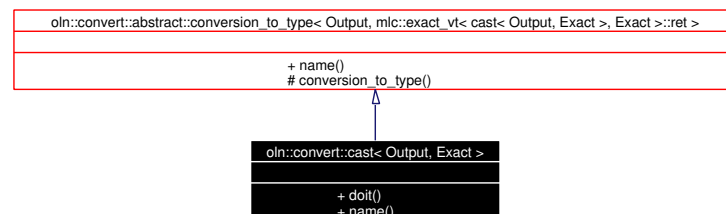
The documentation for this class was generated from the following file:

- attributes.hh

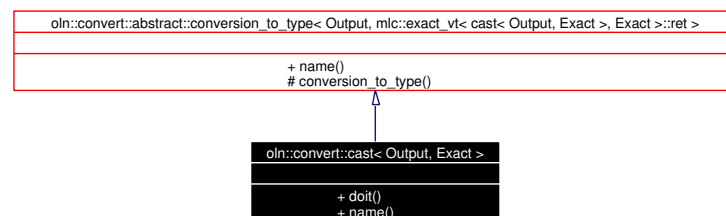
7.36 oln::convert::cast< Output, Exact > Struct Template Reference

```
#include <cast.hh>
```

Inheritance diagram for oln::convert::cast< Output, Exact >:



Collaboration diagram for oln::convert::cast< Output, Exact >:



Public Member Functions

- `template<class Input> Output doit (const Input &v) const`

Static Public Member Functions

- `std::string name ()`

7.36.1 Detailed Description

`template<class Output, class Exact = mlc::final> struct oln::convert::cast< Output, Exact >`

Cast to an output.

Definition at line 40 of file `olena/oln/convert/cast.hh`.

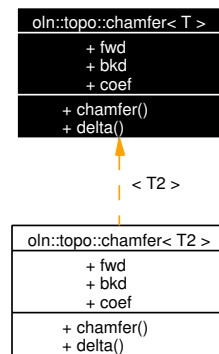
The documentation for this struct was generated from the following file:

- `olena/oln/convert/cast.hh`

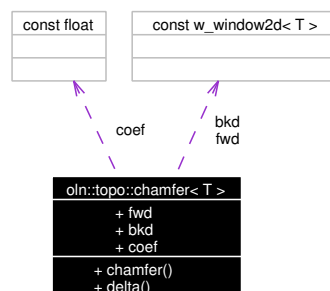
7.37 oln::topo::chamfer< T > Struct Template Reference

```
#include <dmap.hh>
```

Inheritance diagram for oln::topo::chamfer< T >:



Collaboration diagram for oln::topo::chamfer< T >:



Public Member Functions

- **chamfer** (const `w_window2d< T >` &`fwd`, const `w_window2d< T >` &`bkd`, float `coef`)
- **coord delta** () const

Public Attributes

- const `w_window2d< T >` **fwd**
- const `w_window2d< T >` **bkd**
- const float **coef**

7.37.1 Detailed Description

```
template<class T> struct oln::topo::chamfer< T >
```

Chamfer mask.

The Chamfer mask is a weighted masks that provides an approximation of the real Euclidean distance in the discrete space.

Definition at line 55 of file `dmap.hh`.

The documentation for this struct was generated from the following files:

- [dmap.hh](#)
- `dmap.hxx`

7.38 oln::morpho::attr::change_exact< integral_type< T, OldExact >, NewExact > Struct Template Reference

Change the exact type of an attribute.

```
#include <attributes.hh>
```

Public Types

- typedef [integral_type](#)< T, NewExact > **ret**

7.38.1 Detailed Description

```
template<class NewExact, class OldExact, class T> struct oln::morpho::attr::change_exact<  
integral_type< T, OldExact >, NewExact >
```

Change the exact type of an attribute.

Traits to change [integral_type](#) exact type.

Definition at line 1905 of file attributes.hh.

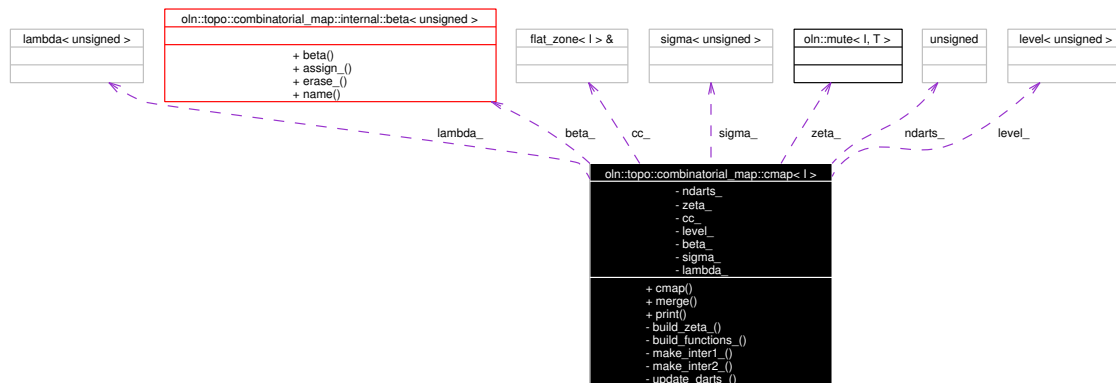
The documentation for this struct was generated from the following file:

- attributes.hh

7.39 oln::topo::combinatorial_map::cmap< I > Class Template Reference

```
#include <cmap.hh>
```

Collaboration diagram for oln::topo::combinatorial_map::cmap< I >:



Public Types

- typedef mlc::exact< I >::ret::dpoint_type **dpoint_type**
- typedef mlc::exact< I >::ret::point_type **point_type**
- typedef oln::topo::inter_pixel::fwd_dir_iter< I::dim > **fwd_dir_iter_type**
- typedef oln::topo::inter_pixel::bkd_dir_iter< I::dim > **bkd_dir_iter_type**
- typedef std::pair< typename mlc::exact< I >::ret::point_type, typename oln::topo::inter_pixel::internal::dir_traits< I::dim >::ret > **head_type**
- typedef oln::mute< I, oln::topo::combinatorial_map::internal::zeta >::ret **zeta_type**

Public Member Functions

- **cmap** (const I &input, const inter_pixel::interpixel< I > &ip, tarjan::obsolete::flat_zone< I > &cc)
- void **merge** (const unsigned l1, const unsigned l2)
Merging algorithm.
- std::ostream & **print** (std::ostream &ostr) const
Print.

7.39.1 Detailed Description

```
template<class I> class oln::topo::combinatorial_map::cmap< I >
```

Combinatorial map.

REF: Braquelaire, J. P. and Brun, L. Image Segmentation with Topological Maps and Inter-pixel Representation}, Journal of Visual Communication and Image representation, 1998, vol. 9

Definition at line 63 of file cmap.hh.

The documentation for this class was generated from the following file:

- [cmap.hh](#)

7.40 oln::morpho::cmp_queue_elt< T > Struct Template Reference

```
#include <watershed.hxx>
```

Public Member Functions

- **bool operator()** (const std::pair< typename mlc::exact< T >::ret::point_type, typename mlc::exact< T >::ret::value_type > &l, const std::pair< typename mlc::exact< T >::ret::point_type, typename mlc::exact< T >::ret::value_type > &r) const

7.40.1 Detailed Description

template<class T> struct oln::morpho::cmp_queue_elt< T >

[cmp_queue_elt](#) is a comparison function for the elements from the priority queue used in watershed_seg_or. Note that we return true when l is greater than r, because we when the queue sorted in increasing order.

Definition at line 301 of file watershed.hxx.

The documentation for this struct was generated from the following file:

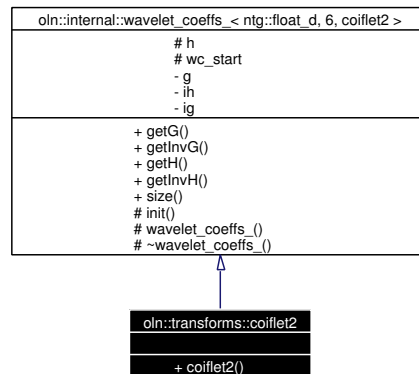
- watershed.hxx

7.41 oln::transforms::coiflet2 Struct Reference

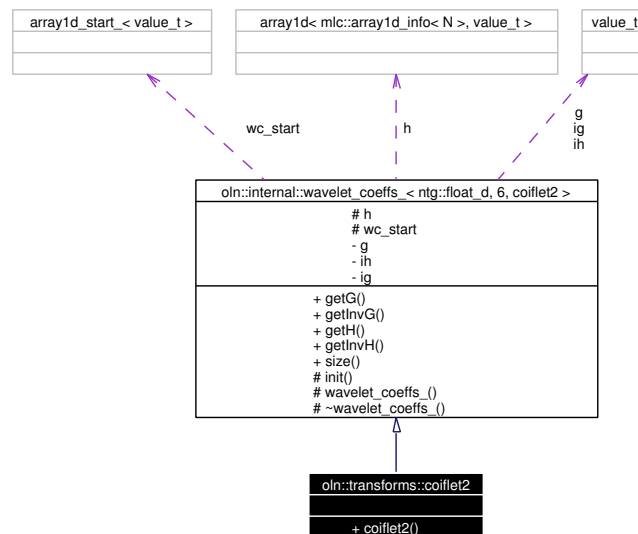
Coifman wavelet coefficients.

```
#include <wavelet_coeffs.hh>
```

Inheritance diagram for oln::transforms::coiflet2:



Collaboration diagram for oln::transforms::coiflet2:



7.41.1 Detailed Description

Coifman wavelet coefficients.

Six coefficients version.

Definition at line 191 of file `wavelet_coeffs.hh`.

The documentation for this struct was generated from the following file:

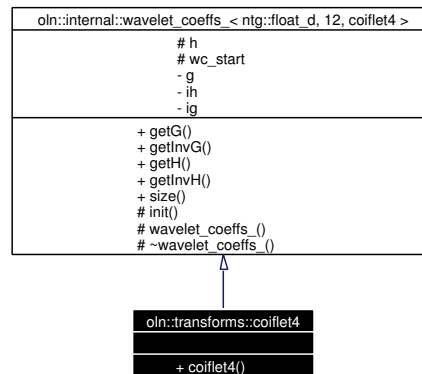
- `wavelet_coeffs.hh`

7.42 oln::transforms::coiflet4 Struct Reference

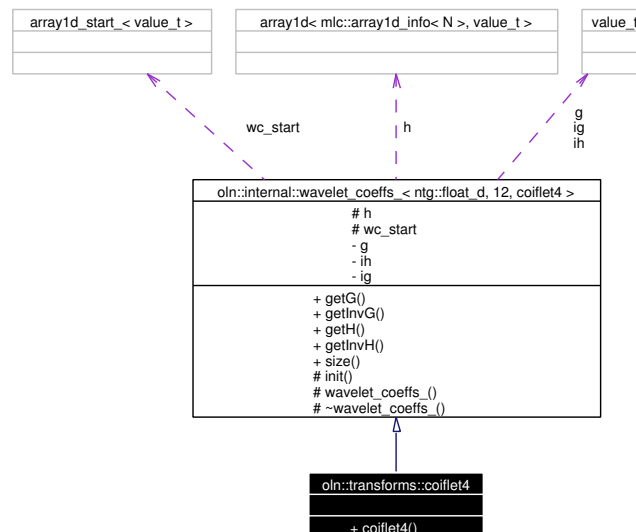
Coifman wavelet coefficients.

```
#include <wavelet_coeffs.hh>
```

Inheritance diagram for oln::transforms::coiflet4:



Collaboration diagram for oln::transforms::coiflet4:



7.42.1 Detailed Description

Coifman wavelet coefficients.

Twelve coefficients version.

Definition at line 211 of file `wavelet_coeffs.hh`.

The documentation for this struct was generated from the following file:

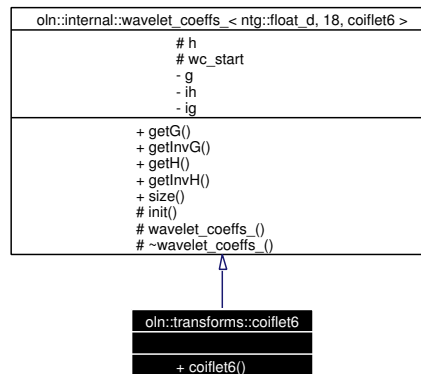
- `wavelet_coeffs.hh`

7.43 oln::transforms::coiflet6 Struct Reference

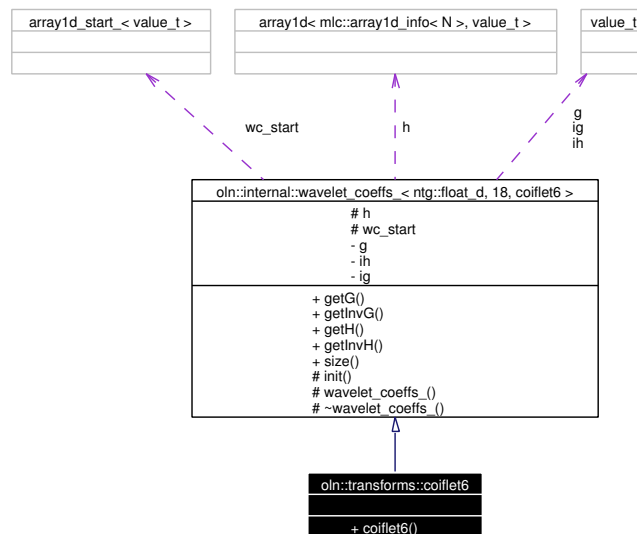
Coifman wavelet coefficients.

```
#include <wavelet_coeffs.hh>
```

Inheritance diagram for oln::transforms::coiflet6:



Collaboration diagram for oln::transforms::coiflet6:



7.43.1 Detailed Description

Coifman wavelet coefficients.

Eighteen coefficients version.

Definition at line 234 of file `wavelet_coeffs.hh`.

The documentation for this struct was generated from the following file:

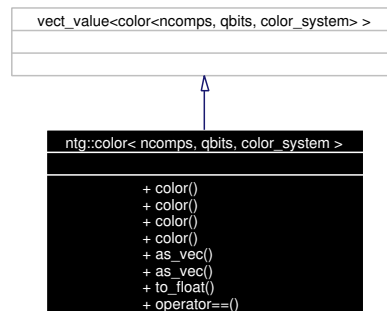
- `wavelet_coeffs.hh`

7.44 ntg::color< ncomps, qbits, color_system > Struct Template Reference

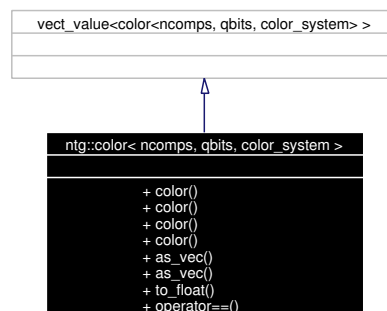
Generic type for color.

```
#include <color.hh>
```

Inheritance diagram for ntg::color< ncomps, qbits, color_system >:



Collaboration diagram for ntg::color< ncomps, qbits, color_system >:



Public Types

- typedef int_u< qbits > **comp_type**
- typedef vec< ncomps, comp_type > **vec_type**
- typedef vec< ncomps, float > **float_vec_type**

Public Member Functions

- **color** (const vec_type &vec)
- **color** (const float_vec_type &vec)
- **color** (const comp_type &c1, const comp_type &c2, const comp_type &c3)
- vec_type & **as_vec** ()
- const vec_type & **as_vec** () const
- float_vec_type **to_float** () const
- bool **operator==** (const color &r) const

7.44.1 Detailed Description

template<unsigned ncomps, unsigned qbits, template< unsigned > class color_system> struct ntg::color< ncomps, qbits, color_system >

Generic type for color.

Specific color types (such as rgb, xyz, etc.) are defined by specifying ncomps, qbits and a color_system trait.

ncomps: number of components. qbits: number of bits of each unsigned integer component. color_system: traits defining the intervals of each component.

Colors are implemented and seen as a vector of components.

Definition at line 182 of file color/color.hh.

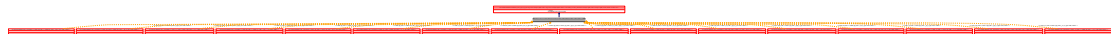
The documentation for this struct was generated from the following file:

- color/color.hh

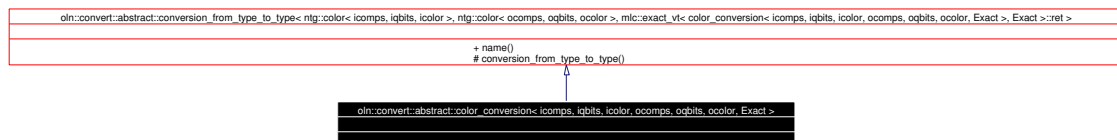
7.45 `oln::convert::abstract::color_conversion< icsmps, iqbits, icolor, ocomps, oqbits, ocolor, Exact > Struct Template Reference`

```
#include <colorconv.hh>
```

Inheritance diagram for `oln::convert::abstract::color_conversion< icsmps, iqbits, icolor, ocomps, oqbits, ocolor, Exact >`:



Collaboration diagram for `oln::convert::abstract::color_conversion< icsmps, iqbits, icolor, ocomps, oqbits, ocolor, Exact >`:



7.45.1 Detailed Description

```
template<unsigned icsmps, unsigned iqbits, template< unsigned > class icolor, unsigned
ocomps, unsigned oqbits, template< unsigned > class ocolor, class Exact = mlc::final> struct
oln::convert::abstract::color_conversion< icsmps, iqbits, icolor, ocomps, oqbits, ocolor, Exact >
```

Base class for color conversion.

Parameters:

- icsmps* Number of components in the input.
- iqbits* Number of bits per components in the input.
- icolor* Input color.
- ocomps* Number of components in the output.
- oqbits* Number of bits per components in the output.
- ocolor* Output color.
- Exact* Exact class.

Definition at line 57 of file `colorconv.hh`.

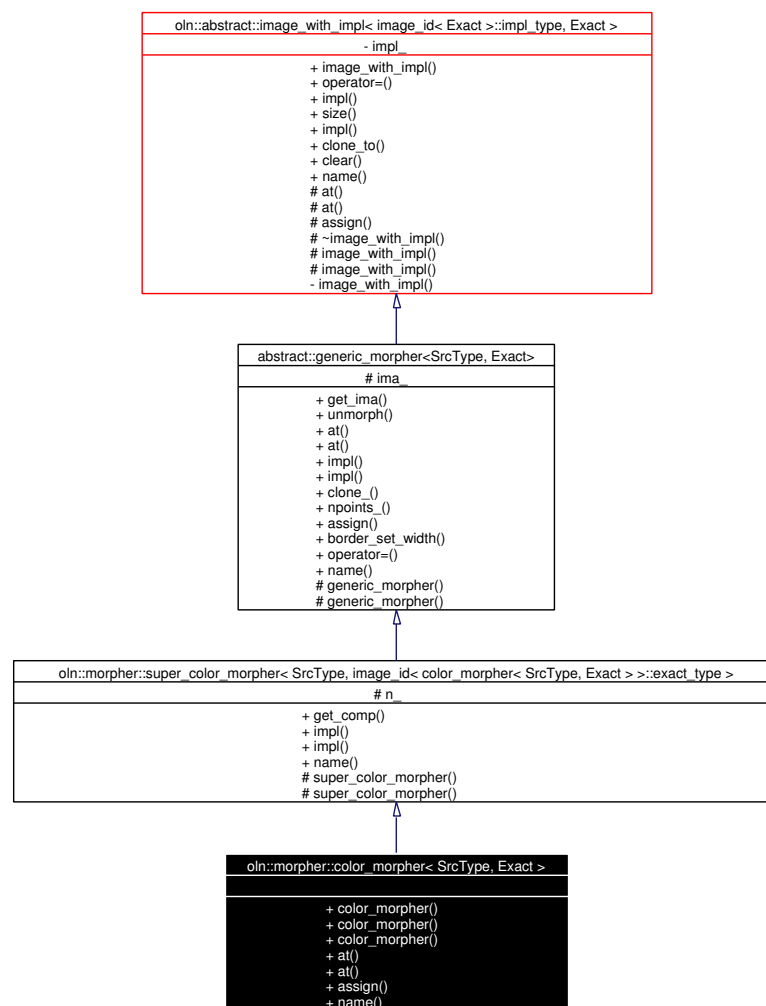
The documentation for this struct was generated from the following file:

- `colorconv.hh`

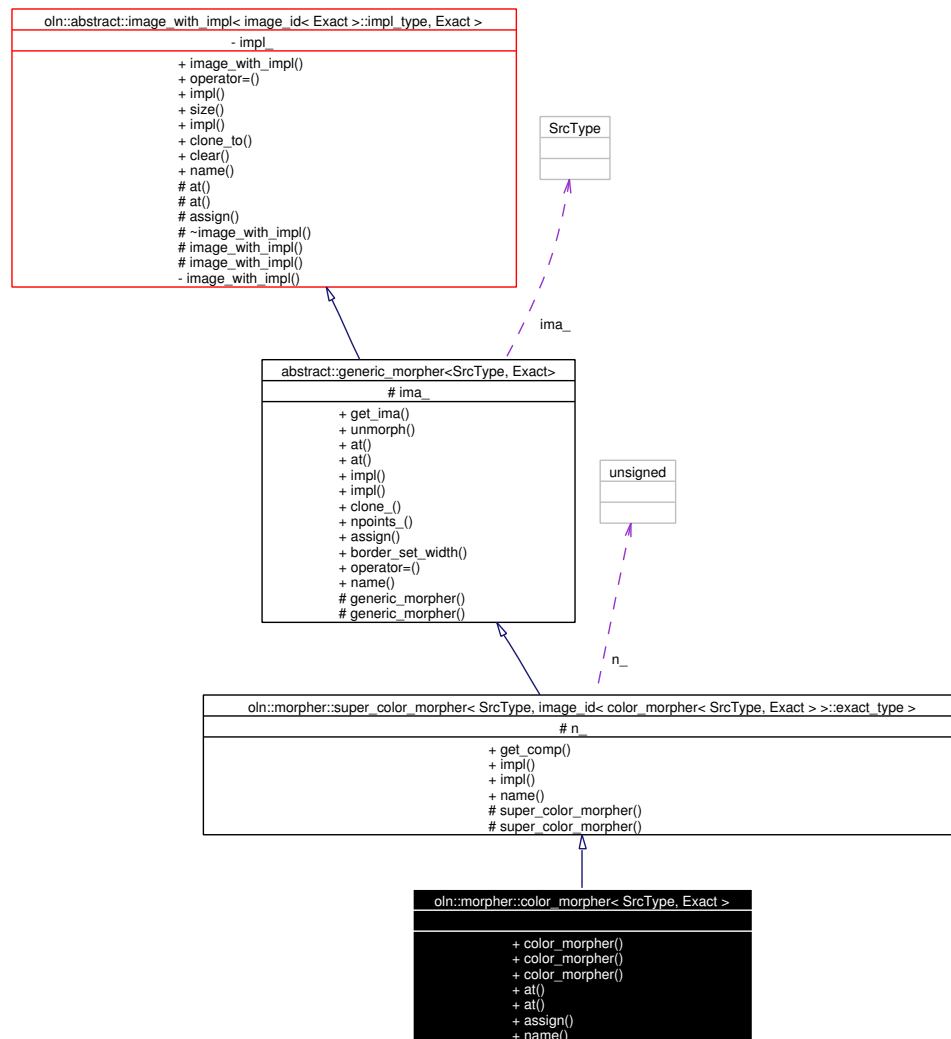
7.46 oln::morpher::color_morpher< SrcType, Exact > Struct Template Reference

```
#include <color_morpher.hh>
```

Inheritance diagram for oln::morpher::color_morpher< SrcType, Exact >:



Collaboration diagram for oln::morpher::color_morpher< SrcType, Exact >:



Public Types

- typedef image_id< [color_morpher](#)< SrcType, Exact > >::exact_type exact_type

The type of the object instantiated. [color_morpher](#) can be derived.

- typedef [color_morpher](#)< SrcType, Exact > self_type

The self type.

- typedef image_id< [exact_type](#) >::iter_type iter_type

The morpher iterator type.

- typedef image_id< [exact_type](#) >::point_type point_type

- typedef image_id< [exact_type](#) >::value_type value_type

- typedef [super_color_morpher](#)< SrcType, exact_type > super_type

Public Member Functions

- [color_morpher](#) (const SrcType &ima, unsigned n)
Construct the [color_morpher](#) with an image ima and a component n.
- [color_morpher](#) (const [color_morpher](#)< SrcType, Exact > &r)
Construct the [color_morpher](#) with another [color_morpher](#).
- [color_morpher](#) ()
- [value_type](#) & at (const [point_type](#) &p)
- const [value_type](#) at (const [point_type](#) &p) const
- [self_type](#) & assign ([self_type](#) &rhs)

Static Public Member Functions

- std::string [name](#) ()

7.46.1 Detailed Description

template<class SrcType, class Exact> struct oln::morpher::color_morpher< SrcType, Exact >

The default [color_morpher](#) class.

Using this class, an rgb image can be viewed according to one of its component.

Parameters:

SrcType Input Type decorated.

Exact Exact type.

See also:

[oln::morpher::abstract::generic_morpher](#)

Definition at line 177 of file color_morpher.hh.

7.46.2 Member Typedef Documentation

7.46.2.1 template<class SrcType, class Exact> typedef image_id<[exact_type](#)>::[point_type](#) [oln::morpher::color_morpher](#)< SrcType, Exact >::[point_type](#)

<Type of the class iterator.

Reimplemented from [oln::morpher::abstract::generic_morpher](#)< SrcType, Exact >.

Definition at line 188 of file color_morpher.hh.

7.46.2.2 template<class SrcType, class Exact> typedef [super_color_morpher](#)<SrcType, [exact_type](#)> [oln::morpher::color_morpher](#)< SrcType, Exact >::[super_type](#)

<The value type of the decorated image.

Reimplemented from [oln::morpher::super_color_morpher](#)< SrcType, Exact >.

Definition at line 192 of file color_morpher.hh.

7.46.2.3 `template<class SrcType, class Exact> typedef image_id<exact_type>::value_type
oln::morpher::color_morpher< SrcType, Exact >::value_type`

<Type of the class point.

Reimplemented from `oln::morpher::abstract::generic_morpher< SrcType, Exact >`.

Definition at line 190 of file `color_morpher.hh`.

7.46.3 Constructor & Destructor Documentation

7.46.3.1 `template<class SrcType, class Exact> oln::morpher::color_morpher< SrcType, Exact
>::color_morpher (const SrcType & ima, unsigned n) [inline]`

Construct the `color_morpher` with an image *ima* and a component *n*.

<The upper class.

Definition at line 196 of file `color_morpher.hh`.

```
196                                     : super_type(ima, n)
197     {}
```

7.46.3.2 `template<class SrcType, class Exact> oln::morpher::color_morpher< SrcType, Exact
>::color_morpher () [inline]`

Empty constructor.

Needed by `mlc_hierarchy::any_with_diamond`.

Definition at line 207 of file `color_morpher.hh`.

```
208     {}
```

7.46.4 Member Function Documentation

7.46.4.1 `template<class SrcType, class Exact> self_type& oln::morpher::color_morpher<
SrcType, Exact >::assign (self_type & rhs) [inline]`

Perform a shallow copy from the decorated image of *rhs* to the current decorated image. The points will be shared by the two images.

Definition at line 238 of file `color_morpher.hh`.

References `oln::morpher::color_morpher< SrcType, Exact >::at()`.

```
239     {
240         oln_iter_type(SrcType)  it(rhs);
241
242         for_all(it)
243             this->at(it) = rhs[it];
244         return this->exact();
245     }
```


7.46.4.2 `template<class SrcType, class Exact> const value_type oln::morpher::color_morpher< SrcType, Exact >::at (const point_type & p) const` `[inline]`

Return the n_{th} component of the rgb value stored at p .

Warning:

This method should not be used directly. Prefer operator[].

Reimplemented from [oln::abstract::image_with_impl< Impl, Exact >](#).

Definition at line 228 of file color_morpher.hh.

```
229     {  
230         return this->ima_[p][this->n_];  
231     }
```

7.46.4.3 `template<class SrcType, class Exact> value_type& oln::morpher::color_morpher< SrcType, Exact >::at (const point_type & p)` `[inline]`

Return a reference to the n_{th} component of the rgb value stored at p .

Warning:

This method should not be used directly. Prefer operator[].

Reimplemented from [oln::abstract::image_with_impl< Impl, Exact >](#).

Definition at line 217 of file color_morpher.hh.

Referenced by `oln::morpher::color_morpher< SrcType, Exact >::assign()`.

```
218     {  
219         return const_cast<SrcType &>(this->ima_)[p][this->n_];  
220     }
```

The documentation for this struct was generated from the following file:

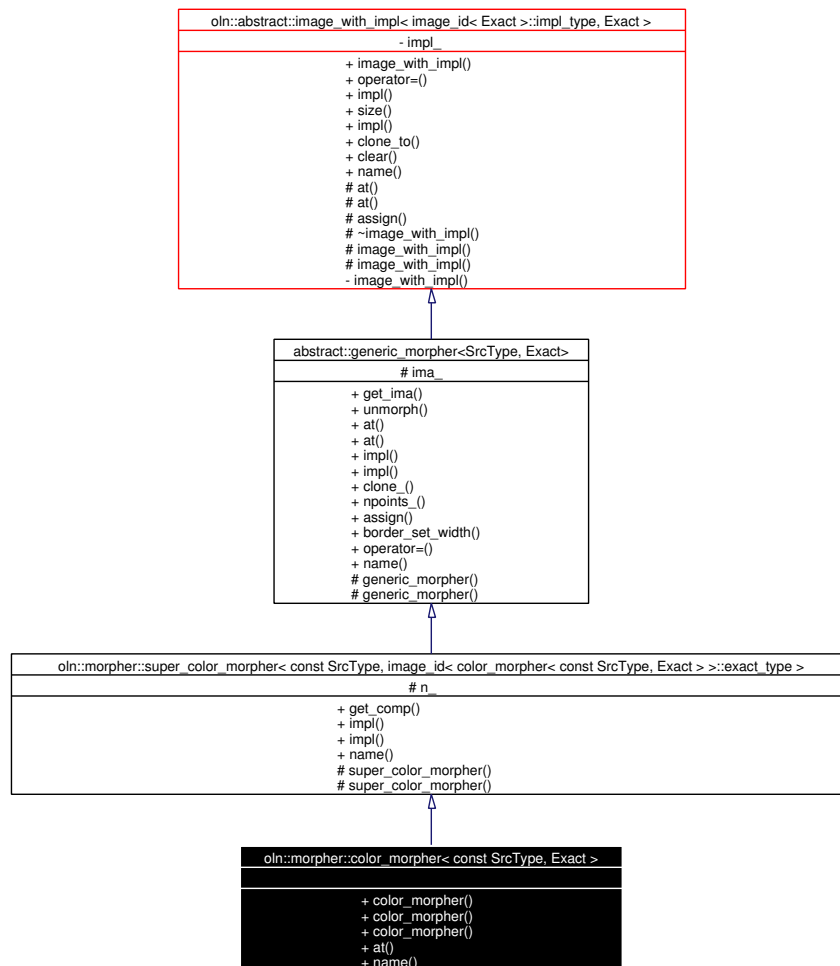
- color_morpher.hh

7.47 oln::morpher::color_morpher< const SrcType, Exact > Struct Template Reference

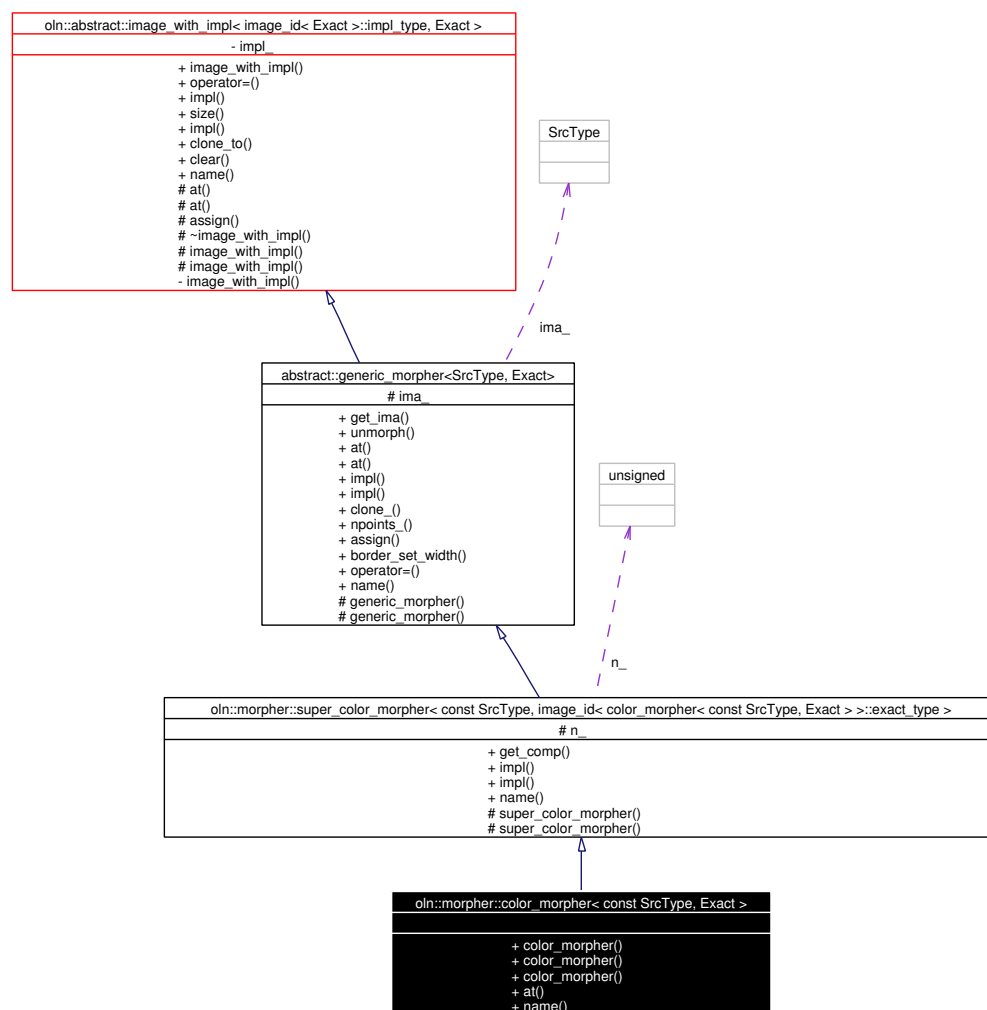
The specialized version for 'const' declared images.

```
#include <color_morpher.hh>
```

Inheritance diagram for oln::morpher::color_morpher< const SrcType, Exact >:



Collaboration diagram for oln::morpher::color_morpher< const SrcType, Exact >:



Public Types

- `typedef image_id< color_morpher< const SrcType, Exact >::exact_type exact_type`
The type of the object instantiated. *color_morpher* can be derived.
- `typedef image_id< exact_type >::point_type point_type`
The morpher point type.
- `typedef image_id< exact_type >::iter_type iter_type`
- `typedef image_id< exact_type >::value_type value_type`
- `typedef super_color_morpher< const SrcType, exact_type > super_type`

Public Member Functions

- `color_morpher` (`const SrcType &ima, unsigned n`)
Construct the *color_morpher* with an image *ima* and a component *n*.

- [color_morpher](#) (const [color_morpher](#)< const SrcType, Exact > &r)
Construct the [color_morpher](#) with another [color_morpher](#).
- [color_morpher](#) ()
- const [value_type](#) at (const [point_type](#) &p) const

Static Public Member Functions

- std::string [name](#) ()

7.47.1 Detailed Description

template<class SrcType, class Exact> struct oln::morpher::color_morpher< const SrcType, Exact >

The specialized version for 'const' declared images.

Parameters:

SrcType Input type decorated.

Exact Exact Type.

See also:

[oln::morpher::abstract::generic_morpher](#)

Definition at line 264 of file color_morpher.hh.

7.47.2 Member Typedef Documentation

7.47.2.1 template<class SrcType, class Exact> typedef image_id<[exact_type](#)>::[iter_type](#)
[oln::morpher::color_morpher](#)< const SrcType, Exact >::[iter_type](#)

<The type of the class point.

Reimplemented from [oln::morpher::abstract::generic_morpher](#)< SrcType, Exact >.

Definition at line 274 of file color_morpher.hh.

7.47.2.2 template<class SrcType, class Exact> typedef [super_color_morpher](#)<const SrcType,
[exact_type](#)> [oln::morpher::color_morpher](#)< const SrcType, Exact >::[super_type](#)

<The value of the decorated image.

Reimplemented from [oln::morpher::super_color_morpher](#)< SrcType, Exact >.

Definition at line 279 of file color_morpher.hh.

7.47.2.3 template<class SrcType, class Exact> typedef image_id<[exact_type](#)>::[value_type](#)
[oln::morpher::color_morpher](#)< const SrcType, Exact >::[value_type](#)

<The type of the class iterator.

Reimplemented from [oln::morpher::abstract::generic_morpher< SrcType, Exact >](#).

Definition at line 276 of file color_morpher.hh.

7.47.3 Member Function Documentation

7.47.3.1 `template<class SrcType, class Exact> const value_type oln::morpher::color_morpher< const SrcType, Exact >::at (const point_type & p) const` [inline]

Return the n_{th} component of the rgb value stored at p .

Warning:

This method should not be used directly. Prefer operator[].

Reimplemented from [oln::abstract::image_with_impl< Impl, Exact >](#).

Definition at line 304 of file color_morpher.hh.

```
305     {
306         return this->ima_[p][this->n_];
307     }
```

7.47.3.2 `template<class SrcType, class Exact> oln::morpher::color_morpher< const SrcType, Exact >::color_morpher ()` [inline]

Empty constructor.

Needed by `mlc_hierarchy::any_with_diamond`.

Definition at line 295 of file color_morpher.hh.

```
296     {}
```

7.47.3.3 `template<class SrcType, class Exact> oln::morpher::color_morpher< const SrcType, Exact >::color_morpher (const SrcType & ima, unsigned n)` [inline]

Construct the `color_morpher` with an image *ima* and a component n .

<The upper class.

Definition at line 283 of file color_morpher.hh.

```
283                                     : super_type(ima, n)
284     {}
```

The documentation for this struct was generated from the following file:

- color_morpher.hh

7.48 oln::morpher::color_mute< T, N > Struct Template Reference

```
#include <subq_morpher.hh>
```

7.48.1 Detailed Description

```
template<class T, unsigned N> struct oln::morpher::color_mute< T, N >
```

Change the color depth of *T*.

For Example, calling [color_mute](#) with `color<3, 8, rgb_traits>`, 5 will give the type : `color<3, 5, rgb_traits>`.

Parameters:

- T* The data type of the image.
- N* The new number of bits by component.

Definition at line 56 of file `subq_morpher.hh`.

The documentation for this struct was generated from the following file:

- `subq_morpher.hh`

7.49 oln::morpher::color_mute< ntg::color< nbcomps_, nbits_, color_system >, N > Struct Template Reference

Specialized version for [ntg::color](#).

```
#include <subq_morpher.hh>
```

Public Types

- typedef [ntg::color](#)< nbcomps_, N, color_system > **ret**
- enum { **nbcomps** = nbcomps_ }

7.49.1 Detailed Description

template<unsigned nbcomps_, unsigned nbits_, template< unsigned > class color_system, unsigned N> struct oln::morpher::color_mute< ntg::color< nbcomps_, nbits_, color_system >, N >

Specialized version for [ntg::color](#).

Definition at line 65 of file subq_morpher.hh.

7.49.2 Member Enumeration Documentation

7.49.2.1 template<unsigned nbcomps_, unsigned nbits_, template< unsigned > class color_system, unsigned N> anonymous enum

<The new value type.

Definition at line 69 of file subq_morpher.hh.

```
69 { nbcomps = nbcomps_ };
```

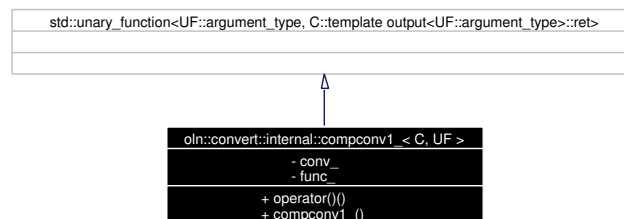
The documentation for this struct was generated from the following file:

- subq_morpher.hh

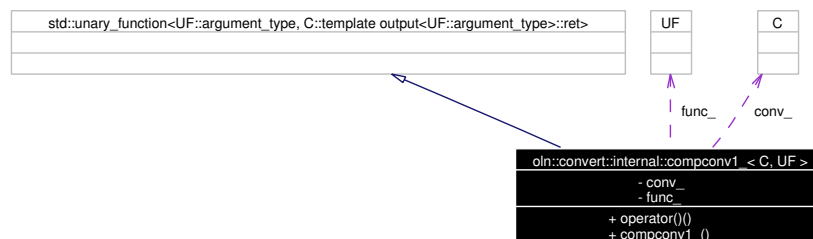
7.50 oln::convert::internal::compconv1_< C, UF > Struct Template Reference

```
#include <conversion.hh>
```

Inheritance diagram for oln::convert::internal::compconv1_< C, UF >:



Collaboration diagram for oln::convert::internal::compconv1_< C, UF >:



Public Types

- typedef [compconv1_self_type](#)

Public Member Functions

- self_type::result_type **operator()** (typename self_type::argument_type arg) const
- **compconv1_** (const C &conv, const UF &func)

7.50.1 Detailed Description

```
template<class C, class UF> struct oln::convert::internal::compconv1_< C, UF >
```

Compose a conversion C and an adaptable unary function UF, producing an adaptable unary function.

Definition at line 66 of file conversion.hh.

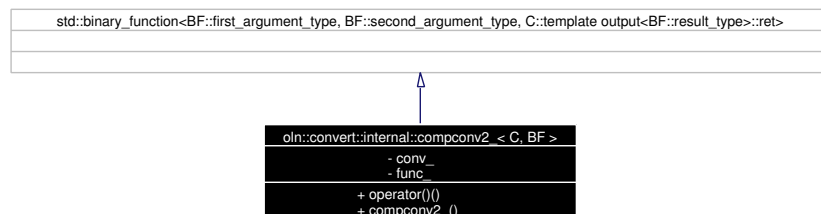
The documentation for this struct was generated from the following file:

- conversion.hh

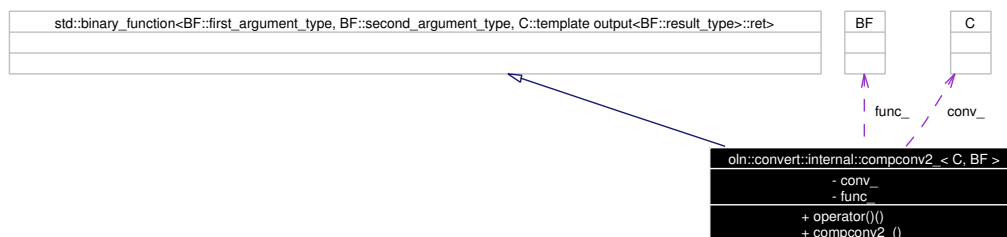
7.51 oln::convert::internal::compconv2_< C, BF > Struct Template Reference

```
#include <conversion.hh>
```

Inheritance diagram for oln::convert::internal::compconv2_< C, BF >:



Collaboration diagram for oln::convert::internal::compconv2_< C, BF >:



Public Types

- typedef `compconv2_ self_type`

Public Member Functions

- `self_type::result_type operator()` (typename `self_type::first_argument_type` arg1, typename `self_type::second_argument_type` arg2) const
- `compconv2_` (const C &conv, const BF &func)

7.51.1 Detailed Description

```
template<class C, class BF> struct oln::convert::internal::compconv2_< C, BF >
```

Compose a conversion C and an adaptable binary function BF, producing an adaptable binary function.

Definition at line 90 of file `conversion.hh`.

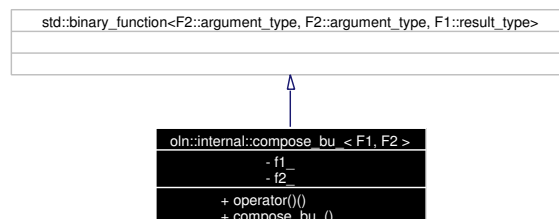
The documentation for this struct was generated from the following file:

- `conversion.hh`

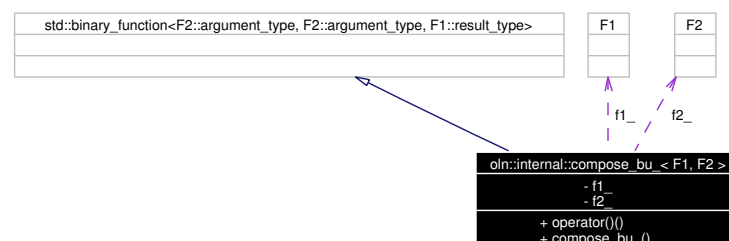
7.52 oln::internal::compose_bu_< F1, F2 > Struct Template Reference

```
#include <compose.hh>
```

Inheritance diagram for oln::internal::compose_bu_< F1, F2 >:



Collaboration diagram for oln::internal::compose_bu_< F1, F2 >:



Public Types

- typedef [compose_bu_self_type](#)

Public Member Functions

- self_type::result_type **operator()** (typename self_type::first_argument_type arg1, typename self_type::second_argument_type arg2) const
- **compose_bu_** (const F1 &f1, const F2 &f2)

7.52.1 Detailed Description

```
template<class F1, class F2> struct oln::internal::compose_bu_< F1, F2 >
```

The operator () of this class performs a composition between a binary functor *F1* and an unary functor *F2*.
Definition at line 103 of file compose.hh.

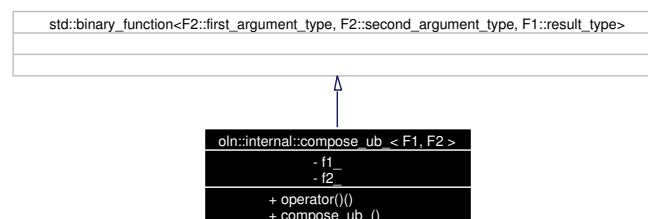
The documentation for this struct was generated from the following file:

- compose.hh

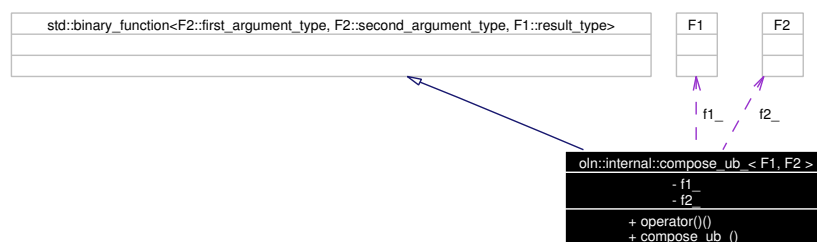
7.53 oln::internal::compose_ub_< F1, F2 > Struct Template Reference

```
#include <compose.hh>
```

Inheritance diagram for oln::internal::compose_ub_< F1, F2 >:



Collaboration diagram for oln::internal::compose_ub_< F1, F2 >:



Public Types

- typedef [compose_ub_ self_type](#)

Public Member Functions

- self_type::result_type **operator()** (typename self_type::first_argument_type arg1, typename self_type::second_argument_type arg2) const
- **compose_ub_** (const F1 &f1, const F2 &f2)

7.53.1 Detailed Description

```
template<class F1, class F2> struct oln::internal::compose_ub_< F1, F2 >
```

The operator () of this class performs a composition between a unary functor *F1* and a binary functor *F2*.

Definition at line 73 of file compose.hh.

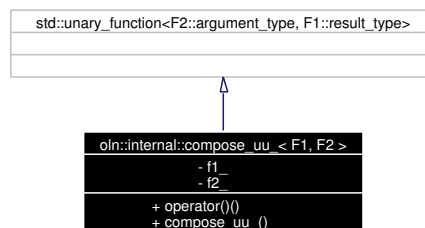
The documentation for this struct was generated from the following file:

- compose.hh

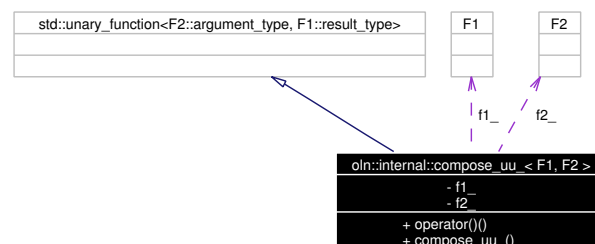
7.54 oln::internal::compose_uu_< F1, F2 > Struct Template Reference

```
#include <compose.hh>
```

Inheritance diagram for oln::internal::compose_uu_< F1, F2 >:



Collaboration diagram for oln::internal::compose_uu_< F1, F2 >:



Public Types

- typedef `compose_uu_self_type`

Public Member Functions

- self_type::result_type **operator()** (typename self_type::argument_type arg) const
- **compose_uu_** (const F1 &f1, const F2 &f2)

7.54.1 Detailed Description

```
template<class F1, class F2> struct oln::internal::compose_uu_< F1, F2 >
```

The operator () of this class performs a composition between two unary functors *F1* & *F2*.

Definition at line 44 of file compose.hh.

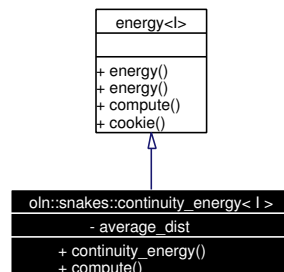
The documentation for this struct was generated from the following file:

- compose.hh

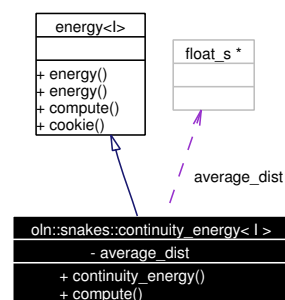
7.55 oln::snakes::continuity_energy< I > Class Template Reference

```
#include <energies.hh>
```

Inheritance diagram for oln::snakes::continuity_energy< I >:



Collaboration diagram for oln::snakes::continuity_energy< I >:



Public Types

- typedef I **image_type**

Public Member Functions

- **continuity_energy** (ntg::float_s *average_dist)
- ntg::float_s **compute** (const I &, const **node**< I > &prev, const **node**< I > ¤t, const **node**< I > &)

7.55.1 Detailed Description

```
template<class I> class oln::snakes::continuity_energy< I >
```

Energy of continuity.

The goal of this energy is to avoid pack of nodes and lack of nodes in some part of the snake. The average distance between two consecutive points is *average_dist*. The more the distance between *prev* and *current* is far from *average_dist*, the higher the energy is.

Definition at line 79 of file energies.hh.

7.55.2 Member Function Documentation

7.55.2.1 `template<class I> ntg::float_s oln::snakes::continuity_energy<I>::compute (const I &, const node<I> & prev, const node<I> & current, const node<I> &) [inline]`

Return the energy.

The first arg is the gradient of the image; the 3 nodes are the previous, the current and the next node.

Reimplemented from [oln::snakes::energy<I>](#).

Definition at line 39 of file energies.hxx.

```
43     {  
44         ntg::float_s d = *average_dist - (current - prev).norm2();  
45         return d > 0 ? d : -d;  
46     }
```

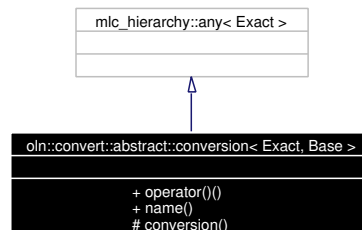
The documentation for this class was generated from the following files:

- energies.hh
- energies.hxx

7.56 oln::convert::abstract::conversion< Exact, Base > Struct Template Reference

```
#include <conversion.hh>
```

Inheritance diagram for oln::convert::abstract::conversion< Exact, Base >: Collaboration diagram for oln::convert::abstract::conversion< Exact, Base >:



Public Member Functions

- `template<class T> output< T >::ret operator() (const T &in) const`
Call the conversion written in the exact class.

Static Public Member Functions

- `std::string name ()`

7.56.1 Detailed Description

```
template<class Exact, class Base> struct oln::convert::abstract::conversion< Exact, Base >
```

Base class for conversion.

Note:

If you write an class derived from this one, you must write the specialization of the output trait.

Definition at line 88 of file `abstract/conversion.hh`.

The documentation for this struct was generated from the following file:

- `abstract/conversion.hh`

7.57 oln::convert::abstract::conversion_from_type_to_type<Argument_Type, Result_Type, Exact, Base > Struct Template Reference

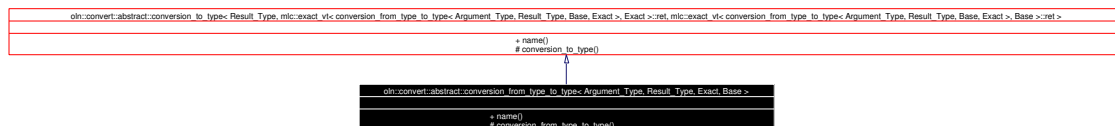
Base class if both input and output types of the conversion are fixed.

```
#include <conversion.hh>
```

Inheritance diagram for oln::convert::abstract::conversion_from_type_to_type<Argument_Type, Result_Type, Exact, Base >:



Collaboration diagram for oln::convert::abstract::conversion_from_type_to_type<Argument_Type, Result_Type, Exact, Base >:



Public Types

- typedef Argument_Type **argument_type**

Static Public Member Functions

- std::string **name** ()

7.57.1 Detailed Description

```
template<class Argument_Type, class Result_Type, class Exact = mlc::final, class Base = mlc::final>
struct oln::convert::abstract::conversion_from_type_to_type<Argument_Type, Result_Type, Exact, Base >
```

Base class if both input and output types of the conversion are fixed.

Definition at line 146 of file abstract/conversion.hh.

The documentation for this struct was generated from the following file:

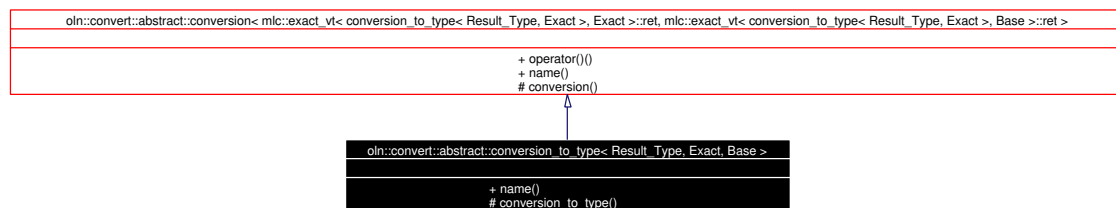
- abstract/conversion.hh

7.58 oln::convert::abstract::conversion_to_type< Result_Type, Exact, Base > Struct Template Reference

Base class for the conversion to a specific type.

```
#include <conversion.hh>
```

Inheritance diagram for oln::convert::abstract::conversion_to_type< Result_Type, Exact, Base >: Collaboration diagram for oln::convert::abstract::conversion_to_type< Result_Type, Exact, Base >:



Public Types

- `typedef Result_Type result_type`

Static Public Member Functions

- `std::string name ()`

7.58.1 Detailed Description

```
template<class Result_Type, class Exact = mlc::final, class Base = mlc::final> struct
oln::convert::abstract::conversion_to_type< Result_Type, Exact, Base >
```

Base class for the conversion to a specific type.

Definition at line 117 of file abstract/conversion.hh.

The documentation for this struct was generated from the following file:

- abstract/conversion.hh

7.59 oln::convert::convoutput< ConvType, Base, InputType > Struct Template Reference

```
#include <conversion.hh>
```

Public Types

- typedef [abstract::conversion](#)< ConvType, Base >::template output< InputType >::ret ret

7.59.1 Detailed Description

template<class ConvType, class Base, class InputType> struct oln::convert::convoutput< ConvType, Base, InputType >

Trait that returns the output of a conversion.

convoutput queries the output type of conversion ConvType for an input of type InputType. This comes handy when computing the return type of a function which takes a conversion function in argument.

Note:

convoutput is exported in the namespace oln for convenience.

Definition at line 55 of file conversion.hh.

The documentation for this struct was generated from the following file:

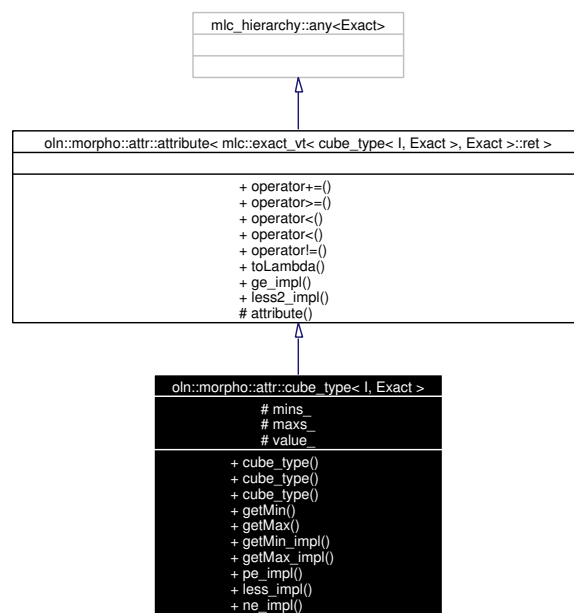
- conversion.hh

7.60 oln::morpho::attr::cube_type< I, Exact > Class Template Reference

Cube attribute.

```
#include <attributes.hh>
```

Inheritance diagram for oln::morpho::attr::cube_type< I, Exact >:



Collaboration diagram for oln::morpho::attr::cube_type< I, Exact >:



Public Types

- typedef `cube_type< I, Exact >` `self_type`
Self type of the class.
- typedef `mlc::exact_vt< self_type, Exact >::ret` `exact_type`
- typedef `oln::morpho::attr::attr_traits< exact_type >::value_type` `value_type`
- typedef `oln::morpho::attr::attr_traits< exact_type >::env_type` `env_type`
- typedef `oln::morpho::attr::attr_traits< exact_type >::lambda_type` `lambda_type`
- typedef `abstract::image< typename mlc::exact< I >::ret >` `im_type`
Image type.
- typedef `mlc::exact< im_type >::ret::point_type` `point_type`
Point type associated to im_type.
- typedef `mlc::exact< im_type >::ret::dpoint_type` `dpoint_type`
Dpoint type associated to im_type.
- enum { **dim** = `point_traits<point_type>::dim` }

Public Member Functions

- `cube_type()`
Basic Ctor.
- `cube_type` (const `lambda_type` &`lambda`)
Ctor from a lambda_type value.

- `cube_type` (const `im_type` &, const `point_type` &p, const `env_type` &)
Ctor from a point and an image.
- `int getMin` (int i) const
Accessor to minimums.
- `int getMax` (int i) const
Accessor to maximums.
- `int getMin_impl` (int i) const
Implementation of getMin(int i).
- `int getMax_impl` (int i) const
Implementation of getMax(int i).
- `void pe_impl` (const `cube_type` &rhs)
+= operator implementation.
- `bool less_impl` (const `lambda_type` &lambda) const
"<" operator implementation.
- `bool ne_impl` (const `lambda_type` &lambda) const
!= operator implementation.

Protected Attributes

- `std::vector< coord > mins_`
- `std::vector< coord > maxs_`
- `value_type value_`

7.60.1 Detailed Description

`template<class I, class Exact = mlc::final> class oln::morpho::attr::cube_type< I, Exact >`

Cube attribute.

Parameters:

I Exact type of images to process.

Exact The exact type.

Definition at line 1434 of file attributes.hh.

7.60.2 Constructor & Destructor Documentation

7.60.2.1 `template<class I, class Exact = mlc::final> oln::morpho::attr::cube_type< I, Exact >::cube_type () [inline]`

Basic Ctor.

Warning:

After this call, the object is only instantiated (not initialized).

Definition at line 1452 of file attributes.hh.

```
1453         {
1454     }
```

7.60.2.2 `template<class I, class Exact = mlc::final> oln::morpho::attr::cube_type< I, Exact >::cube_type (const lambda_type & lambda)` `[inline]`

Ctor from a `lambda_type` value.

- `lambda` Value of the attribute.

Definition at line 1461 of file attributes.hh.

References `oln::coord`.

```
1461                                     :
1462         mins_(dim),
1463         maxs_(dim),
1464         value_(lambda)
1465     {
1466         for (int i = 0; i < point_traits<point_type>::dim; ++i)
1467         {
1468             maxs_[i] = lambda;
1469             mins_[i] = ntg_zero_val(coord);
1470         }
1471     };
```

7.60.2.3 `template<class I, class Exact = mlc::final> oln::morpho::attr::cube_type< I, Exact >::cube_type (const im_type &, const point_type & p, const env_type &)` `[inline]`

Ctor from a point and an image.

- `p` Point to consider in the image.

Definition at line 1478 of file attributes.hh.

```
1480                                     :
1481         mins_(dim), maxs_(dim), value_(ntg_zero_val(value_type))
1482     {
1483         for (int i = 0; i < dim; ++i)
1484             mins_[i] = maxs_[i] = p.nth(i);
1485     }
```

7.60.3 Member Function Documentation

7.60.3.1 `template<class I, class Exact = mlc::final> int oln::morpho::attr::cube_type< I, Exact >::getMax (int i) const` `[inline]`

Accessor to maximums.

Virtual method.

- i Index of the minimum wanted.

Returns:

the i th maximum.

See also:

[getMax_impl\(\)](#)

Definition at line 1508 of file attributes.hh.

Referenced by oln::morpho::attr::cube_type< I, Exact >::pe_impl().

```
1509         {
1510             mlc_dispatch(getMax)(i);
1511         };
```

7.60.3.2 `template<class I, class Exact = mlc::final> int oln::morpho::attr::cube_type< I, Exact >::getMax_impl (int i) const` [inline]

Implementation of getMax(int i).

Override this method in order to provide a new version of getMax(int i).

Warning:

Do not call this method, use [getMax\(\)](#) instead.

Definition at line 1536 of file attributes.hh.

```
1537         {
1538             precondition(i < dim);
1539             return maxs_[i];
1540         };
```

7.60.3.3 `template<class I, class Exact = mlc::final> int oln::morpho::attr::cube_type< I, Exact >::getMin (int i) const` [inline]

Accessor to minimums.

Virtual method.

- i Index of the minimum wanted.

Returns:

the i th minimum.

See also:

[getMin_impl\(\)](#)

Definition at line 1495 of file attributes.hh.

Referenced by oln::morpho::attr::cube_type< I, Exact >::pe_impl().

```
1496         {
1497             mlc_dispatch(getMin)(i);
1498         };
```

7.60.3.4 `template<class I, class Exact = mlc::final> int oln::morpho::attr::cube_type< I, Exact >::getMin_impl (int i) const [inline]`

Implementation of getMin(int i).

Override this method in order to provide a new version of getMin(int i).

Warning:

Do not call this method, use [getMin\(\)](#) instead.

Definition at line 1522 of file attributes.hh.

```
1523         {
1524             precondition(i < dim);
1525             return mins_[i];
1526         };
```

7.60.3.5 `template<class I, class Exact = mlc::final> bool oln::morpho::attr::cube_type< I, Exact >::less_impl (const lambda_type & lambda) const [inline]`

"<" operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 1569 of file attributes.hh.

```
1570         {
1571             return value_ < lambda;
1572         };
```

7.60.3.6 `template<class I, class Exact = mlc::final> bool oln::morpho::attr::cube_type< I, Exact >::ne_impl (const lambda_type & lambda) const [inline]`

!= operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 1581 of file attributes.hh.

```
1582         {
1583             return value_ != lambda;
1584         };
```


7.60.3.7 `template<class I, class Exact = mlc::final> void oln::morpho::attr::cube_type< I, Exact >::pe_impl (const cube_type< I, Exact > & rhs) [inline]`

`+=` operator implementation.

This is an implementation of the `+=` operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 1549 of file attributes.hh.

References `oln::morpho::attr::cube_type< I, Exact >::getMax()`, and `oln::morpho::attr::cube_type< I, Exact >::getMin()`.

```

1550     {
1551         for (int i = 0; i < dim; ++i)
1552         {
1553             mins_[i] = ntg::min(mins_[i], rhs.getMin(i));
1554             maxs_[i] = ntg::max(maxs_[i], rhs.getMax(i));
1555         }
1556         value_ = maxs_[0] - mins_[0];
1557         for (int i = 1; i < dim; ++i)
1558             if (value_ < value_type(maxs_[i] - mins_[i]))
1559                 value_ = maxs_[i] - mins_[i];
1560     }

```

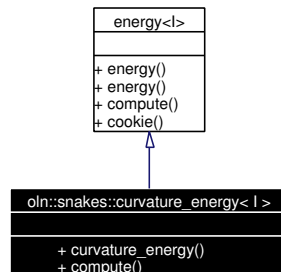
The documentation for this class was generated from the following file:

- attributes.hh

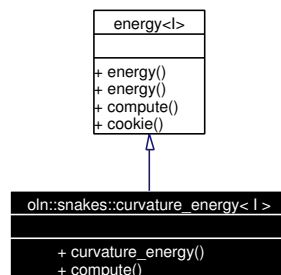
7.61 oln::snakes::curvature_energy< I > Class Template Reference

```
#include <energies.hh>
```

Inheritance diagram for oln::snakes::curvature_energy< I >:



Collaboration diagram for oln::snakes::curvature_energy< I >:



Public Types

- `typedef I image_type`

Public Member Functions

- `curvature_energy (void *)`

Static Public Member Functions

- `ntg::float_s compute (const I &, const node< I > &prev, const node< I > ¤t, const node< I > &next)`

7.61.1 Detailed Description

```
template<class I> class oln::snakes::curvature_energy< I >
```

Energy of curvature.

The snake is supposed to be applied on object that have smooth edge (example: an egg). The more the *prev* *current* *next* nodes are aligned, the less the energy is.

Definition at line 109 of file energies.hh.

7.61.2 Member Function Documentation

7.61.2.1 `template<class I> ntg::float_s oln::snakes::curvature_energy< I >::compute (const I &, const node< I > & prev, const node< I > & current, const node< I > & next) [inline, static]`

Return the energy.

The first arg is the gradient of the image; the 3 nodes are the previous, the current and the next node.

Reimplemented from [oln::snakes::energy< I >](#).

Definition at line 51 of file energies.hxx.

```
55     {
56         typename I::point_type twice_current;
57
58         twice_current.row() = 2 * current.row();
59         twice_current.col() = 2 * current.col();
60         return (next + (prev - twice_current) -
61                 typename I::point_type(0,0)).norm2();
62     }
```

The documentation for this class was generated from the following files:

- energies.hh
- energies.hxx

7.62 ntg::cycle_behavior Struct Reference

Apply a modulus when an overflow occurs.

```
#include <behavior.hh>
```

Static Public Member Functions

- `std::string name ()`

7.62.1 Detailed Description

Apply a modulus when an overflow occurs.

This behavior is not really useful, but implement `cycle<>` internal calculus. You should note that a `range<int_u, ..., cycle_behavior>` is different than `cycle<int_u, ...>`. Refer to the documentation for more details.

Definition at line 342 of file `integre/ntg/real/behavior.hh`.

The documentation for this struct was generated from the following file:

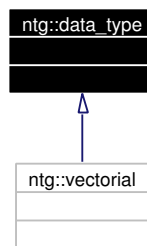
- `integre/ntg/real/behavior.hh`

7.63 ntg::data_type Class Reference

Top of the hierarchy.

```
#include <abstract_hierarchy.hh>
```

Inheritance diagram for ntg::data_type:



7.63.1 Detailed Description

Top of the hierarchy.

Definition at line 47 of file `abstract_hierarchy.hh`.

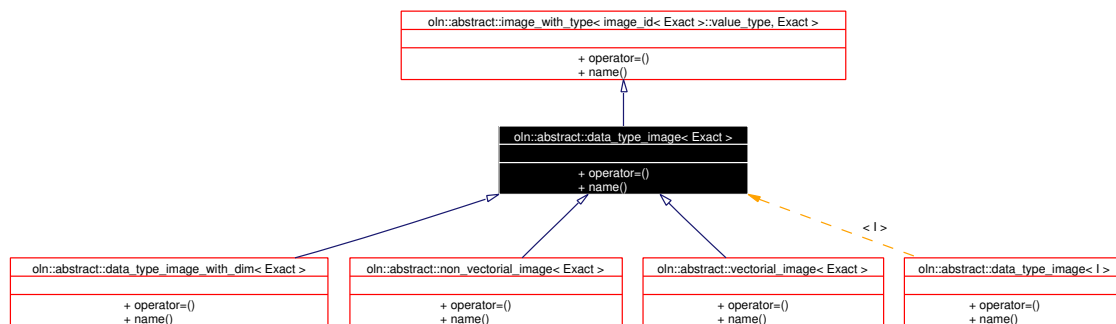
The documentation for this class was generated from the following file:

- `abstract_hierarchy.hh`

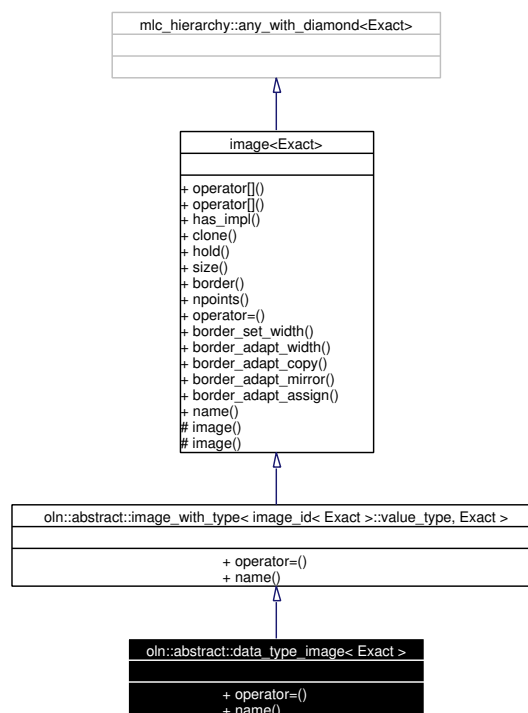
7.64 oln::abstract::data_type_image< Exact > Struct Template Reference

```
#include <image_with_type_with_dim.hh>
```

Inheritance diagram for oln::abstract::data_type_image< Exact >:



Collaboration diagram for oln::abstract::data_type_image< Exact >:



Public Types

- typedef `data_type_image< Exact > self_type`
- typedef `Exact exact_type`

Public Member Functions

- `exact_type & operator= (self_type rhs)`

Perform a shallow copy between rhs and the current point, the points will not be duplicated but shared between the two images.

Static Public Member Functions

- `std::string name ()`

7.64.1 Detailed Description

`template<class Exact> struct oln::abstract::data_type_image< Exact >`

This class is used as a proxy towards [image_with_type](#).

Note:

This shouldn't be used.

Definition at line 115 of file `image_with_type_with_dim.hh`.

7.64.2 Member Function Documentation

7.64.2.1 `template<class Exact> exact_type& oln::abstract::data_type_image< Exact >::operator= (self_type rhs) [inline]`

Perform a shallow copy between *rhs* and the current point, the points will not be duplicated but shared between the two images.

See also:

[image::clone\(\)](#)

Reimplemented from [oln::abstract::image_with_type< T, Exact >](#).

Reimplemented in [oln::abstract::data_type_image_with_dim< Dim, Exact >](#), [oln::abstract::vectorial_image< Exact >](#), [oln::abstract::non_vectorial_image< Exact >](#), [oln::abstract::non_vectorial_image_with_dim< Dim, Exact >](#), [oln::abstract::binary_image< Exact >](#), [oln::abstract::binary_image_with_dim< Dim, Exact >](#), [oln::abstract::integer_image< Exact >](#), [oln::abstract::integer_image_with_dim< Dim, Exact >](#), [oln::abstract::decimal_image< Exact >](#), [oln::abstract::decimal_image_with_dim< Dim, Exact >](#), and [oln::abstract::non_vectorial_image< I >](#).

Definition at line 132 of file `image_with_type_with_dim.hh`.

```

133     {
134         return this->exact().assign(rhs.exact());
135     }
```

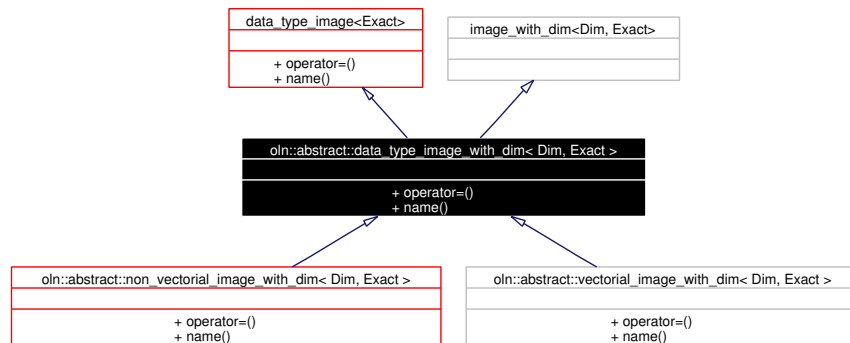
The documentation for this struct was generated from the following file:

- `image_with_type_with_dim.hh`

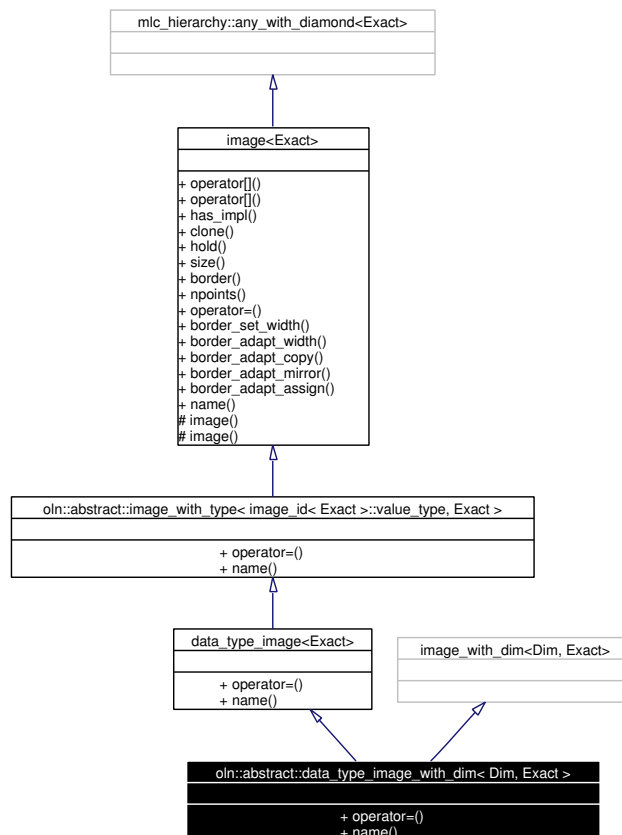
7.65 oln::abstract::data_type_image_with_dim< Dim, Exact > Struct Template Reference

```
#include <image_with_type_with_dim.hh>
```

Inheritance diagram for oln::abstract::data_type_image_with_dim< Dim, Exact >:



Collaboration diagram for oln::abstract::data_type_image_with_dim< Dim, Exact >:



Public Types

- typedef [data_type_image](#)< Exact > **self_type**
- typedef Exact **exact_type**

Public Member Functions

- **exact_type** & [operator=](#) (self_type rhs)
Perform a shallow copy between rhs and the current point, the points will not be duplicated but shared between the two images.

Static Public Member Functions

- std::string **name** ()

7.65.1 Detailed Description

template<unsigned Dim, class Exact> **struct** oln::abstract::data_type_image_with_dim< Dim, Exact >

This class when used in a function declaration, force the function to only accept images with a dimension equal to *Dim*.

Definition at line 156 of file image_with_type_with_dim.hh.

7.65.2 Member Function Documentation

7.65.2.1 **template**<unsigned Dim, class Exact> **exact_type**& [olin::abstract::data_type_image_with_dim](#)< Dim, Exact >::operator= (self_type rhs)
 [inline]

Perform a shallow copy between *rhs* and the current point, the points will not be duplicated but shared between the two images.

See also:

[image::clone\(\)](#)

Reimplemented from [olin::abstract::data_type_image](#)< Exact >.

Reimplemented in [olin::abstract::non_vectorial_image_with_dim](#)< Dim, Exact >, [olin::abstract::binary_image_with_dim](#)< Dim, Exact >, [olin::abstract::integer_image_with_dim](#)< Dim, Exact >, and [olin::abstract::decimal_image_with_dim](#)< Dim, Exact >.

Definition at line 173 of file image_with_type_with_dim.hh.

```

174      {
175      return this->exact().assign(rhs.exact());
176      }
```

The documentation for this struct was generated from the following file:

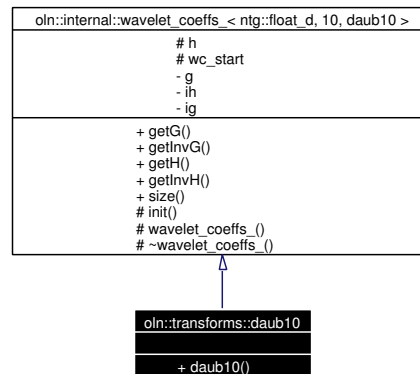
- image_with_type_with_dim.hh

7.66 oln::transforms::daub10 Struct Reference

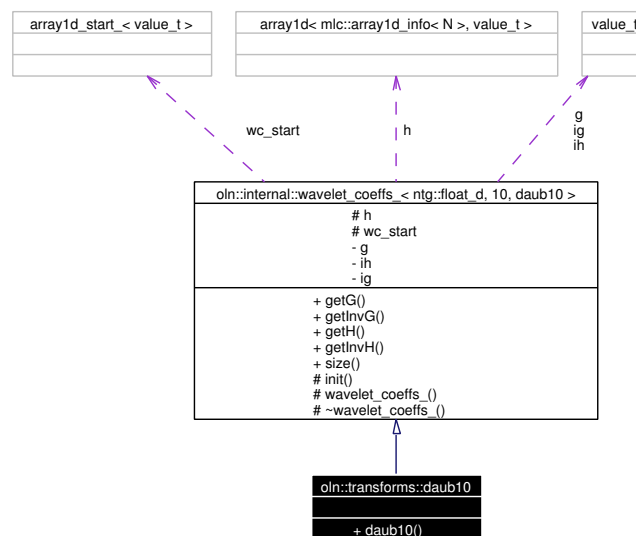
Daubechies wavelet coefficients.

```
#include <wavelet_coeffs.hh>
```

Inheritance diagram for oln::transforms::daub10:



Collaboration diagram for oln::transforms::daub10:



7.66.1 Detailed Description

Daubechies wavelet coefficients.

Ten coefficients version.

Definition at line 115 of file `wavelet_coeffs.hh`.

The documentation for this struct was generated from the following file:

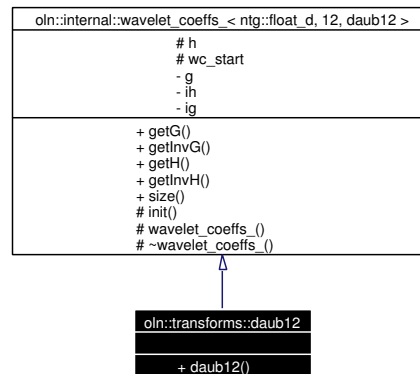
- `wavelet_coeffs.hh`

7.67 oln::transforms::daub12 Struct Reference

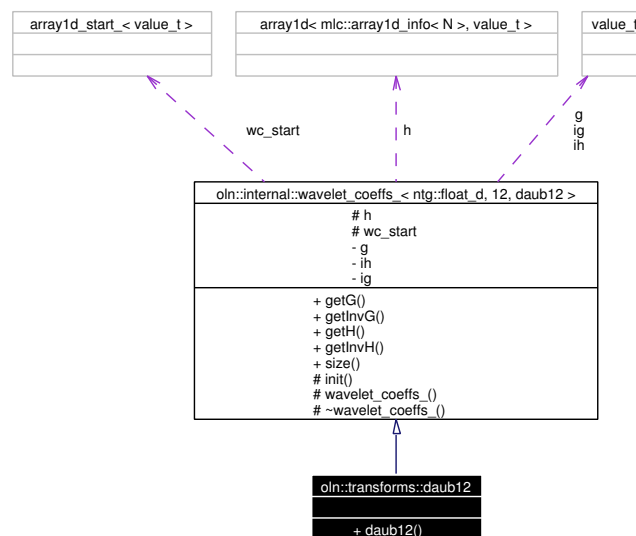
Daubechies wavelet coefficients.

```
#include <wavelet_coeffs.hh>
```

Inheritance diagram for oln::transforms::daub12:



Collaboration diagram for oln::transforms::daub12:



7.67.1 Detailed Description

Daubechies wavelet coefficients.

Twelve coefficients version.

Definition at line 136 of file `wavelet_coeffs.hh`.

The documentation for this struct was generated from the following file:

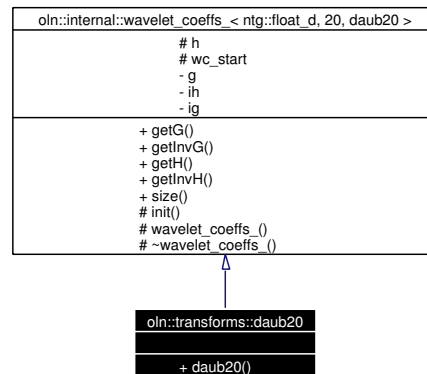
- `wavelet_coeffs.hh`

7.68 oln::transforms::daub20 Struct Reference

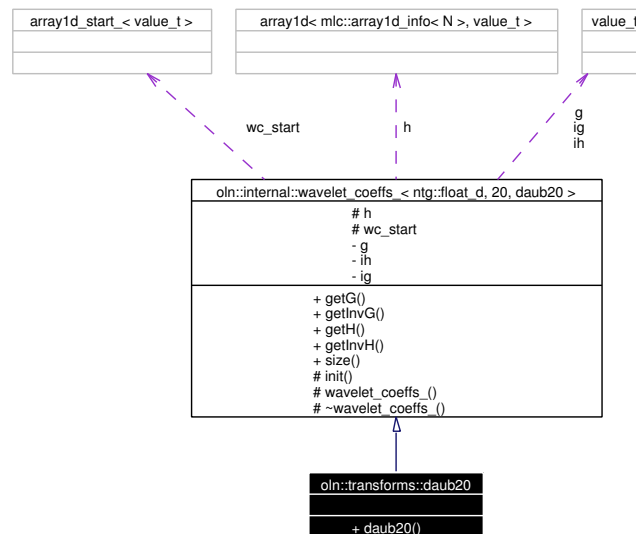
Daubechies wavelet coefficients.

```
#include <wavelet_coeffs.hh>
```

Inheritance diagram for oln::transforms::daub20:



Collaboration diagram for oln::transforms::daub20:



7.68.1 Detailed Description

Daubechies wavelet coefficients.

Twenty coefficients version.

Definition at line 159 of file `wavelet_coeffs.hh`.

The documentation for this struct was generated from the following file:

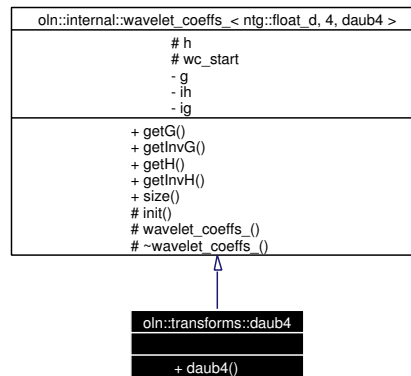
- `wavelet_coeffs.hh`

7.69 oln::transforms::daub4 Struct Reference

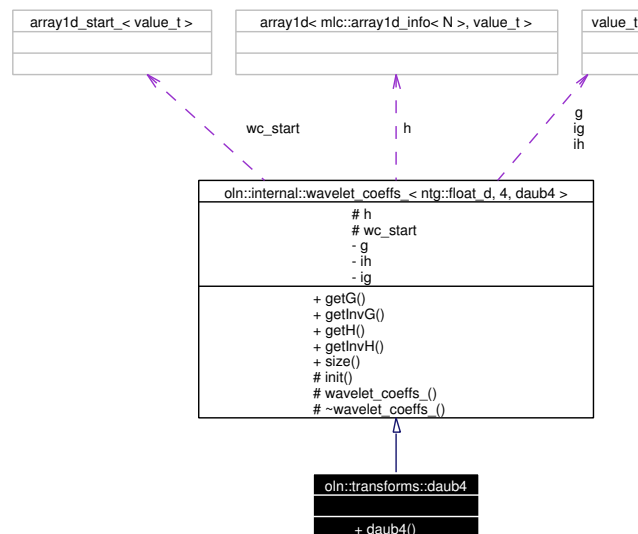
Daubechies wavelet coefficients.

```
#include <wavelet_coeffs.hh>
```

Inheritance diagram for oln::transforms::daub4:



Collaboration diagram for oln::transforms::daub4:



7.69.1 Detailed Description

Daubechies wavelet coefficients.

Four coefficients version.

Definition at line 61 of file `wavelet_coeffs.hh`.

The documentation for this struct was generated from the following file:

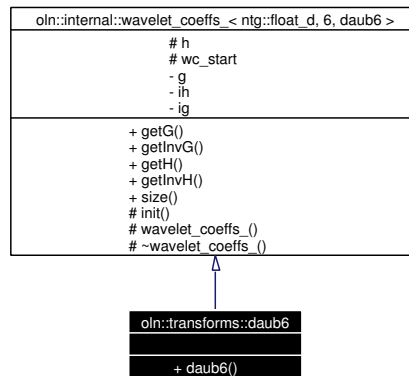
- `wavelet_coeffs.hh`

7.70 oln::transforms::daub6 Struct Reference

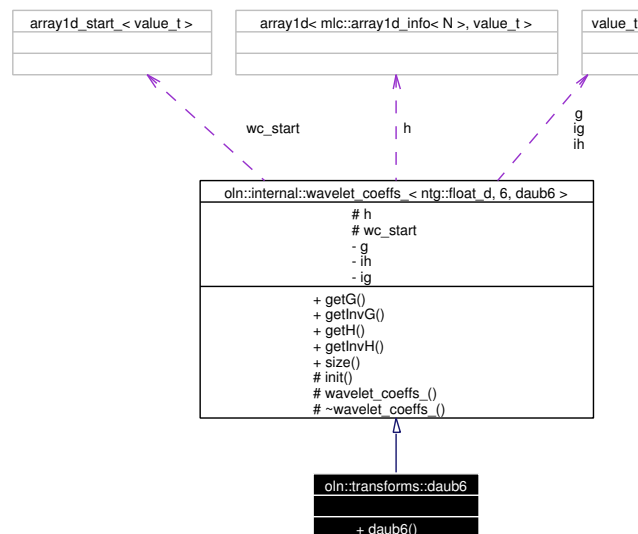
Daubechies wavelet coefficients.

```
#include <wavelet_coeffs.hh>
```

Inheritance diagram for oln::transforms::daub6:



Collaboration diagram for oln::transforms::daub6:



7.70.1 Detailed Description

Daubechies wavelet coefficients.

Six coefficients version.

Definition at line 79 of file `wavelet_coeffs.hh`.

The documentation for this struct was generated from the following file:

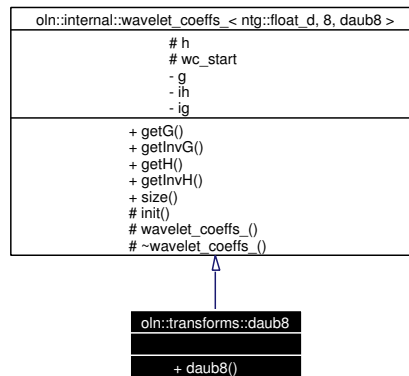
- `wavelet_coeffs.hh`

7.71 oln::transforms::daub8 Struct Reference

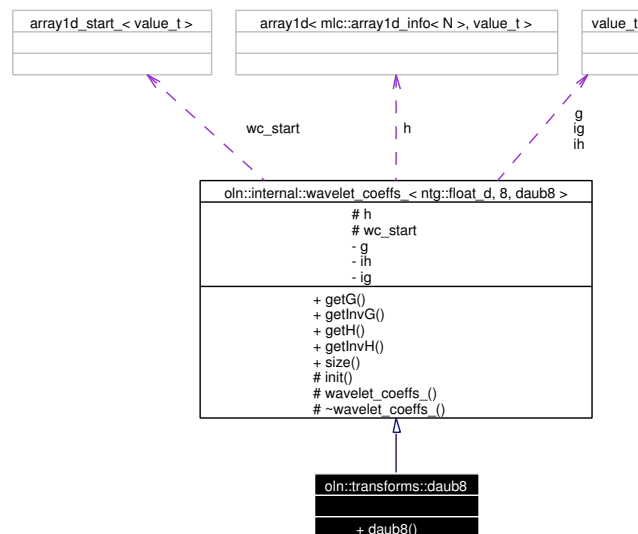
Daubechies wavelet coefficients.

```
#include <wavelet_coeffs.hh>
```

Inheritance diagram for oln::transforms::daub8:



Collaboration diagram for oln::transforms::daub8:



7.71.1 Detailed Description

Daubechies wavelet coefficients.

Eight coefficients version.

Definition at line 96 of file `wavelet_coeffs.hh`.

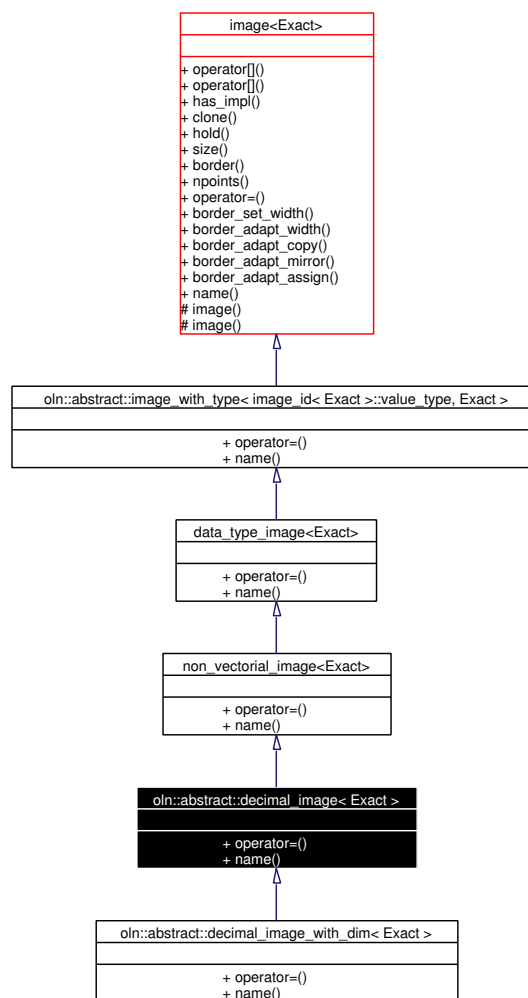
The documentation for this struct was generated from the following file:

- `wavelet_coeffs.hh`

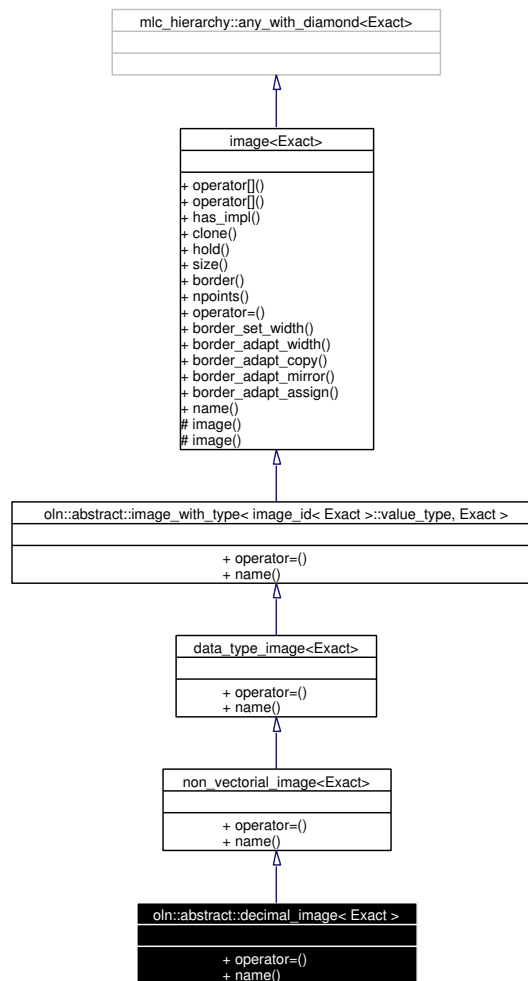
7.72 oln::abstract::decimal_image< Exact > Class Template Reference

```
#include <image_with_type_with_dim.hh>
```

Inheritance diagram for oln::abstract::decimal_image< Exact >:



Collaboration diagram for oln::abstract::decimal_image< Exact >:



Public Types

- typedef `decimal_image< Exact >` `self_type`
- typedef `Exact` `exact_type`

Public Member Functions

- `exact_type & operator= (self_type rhs)`

Perform a shallow copy between rhs and the current point, the points will not be duplicated but shared between the two images.

Static Public Member Functions

- `std::string name ()`

7.72.1 Detailed Description

template<class Exact> class oln::abstract::decimal_image< Exact >

This class, when used in a function declaration, forces the function to only accept decimal images.

Definition at line 253 of file image_with_type_with_dim.hh.

7.72.2 Member Function Documentation

7.72.2.1 template<class Exact> exact_type& oln::abstract::decimal_image< Exact >::operator=(self_type rhs) [inline]

Perform a shallow copy between *rhs* and the current point, the points will not be duplicated but shared between the two images.

See also:

[image::clone\(\)](#)

Reimplemented from [oln::abstract::non_vectorial_image< Exact >](#).

Reimplemented in [oln::abstract::decimal_image_with_dim< Dim, Exact >](#).

Definition at line 253 of file image_with_type_with_dim.hh.

273 {

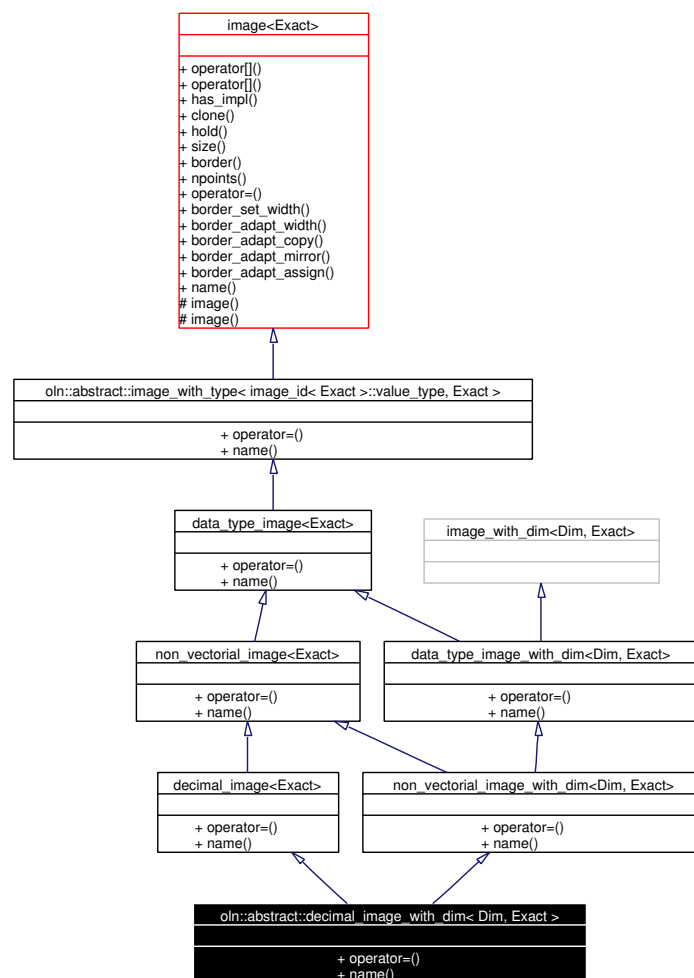
The documentation for this class was generated from the following file:

- image_with_type_with_dim.hh

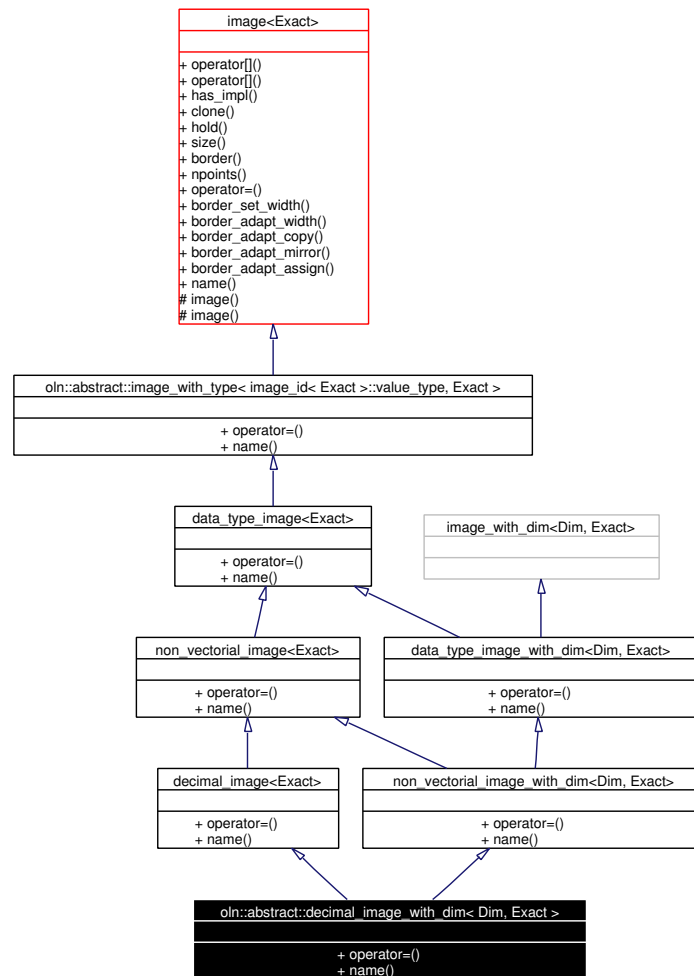
7.73 oln::abstract::decimal_image_with_dim< Dim, Exact > Class Template Reference

```
#include <image_with_type_with_dim.hh>
```

Inheritance diagram for oln::abstract::decimal_image_with_dim< Dim, Exact >:



Collaboration diagram for oln::abstract::decimal_image_with_dim< Dim, Exact >:



Public Types

- typedef `decimal_image_with_dim< Dim, Exact >` **self_type**
- typedef `Exact` **exact_type**

Public Member Functions

- `exact_type & operator= (self_type rhs)`

Perform a shallow copy between rhs and the current point, the points will not be duplicated but shared between the two images.

Static Public Member Functions

- `std::string name ()`

7.73.1 Detailed Description

```
template<unsigned Dim, class Exact> class oln::abstract::decimal_image_with_dim< Dim, Exact >
```

This class, when used in a function declaration, forces the function to only accept decimal images with a dimension of 'Dim'.

Definition at line 253 of file image_with_type_with_dim.hh.

7.73.2 Member Function Documentation

7.73.2.1 `template<unsigned Dim, class Exact> exact_type& oln::abstract::decimal_image_with_dim< Dim, Exact >::operator= (self_type rhs) [inline]`

Perform a shallow copy between *rhs* and the current point, the points will not be duplicated but shared between the two images.

See also:

[image::clone\(\)](#)

Reimplemented from [oln::abstract::non_vectorial_image_with_dim< Dim, Exact >](#).

Definition at line 253 of file image_with_type_with_dim.hh.

273 {

The documentation for this class was generated from the following file:

- image_with_type_with_dim.hh

7.74 ntg::internal::deduce_from_traits< Op, T, U > Struct Template Reference

Find the good [operator_traits](#), following a simple algorithm.

```
#include <global_ops_traits.hh>
```

Public Types

- typedef [operator_traits](#)< Op, traits_lhs_type, traits_rhs_type > **deduced_traits**
- typedef deduced_type::lhs_type **lhs_type**
- typedef deduced_type::rhs_type **rhs_type**
- typedef deduced_traits::ret **ret**
- typedef deduced_traits::impl **impl**

7.74.1 Detailed Description

```
template<class Op, class T, class U> struct ntg::internal::deduce_from_traits< Op, T, U >
```

Find the good [operator_traits](#), following a simple algorithm.

[deduce_from_traits](#) should generally be used instead of [operator_traits](#). Indeed, it has a handy algorithm to find return types:

1) Convert T1 and T2 to ntg types. 2) Check if traits<T1, T2> is defined. 3) Else, check if traits<T2, T1> is defined AND traits<T2, T1>::commutative is true.

[deduce_from_traits](#) defines several types:

- lhs_type and rhs_type: The types into which the first and second paramaters should be converted before called the implementation.
- ret: The return type.
- impl: The implementation type.
- deduced_traits: A pointer to the good operator_traits<> specialization. This can be useful sometimes.

Definition at line 174 of file global_ops_traits.hh.

The documentation for this struct was generated from the following file:

- global_ops_traits.hh

7.75 `ntg::internal::deduce_op_behavior< B1, B2 >` Struct Template Reference

Determine the resulting behavior of an operator return type.

```
#include <behavior.hh>
```

Public Types

- typedef `strict ret`

7.75.1 Detailed Description

```
template<class B1, class B2> struct ntg::internal::deduce_op_behavior< B1, B2 >
```

Determine the resulting behavior of an operator return type.

The algorithm is quite simple and arbitrary, is the two behaviors are identicals, then use it for the return type. Else use a strict behavior.

Definition at line 425 of file `integre/ntg/real/behavior.hh`.

The documentation for this struct was generated from the following file:

- `integre/ntg/real/behavior.hh`

7.76 ntg::internal::default_less< T > Struct Template Reference

```
#include <pred_succ.hh>
```

Public Types

- typedef T **arg_type**

Public Member Functions

- bool **operator()** (const T &l, const T &r) const

7.76.1 Detailed Description

template<typename T> struct ntg::internal::default_less< T >

Default less.

Specialization of this class provides a less functor even for types that do not support the operator "<".

Definition at line 83 of file pred_succ.hh.

The documentation for this struct was generated from the following file:

- pred_succ.hh

7.77 oln::internal::default_less< abstract::dpoint< Exact > > Struct Template Reference

```
#include <dpoint.hh>
```

Public Member Functions

- bool [operator\(\)](#) (const [abstract::dpoint](#)< Exact > &l, const [abstract::dpoint](#)< Exact > &r) const
Test if the coordinates of a dpoint l are not greater than the coordinates of a dpoint r.

7.77.1 Detailed Description

`template<class Exact> struct oln::internal::default_less< abstract::dpoint< Exact > >`

The specialized version for < abstract::dpoint<Exact> >.

Definition at line 262 of file dpoint.hh.

7.77.2 Member Function Documentation

7.77.2.1 `template<class Exact> bool oln::internal::default_less< abstract::dpoint< Exact >
>::operator() (const abstract::dpoint< Exact > &l, const abstract::dpoint< Exact > &
r) const` [inline]

Test if the coordinates of a dpoint l are not greater than the coordinates of a dpoint r.

Returns:

True if the coordinates of l are not greater than the coordinates of r.

Definition at line 271 of file dpoint.hh.

References `oln::abstract::dpoint< Exact >::nth()`.

```
273     {
274         for (unsigned i = 0; i < abstract::dpoint<Exact>::dim; ++i)
275             if (l.nth(i) < r.nth(i))
276                 return true;
277             else if (l.nth(i) > r.nth(i))
278                 return false;
279         return false;
280     }
```

The documentation for this struct was generated from the following file:

- dpoint.hh

7.78 oln::internal::default_less< abstract::point< Exact > > Struct Template Reference

```
#include <point.hh>
```

Public Member Functions

- bool [operator\(\)](#) (const [abstract::point](#)< Exact > &l, const [abstract::point](#)< Exact > &r) const
Test if the coordinates of a point l are not greater than the coordinates of a point r.

7.78.1 Detailed Description

template<class Exact> struct oln::internal::default_less< abstract::point< Exact > >

The specialized version for [abstract::point](#).

Definition at line 242 of file point.hh.

7.78.2 Member Function Documentation

7.78.2.1 template<class Exact> bool oln::internal::default_less< [abstract::point](#)< Exact > >::operator() (const [abstract::point](#)< Exact > &l, const [abstract::point](#)< Exact > &r) const [inline]

Test if the coordinates of a point l are not greater than the coordinates of a point r.

- l A point.
- r Another point.

Returns:

True if the coordinates of l are not greater than the coordinates of r.

Definition at line 256 of file point.hh.

References [oln::abstract::point< Exact >::nth\(\)](#).

```
258     {
259         for (unsigned i = 0; i < abstract::point<Exact>::dim; ++i)
260             if (l.nth(i) < r.nth(i))
261                 return true;
262             else if (l.nth(i) > r.nth(i))
263                 return false;
264         return false;
265     }
```

The documentation for this struct was generated from the following file:

- point.hh

7.79 oln::internal::default_less< coord > Struct Template Reference

[default_less<coord>](#)

```
#include <coord.hh>
```

Public Member Functions

- `bool operator() (const coord &lhs, const coord &rhs) const`

7.79.1 Detailed Description

`template<> struct oln::internal::default_less< coord >`

[default_less<coord>](#)

Specialized version for *coord*. This functor test if a coord *lhs* is less than another coord *rhs*.

Definition at line 51 of file coord.hh.

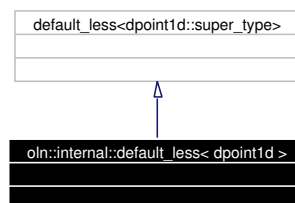
The documentation for this struct was generated from the following file:

- coord.hh

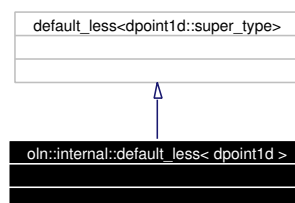
7.80 oln::internal::default_less< dpoint1d > Struct Template Reference

```
#include <dpoint1d.hh>
```

Inheritance diagram for oln::internal::default_less< dpoint1d >:



Collaboration diagram for oln::internal::default_less< dpoint1d >:



7.80.1 Detailed Description

template<> struct oln::internal::default_less< dpoint1d >

The specialized version for [dpoint1d](#).

Definition at line 153 of file dpoint1d.hh.

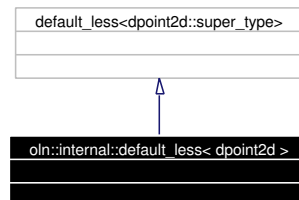
The documentation for this struct was generated from the following file:

- dpoint1d.hh

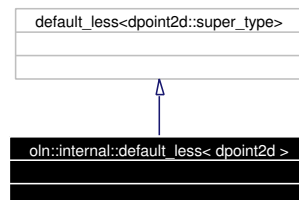
7.81 oln::internal::default_less< dpoint2d > Struct Template Reference

```
#include <dpoint2d.hh>
```

Inheritance diagram for oln::internal::default_less< dpoint2d >:



Collaboration diagram for oln::internal::default_less< dpoint2d >:



7.81.1 Detailed Description

template<> struct oln::internal::default_less< dpoint2d >

The specialized version for [dpoint2d](#).

Definition at line 161 of file dpoint2d.hh.

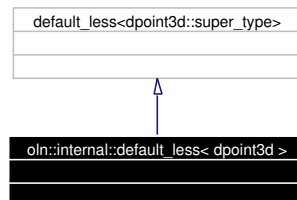
The documentation for this struct was generated from the following file:

- dpoint2d.hh

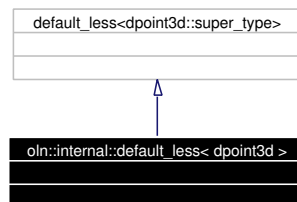
7.82 oln::internal::default_less< dpoint3d > Struct Template Reference

```
#include <dpoint3d.hh>
```

Inheritance diagram for oln::internal::default_less< dpoint3d >:



Collaboration diagram for oln::internal::default_less< dpoint3d >:



7.82.1 Detailed Description

template<> struct oln::internal::default_less< dpoint3d >

The specialized version for [dpoint3d](#).

Definition at line 170 of file dpoint3d.hh.

The documentation for this struct was generated from the following file:

- dpoint3d.hh

7.83 `ntg::internal::default_less< ntg::color< ncomps, qbits, color_system > >` Struct Template Reference

```
#include <color.hh>
```

Public Types

- typedef `ntg::color< ncomps, qbits, color_system >` **arg_type**

Public Member Functions

- `bool operator() (const arg_type &l, const arg_type &r) const`

7.83.1 Detailed Description

`template<unsigned ncomps, unsigned qbits, template< unsigned > class color_system> struct ntg::internal::default_less< ntg::color< ncomps, qbits, color_system > >`

The specialized version of `default_less` for colors.

Warning:

This class is only provided to build classes that need a less class, it does not correspond to the reality.
Example of a `std::set` of RGB colors:

```
** std::set<ntg::rgb_8,  
**      ntg::internal::default_less<ntg::rgb8> > s;  
** s.insert(ntg::rgb_8(10, 16, 64));  
**
```

Definition at line 290 of file `color/color.hh`.

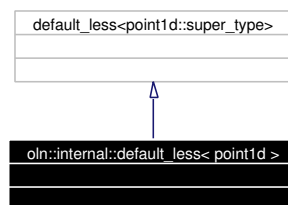
The documentation for this struct was generated from the following file:

- `color/color.hh`

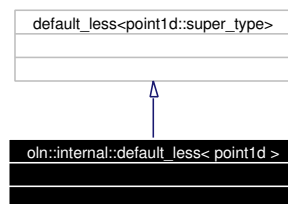
7.84 oln::internal::default_less< point1d > Struct Template Reference

```
#include <point1d.hh>
```

Inheritance diagram for oln::internal::default_less< point1d >:



Collaboration diagram for oln::internal::default_less< point1d >:



7.84.1 Detailed Description

template<> struct oln::internal::default_less< point1d >

The specialized version for [point1d](#).

Definition at line 155 of file `point1d.hh`.

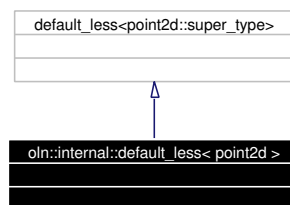
The documentation for this struct was generated from the following file:

- `point1d.hh`

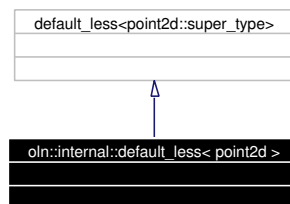
7.85 oln::internal::default_less< point2d > Struct Template Reference

```
#include <point2d.hh>
```

Inheritance diagram for oln::internal::default_less< point2d >:



Collaboration diagram for oln::internal::default_less< point2d >:



7.85.1 Detailed Description

template<> struct oln::internal::default_less< point2d >

The specialized version for [point2d](#).

Definition at line 163 of file `point2d.hh`.

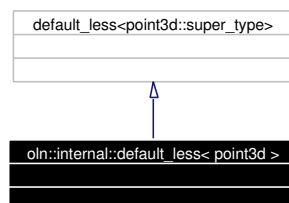
The documentation for this struct was generated from the following file:

- `point2d.hh`

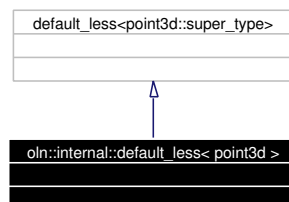
7.86 oln::internal::default_less< point3d > Struct Template Reference

```
#include <point3d.hh>
```

Inheritance diagram for oln::internal::default_less< point3d >:



Collaboration diagram for oln::internal::default_less< point3d >:



7.86.1 Detailed Description

template<> struct oln::internal::default_less< point3d >

The specialized version for [point3d](#).

Definition at line 178 of file point3d.hh.

The documentation for this struct was generated from the following file:

- point3d.hh

7.87 oln::internal::dim_iterate_rec_< dim, skip > Struct Template Reference

Iterate over all dimensions except one.

```
#include <dwt.hh>
```

Static Public Member Functions

- `template<class I, class K> void doit (abstract::image< I > &im, typename mlc::exact< I >::ret::point_type &p, const unsigned l1, const unsigned l2, const K &coeffs, dwt_transform_dir_d)`

Iterate over all dimensions except one.

7.87.1 Detailed Description

```
template<unsigned dim, unsigned skip> struct oln::internal::dim_iterate_rec_< dim, skip >
```

Iterate over all dimensions except one.

Parameters:

dim Number of dimension to process.

skip Dimension to skip.

Definition at line 356 of file dwt.hh.

7.87.2 Member Function Documentation

7.87.2.1 `template<unsigned dim, unsigned skip> template<class I, class K> void oln::internal::dim_iterate_rec_< dim, skip >::doit (abstract::image< I > &im, typename mlc::exact< I >::ret::point_type &p, const unsigned l1, const unsigned l2, const K &coeffs, dwt_transform_dir_d) [inline, static]`

Iterate over all dimensions except one.

Parameters:

I Exact type of the image to process.

K Type of coefficients.

Definition at line 365 of file dwt.hh.

```
371      {
372          dim_skip_iterate_rec_<dim, skip>::doit(im, p, l1, l2, coeffs, d);
373          dim_iterate_rec_<dim, skip - 1>::doit(im, p, l1, l2, coeffs, d);
374      }
```

The documentation for this struct was generated from the following file:

- dwt.hh

7.88 oln::internal::dim_iterate_rec_< dim, 0 > Struct Template Reference

Specialization of [dim_iterate_rec_](#) with skip dimension set to 0.

```
#include <dwt.hh>
```

Static Public Member Functions

- `template<class I, class K> void doit (abstract::image< I > &, typename mlc::exact< I >::ret::point_type &, const unsigned, const unsigned, const K &, dwt_transform_dir_)`

Iterate over all dimensions except one.

7.88.1 Detailed Description

`template<unsigned dim> struct oln::internal::dim_iterate_rec_< dim, 0 >`

Specialization of [dim_iterate_rec_](#) with skip dimension set to 0.

Parameters:

dim Number of dimension to process.

Definition at line 384 of file `dwt.hh`.

7.88.2 Member Function Documentation

7.88.2.1 `template<unsigned dim> template<class I, class K> void oln::internal::dim_iterate_rec_< dim, 0 >::doit (abstract::image< I > &, typename mlc::exact< I >::ret::point_type &, const unsigned, const unsigned, const K &, dwt_transform_dir_)`
`[inline, static]`

Iterate over all dimensions except one.

Parameters:

I Exact type of the image to process.

K Type of coefficients.

End of recursion.

Definition at line 396 of file `dwt.hh`.

```
402     {
403         // end of recursion
404     }
```

The documentation for this struct was generated from the following file:

- `dwt.hh`

7.89 oln::internal::dim_skip_iterate_rec_< dim, skip, current > Struct Template Reference

Functions used to iterate over all dimensions except one.

```
#include <dwt.hh>
```

Static Public Member Functions

- template<class I, class K> void [doit](#) ([abstract::image](#)< I > &im, typename mlc::exact< I >::ret::point_type &p, const unsigned l1, const unsigned l2, const K &coeffs, [dwt_transform_dir_d](#))

Iterate over all dimensions except one.

7.89.1 Detailed Description

```
template<unsigned dim, unsigned skip, unsigned current = dim> struct oln::internal::dim_skip_iterate_rec_< dim, skip, current >
```

Functions used to iterate over all dimensions except one.

Parameters:

dim Number of dimension to treat.

skip Dimension to skip.

current Current dimension.

Definition at line 281 of file dwt.hh.

The documentation for this struct was generated from the following file:

- dwt.hh

7.90 oln::internal::dim_skip_iterate_rec_< dim, skip, 0 > Struct Template Reference

Specialization of [dim_skip_iterate_rec_](#) with current dimension set to 0.

```
#include <dwt.hh>
```

Static Public Member Functions

- `template<class I, class K> void doit (abstract::image< I > &im, typename mlc::exact< I >::ret::point_type &p, const unsigned l1, const unsigned l2, const K &coeffs, dwt_transform_dir_d)`

Iterate over all dimensions except one.

7.90.1 Detailed Description

```
template<unsigned dim, unsigned skip> struct oln::internal::dim_skip_iterate_rec_< dim, skip, 0 >
```

Specialization of [dim_skip_iterate_rec_](#) with current dimension set to 0.

Parameters:

dim Number of dimension to process.

skip Dimension to skip.

Definition at line 316 of file dwt.hh.

7.90.2 Member Function Documentation

7.90.2.1 `template<unsigned dim, unsigned skip> template<class I, class K> void oln::internal::dim_skip_iterate_rec_< dim, skip, 0 >::doit (abstract::image< I > &im, typename mlc::exact< I >::ret::point_type &p, const unsigned l1, const unsigned l2, const K &coeffs, dwt_transform_dir_d) [inline, static]`

Iterate over all dimensions except one.

Parameters:

I Exact type of the image to process.

K Type of coefficients.

It is leaf call, thus you can call a dwt transform step.

Definition at line 327 of file dwt.hh.

References [oln::internal::dwt_transform_inv_step_\(\)](#), and [oln::internal::dwt_transform_step_\(\)](#).

```
333     {
334         unsigned n;
335
336         switch (d) {
```

```
337         case dwt_fwd:
338             for (n = 12; n >= 11; n >>= 1)
339                 dwt_transform_step_(im, p, skip - 1, n, coeffs);
340             break;
341         case dwt_bwd:
342             for (n = 11; n <= 12; n <<= 1)
343                 dwt_transform_inv_step_(im, p, skip - 1, n, coeffs);
344             break;
345     };
346 }
```

The documentation for this struct was generated from the following file:

- dwt.hh

7.91 oln::dim_traits< Dim, T, Exact > Struct Template Reference

```
#include <image.hh>
```

7.91.1 Detailed Description

template<unsigned Dim, class T, class Exact = mlc::final> struct oln::dim_traits< Dim, T, Exact >

Define `img_type` equal to the image of dim *Dim* the generic declaration defines nothing this class will be specialized for each dimension.

Definition at line 191 of file `core/image.hh`.

The documentation for this struct was generated from the following file:

- `core/image.hh`

7.92 oln::dim_traits< 1, T, Exact > Struct Template Reference

```
#include <image1d.hh>
```

Public Types

- typedef [image1d](#)< T, Exact > **img_type**

7.92.1 Detailed Description

template<class T, class Exact> struct oln::dim_traits< 1, T, Exact >

Define `img_type` equal to `image1d<T, Exact>`.

Definition at line 230 of file `image1d.hh`.

The documentation for this struct was generated from the following file:

- `image1d.hh`

7.93 oln::dim_traits< 2, T, Exact > Struct Template Reference

```
#include <image2d.hh>
```

Public Types

- typedef [image2d](#)< T, Exact > **img_type**

7.93.1 Detailed Description

```
template<class T, class Exact> struct oln::dim_traits< 2, T, Exact >
```

Define `img_type` equal to `image2d<T, Exact>`.

Definition at line 232 of file `image2d.hh`.

The documentation for this struct was generated from the following file:

- `image2d.hh`

7.94 oln::dim_traits< 3, T, Exact > Struct Template Reference

```
#include <image3d.hh>
```

Public Types

- typedef [image3d](#)< T, Exact > **img_type**

7.94.1 Detailed Description

```
template<class T, class Exact> struct oln::dim_traits< 3, T, Exact >
```

Define `img_type` equal to `image3d<T, Exact>`.

Definition at line 231 of file `image3d.hh`.

The documentation for this struct was generated from the following file:

- `image3d.hh`

7.95 oln::topo::inter_pixel::internal::dir_traits< Dim > Struct Template Reference

Provides the enum dir.

```
#include <dir.hh>
```

7.95.1 Detailed Description

template<unsigned Dim> struct oln::topo::inter_pixel::internal::dir_traits< Dim >

Provides the enum dir.

Definition at line 44 of file dir.hh.

The documentation for this struct was generated from the following file:

- dir.hh

7.96 oln::topo::inter_pixel::internal::dir_traits< 2 > Struct Template Reference

Provides the enum dir for 2D.

```
#include <dir.hh>
```

Public Types

- typedef enum [oln::topo::inter_pixel::internal::dir_traits< 2 >::dir](#) **ret**
- enum **dir** { **east**, **north**, **west**, **south** }

Static Public Member Functions

- [ret first](#) ()
First direction.
- [ret last](#) ()
Last direction.
- [ret prev](#) ([ret i](#))
Prev direction (with `Prev(first()) == last()`).
- [ret next](#) ([ret i](#))
Next direction (with `next(last()) == first()`).
- [ret opposite](#) ([ret i](#))

7.96.1 Detailed Description

template<> struct oln::topo::inter_pixel::internal::dir_traits< 2 >

Provides the enum dir for 2D.

Definition at line 48 of file dir.hh.

7.96.2 Member Function Documentation

7.96.2.1 [ret oln::topo::inter_pixel::internal::dir_traits< 2 >::opposite](#) ([ret i](#)) [`inline`, `static`]

Opposit direction.

Todo

FIXME: no modulus.

Definition at line 85 of file dir.hh.

```
86         {  
87             return ret((i + 2) % 4);  
88         }
```

The documentation for this struct was generated from the following file:

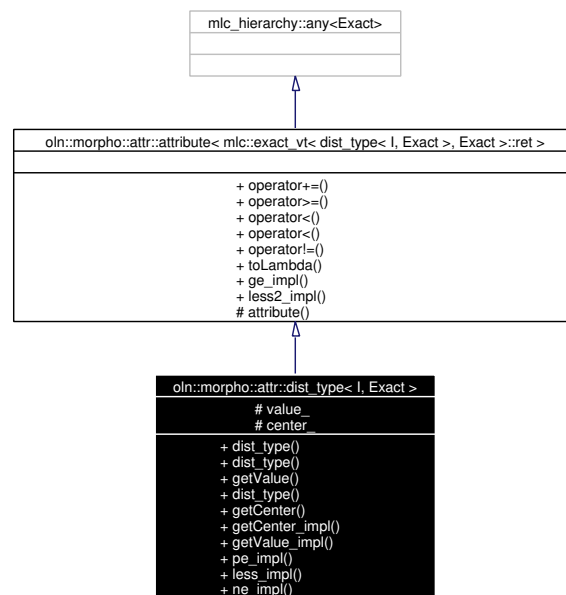
- dir.hh

7.97 oln::morpho::attr::dist_type< I, Exact > Class Template Reference

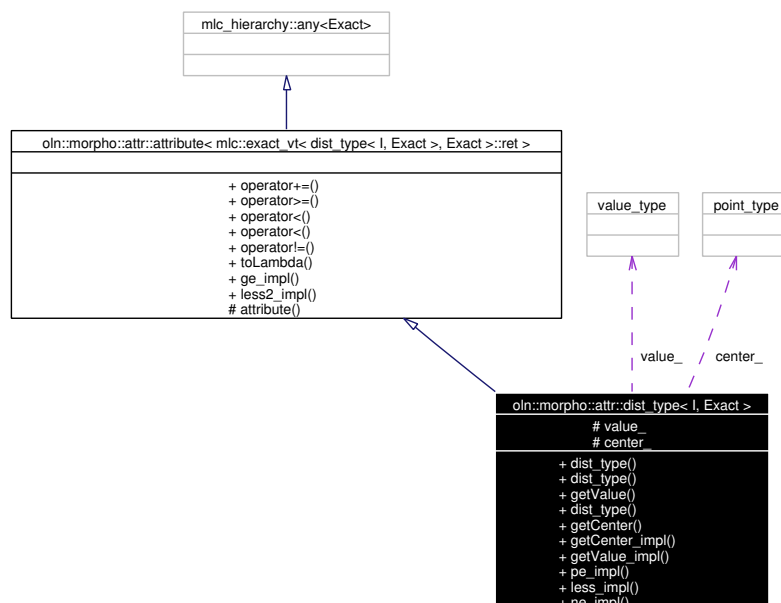
Dist attribute.

```
#include <attributes.hh>
```

Inheritance diagram for oln::morpho::attr::dist_type< I, Exact >:



Collaboration diagram for oln::morpho::attr::dist_type< I, Exact >:



Public Types

- typedef [dist_type](#)< I, Exact > [self_type](#)
Self type of the class.
- typedef mlc::exact_vt< [self_type](#), Exact >::ret [exact_type](#)
- typedef oln::morpho::attr::attr_traits< exact_type >::value_type [value_type](#)
- typedef oln::morpho::attr::attr_traits< exact_type >::env_type [env_type](#)
- typedef oln::morpho::attr::attr_traits< exact_type >::lambda_type [lambda_type](#)
- typedef [abstract::image](#)< typename mlc::exact< I >::ret > [im_type](#)
Image type.
- typedef mlc::exact< [im_type](#) >::ret::point_type [point_type](#)
Point type associated to im_type.
- typedef mlc::exact< [im_type](#) >::ret::dpoint_type [dpoint_type](#)
Dpoint type associated to im_type.

Public Member Functions

- [dist_type](#) ()
Basic Ctor.
- [dist_type](#) (const [im_type](#) &, const [point_type](#) &p, const env_type &)
Ctor from a point and an image.
- const value_type & [getValue](#) () const
Accessor to value_.
- [dist_type](#) (const lambda_type lambda)
Ctor from a lambda_type value.
- const [point_type](#) & [getCenter](#) () const
Accessor to center_.
- const [point_type](#) & [getCenter_impl](#) () const
Implementation of [getCenter\(\)](#).
- const value_type & [getValue_impl](#) () const
Implementation of [getValue\(\)](#).
- void [pe_impl](#) (const [dist_type](#) &rhs)
+= operator implementation.
- bool [less_impl](#) (const lambda_type &lambda) const
"<" operator implementation.
- bool [ne_impl](#) (const lambda_type &lambda) const
!= operator implementation.

Protected Attributes

- value_type [value_](#)
Current value of the attribute.
- point_type [center_](#)
Center of the attribute.

7.97.1 Detailed Description

template<class I, class Exact = mlc::final> class oln::morpho::attr::dist_type< I, Exact >

Dist attribute.

Parameters:

I Exact type of images to process.

Exact The exact type.

Definition at line 1282 of file attributes.hh.

7.97.2 Constructor & Destructor Documentation

7.97.2.1 **template<class I, class Exact = mlc::final> [oln::morpho::attr::dist_type](#)< I, Exact >::[dist_type](#) ()** `[inline]`

Basic Ctor.

Warning:

After this call, the object is only instantiated (not initialized).

Definition at line 1298 of file attributes.hh.

```
1299         {
1300     };
```

7.97.2.2 **template<class I, class Exact = mlc::final> [oln::morpho::attr::dist_type](#)< I, Exact >::[dist_type](#) (const [im_type](#) &, const [point_type](#) & *p*, const env_type &)** `[inline]`

Ctor from a point and an image.

- *p* Point to consider in the image.

Definition at line 1307 of file attributes.hh.

References [oln::morpho::attr::dist_type< I, Exact >::center_](#).

```
1309                                     :
1310     value_(ntg_zero_val(value_type)),
1311     center_(p)
1312     {
1313     };
```

7.97.2.3 `template<class I, class Exact = mlc::final> oln::morpho::attr::dist_type< I, Exact >::dist_type (const lambda_type lambda)` `[inline]`

Ctor from a `lambda_type` value.

- `lambda` Value of the attribute.

Definition at line 1331 of file `attributes.hh`.

```
1331                                     : value_(lambda)
1332     {
1333     };
```

7.97.3 Member Function Documentation

7.97.3.1 `template<class I, class Exact = mlc::final> const point_type& oln::morpho::attr::dist_type< I, Exact >::getCenter () const` `[inline]`

Accessor to `center_`.

Virtual method.

See also:

[getCenter_impl\(\)](#)

Definition at line 1341 of file `attributes.hh`.

Referenced by `oln::morpho::attr::dist_type< I, Exact >::pe_impl()`.

```
1342     {
1343         mlc_dispatch(getCenter)();
1344     };
```

7.97.3.2 `template<class I, class Exact = mlc::final> const point_type& oln::morpho::attr::dist_type< I, Exact >::getCenter_impl () const` `[inline]`

Implementation of [getCenter\(\)](#).

Override this method in order to provide a new version of [getCenter\(\)](#).

Warning:

Do not call this method, use [getCenter\(\)](#) instead.

Definition at line 1355 of file `attributes.hh`.

References `oln::morpho::attr::dist_type< I, Exact >::center_`.

```
1356     {
1357         return center_;
1358     };
```

7.97.3.3 `template<class I, class Exact = mlc::final> const value_type&
oln::morpho::attr::dist_type< I, Exact >::getValue () const [inline]`

Accessor to value_.

Virtual method.

See also:

[getValue_impl\(\)](#)

Definition at line 1321 of file attributes.hh.

Referenced by `oln::morpho::attr::dist_type< I, Exact >::pe_impl()`.

```
1322         {
1323             mlc_dispatch(getValue)();
1324         };
```

7.97.3.4 `template<class I, class Exact = mlc::final> const value_type&
oln::morpho::attr::dist_type< I, Exact >::getValue_impl () const [inline]`

Implementation of [getValue\(\)](#).

Override this method in order to provide a new version of [getValue\(\)](#).

Warning:

Do not call this method, use [getValue\(\)](#) instead.

Definition at line 1368 of file attributes.hh.

```
1369         {
1370             return value_;
1371         };
```

7.97.3.5 `template<class I, class Exact = mlc::final> bool oln::morpho::attr::dist_type< I, Exact
 >::less_impl (const lambda_type & lambda) const [inline]`

"<" operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 1400 of file attributes.hh.

```
1401         {
1402             return value_ < lambda;
1403         };
```

7.97.3.6 `template<class I, class Exact = mlc::final> bool oln::morpho::attr::dist_type< I, Exact >::ne_impl (const lambda_type & lambda) const` [inline]

!= operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 1412 of file attributes.hh.

```
1413         {
1414             return value_ != lambda;
1415         };
```

7.97.3.7 `template<class I, class Exact = mlc::final> void oln::morpho::attr::dist_type< I, Exact >::pe_impl (const dist_type< I, Exact > & rhs)` [inline]

+= operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 1380 of file attributes.hh.

References `oln::morpho::attr::dist_type< I, Exact >::center_`, `oln::morpho::attr::dist_type< I, Exact >::getCenter()`, and `oln::morpho::attr::dist_type< I, Exact >::getValue()`.

```
1381         {
1382             value_type last = value_;
1383             dpoint_type p = center_ - rhs.getCenter();
1384
1385             value_ = ntg_zero_val(value_type);
1386             for (int i = 0; i < point_traits<point_type>::dim; ++i)
1387                 value_ += p.nth(i) * p.nth(i);
1388             value_ = sqrt(value_);
1389             value_ = ntg::max(value_, last);
1390             value_ = ntg::max(value_, rhs.getValue());
1391         };
```

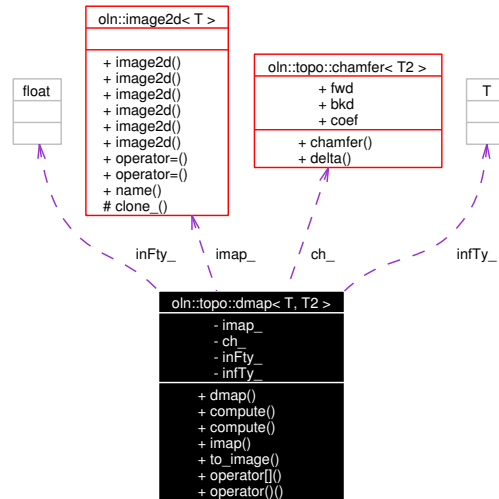
The documentation for this class was generated from the following file:

- attributes.hh

7.98 oln::topo::dmap< T, T2 > Class Template Reference

```
#include <dmap.hh>
```

Collaboration diagram for oln::topo::dmap< T, T2 >:



Public Types

- typedef `image2d< ntg::bin >::point_type` `point_type`

Public Member Functions

- `dmap` (const `image2d_size` &size, const `chamfer< T2 >` &ch)
- template<class V> void `compute` (const `image2d< V >` &input, float infty=0.f)
Compute the distance map.
- template<class V> void `compute` (const `image2d< V >` &input, `image2d< point2d >` &nearest_point_map, float infty=0.f)
Compute the distance map.
- const `image2d< T >` &`imap` () const
Return the distance map of type T.
- `image2d< float >` `to_image` () const
Return the distance map divided by the Chamfer coefficient.
- const T &`operator[]` (const `point_type` &p) const
Distance of a point p.
- const T &`operator()` (`coord` row, `coord` col) const
Distance of a point2d(row, col).

7.98.1 Detailed Description

`template<class T, class T2> class oln::topo::dmap< T, T2 >`

Distance map

Parameters:

T Type of the distance.

T2 Type of the chamfer distance.

Note:

Do not forget to call *compute*.

```
#include <oln/basics2d.hh>
#include <oln/topo/dmap.hh>
#include <oln/convert/stretch.hh>

int main()
{
    oln::image2d<ntg::bin> in = oln::load(IMG_IN "face_se.pbm");

    oln::topo::dmap<ntg::int_u<16>, int> m(in.size(), oln::topo::chessboard());
    m.compute(in);
    save(oln::convert::stretch_balance<ntg::int_u8>(m.imap()),
        IMG_OUT "oln_topo_dmap.pgm");
}
```



Definition at line 660 of file dmap.hh.

7.98.2 Constructor & Destructor Documentation

7.98.2.1 `template<class T, class T2> oln::topo::dmap< T, T2 >::dmap (const image2d_size & size, const chamfer< T2 > & ch)`

Constructor.

- size Size of the image on which the dmap will be compute.
- ch Chamfer distance used.

Definition at line 203 of file dmap.hxx.

```
204                                     :
205     imap_(size),
206     ch_(ch)
207     {
208         // FIXME: if T is float then precondition(ch.coef == 1.f)
209     }
```

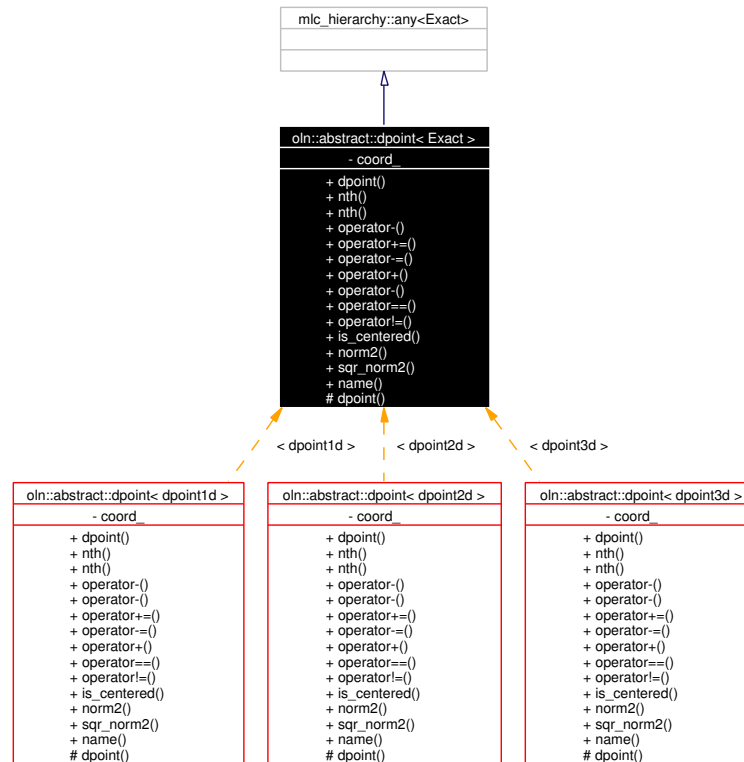
The documentation for this class was generated from the following files:

- [dmap.hh](#)
- [dmap.hxx](#)

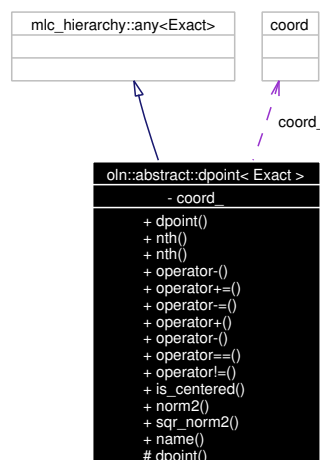
7.99 oln::abstract::dpoint< Exact > Struct Template Reference

```
#include <dpoint.hh>
```

Inheritance diagram for oln::abstract::dpoint< Exact >:



Collaboration diagram for oln::abstract::dpoint< Exact >:



Public Types

- typedef Exact **exact_type**
- typedef **dpoint**< Exact > **self_type**
- typedef dpoint_traits< exact_type >::point_type **point_type**
- enum { **dim** = dpoint_traits<exact_type>::dim }

Public Member Functions

- **dpoint** (const **abstract::point**< point_type > &p)
Construct a dpoint with the same coordinates as the point p.
- **coord_nth** (const unsigned d) const
Return the dth coordinate of the current dpoint.
- **coord & nth** (const unsigned d)
Return a reference to the dth coordinate of the current dpoint.
- exact_type **operator-** () const
Return a dpoint whose coordinates are the opposite of the current dpoint coordinates.
- exact_type & **operator+=** (const self_type &dp)
Add a dpoint dp to the current dpoint.
- exact_type & **operator-=** (const self_type &dp)
Subtract a dpoint dp from the current dpoint.
- exact_type **operator+** (const self_type &dp) const
Add a dpoint dp to the current dpoint.
- exact_type **operator-** (const self_type &dp) const
Subtract a dpoint dp from the current dpoint.
- bool **operator==** (const self_type &dp) const
Test if two dpoints have the same coordinates.
- bool **operator!=** (const self_type &dp) const
Test if two dpoints do not have the same coordinates.
- bool **is_centered** (void) const
Test if all the dpoint coordinates are set to zero.
- ntg::float_d **norm2** (void) const
Return the norm of the current dpoint.
- ntg::float_d **sqr_norm2** (void) const
Return the square of the norm of the current dpoint.

Static Public Member Functions

- `std::string name ()`

7.99.1 Detailed Description

`template<class Exact> struct oln::abstract::dpoint< Exact >`

The abstract dpoint class from whom derives all the others dpoint classes.

Definition at line 71 of file dpoint.hh.

7.99.2 Member Function Documentation

7.99.2.1 `template<class Exact> bool oln::abstract::dpoint< Exact >::is_centered (void) const`
[inline]

Test if all the dpoint coordinates are set to zero.

Returns:

True if all the coordinates of the current dpoint are set to zero.

Definition at line 199 of file dpoint.hh.

```

200     {
201         for (unsigned i = 0; i < dim; ++i)
202             if (nth(i) != 0)
203                 return false;
204         return true;
205     }
```

7.99.2.2 `template<class Exact> bool oln::abstract::dpoint< Exact >::operator!= (const self_type & dp) const` [inline]

Test if two dpoints do not have the same coordinates.

- `dp` A dpoint.

Returns:

False if `dp` and the current point have the same coordinate, true otherwise.

Definition at line 184 of file dpoint.hh.

```

185     {
186         for (unsigned i = 0; i < dim; ++i)
187             if (dp.nth(i) != nth(i))
188                 return true;
189         return false;
190     }
```

7.99.2.3 `template<class Exact> exact_type oln::abstract::dpoint< Exact >::operator+ (const self_type & dp) const` [inline]

Add a dpoint *dp* to the current dpoint.

Returns:

The value of the current point plus *dp*.

Definition at line 142 of file dpoint.hh.

```
143     {  
144         return this->exact().plus_dp(dp.exact());  
145     }
```

7.99.2.4 `template<class Exact> exact_type& oln::abstract::dpoint< Exact >::operator+= (const self_type & dp)` [inline]

Add a dpoint *dp* to the current dpoint.

Returns:

A reference to the current dpoint plus *dp*.

Definition at line 120 of file dpoint.hh.

```
121     {  
122         return this->exact().plus_assign_dp(dp.exact());  
123     }
```

7.99.2.5 `template<class Exact> exact_type oln::abstract::dpoint< Exact >::operator- (const self_type & dp) const` [inline]

Subtract a dpoint *dp* from the current dpoint.

Returns:

The value of the current point minus *dp*.

Definition at line 153 of file dpoint.hh.

```
154     {  
155         return this->exact().minus_dp(dp.exact());  
156     }
```

7.99.2.6 `template<class Exact> exact_type& oln::abstract::dpoint< Exact >::operator-= (const self_type & dp)` [inline]

Subtract a dpoint *dp* from the current dpoint.

Returns:

A reference to the current point minus *dp*.

Definition at line 131 of file dpoint.hh.

```
132     {  
133         return this->exact().minus_assign_dp(dp.exact());  
134     }
```

7.99.2.7 `template<class Exact> bool oln::abstract::dpoint< Exact >::operator==(const self_type & dp) const` [inline]

Test if two dpoints have the same coordinates.

- dp A dpoint.

Returns:

True if *dp* and the current point have the same coordinate, false otherwise.

Definition at line 167 of file dpoint.hh.

```
168     {  
169         for (unsigned i = 0; i < dim; ++i)  
170             if (dp.nth(i) != nth(i))  
171                 return false;  
172         return true;  
173     }
```

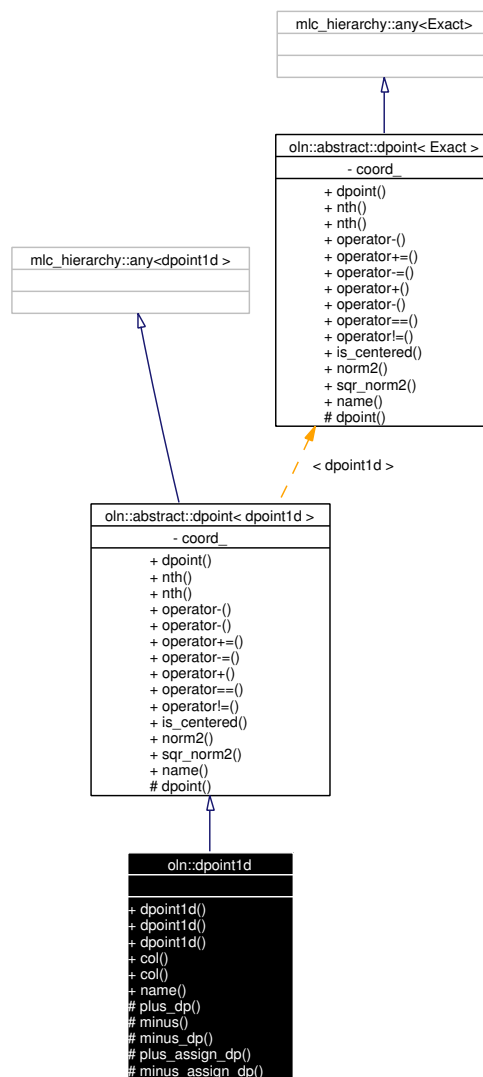
The documentation for this struct was generated from the following file:

- dpoint.hh

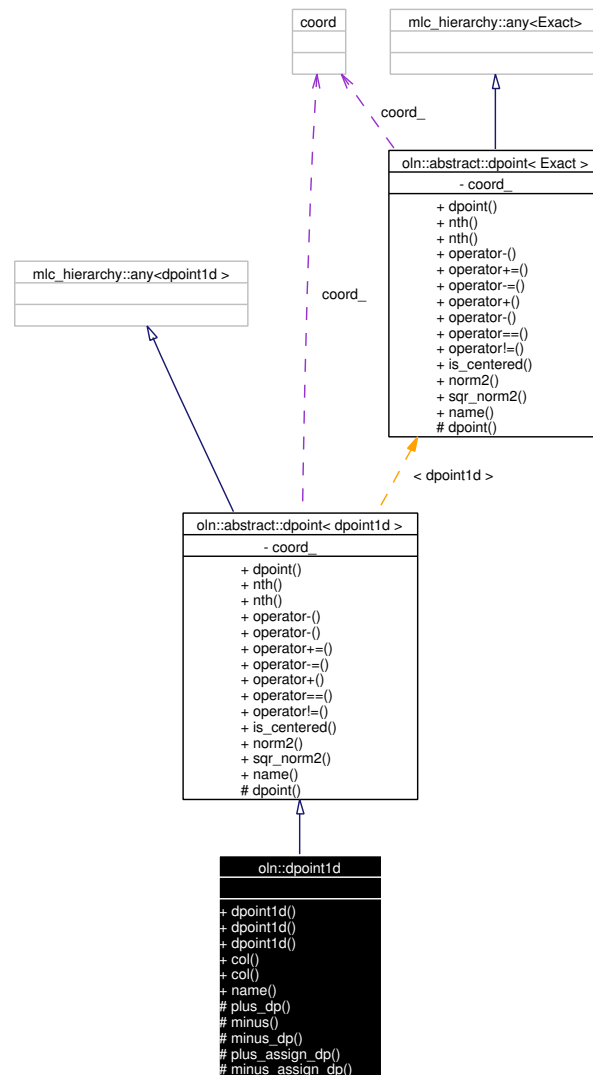
7.100 oln::dpoint1d Class Reference

```
#include <dpoint1d.hh>
```

Inheritance diagram for oln::dpoint1d:



Collaboration diagram for oln::dpoint1d:



Public Types

- typedef `abstract::dpoint< dpoin1d >` `super_type`

Public Member Functions

- `dpoin1d (coord c)`
The coordinate of the `dpoin1d` is set to `c`.
- `dpoin1d (const point1d &p)`
The coordinate of the `dpoin1d` is set to the `coordinate`.
- `coord col () const`
Return the value of the `dpoin1d` coordinate.

- `coord & col ()`

Return a reference to the `dpoint1d` coordinate.

Static Public Member Functions

- `std::string name ()`

Protected Member Functions

- `dpoint1d plus_dp (const dpoint1d &dp) const`

Return a `dpoint1d` whose coordinate is equal to `dp` coordinate plus the current `dpoint1d` coordinate.

- `dpoint1d minus () const`

Return a `dpoint1d` whose coordinate is equal to the opposite of the current `dpoint1d` coordinate.

- `dpoint1d minus_dp (const dpoint1d &dp) const`

Return a `dpoint1d` whose coordinate is equal to the current `dpoint1d` coordinate minus '`dp`' coordinate.

- `dpoint1d & plus_assign_dp (const dpoint1d &dp)`

Return a reference to the current `dpoint1d` plus '`dp`'.

- `dpoint1d & minus_assign_dp (const dpoint1d &dp)`

Return a reference to the current `dpoint1d` minus '`dp`'.

Friends

- `class abstract::dpoint< dpoint1d >`

7.100.1 Detailed Description

Subclass of `abstract::dpoint`, declaration of `dpoint` for `image1d`. To instantiate a `dpoint1d` on an `oln::image1d<ntg::rgb_8>` for example, use the macro `oln_dpoint_type(I)`.

```
oln_dpoint_type(oln::image1d<ntg::rgb_8>) p();
```

or

```
oln_dpoint_type(oln::image1d<ntg::rgb_8>) p(1);
```

Definition at line 68 of file `dpoint1d.hh`.

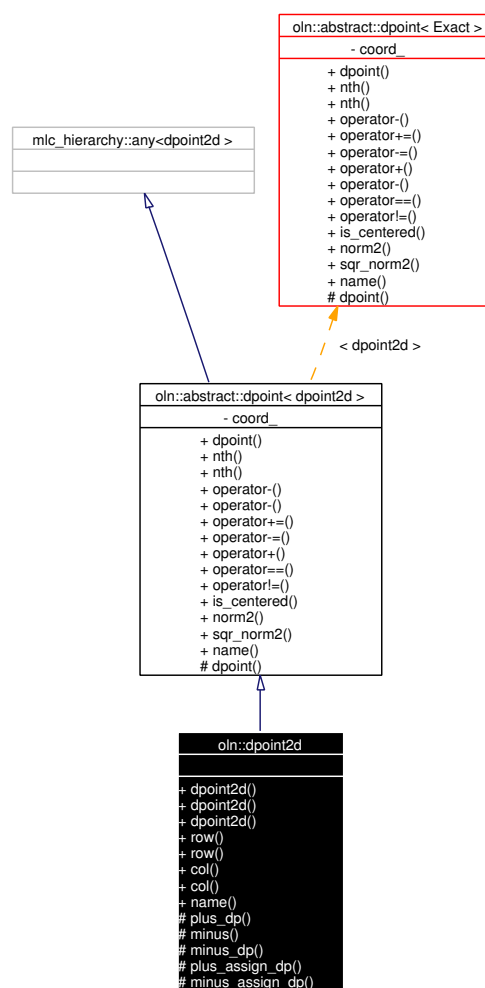
The documentation for this class was generated from the following files:

- `dpoint1d.hh`
- `dpoint1d.hxx`

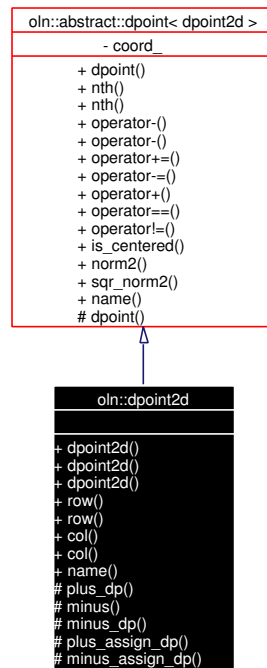
7.101 oln::dpoint2d Class Reference

```
#include <dpoint2d.hh>
```

Inheritance diagram for oln::dpoint2d:



Collaboration diagram for oln::dpoint2d:



Public Types

- typedef `abstract::dpoint< dpoint2d >` `super_type`

Public Member Functions

- `dpoint2d` (`coord` row, `coord` col)
The coordinates of the `dpoint2d` are set to row and col.
- `dpoint2d` (const `point2d` &p)
The coordinates of the `dpoint2d` are set to the p coordinates.
- `coord` row () const
Return the value of the `dpoint2d` row coordinate.
- `coord` & row ()
Return a reference to the `dpoint2d` row coordinate.
- `coord` col () const
Return the value of the `dpoint2d` column coordinate.
- `coord` & col ()
Return a reference to the `dpoint2d` column coordinate.

Static Public Member Functions

- `std::string name ()`

Protected Member Functions

- `dpoint2d plus_dp (const dpoint2d &dp) const`
Return a [dpoint2d](#) whose coordinates are equal to dp coordinates plus the current [dpoint2d](#) coordinates.
- `dpoint2d minus () const`
Return a [dpoint2d](#) whose coordinates are equal to the opposite of the current [dpoint2d](#) coordinates.
- `dpoint2d minus_dp (const dpoint2d &dp) const`
Return a [dpoint2d](#) whose coordinates are equal to the current [dpoint2d](#) coordinates minus dp coordinates.
- `dpoint2d & plus_assign_dp (const dpoint2d &dp)`
Return a reference to the current [dpoint2d](#) plus dp.
- `dpoint2d & minus_assign_dp (const dpoint2d &dp)`
Return a reference to the current [dpoint2d](#) minus dp.

Friends

- class `abstract::dpoint< dpoint2d >`

7.101.1 Detailed Description

Subclass of [abstract::dpoint](#), declaration of dpoint for [image2d](#). To instantiate a [dpoint2d](#) on an `oln::image2d<ntg::rgb_8>` for example, use the macro `oln_dpoint_type(I)`.

```
oln_dpoint_type(oln::image2d<ntg::rgb_8>) dp();
```

or

```
oln_dpoint_type(oln::image2d<ntg::rgb_8>) dp(1, 2);
```

Definition at line 68 of file `dpoint2d.hh`.

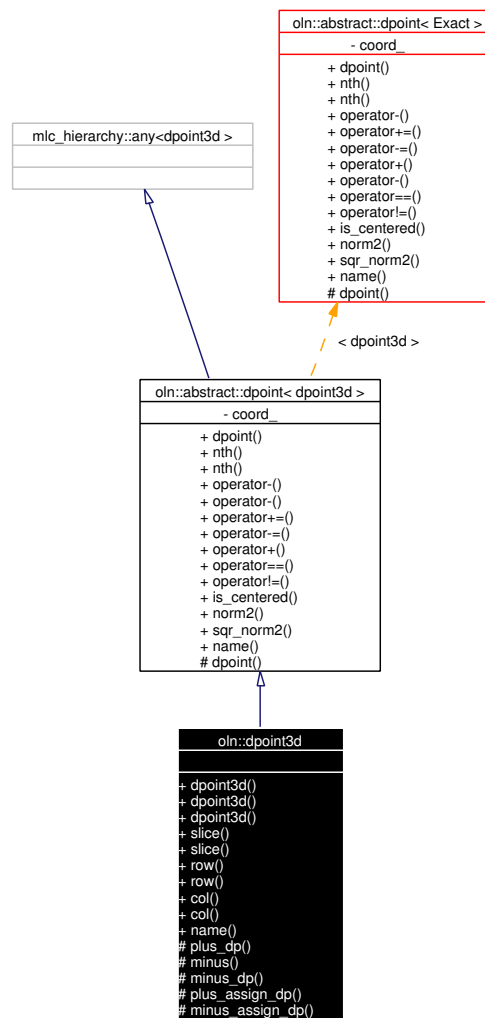
The documentation for this class was generated from the following files:

- `dpoint2d.hh`
- `dpoint2d.hxx`

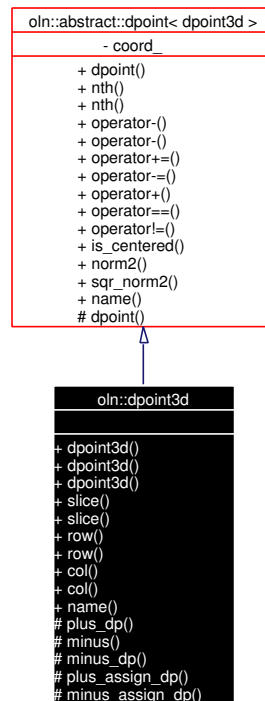
7.102 oln::dpoint3d Class Reference

```
#include <dpoint3d.hh>
```

Inheritance diagram for oln::dpoint3d:



Collaboration diagram for oln::dpoint3d:



Public Types

- typedef `abstract::dpoint< dpoint3d >` `super_type`

Public Member Functions

- `dpoint3d (coord slice, coord row, coord col)`
The coordinates of the `dpoint3d` are set to slice, row, and col.
- `dpoint3d (const point3d &p)`
The coordinates of the `dpoint3d` are set to the p coordinates.
- `coord slice () const`
Return the value of the `dpoint3d` slice coordinate.
- `coord & slice ()`
Return a reference to the `dpoint3d` slice coordinate.
- `coord row () const`
Give the value of the `dpoint3d` row coordinate.
- `coord & row ()`
Return a reference to the `dpoint3d` row coordinate.
- `coord col () const`
Return the value of the `dpoint3d` column coordinate.

- `coord & col ()`

Return a reference to the [dpoint3d](#) column coordinate.

Static Public Member Functions

- `std::string name ()`

Protected Member Functions

- `dpoint3d plus_dp (const dpoint3d &dp) const`

Return a [dpoint3d](#) whose coordinates are equal to dp coordinates plus the current [dpoint3d](#) coordinates.

- `dpoint3d minus () const`

Return a [dpoint3d](#) whose coordinates are equal to the opposite of the current [dpoint3d](#) coordinates.

- `dpoint3d minus_dp (const dpoint3d &dp) const`

Return a [dpoint3d](#) whose coordinates are equal to the current [dpoint3d](#) coordinates minus dp coordinates.

- `dpoint3d & plus_assign_dp (const dpoint3d &dp)`

Return a reference to the current [dpoint3d](#) plus dp.

- `dpoint3d & minus_assign_dp (const dpoint3d &dp)`

Return a reference to the current [dpoint3d](#) minus 'dp'.

Friends

- class `abstract::dpoint< dpoint3d >`

7.102.1 Detailed Description

Subclass of [abstract::dpoint](#), declaration of dpoint for [image3d](#). To instantiate a [dpoint3d](#) on an `oln::image3d<ntg::rgb_8>` for example, use the macro `oln_dpoint_type(I)`.

```
oln_dpoint_type(oln::image3d<ntg::rgb_8>) dp();
```

or

```
oln_dpoint_type(oln::image3d<ntg::rgb_8>) dp(1, 2, 3);
```

Definition at line 65 of file `dpoint3d.hh`.

The documentation for this class was generated from the following files:

- `dpoint3d.hh`
- `dpoint3d.hxx`

7.103 oln::dpoint_traits< abstract::dpoint< Exact > > Struct Template Reference

```
#include <dpoint.hh>
```

7.103.1 Detailed Description

template<class Exact> struct oln::dpoint_traits< abstract::dpoint< Exact > >

The specialized version for abstract::dpoint<Exact>.

Definition at line 56 of file dpoint.hh.

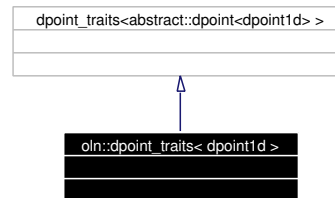
The documentation for this struct was generated from the following file:

- dpoint.hh

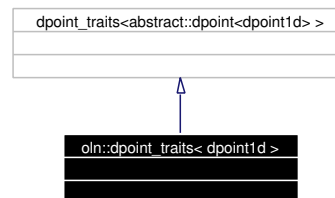
7.104 oln::dpoint_traits< dpoint1d > Struct Template Reference

```
#include <dpoint1d.hh>
```

Inheritance diagram for oln::dpoint_traits< dpoint1d >:



Collaboration diagram for oln::dpoint_traits< dpoint1d >:



Public Types

- typedef [point1d](#) point_type
- enum { **dim** = 1 }

7.104.1 Detailed Description

template<> struct oln::dpoint_traits< dpoint1d >

The specialized version for [dpoint1d](#).

Definition at line 50 of file dpoint1d.hh.

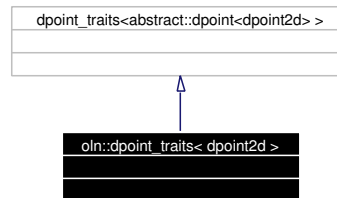
The documentation for this struct was generated from the following file:

- dpoint1d.hh

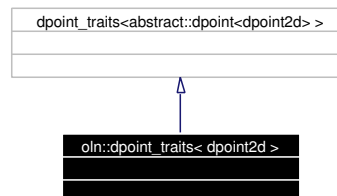
7.105 oln::dpoint_traits< dpoint2d > Struct Template Reference

```
#include <dpoint2d.hh>
```

Inheritance diagram for oln::dpoint_traits< dpoint2d >:



Collaboration diagram for oln::dpoint_traits< dpoint2d >:



Public Types

- typedef [point2d](#) point_type
- enum { **dim** = 2 }

7.105.1 Detailed Description

template<> struct oln::dpoint_traits< dpoint2d >

The specialized version for [dpoint2d](#).

Definition at line 49 of file dpoint2d.hh.

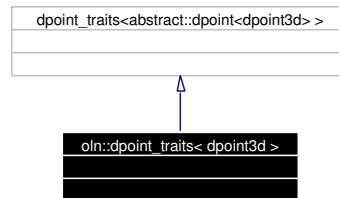
The documentation for this struct was generated from the following file:

- dpoint2d.hh

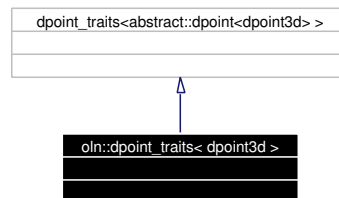
7.106 oln::dpoint_traits< dpoint3d > Struct Template Reference

```
#include <dpoint3d.hh>
```

Inheritance diagram for oln::dpoint_traits< dpoint3d >:



Collaboration diagram for oln::dpoint_traits< dpoint3d >:



Public Types

- typedef [point3d](#) `point_type`
- enum { `dim` = 3 }

7.106.1 Detailed Description

template<> struct oln::dpoint_traits< dpoint3d >

The specialized version for [dpoint3d](#).

Definition at line 47 of file `dpoint3d.hh`.

The documentation for this struct was generated from the following file:

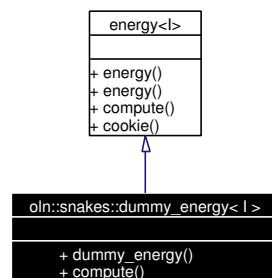
- `dpoint3d.hh`

7.107 oln::snakes::dummy_energy< I > Class Template Reference

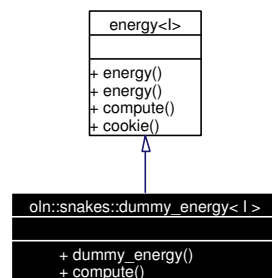
Default external energy.

```
#include <energies.hh>
```

Inheritance diagram for oln::snakes::dummy_energy< I >:



Collaboration diagram for oln::snakes::dummy_energy< I >:



Public Types

- typedef I **image_type**

Public Member Functions

- **dummy_energy** (void *)
- `::ntg::float_s` **compute** (const I &gradient, const::oln::snakes::node< I > &, const::oln::snakes::node< I > &, const::oln::snakes::node< I > &)

7.107.1 Detailed Description

```
template<class I> class oln::snakes::dummy_energy< I >
```

Default external energy.

Definition at line 183 of file energies.hh.

The documentation for this class was generated from the following file:

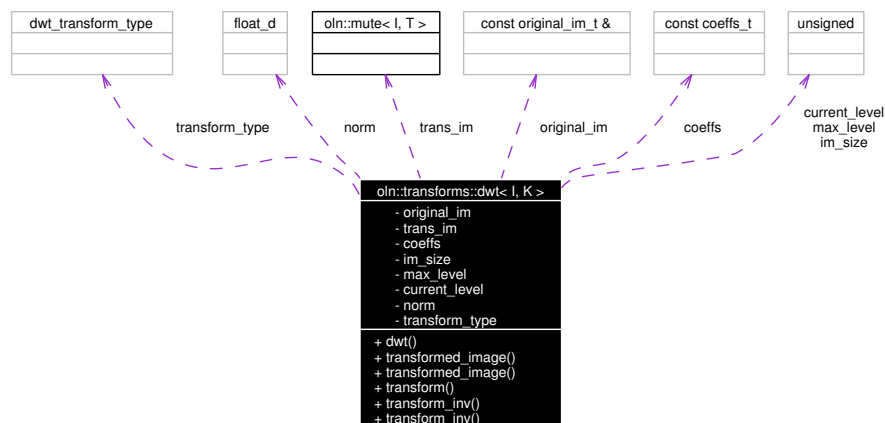
- [energies.hh](#)

7.108 oln::transforms::dwt< I, K > Class Template Reference

Object to compute dwt transforms.

```
#include <dwt.hh>
```

Collaboration diagram for oln::transforms::dwt< I, K >:



Public Types

- typedef `I` `original_im_t`
Exact of the image.
- typedef `mlc::exact< I >::ret::value_type` `original_im_value_t`
Original data type of the image.
- typedef `mute< I, ntg::float_d >::ret` `trans_im_t`
Type of the transformed image.
- typedef `K::self_t` `coeffs_t`
Data type of coefficients.

Public Member Functions

- `dwt` (`const original_im_t &im`)
Constructor from an image.
- `const trans_im_t transformed_image () const`
Accessor to the transformed image.
- `trans_im_t & transformed_image ()`
Accessor to the transformed image.
- `trans_im_t transform (dwt_transform_type t=dwt_non_std, bool normalized=true, unsigned l=0)`

Compute the dwt transform.

- [trans_im_t transform_inv](#) (unsigned l=0)

Compute the invert dwt transform.

- `template<class T1> mute< I, T1 >::ret transform_inv` (unsigned l=0)

Compute the invert dwt transform.

7.108.1 Detailed Description

`template<class I, class K> class oln::transforms::dwt< I, K >`

Object to compute dwt transforms.

Parameters:

I Exact type of the image to process.

K Type of coefficients.

Definition at line 470 of file dwt.hh.

7.108.2 Constructor & Destructor Documentation

7.108.2.1 `template<class I, class K> oln::transforms::dwt< I, K >::dwt (const original_im_t & im)` [inline]

Constructor from an image.

Initialization of dwt transform.

- *im* Image to process.

Definition at line 486 of file dwt.hh.

References `oln::transforms::dwt< I, K >::original_im_t`, `oln::transforms::dwt< I, K >::original_im_value_t`, and `oln::transforms::dwt< I, K >::trans_im_t`.

```

486                                     : original_im(im)
487     {
488         ntg_is_a(original_im_value_t, ntg::real)::ensure();
489
490         im_size = im.size().nth(0);
491         assertion(im_size >= coeffs.size());
492         for (unsigned i = 1; i < original_im_t::dim; i++)
493             assertion(im_size == static_cast<unsigned>(im.size().nth(i)));
494
495         max_level = static_cast<unsigned>(log(2 * im_size / coeffs.size()) /
496                                         internal::ln_2_);
497
498         current_level = max_level;
499
500         assertion(!(im_size % (1 << max_level)));
501
502         trans_im = trans_im_t(im.size());
503     }
```

7.108.3 Member Function Documentation

7.108.3.1 `template<class I, class K> trans_im_t oln::transforms::dwt< I, K >::transform(dwt_transform_type t = dwt_non_std, bool normalized = true, unsigned l = 0) [inline]`

Compute the dwt transform.

- *t* Type of the transform (standard or non standard)
- *normalized* Do you want a normalized result ?
- *l* Level.

Definition at line 532 of file `dwt.hh`.

References `oln::transforms::dwt< I, K >::trans_im_t`.

```

534     {
535         if (l == 0) {
536             l = max_level;
537             current_level = max_level;
538         } else {
539             if (l > max_level)
540                 l = max_level;
541             current_level = l;
542         }
543
544         oln_iter_type(trans_im_t) it(trans_im);
545
546         if (normalized) {
547             norm = pow(sqrt(2), original_im_t::dim * l);
548             for_all(it)
549                 trans_im[it] = original_im[it] / norm;
550         }
551         else {
552             norm = 1;
553             for_all(it)
554                 trans_im[it] = original_im[it];
555         }
556
557         unsigned lim = im_size >> (l - 1);
558         transform_type = t;
559
560         internal::dwt_transform_(trans_im, lim, im_size, coeffs, t);
561
562         return trans_im;
563     }

```

7.108.3.2 `template<class I, class K> template<class T1> mute<I, T1>::ret oln::transforms::dwt< I, K >::transform_inv (unsigned l = 0) [inline]`

Compute the invert dwt transform.

Parameters:

TI Data type you want the output image contains.

- *l* Level.

Definition at line 604 of file dwt.hh.

References oln::transforms::dwt< I, K >::trans_im_t, and oln::transforms::dwt< I, K >::transform_inv().

```

605     {
606         ntg_is_a(T1, ntg::real)::ensure();
607
608         trans_im_t tmp_im = transform_inv(l);
609         typename mute<I, T1>::ret new_im(tmp_im.size());
610
611         oln_iter_type(trans_im_t) it(tmp_im);
612
613         for_all(it)
614             new_im[it] = (tmp_im[it] >= ntg_inf_val(T1) ?
615                          (tmp_im[it] <= ntg_sup_val(T1) ?
616                           tmp_im [it] :
617                            ntg_sup_val(T1)) :
618                          ntg_inf_val(T1));
619     return new_im;
620     }
```

7.108.3.3 template<class I, class K> [trans_im_t](#) oln::transforms::dwt< I, K >::transform_inv (unsigned l = 0) [inline]

Compute the invert dwt transform.

- l Level.

Definition at line 570 of file dwt.hh.

References oln::transforms::dwt< I, K >::trans_im_t.

Referenced by oln::transforms::dwt< I, K >::transform_inv().

```

571     {
572         if (l == 0)
573             l = current_level;
574         else if (l > current_level)
575             l = current_level;
576
577         trans_im_t new_im(trans_im.size());
578         oln_iter_type(trans_im_t) it(trans_im);
579
580         if (norm != 1) {
581             for_all(it)
582                 new_im[it] = trans_im[it] * norm;
583         }
584         else
585             for_all(it)
586                 new_im[it] = trans_im[it];
587
588         unsigned lim1 = im_size >> (current_level - 1);
589         unsigned lim2 = im_size >> (current_level - 1);
590
591         internal::dwt_transform_inv(new_im, lim1, lim2, coeffs, transform_type);
592
593     return new_im;
594     }
```

7.108.3.4 `template<class I, class K> trans_im_t & oln::transforms::dwt< I, K >::transformed_image () [inline]`

Accessor to the transformed image.

Non const version.

Definition at line 520 of file dwt.hh.

References [oln::transforms::dwt](#)< I, K >::trans_im_t.

```
521     {  
522         return trans_im;  
523     }
```

7.108.3.5 `template<class I, class K> const trans_im_t oln::transforms::dwt< I, K >::transformed_image () const [inline]`

Accessor to the transformed image.

Const version.

Definition at line 510 of file dwt.hh.

References [oln::transforms::dwt](#)< I, K >::trans_im_t.

```
511     {  
512         return trans_im;  
513     }
```

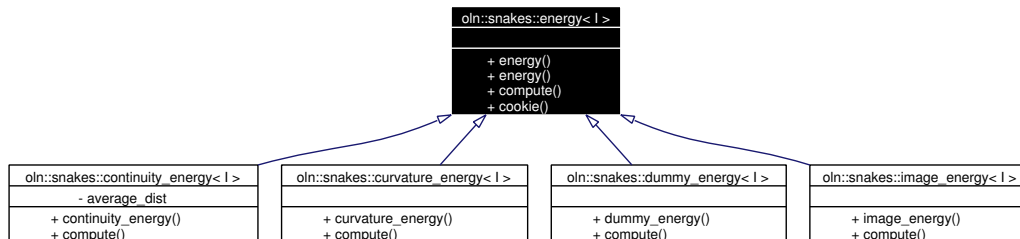
The documentation for this class was generated from the following file:

- dwt.hh

7.109 oln::snakes::energy< I > Class Template Reference

```
#include <energies.hh>
```

Inheritance diagram for oln::snakes::energy< I >:



Public Member Functions

- **energy** (void *)
- ntg::float_s **compute** (const I &, const [node](#)< I > &, const [node](#)< I > &, const [node](#)< I > &)

Static Public Member Functions

- void * **cookie** ()
FIXME: What is that?

7.109.1 Detailed Description

template<class I> class oln::snakes::energy< I >

Base class for energy.

Definition at line 38 of file energies.hh.

7.109.2 Member Function Documentation

7.109.2.1 **template<class I> ntg::float_s oln::snakes::energy< I >::compute** (const I &, const [node](#)< I > &, const [node](#)< I > &, const [node](#)< I > &) [inline]

Return the energy.

The first arg is the gradient of the image; the 3 nodes are the previous, the current and the next node.

Reimplemented in [oln::snakes::continuity_energy< I >](#), [oln::snakes::curvature_energy< I >](#), and [oln::snakes::image_energy< I >](#).

Definition at line 51 of file energies.hh.

```

52     {
53         // This is intended to cause an error. The user must define a
54         // member function named 'compute()' for each external energy,
55         // otherwise the following method will be compiled and cause an
  
```

```
56         // error.  
57         user_defined_external_energy_functor::compute();  
58         return 0.f;  
59     }
```

The documentation for this class was generated from the following file:

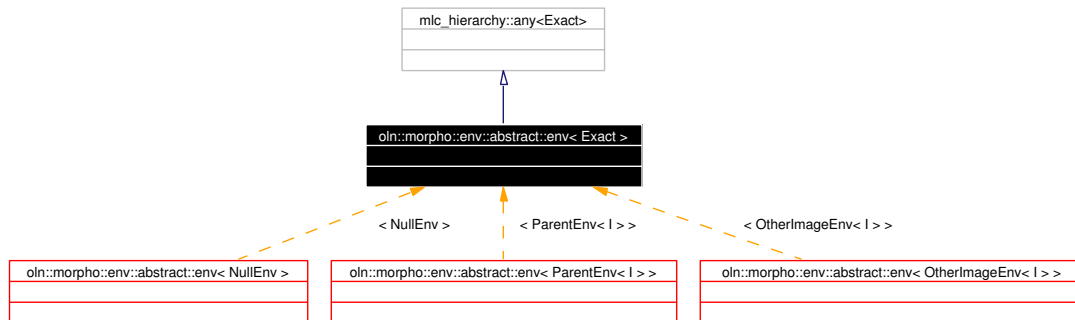
- energies.hh

7.110 oln::morpho::env::abstract::env< Exact > Struct Template Reference

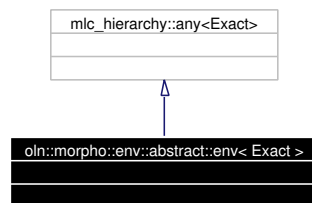
Top of environment hierarchy.

```
#include <environments.hh>
```

Inheritance diagram for oln::morpho::env::abstract::env< Exact >:



Collaboration diagram for oln::morpho::env::abstract::env< Exact >:



7.110.1 Detailed Description

template<class Exact> struct oln::morpho::env::abstract::env< Exact >

Top of environment hierarchy.

Definition at line 49 of file environments.hh.

The documentation for this struct was generated from the following file:

- environments.hh

7.111 oln::math::f_abs< T > Struct Template Reference

Absolute value fctor.

```
#include <macros.hh>
```

Public Types

- typedef T **output_t**

Public Member Functions

- const T **operator()** (const T &val) const

7.111.1 Detailed Description

```
template<class T> struct oln::math::f_abs< T >
```

Absolute value fctor.

Definition at line 68 of file olena/oln/math/macros.hh.

The documentation for this struct was generated from the following file:

- olena/oln/math/macros.hh

7.112 `oln::math::internal::f_dot_product_nv< DestValue, I, J >` Struct Template Reference

Dot product for non-vectorial types.

```
#include <macros.hh>
```

Public Types

- `typedef mlc::internal::wrap< typename mlc::internal::is_a< sizeof(mlc::form::get< ntg::non_vectorial >)) >::check< typename ntg::type_traits< I >::abstract_type, ntg::non_vectorial >::ensure_type t`

Static Public Member Functions

- `DestValue product (const I &i, const J &j)`

7.112.1 Detailed Description

`template<typename DestValue, typename I, typename J> struct oln::math::internal::f_dot_product_nv< DestValue, I, J >`

Dot product for non-vectorial types.

Definition at line 92 of file `olena/oln/math/macros.hh`.

The documentation for this struct was generated from the following file:

- `olena/oln/math/macros.hh`

7.113 `oln::math::internal::f_dot_product_v`< `DestValue`, `I`, `J` > Struct Template Reference

Dot product for vectorial types.

```
#include <macros.hh>
```

Public Types

- `typedef mlc::internal::wrap< typename mlc::internal::is_a< sizeof(mlc::form::get< ntg::vectorial >)) >::check< typename ntg::type_traits< I >::abstract_type, ntg::vectorial > >::ensure_type t`

Static Public Member Functions

- `DestValue product` (const I &i, const J &j)

7.113.1 Detailed Description

```
template<typename DestValue, typename I, typename J> struct oln::math::internal::f_dot_product_v< DestValue, I, J >
```

Dot product for vectorial types.

Definition at line 105 of file `olena/oln/math/macros.hh`.

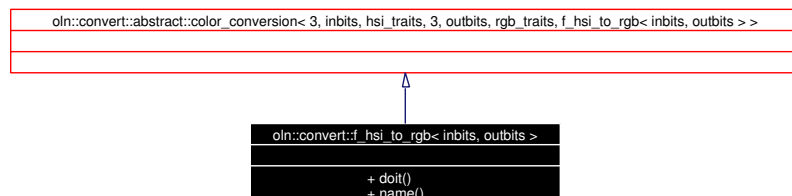
The documentation for this struct was generated from the following file:

- `olena/oln/math/macros.hh`

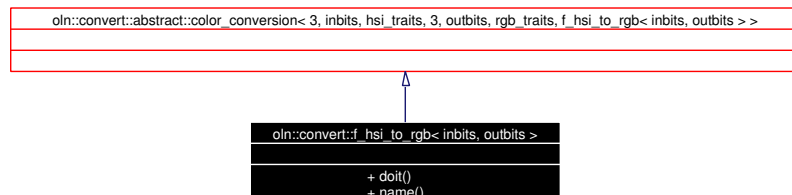
7.114 oln::convert::f_hsi_to_rgb< inbits, outbits > Struct Template Reference

```
#include <rgbhsi.hh>
```

Inheritance diagram for oln::convert::f_hsi_to_rgb< inbits, outbits >:



Collaboration diagram for oln::convert::f_hsi_to_rgb< inbits, outbits >:



Public Member Functions

- `color< 3, outbits, rgb_traits > doit (const color< 3, inbits, hsi_traits > &v) const`

Static Public Member Functions

- `std::string name ()`

7.114.1 Detailed Description

```
template<unsigned inbits, unsigned outbits> struct oln::convert::f_hsi_to_rgb< inbits, outbits >
```

Functor conversion from HSI to RGB color space.

See also:

[f_rgb_to_hsl](#)

Definition at line 110 of file `rgbhsi.hh`.

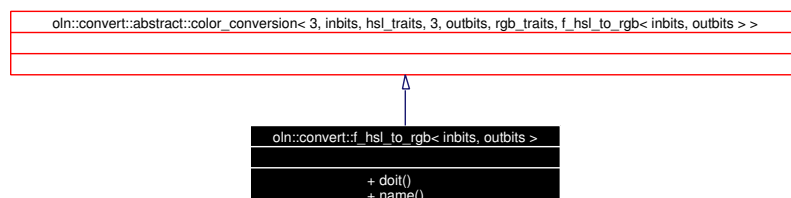
The documentation for this struct was generated from the following file:

- [rgbhsi.hh](#)

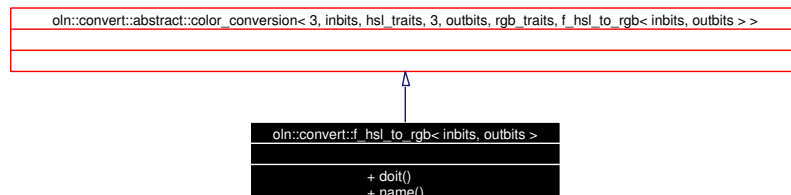
7.115 oln::convert::f_hsl_to_rgb< inbits, outbits > Struct Template Reference

```
#include <rgbhsl.hh>
```

Inheritance diagram for oln::convert::f_hsl_to_rgb< inbits, outbits >:



Collaboration diagram for oln::convert::f_hsl_to_rgb< inbits, outbits >:



Public Member Functions

- `color< 3, outbits, rgb_traits > doit (const color< 3, inbits, hsl_traits > &v) const`

Static Public Member Functions

- `std::string name ()`

7.115.1 Detailed Description

`template<unsigned inbits, unsigned outbits> struct oln::convert::f_hsl_to_rgb< inbits, outbits >`

Functor for conversion from HSL to RGB color space.

See also:

[f_rgb_to_hsl](#)

Definition at line 172 of file `rgbhsl.hh`.

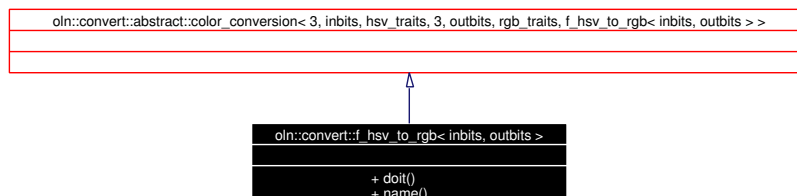
The documentation for this struct was generated from the following file:

- [rgbhsl.hh](#)

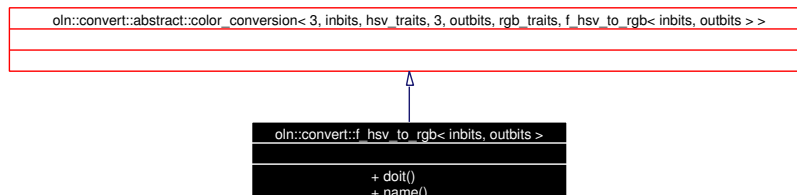
7.116 oln::convert::f_hsv_to_rgb< inbits, outbits > Struct Template Reference

```
#include <rgbhsv.hh>
```

Inheritance diagram for oln::convert::f_hsv_to_rgb< inbits, outbits >:



Collaboration diagram for oln::convert::f_hsv_to_rgb< inbits, outbits >:



Public Member Functions

- `color< 3, outbits, rgb_traits > doit (const color< 3, inbits, hsv_traits > &v) const`

Static Public Member Functions

- `std::string name ()`

7.116.1 Detailed Description

```
template<unsigned inbits, unsigned outbits> struct oln::convert::f_hsv_to_rgb< inbits, outbits >
```

Functor conversion from HSV to RGB.

See also:

[f_rgb_to_hsl](#)

Definition at line 122 of file `rgbhsv.hh`.

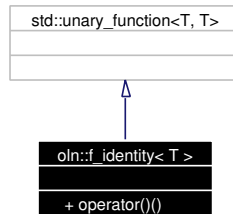
The documentation for this struct was generated from the following file:

- [rgbhsv.hh](#)

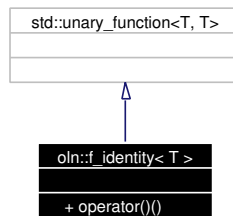
7.117 oln::f_identity< T > Struct Template Reference

```
#include <compose.hh>
```

Inheritance diagram for oln::f_identity< T >:



Collaboration diagram for oln::f_identity< T >:



Public Member Functions

- **T operator()** (T t) const

7.117.1 Detailed Description

template<class T> struct oln::f_identity< T >

This functor returns its argument.

Definition at line 159 of file `compose.hh`.

The documentation for this struct was generated from the following file:

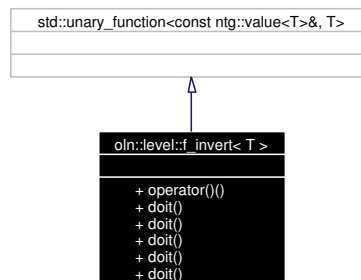
- `compose.hh`

7.118 oln::level::f_invert< T > Struct Template Reference

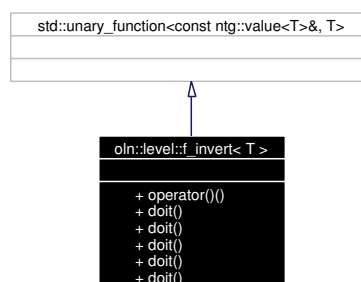
Fctor to invert a value.

```
#include <invert.hh>
```

Inheritance diagram for oln::level::f_invert< T >:



Collaboration diagram for oln::level::f_invert< T >:



Public Types

- typedef [f_invert](#) self

Public Member Functions

- const self::result_type **operator()** (typename self::argument_type val) const

Static Public Member Functions

- template<unsigned N, class B> const ntg::int_s< N, B > **doit** (const ntg::int_s< N, B > &val)
- const ntg::float_d **doit** (const ntg::float_d &val)
- const ntg::float_s **doit** (const ntg::float_s &val)
- template<unsigned N, class B> const ntg::int_u< N, B > **doit** (const ntg::int_u< N, B > &val)
- const ntg::bin **doit** (ntg::bin val)

7.118.1 Detailed Description

template<class T> struct oln::level::f_invert< T >

Fctor to invert a value.

See also:

[invert](#).

Todo

FIXME: the specialisation is done within the class.

Definition at line 48 of file invert.hh.

The documentation for this struct was generated from the following file:

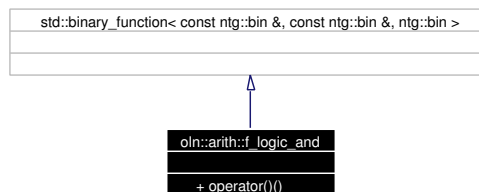
- invert.hh

7.119 oln::arith::f_logic_and Struct Reference

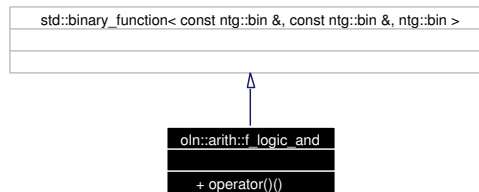
Functor AND operators.

```
#include <logic.hh>
```

Inheritance diagram for oln::arith::f_logic_and:



Collaboration diagram for oln::arith::f_logic_and:



Public Member Functions

- `const result_type operator() (first_argument_type val1, second_argument_type val2) const`

7.119.1 Detailed Description

Functor AND operators.

Definition at line 44 of file `logic.hh`.

The documentation for this struct was generated from the following file:

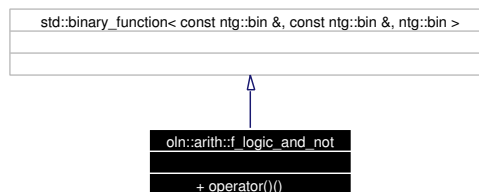
- `logic.hh`

7.120 oln::arith::f_logic_and_not Struct Reference

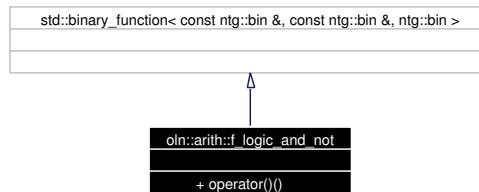
Functor AND NOT operators.

```
#include <logic.hh>
```

Inheritance diagram for oln::arith::f_logic_and_not:



Collaboration diagram for oln::arith::f_logic_and_not:



Public Member Functions

- `const result_type operator() (first_argument_type val1, second_argument_type val2) const`

7.120.1 Detailed Description

Functor AND NOT operators.

Definition at line 53 of file `logic.hh`.

The documentation for this struct was generated from the following file:

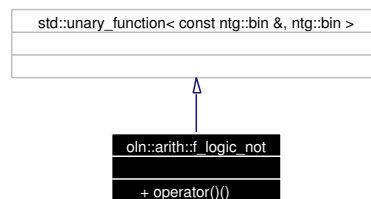
- `logic.hh`

7.121 oln::arith::f_logic_not Struct Reference

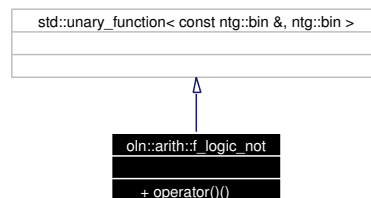
Functor NOT operator.

```
#include <logic.hh>
```

Inheritance diagram for oln::arith::f_logic_not:



Collaboration diagram for oln::arith::f_logic_not:



Public Member Functions

- `const result_type operator() (argument_type val) const`

7.121.1 Detailed Description

Functor NOT operator.

Definition at line 56 of file `logic.hh`.

The documentation for this struct was generated from the following file:

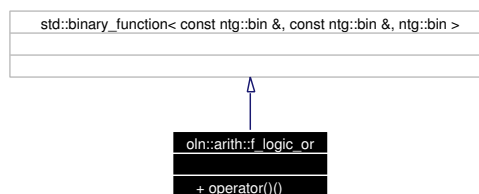
- `logic.hh`

7.122 oln::arith::f_logic_or Struct Reference

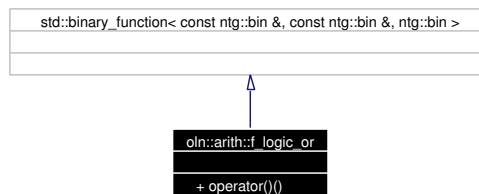
Functor OR operators.

```
#include <logic.hh>
```

Inheritance diagram for oln::arith::f_logic_or:



Collaboration diagram for oln::arith::f_logic_or:



Public Member Functions

- `const result_type` **operator()** (`first_argument_type val1`, `second_argument_type val2`) `const`

7.122.1 Detailed Description

Functor OR operators.

Definition at line 49 of file `logic.hh`.

The documentation for this struct was generated from the following file:

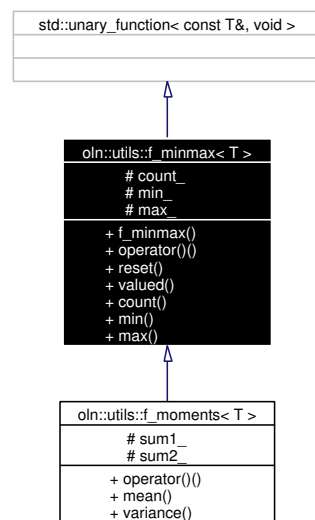
- `logic.hh`

7.123 oln::utils::f_minmax< T > Struct Template Reference

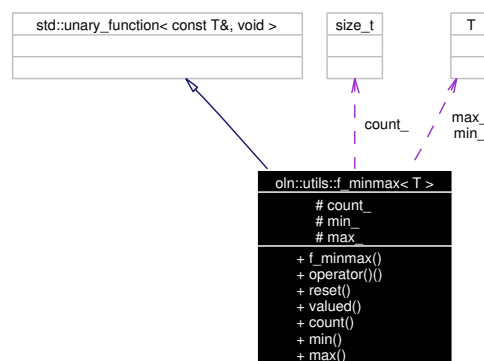
Unary function that stores the min and the max.

```
#include <stat.hh>
```

Inheritance diagram for oln::utils::f_minmax< T >:



Collaboration diagram for oln::utils::f_minmax< T >:



Public Member Functions

- void **operator**() (const T &val)
- void **reset** ()
- bool **valued** () const
True if a value has been tested.
- size_t **count** () const
Number of value has been tested.

- `const T min () const`
Minimum found.
- `const T max () const`
Maximum found.

Protected Attributes

- `size_t count_`
- `T min_`
- `T max_`

7.123.1 Detailed Description

`template<class T> struct oln::utils::f_minmax< T >`

Unary function that stores the min and the max.

Definition at line 40 of file `utils/stat.hh`.

The documentation for this struct was generated from the following file:

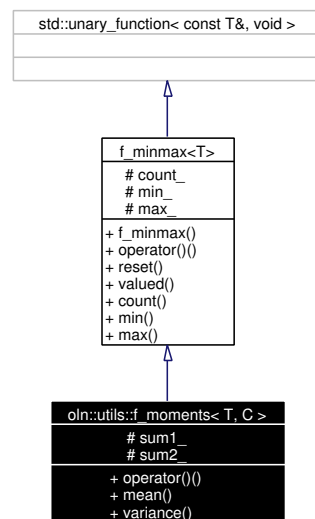
- `utils/stat.hh`

7.124 oln::utils::f_moments< T, C > Struct Template Reference

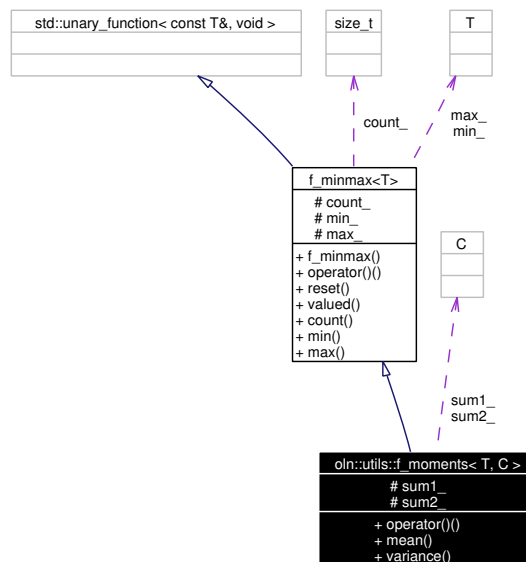
Computes the mean, the variance and store the min, the max.

```
#include <stat.hh>
```

Inheritance diagram for oln::utils::f_moments< T, C >:



Collaboration diagram for oln::utils::f_moments< T, C >:



Public Types

- typedef `f_minmax< T >` **super**

Public Member Functions

- void **operator()** (const T &val)
- const C **mean** () const
Return the mean value.
- const C **variance** () const
Return the variance.

Protected Attributes

- C **sum1_**
- C **sum2_**

7.124.1 Detailed Description

template<class T, class C = ntg::float_s> struct oln::utils::f_moments< T, C >

Computes the mean, the variance and store the min, the max.

Definition at line 102 of file utils/stat.hh.

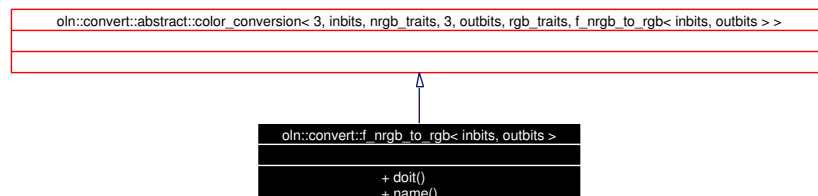
The documentation for this struct was generated from the following file:

- utils/stat.hh

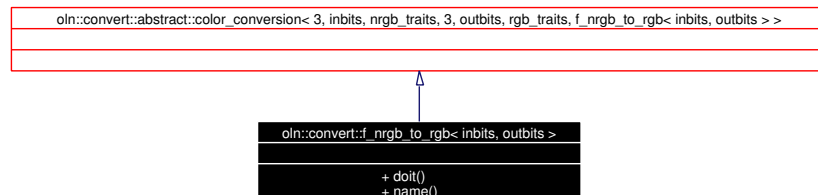
7.125 oln::convert::f_nrgb_to_rgb< inbits, outbits > Struct Template Reference

```
#include <rgbnrgb.hh>
```

Inheritance diagram for oln::convert::f_nrgb_to_rgb< inbits, outbits >:



Collaboration diagram for oln::convert::f_nrgb_to_rgb< inbits, outbits >:



Public Member Functions

- `color< 3, outbits, rgb_traits > doit (const color< 3, inbits, nrgb_traits > &v) const`

Static Public Member Functions

- `std::string name ()`

7.125.1 Detailed Description

`template<unsigned inbits, unsigned outbits> struct oln::convert::f_nrgb_to_rgb< inbits, outbits >`

Functor for conversion from N-RGB to RGB.

See also:

[f_rgb_to_hsl](#)

Definition at line 102 of file `rgbnrgb.hh`.

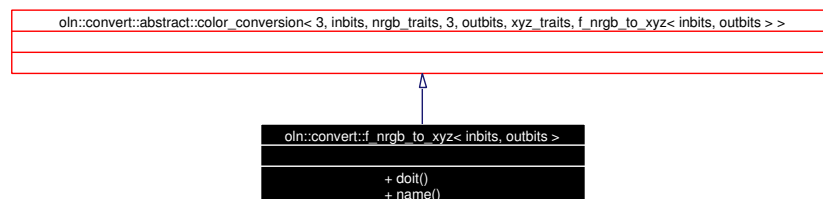
The documentation for this struct was generated from the following file:

- [rgbnrgb.hh](#)

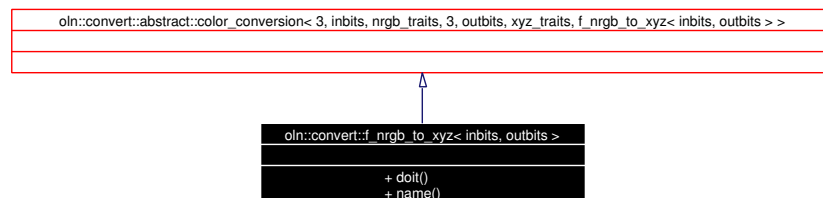
7.126 oln::convert::f_nrgb_to_xyz< inbits, outbits > Struct Template Reference

```
#include <nrgbxyz.hh>
```

Inheritance diagram for oln::convert::f_nrgb_to_xyz< inbits, outbits >:



Collaboration diagram for oln::convert::f_nrgb_to_xyz< inbits, outbits >:



Public Member Functions

- `color< 3, outbits, xyz_traits > doit (const color< 3, inbits, nrgb_traits > &v) const`

Static Public Member Functions

- `std::string name ()`

7.126.1 Detailed Description

`template<unsigned inbits, unsigned outbits> struct oln::convert::f_nrgb_to_xyz< inbits, outbits >`

Functor for conversion from N-RGB to XYZ color space.

Deprecated

A composition should be performed with `nrgb->rgb` and `rgb->xyz`. It has not been replaced within the function because a double conversion 'reduces' the color space. See the following example:

```
// Obsolete:
//
// #include <oln/convert/nrgbxyz.hh>
// #include <ntg/all.hh>
// int main(int argc, char **argv)
// {
//     ntg::nrgb_8 in(100, 60, 64);
```

```
//  ntg::xyz_8 out = oln::convert::f_nrgb_to_xyz<8, 8>()(in);  
// }  
//  
// Should be replaced by:  
//  
#include <oln/convert/rgbxyz.hh>  
#include <oln/convert/rgbnrgb.hh>  
#include <ntg/all.hh>  
int main()  
{  
  ntg::nrgb_8 in(100, 60, 64);  
  ntg::xyz_8 out = oln::convert::f_rgb_to_xyz<8, 8>()  
  (oln::convert::f_nrgb_to_rgb<8, 8>()(in));  
}
```

Definition at line 83 of file nrgbxyz.hh.

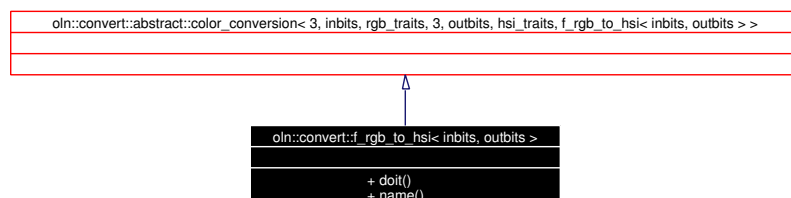
The documentation for this struct was generated from the following file:

- [nrgbxyz.hh](#)

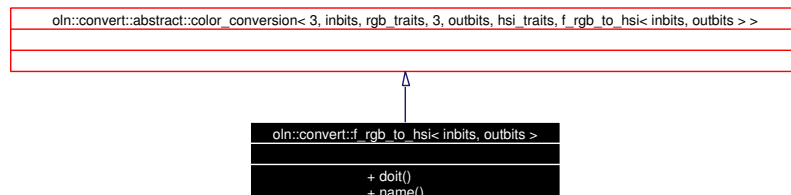
7.127 oln::convert::f_rgb_to_hsi< inbits, outbits > Struct Template Reference

```
#include <rgbhsi.hh>
```

Inheritance diagram for oln::convert::f_rgb_to_hsi< inbits, outbits >:



Collaboration diagram for oln::convert::f_rgb_to_hsi< inbits, outbits >:



Public Member Functions

- `color< 3, inbits, hsi_traits > doit (const color< 3, outbits, rgb_traits > &v) const`

Static Public Member Functions

- `std::string name ()`

7.127.1 Detailed Description

```
template<unsigned inbits, unsigned outbits> struct oln::convert::f_rgb_to_hsi< inbits, outbits >
```

Functor for conversion from RGB to HSI color space.

See also:

[f_rgb_to_hsl](#)

Definition at line 61 of file `rgbhsi.hh`.

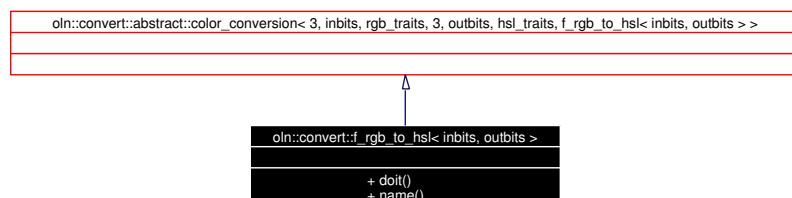
The documentation for this struct was generated from the following file:

- [rgbhsi.hh](#)

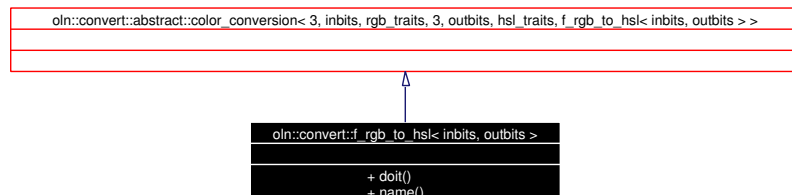
7.128 oln::convert::f_rgb_to_hsl< inbits, outbits > Struct Template Reference

```
#include <rgbhsl.hh>
```

Inheritance diagram for oln::convert::f_rgb_to_hsl< inbits, outbits >:



Collaboration diagram for oln::convert::f_rgb_to_hsl< inbits, outbits >:



Public Member Functions

- `color< 3, outbits, hsl_traits > doit (const color< 3, inbits, rgb_traits > &v) const`

Static Public Member Functions

- `std::string name ()`

7.128.1 Detailed Description

```
template<unsigned inbits, unsigned outbits> struct oln::convert::f_rgb_to_hsl< inbits, outbits >
```

Functor for conversion from RGB to HSL color space.

Note:

Every conversion should go through the RGB color space.

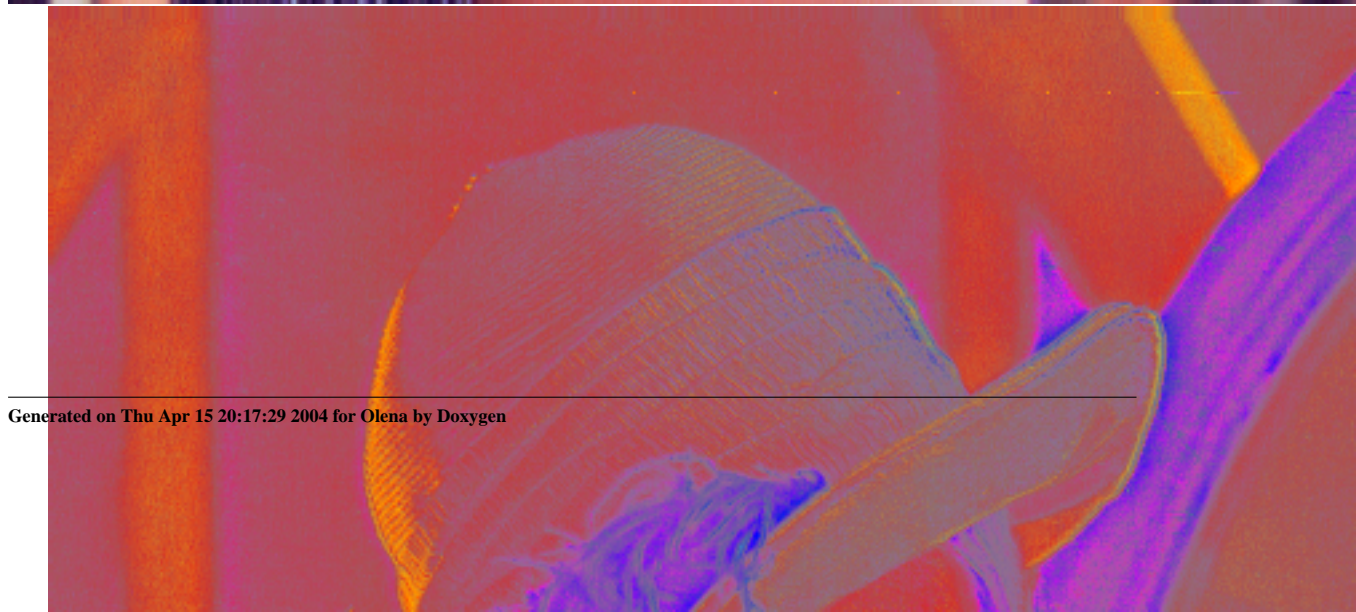
```

#include <oln/basics2d.hh>
#include <oln/convert/rgbhsl.hh>
#include <ntg/all.hh>

int main()
{
    oln::image2d<ntg::rgb_8> lena_rgb = oln::load(IMG_IN "lena.ppm");
}
  
```

```
oln::image2d<ntg::hsl_8> lena_hsl = apply(oln::convert::f_rgb_to_hsl<8, 8>(), lena_rgb);
oln::image2d<ntg::hsl_8>::iter_type it(lena_hsl);
for_all(it)
    lena_hsl[it][ntg::hsl_L] = 127;

oln::io::save(apply(oln::convert::f_hsl_to_rgb<8, 8>(), lena_hsl),
              IMG_OUT "oln_convert_f_rgb_to_hsl.pgm");
}
```



Definition at line 83 of file rgbhsl.hh.

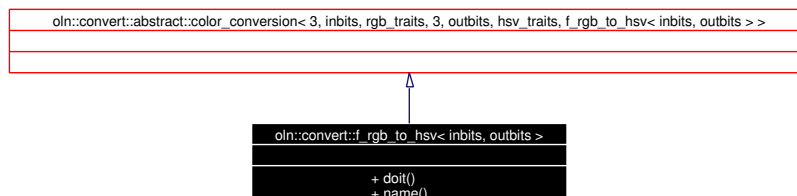
The documentation for this struct was generated from the following file:

- [rgbhsl.hh](#)

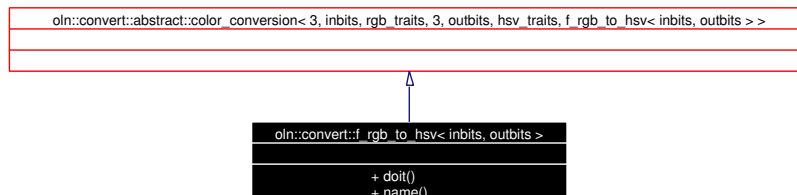
7.129 oln::convert::f_rgb_to_hsv< inbits, outbits > Struct Template Reference

```
#include <rgbhsv.hh>
```

Inheritance diagram for oln::convert::f_rgb_to_hsv< inbits, outbits >:



Collaboration diagram for oln::convert::f_rgb_to_hsv< inbits, outbits >:



Public Member Functions

- [color](#)< 3, outbits, hsv_traits > **doit** (const [color](#)< 3, inbits, rgb_traits > &v) const

Static Public Member Functions

- std::string **name** ()

7.129.1 Detailed Description

```
template<unsigned inbits, unsigned outbits> struct oln::convert::f_rgb_to_hsv< inbits, outbits >
```

Functor for conversion from RGB to HSV.

See also:

[f_rgb_to_hsl](#)

Definition at line 58 of file rgbhsv.hh.

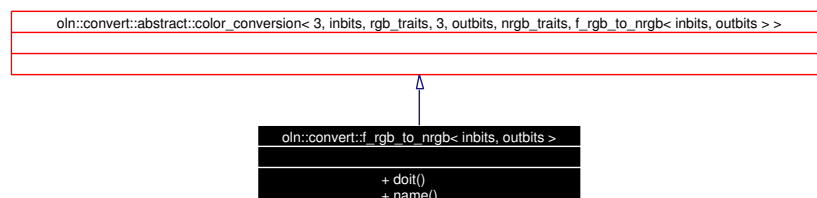
The documentation for this struct was generated from the following file:

- [rgbhsv.hh](#)

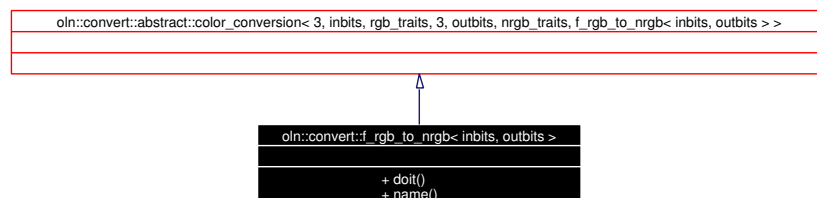
7.130 oln::convert::f_rgb_to_nrgb< inbits, outbits > Struct Template Reference

```
#include <rgbnrgb.hh>
```

Inheritance diagram for oln::convert::f_rgb_to_nrgb< inbits, outbits >:



Collaboration diagram for oln::convert::f_rgb_to_nrgb< inbits, outbits >:



Public Member Functions

- `color< 3, outbits, nrgb_traits > doit (const color< 3, inbits, rgb_traits > &v) const`

Static Public Member Functions

- `std::string name ()`

7.130.1 Detailed Description

`template<unsigned inbits, unsigned outbits> struct oln::convert::f_rgb_to_nrgb< inbits, outbits >`

Functor for conversion from RGB to N-RGB.

See also:

[f_rgb_to_hsl](#)

Definition at line 57 of file `rgbnrgb.hh`.

The documentation for this struct was generated from the following file:

- [rgbnrgb.hh](#)

7.131 oln::math::f_sqr< T > Struct Template Reference

Square fctor.

```
#include <macros.hh>
```

Public Types

- typedef T **output_t**

Public Member Functions

- const T **operator()** (const T &val) const

7.131.1 Detailed Description

```
template<class T> struct oln::math::f_sqr< T >
```

Square fctor.

Definition at line 49 of file olena/oln/math/macros.hh.

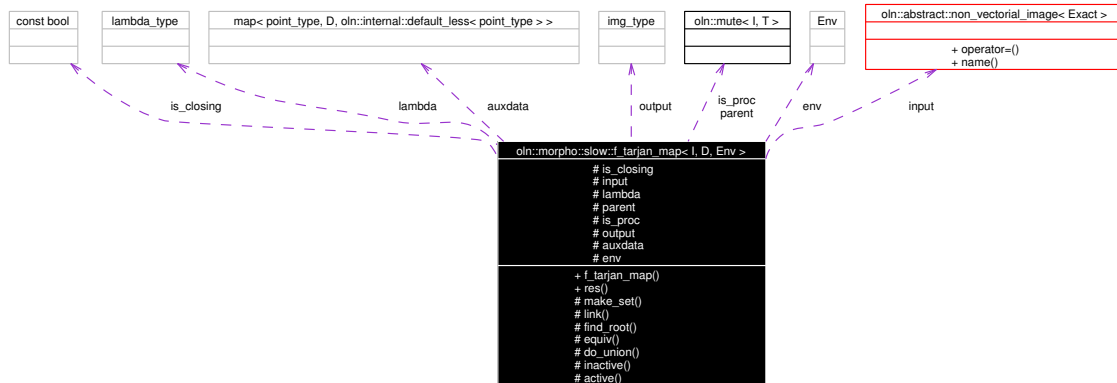
The documentation for this struct was generated from the following file:

- olena/oln/math/macros.hh

7.132 oln::morpho::slow::f_tarjan_map< I, D, Env > Struct Template Reference

```
#include <attribute_closing_opening_map.hh>
```

Collaboration diagram for oln::morpho::slow::f_tarjan_map< I, D, Env >:



Public Types

- typedef `abstract::non_vectorial_image< I >` `input_type`
Image type (abstract).
- typedef `oln::mute< I >::ret` `img_type`
Image type (concrete).
- typedef `mlc::exact< input_type >::ret::point_type` `point_type`
Associated point type.
- typedef `mlc::exact< input_type >::ret::value_type` `value_type`
Value type for the image.
- typedef `mute< input_type, point_type >::ret` `parent_type`
Image of points.
- typedef `mute< input_type, bool >::ret` `is_proc_type`
Image of bool.
- typedef `oln::morpho::attr::attr_traits< D >::lambda_type` `lambda_type`
Threshold type.

Public Member Functions

- template<class N> `f_tarjan_map` (bool `is_closing`, const `input_type` &`input`, const `abstract::neighborhood< N >` &`ng`, const `lambda_type` &`lambda`, const Env &`env`=Env())

Perform an attribute opening/closing.

- `oln::mute< I >::ret res ()`

Return the result of the opening/closing.

Protected Member Functions

- `void make_set (const point_type &x)`

Make a new component from a point.

– *x Root of the component.*

- `void link (const point_type &x, const point_type &y)`

link two components

– *x A point of the first component.*

– *y A point of the second component.*

- `point_type find_root (const point_type &x)`

find the root of a component.

– *x A point of the component.*

- `bool equiv (const point_type &x, const point_type &y) const`

check if two components are equivalent.

- `void do_union (const point_type &n, const point_type &p)`

link two components if they have to be linked

– *n A point of the first component.*

– *p A point of the second component.*

Static Protected Member Functions

- `const point_type inactive ()`

Return the value of an inactive point.

- `const point_type active ()`

Return the value of an active point.

Protected Attributes

- `const bool is_closing`

Do you want a closing or an openng ?

- `const input_type & input`

Input image.

- `lambda_type lambda`

Trheshold.

- [parent_type parent](#)

Give a parent of a point.

- [is_proc_type is_proc](#)

Tell if a point has already been preceded.

- [img_type output](#)

Image to store the result.

- `std::map< point_type, D, oln::internal::default_less< point_type > > auxdata`

Map to store attributes.

- Env [env](#)

The environment.

7.132.1 Detailed Description

template<class I, class D, class Env = morpho::env::NullEnv> struct oln::morpho::slow::f_tarjan_map< I, D, Env >

Attribute closing using map. Smaller memory usage, but slower computation than the `attribute_closing_opening`

See "Fast morphological attribute operations using Tarjan's union-find algorithm" by Michael H. F. Wilkinson and Jos B. T. M. Roerdink

Parameters:

I Image exact type.

D Attribute exact type.

Env Type of environment.

Definition at line 65 of file `attribute_closing_opening_map.hh`.

7.132.2 Constructor & Destructor Documentation

7.132.2.1 `template<class I, class D, class Env = morpho::env::NullEnv> template<class N> oln::morpho::slow::f_tarjan_map< I, D, Env >::f_tarjan_map (bool is_closing, const input_type & input, const abstract::neighborhood< N > & ng, const lambda_type & lambda, const Env & env = Env())`

Perform an attribute opening/closing.

Parameters:

N Exact type of neighborhood

- *is_closing* Choose between closing and opening.
- *input* Input image.

- ng Neighborhood to use.
- lambda Threshold.
- env Environment.

The documentation for this struct was generated from the following files:

- attribute_closing_opening_map.hh
- attribute_closing_opening_map.hxx

7.133 oln::utils::internal::f_to_float_< DestT, SrcT > Struct Template Reference

```
#include <se_stat.hh>
```

Static Public Member Functions

- DestT **doit** (const SrcT &s)

7.133.1 Detailed Description

template<typename DestT, typename SrcT> struct oln::utils::internal::f_to_float_< DestT, SrcT >

This class is used to transform a type to a floating type equivalent.

Todo

FIXME: There should be a way to use the standard conversion, but I failed to find a good one. The problem is that a color<...> derived of vec<..> but the conversion between color and vec should be done through c.to_float().

Example:

```
** rgb_u8::float_vec_type v = rgb_u8(127, 0, 255);
** // v contains 127., 0., 255 but should contain 0.5, 0,1.
** rgb_u8 c = v;
** // c = (255., 255., 255). Error.
**
** rgb_u8::float_vec_type v2 = f_to_float_<rgb_u8::float_vec_type>
**   rgb_u8(127, 0, 255);
** rgb_u8 c2 = v2;
** // c2 = (127, 0, 255). Ok.
**
```

Definition at line 65 of file se_stat.hh.

The documentation for this struct was generated from the following file:

- se_stat.hh

7.134 oln::utils::internal::f_to_float_< typename ntg::color< ncomps, qbits, color_system >::float_vec_type, ntg::color< ncomps, qbits, color_system > > Struct Template Reference

Specialization of the *f_to_float* struct for the colors.

```
#include <se_stat.hh>
```

Static Public Member Functions

- `ntg::color< ncomps, qbits, color_system >::float_vec_type doit` (const `ntg::color< ncomps, qbits, color_system > &s`)

7.134.1 Detailed Description

```
template<unsigned ncomps, unsigned qbits, template< unsigned > class color_system> struct  
oln::utils::internal::f_to_float_< typename ntg::color< ncomps, qbits, color_system >::float_vec_  
type, ntg::color< ncomps, qbits, color_system > >
```

Specialization of the *f_to_float* struct for the colors.

Definition at line 79 of file `se_stat.hh`.

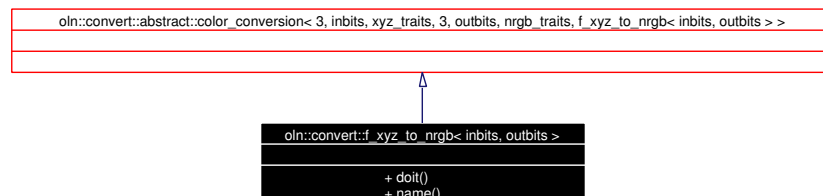
The documentation for this struct was generated from the following file:

- `se_stat.hh`

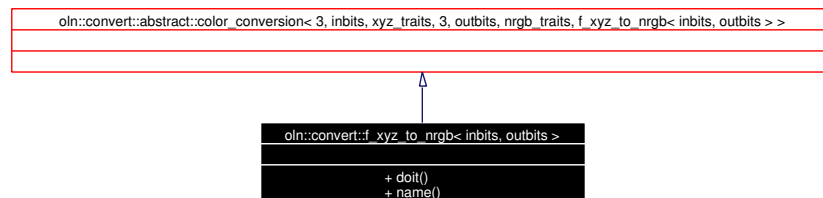
7.135 oln::convert::f_xyz_to_nrgb< inbits, outbits > Struct Template Reference

```
#include <nrgbxyz.hh>
```

Inheritance diagram for oln::convert::f_xyz_to_nrgb< inbits, outbits >:



Collaboration diagram for oln::convert::f_xyz_to_nrgb< inbits, outbits >:



Public Member Functions

- [color](#)< 3, outbits, nrgb_traits > **doit** (const [color](#)< 3, inbits, xyz_traits > &v) const

Static Public Member Functions

- std::string **name** ()

7.135.1 Detailed Description

`template<unsigned inbits, unsigned outbits> struct oln::convert::f_xyz_to_nrgb< inbits, outbits >`

Functor for conversion from XYZ to N-RGB color space.

Deprecated

A composition should be performed with `xyz->rgb` and `rgb->nrgb`.

See also:

[f_nrgb_to_xyz](#) for more information.

Definition at line 135 of file `nrgbxyz.hh`.

The documentation for this struct was generated from the following file:

- [nrgbxyz.hh](#)

7.136 oln::morpho::internal::fast_morpho_inner< NP1, Dim, I, S, H, B, P, O > Struct Template Reference

```
#include <fast_morpho.hxx>
```

Static Public Member Functions

- void [doit](#) (I &input, S &size, H &hist, B *se_add, B *se_rem, B *se_add_back, B *se_rem_back, P &p, O &output, const unsigned *dims)

7.136.1 Detailed Description

```
template<unsigned NP1, unsigned Dim, typename I, typename S, typename H, typename B, type-
name P, typename O> struct oln::morpho::internal::fast_morpho_inner< NP1, Dim, I, S, H, B, P, O
>
```

We will zigzag over the image so that only one coordinate changes at each step. The path looks as follow on 2D images:

```
—————\ | /—————/ | \—————\ | —————/
```

(The algorithm below handles the n-dimensional case.)

Definition at line 161 of file fast_morpho.hxx.

7.136.2 Member Function Documentation

7.136.2.1 template<unsigned NP1, unsigned Dim, typename I, typename S, typename H, typename B, typename P, typename O> void [oln::morpho::internal::fast_morpho_inner](#)< NP1, Dim, I, S, H, B, P, O >::doit (I &input, S &size, H &hist, B *se_add, B *se_rem, B *se_add_back, B *se_rem_back, P &p, O &output, const unsigned *dims) [inline, static]

Perform the action.

Definition at line 167 of file fast_morpho.hxx.

References [oln::morpho::internal::hist_update\(\)](#).

```
170     {
171         const unsigned N = *dims;
172
173         fast_morpho_inner<NP1 + 1, Dim,
174             I, S, H, B, P, O>::doit(input, size, hist,
175                                     se_add, se_rem,
176                                     se_add_back, se_rem_back, p,
177                                     output, dims + 1);
178         if (p.nth(N) == 0) { // Go forward
179             for(++p.nth(N); p.nth(N) < size.nth(N); ++p.nth(N)) {
180                 hist_update(hist, input, p, se_rem[N], se_add[N]);
181                 output[p] = hist.res();
182                 fast_morpho_inner<NP1 + 1, Dim,
183                     I, S, H, B, P, O>::doit(input, size, hist,
184                                             se_add, se_rem,
185                                             se_add_back,
186                                             se_rem_back,
```

```
187                                     p, output, dims + 1);
188     }
189     --p.nth(N);
190 } else {                               // Go backward
191     for(--p.nth(N); p.nth(N) >= 0; --p.nth(N)) {
192         hist_update(hist, input, p, se_rem_back[N], se_add_back[N]);
193         output[p] = hist.res();
194         fast_morpho_inner<NP1 + 1, Dim,
195             I, S, H, B, P, O>::doit(input, size, hist,
196                                     se_add, se_rem,
197                                     se_add_back,
198                                     se_rem_back,
199                                     p, output, dims + 1);
200     }
201     ++p.nth(N);
202 }
203 return;
204 }
```

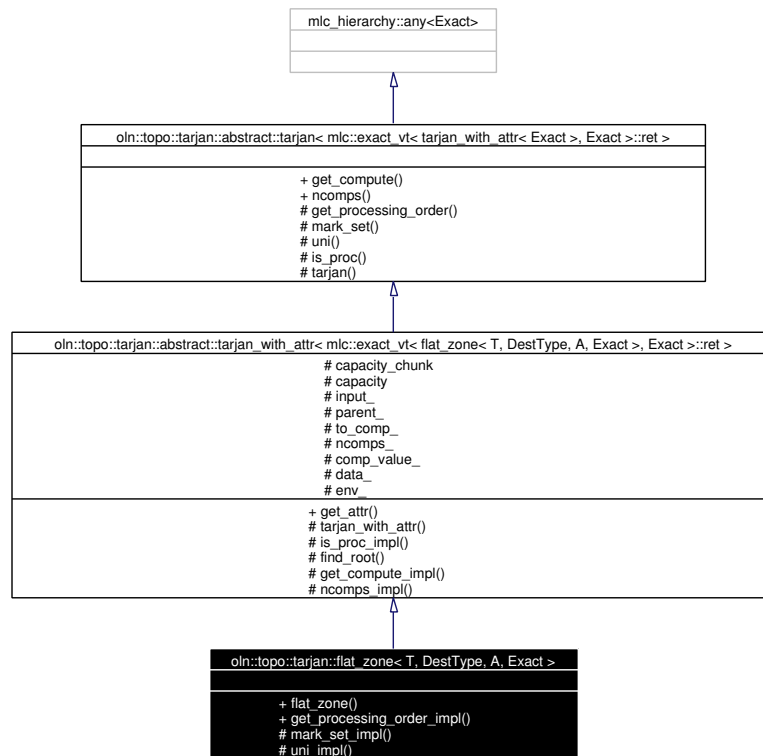
The documentation for this struct was generated from the following file:

- fast_morpho.hxx

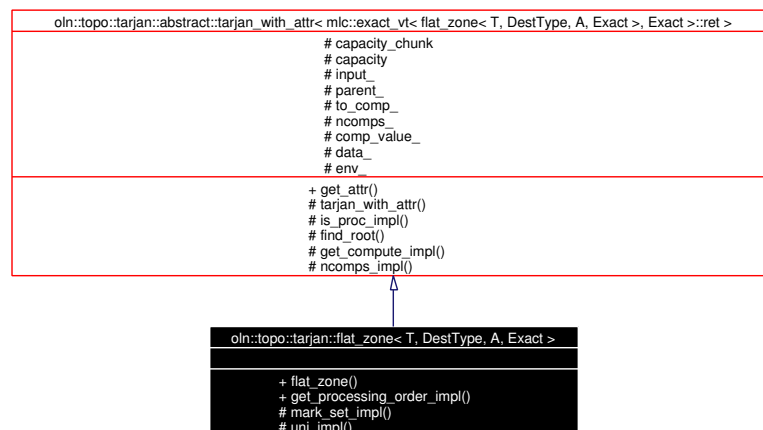
7.137 oln::topo::tarjan::flat_zone< T, DestType, A, Exact > Struct Template Reference

```
#include <flat-zone.hh>
```

Inheritance diagram for oln::topo::tarjan::flat_zone< T, DestType, A, Exact >:



Collaboration diagram for oln::topo::tarjan::flat_zone< T, DestType, A, Exact >:



Public Types

- typedef `mlc::exact< T >::ret::point_type` [point_type](#)
Type of input image.
- typedef `mlc::exact< T >::ret::value_type` [data_type](#)
Data type of the input image.
- typedef `oln::mute< T >::ret image_type`
Concrete type of the input image.
- typedef `DestType` [comp_type](#)
Type of components.
- typedef `flat_zone< T, DestType, A, Exact >` [self_type](#)
Self type of the class.
- typedef `mlc::exact_vt< self_type, Exact >::ret exact_type`
Exact type of the class.
- typedef `abstract::tarjan_with_attr< exact_type >` [super_type](#)
Type of parent class.

Public Member Functions

- [flat_zone](#) (const [image_type](#) &ima, const typename `oln::morpho::attr::attr_traits< A >::env_type` &env=typename `oln::morpho::attr::attr_traits< A >::env_type()`)
Constructor.
- `std::vector< point_type >` [get_processing_order_impl](#) ()

Protected Member Functions

- void [mark_set_impl](#) (const [point_type](#) &x)
Implementation of [mark_set\(\)](#).
- void [uni_impl](#) (const [point_type](#) &n, const [point_type](#) &p)
Implementation of [uni\(\)](#).

Friends

- class `abstract::tarjan< exact_type >`

7.137.1 Detailed Description

template<class T, class DestType = unsigned, class A = oln::morpho::attr::card_type<>, class Exact = mlc::final> **struct** oln::topo::tarjan::flat_zone< T, DestType, A, Exact >

Create an image of label of the flat zones.

Parameters:

T Type of the input image.

DestType Data type of the output image (label type).

A Attribute you want to compute.

Exact Exact type of the class.

```
#include <oln/basics2d.hh>
#include <oln/topo/tarjan/flat-zone.hh>
#include <oln/convert/stretch.hh>

int main()
{
    typedef oln::image2d<ntg::int_u8> img_type;
    img_type in = oln::load(IMG_IN "test-cmap.pgm");
    oln::topo::tarjan::flat_zone<img_type> z(in);
    save(oln::convert::stretch_balance<ntg::int_u8>(z.get_compute(oln::neighb_c4()), 0, 255),
        IMG_OUT "oln_topo_flat_zone.pgm");
}
```



Figure 7.1: input image

=>



Figure 7.2: output image

Definition at line 75 of file flat-zone.hh.

7.137.2 Constructor & Destructor Documentation

7.137.2.1 **template**<class T, class DestType = unsigned, class A = oln::morpho::attr::card_type<>, class Exact = mlc::final> **oln::topo::tarjan::flat_zone**< T, DestType, A, Exact >::flat_zone (const **image_type** & *ima*, const typename oln::morpho::attr::attr_traits< A >::env_type & *env* = typename oln::morpho::attr::attr_traits< A >::env_type()) [inline]

Constructor.

- *ima* Input image.
- *env* Environment to use to compute the attributes.

Definition at line 98 of file flat-zone.hh.

```

99                                     : super_type(ima, env)
100     {
101     }
```

7.137.3 Member Function Documentation

7.137.3.1 `template<class T, class DestType = unsigned, class A = oln::morpho::attr::card_type<>, class Exact = mlc::final> void oln::topo::tarjan::flat_zone< T, DestType, A, Exact >::mark_set_impl (const point_type &x) [inline, protected]`

Implementation of [mark_set\(\)](#).

- x Root of the new component.

Warning:

Do not call this method, use [mark_set\(\)](#) instead.

Definition at line 125 of file flat-zone.hh.

```

126     {
127         if (parent_.size() == parent_.capacity())
128         {
129             capacity = parent_.capacity() + capacity_chunk;
130             parent_.reserve(capacity);
131             comp_value_.reserve(capacity);
132         }
133         to_comp_[x] = ncomps_ + 1;
134         data_.push_back(A(input_, x, env_));
135         parent_.push_back(ncomps_ + 1);
136         comp_value_.push_back(ncomps_ + 1);
137     }
```

7.137.3.2 `template<class T, class DestType = unsigned, class A = oln::morpho::attr::card_type<>, class Exact = mlc::final> void oln::topo::tarjan::flat_zone< T, DestType, A, Exact >::uni_impl (const point_type &n, const point_type &p) [inline, protected]`

Implementation of [uni\(\)](#).

- n A point of the first component.
- p A point of the second component.

Warning:

Do not call this method, use [uni\(\)](#) instead.

Definition at line 148 of file flat-zone.hh.

References `oln::topo::tarjan::flat_zone< T, DestType, A, Exact >::comp_type`, and `oln::topo::tarjan::abstract::tarjan_with_attr< Exact >::find_root()`.

```
149     {
150         comp_type          r = find_root(to_comp_[n]);
151         precondition(to_comp_[n] <= ncomps_);
152         precondition(to_comp_[p] <= (ncomps_ + 1));
153         if (r != to_comp_[p])
154             if (input_[n] == input_[p])
155                 {
156                     if (to_comp_[p] == (ncomps_ + 1)) // first merge of p component
157                         {
158                             precondition(r < comp_value_.capacity());
159                             data_[r] += data_[to_comp_[p]];
160                             precondition(r <= ncomps_);
161                             to_comp_[p] = r;
162                         }
163                     else
164                         {
165                             precondition(r < parent_.capacity());
166                             data_[parent_[to_comp_[p]]] += data_[parent_[r]];
167                             parent_[r] = parent_[to_comp_[p]];
168                         }
169                 }
170     }
171 }
```

The documentation for this struct was generated from the following file:

- flat-zone.hh

7.138 oln::topo::tarjan::obsolete::flat_zone< I > Struct Template Reference

```
#include <flat-zone.hh>
```

Collaboration diagram for oln::topo::tarjan::obsolete::flat_zone< I >:



Public Types

- typedef `mlc::exact< I >::ret::point_type` **point_type**
- typedef `mlc::exact< I >::ret::value_type` **data_type**
- typedef `oln::mute< I >::ret` **image_type**
- typedef `tarjan::tarjan_set< image_type, tarjan::empty_class >` **tarjan_cc**

Public Member Functions

- `flat_zone` (const image_type &input_)
- void `doit` ()
Compute the image of label.
- const unsigned `get_label` (const point_type &p) const
Get the label of a point p.
- const point_type & `get_root` (unsigned l) const
Get the root point of a label l.
- const unsigned `nlabels` () const
Number of label.
- void `merge` (const int l1, const int l2)

Public Attributes

- const image_type & **input**
- [tarjan_cc](#) cc
- [image2d](#)< unsigned > [label](#)
output image.
- std::vector< point_type > **look_up_table**
- [image2d](#)< std::vector< [oln::point2d](#) > > **ima_region**
- unsigned **nlabels_**

7.138.1 Detailed Description

`template<class I> struct oln::topo::tarjan::obsolete::flat_zone< I >`

Create an image of label of the flat zones.

Todo

FIXME: many assertions are missing.

```
#include <oln/basics2d.hh>
#include <oln/topo/tarjan/flat-zone.hh>
#include <oln/convert/stretch.hh>

int main()
{
    typedef oln::image2d<ntg::int_u8> img_type;
    img_type in = oln::load(IMG_IN "test-cmap.pgm");
    oln::topo::tarjan::obsolete::flat_zone<img_type> z(in);
    save(oln::convert::stretch_balance<ntg::int_u8>(z.label, 0, 255),
        IMG_OUT "oln_topo_flat_zone.pgm");
}
```



Figure 7.3: input image

=>



Figure 7.4: output image

Definition at line 219 of file flat-zone.hh.

7.138.2 Constructor & Destructor Documentation

7.138.2.1 `template<class I> oln::topo::tarjan::obsolete::flat_zone< I >::flat_zone (const image_type & input_) [inline]`

Initialize the flat-zone with an image.

doit is called.

Definition at line 241 of file flat-zone.hh.

References `oln::topo::tarjan::obsolete::flat_zone< I >::doit()`, and `oln::topo::tarjan::obsolete::flat_zone< I >::label`.

```

241                                     : input(input_), cc(input_),
242                                     label(input_.size()),
243                                     ima_region(input_.size()),
244                                     nlabels_(0)
245     {
246         doit();
247     }
```

7.138.3 Member Function Documentation

7.138.3.1 `template<class I> void oln::topo::tarjan::obsolete::flat_zone< I >::merge (const int l1, const int l2) [inline]`

Merge two flat zone.

Note:

FIMXE: should be protected, shouldn't it?

Definition at line 361 of file flat-zone.hh.

References `oln::topo::tarjan::tarjan_set< I, aux_data_type >::is_root()`, `oln::topo::tarjan::obsolete::flat_zone< I >::label`, and `oln::topo::tarjan::tarjan_set< I, aux_data_type >::uni()`.


```
362     {
363         point_type root_l1 = look_up_table[l1];
364         point_type root_l2 = look_up_table[l2];
365         assertion(cc.is_root(root_l1));
366         assertion(cc.is_root(root_l2));
367         // merge
368         cc.uni(root_l2, root_l1);
369         // update our tables
370         look_up_table[l2] = root_l1;
371         for (typename std::vector<point_type>::iterator
372              i = ima_region[root_l2].begin();
373              i != ima_region[root_l2].end(); ++i)
374             label[*i] = l1;
375         ima_region[root_l1].insert(ima_region[root_l1].end(),
376                                   ima_region[root_l1].begin(),
377                                   ima_region[root_l1].end());
378         ima_region[root_l2].clear();
379
380         --nlabels_;
381     }
```

The documentation for this struct was generated from the following file:

- flat-zone.hh

7.139 ntg::force Struct Reference

Force the value to be assigned without checks.

```
#include <behavior.hh>
```

Static Public Member Functions

- `std::string name ()`

7.139.1 Detailed Description

Force the value to be assigned without checks.

Quite similar to `unsafe`, but even if the destination type has a strict behavior, by using `cast::force` we ensure that no check will be performed.

Example:

```
int_u<8, strict> a; a = force::get<int_u<8, strict> >::check_plus(5, 6);
```

=> no check

This construction is useful when we want to use code from a particular behavior to a type defined with another behavior.

Definition at line 137 of file `integre/ntg/real/behavior.hh`.

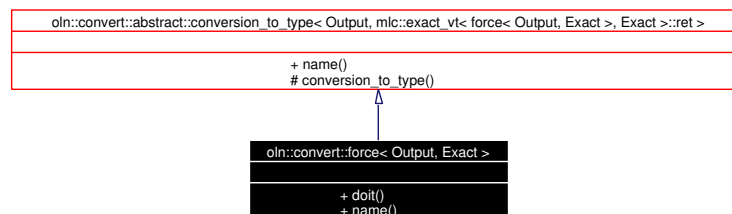
The documentation for this struct was generated from the following file:

- `integre/ntg/real/behavior.hh`

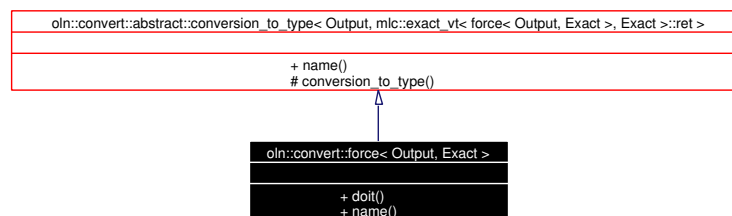
7.140 oln::convert::force< Output, Exact > Struct Template Reference

```
#include <force.hh>
```

Inheritance diagram for oln::convert::force< Output, Exact >:



Collaboration diagram for oln::convert::force< Output, Exact >:



Public Member Functions

- `template<class Input> Output doit (const Input &v) const`

Static Public Member Functions

- `std::string name ()`

7.140.1 Detailed Description

`template<class Output, class Exact = mlc::final> struct oln::convert::force< Output, Exact >`

Like `cast::force`, but as a conversion functor.

Definition at line 40 of file `force.hh`.

The documentation for this struct was generated from the following file:

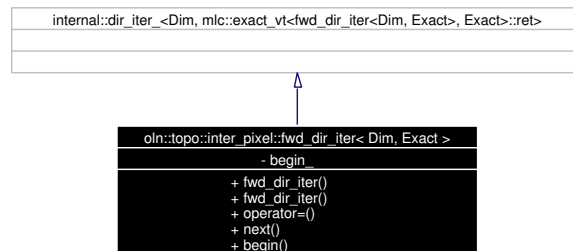
- `force.hh`

7.141 oln::topo::inter_pixel::fwd_dir_iter< Dim, Exact > Class Template Reference

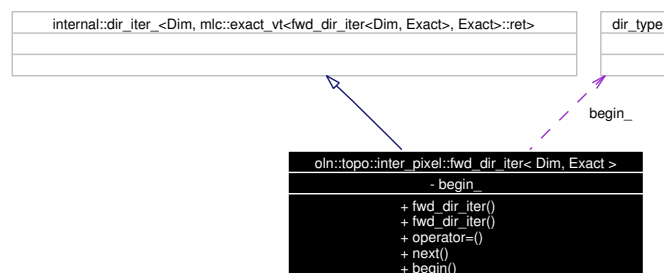
Backward iterator on direction.

```
#include <fwd-dir-iter.hh>
```

Inheritance diagram for oln::topo::inter_pixel::fwd_dir_iter< Dim, Exact >:



Collaboration diagram for oln::topo::inter_pixel::fwd_dir_iter< Dim, Exact >:



Public Member Functions

- **fwd_dir_iter** (dir_type i)
- **template<class U> U operator=** (U u)
- **dir_type next** ()
Next direction.
- **dir_type begin** ()
First direction.

7.141.1 Detailed Description

```
template<unsigned Dim, class Exact> class oln::topo::inter_pixel::fwd_dir_iter< Dim, Exact >
```

Backward iterator on direction.

Definition at line 59 of file fwd-dir-iter.hh.

7.141.2 Member Function Documentation

7.141.2.1 `template<unsigned Dim, class Exact> template<class U> U
oln::topo::inter_pixel::fwd_dir_iter< Dim, Exact >::operator= (U u) [inline]`

Assignment.

Todo

FIXME: I am not sure that this respect the new paradigm.

Definition at line 76 of file fwd-dir-iter.hh.

```
77         {  
78             return super::operator=(u);  
79         }
```

The documentation for this class was generated from the following file:

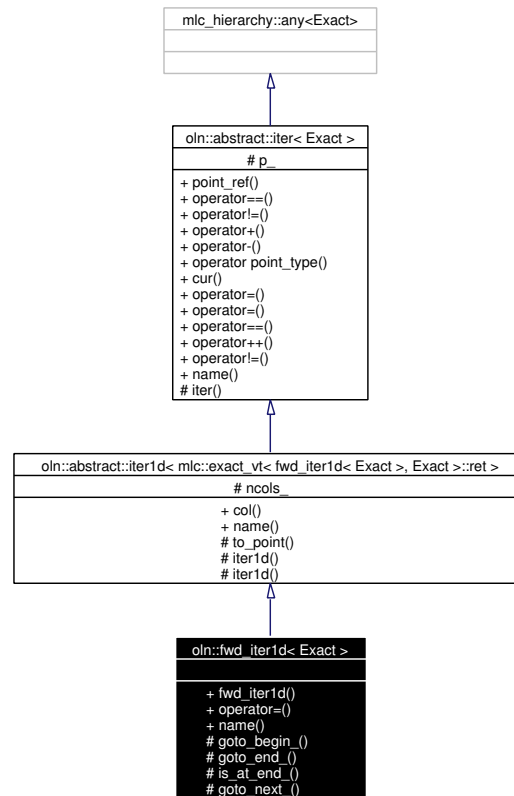
- fwd-dir-iter.hh

7.142 oln::fwd_iter1d< Exact > Class Template Reference

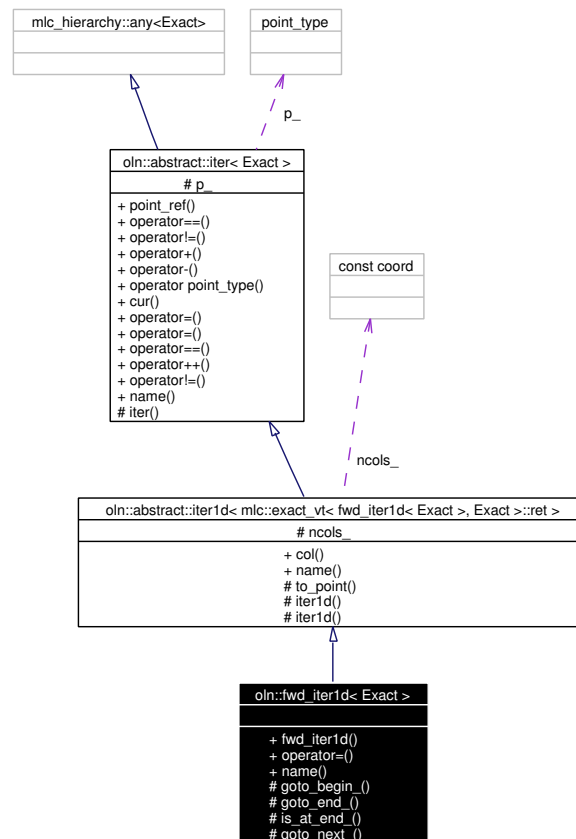
Forward Iterator on image 1 dimension.

```
#include <fwd_iter1d.hh>
```

Inheritance diagram for oln::fwd_iter1d< Exact >:



Collaboration diagram for oln::fwd_iter1d< Exact >:



Public Types

- typedef `mlc::exact_vt< fwd_iter1d< Exact >, Exact >::ret exact_type`
The exact type.
- typedef `abstract::iter1d< exact_type > super_type`
The super type.
- typedef `abstract::iter< exact_type > super_iter_type`
The super iterator type.
- typedef `iter_traits< exact_type >::point_type point_type`
The associate image's type of point.
- enum { **dim** = `iter_traits<exact_type>::dim` }

Public Member Functions

- template<class Image> `fwd_iter1d` (const Image &ima)
Construct a forward iterator (1 dimension).
– ima The image to iterate.

- `template<class U> U operator= (U u)`
Set current iterator's point.

Static Public Member Functions

- `std::string name ()`
Return the name of the type.

Protected Member Functions

- `void goto_begin_ ()`
Set current point to the first iterator's point.
- `void goto_end_ ()`
Set current point to the last iterator's point.
- `bool is_at_end_ () const`
Test if iterator's current point is the last one.
- `void goto_next_ ()`
Go to the next iterator's point.

Friends

- `class abstract::iter< exact_type >`
- `class abstract::iter1d< exact_type >`

7.142.1 Detailed Description

`template<class Exact> class oln::fwd_iter1d< Exact >`

Forward Iterator on image 1 dimension.

Allow iterable object (like image, window, ...) of 1 dimension forward traversing.

See also:

[iter](#)

Definition at line 55 of file `fwd_iter1d.hh`.

7.142.2 Member Typedef Documentation

7.142.2.1 `template<class Exact> typedef iter_traits<exact_type>::point_type oln::fwd_iter1d< Exact >::point_type`

The associate image's type of point.

Warning:

Prefer the macros `oln_point_type(Pointable)` and `oln_point_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented from [oln::abstract::iter< Exact >](#).

Definition at line 74 of file `fwd_iter1d.hh`.

7.142.3 Member Function Documentation**7.142.3.1** `template<class Exact> void oln::fwd_iter1d< Exact >::goto_begin_() [inline, protected]`

Set current point to the first iterator's point.

Set current point of iterator to the first iterator's point.

Definition at line 115 of file `fwd_iter1d.hh`.

```
116     {
117         this->p_.col() = 0;
118     }
```

7.142.3.2 `template<class Exact> void oln::fwd_iter1d< Exact >::goto_end_() [inline, protected]`

Set current point to the last iterator's point.

Set current point of iterator to the last iterator's point.

Definition at line 126 of file `fwd_iter1d.hh`.

```
127     {
128         this->p_.col() = this->ncols_;
129     }
```

7.142.3.3 `template<class Exact> bool oln::fwd_iter1d< Exact >::is_at_end_() const [inline, protected]`

Test if iterator's current point is the last one.

Returns:

True if current point is the last one.

Definition at line 136 of file `fwd_iter1d.hh`.

```
137     {
138         return this->p_.col() == this->ncols_;
139     }
```

7.142.3.4 `template<class Exact> template<class U> U oln::fwd_iter1d< Exact >::operator= (U u) [inline]`

Set current iterator's point.

Set current point of iterator to the first iterator's point.

Definition at line 95 of file fwd_iter1d.hh.

```
96     {  
97         return super_iter_type::operator=(u);  
98     }
```

The documentation for this class was generated from the following file:

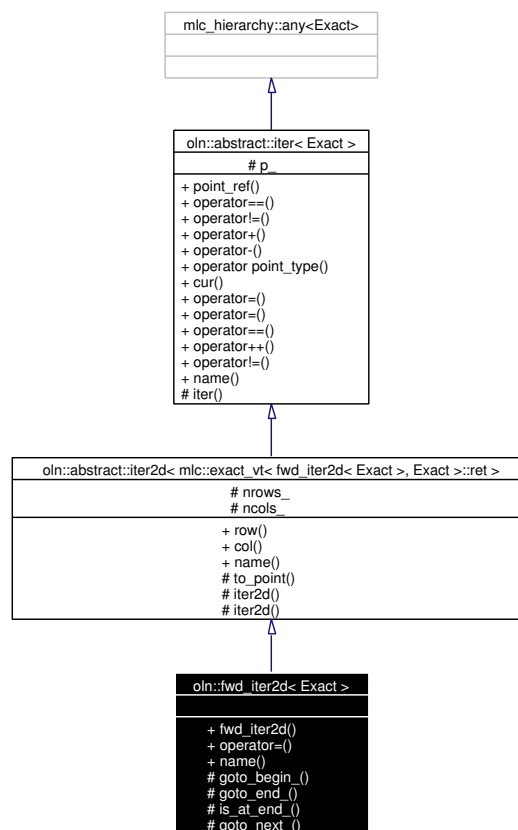
- fwd_iter1d.hh

7.143 oln::fwd_iter2d< Exact > Class Template Reference

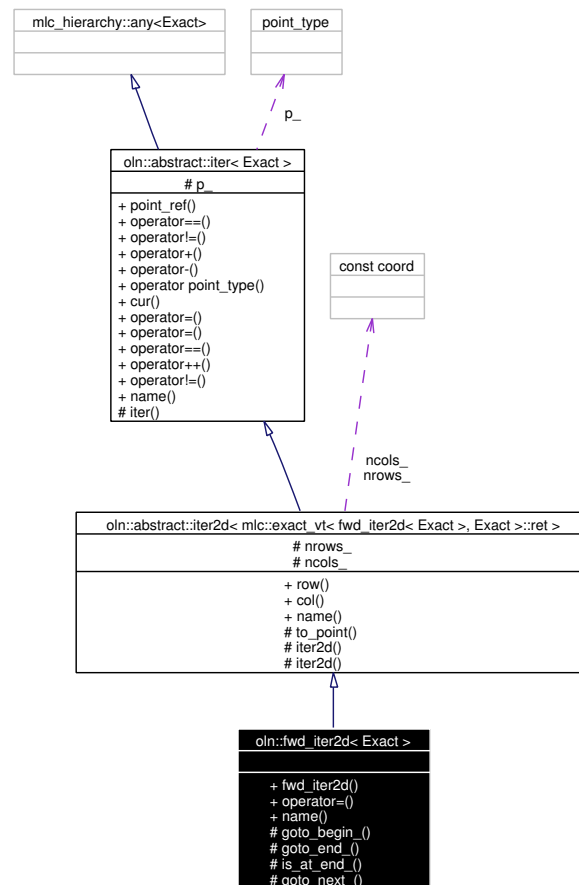
Backward Iterator on image 2 dimension.

```
#include <fwd_iter2d.hh>
```

Inheritance diagram for oln::fwd_iter2d< Exact >:



Collaboration diagram for oln::fwd_iter2d< Exact >:



Public Types

- typedef `mlc::exact_vt< fwd_iter2d< Exact >, Exact >::ret exact_type`
The exact type.
- typedef `abstract::iter2d< exact_type > super_type`
The super type.
- typedef `abstract::iter< exact_type > super_iter_type`
The super iterator type.
- typedef `iter_traits< exact_type >::point_type point_type`
The associate image's type of point.
- enum { **dim** = `iter_traits<exact_type>::dim` }

Public Member Functions

- template<class Image> `fwd_iter2d` (const Image &ima)
Construct a forward iterator (2 dimension).
– ima The image to iterate.

- `template<class U> U operator= (U u)`

Set current iterator's point.

– *u New current point.*

Static Public Member Functions

- `std::string name ()`

Return the name of the type.

Protected Member Functions

- `void goto_begin_ ()`

Set current point to the first iterator's point.

- `void goto_end_ ()`

Set current point to the last iterator's point.

- `bool is_at_end_ () const`

Test if iterator's current point is the last one.

- `void goto_next_ ()`

Go to the next iterator's point.

Friends

- `class abstract::iter< exact_type >`
- `class abstract::iter2d< exact_type >`

7.143.1 Detailed Description

`template<class Exact> class oln::fwd_iter2d< Exact >`

Backward Iterator on image 2 dimension.

Allow iterable object (like image, window, ...) of 2 dimensions forward traversing.

See also:

[iter](#)

Definition at line 58 of file fwd_iter2d.hh.

7.143.2 Member Typedef Documentation

7.143.2.1 `template<class Exact> typedef iter_traits<exact_type>::point_type oln::fwd_iter2d<Exact>::point_type`

The associate image's type of point.

Warning:

Prefer the macros `oln_point_type(Pointable)` and `oln_point_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented from `oln::abstract::iter< Exact >`.

Definition at line 77 of file `fwd_iter2d.hh`.

7.143.3 Member Function Documentation

7.143.3.1 `template<class Exact> void oln::fwd_iter2d< Exact >::goto_begin_() [inline, protected]`

Set current point to the first iterator's point.

Set current point of iterator to the first iterator's point.

Definition at line 117 of file `fwd_iter2d.hh`.

```

118     {
119         this->p_.row() = this->p_.col() = 0;
120     }
```

7.143.3.2 `template<class Exact> void oln::fwd_iter2d< Exact >::goto_end_() [inline, protected]`

Set current point to the last iterator's point.

Set current point of iterator to the last iterator's point.

Definition at line 128 of file `fwd_iter2d.hh`.

```

129     {
130         this->p_.row() = this->nrows_;
131     }
```

7.143.3.3 `template<class Exact> bool oln::fwd_iter2d< Exact >::is_at_end_() const [inline, protected]`

Test if iterator's current point is the last one.

Returns:

True if current point is the last one.

Definition at line 138 of file `fwd_iter2d.hh`.

```
139     {  
140         return this->p_.row() == this->nrows_;  
141     }
```

The documentation for this class was generated from the following file:

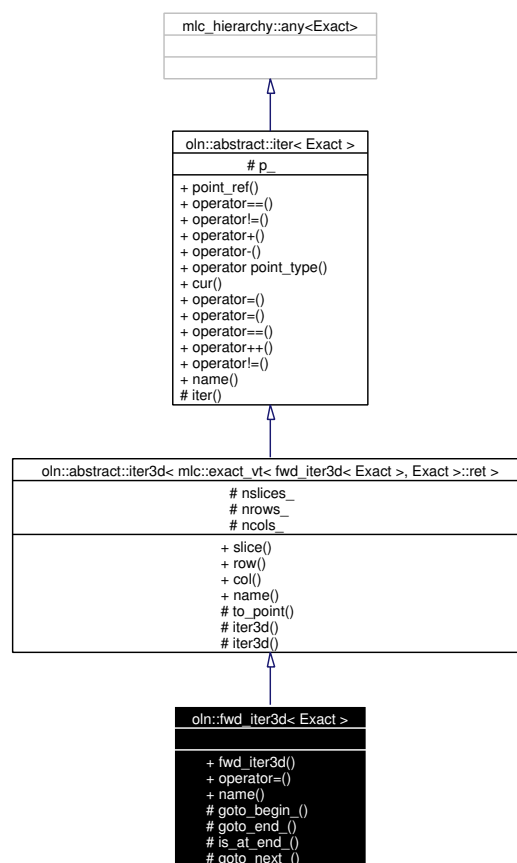
- fwd_iter2d.hh

7.144 oln::fwd_iter3d< Exact > Class Template Reference

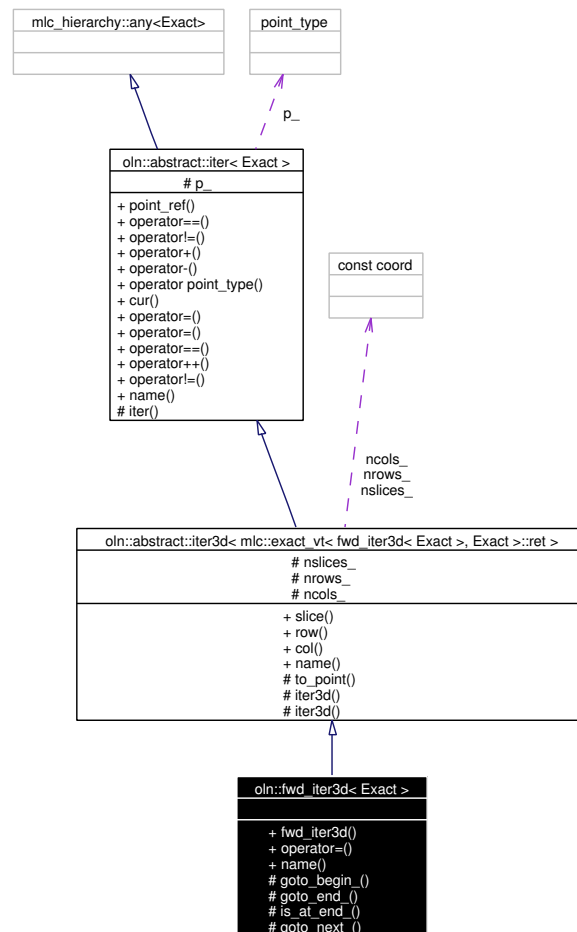
Backward Iterator on image 3 dimension.

```
#include <fwd_iter3d.hh>
```

Inheritance diagram for oln::fwd_iter3d< Exact >:



Collaboration diagram for oln::fwd_iter3d< Exact >:



Public Types

- typedef `mlc::exact_vt< fwd_iter3d< Exact >, Exact >::ret exact_type`
The exact type.
- typedef `abstract::iter3d< exact_type > super_type`
The super type.
- typedef `abstract::iter< exact_type > super_iter_type`
The super iterator type.
- typedef `iter_traits< exact_type >::point_type point_type`
The associate image's type of point.
- enum { **dim** = `iter_traits<exact_type>::dim` }

Public Member Functions

- template<class Image> `fwd_iter3d` (const Image &ima)

Construct a forward iterator (3 dimension).

- *ima* The image to iterate.

- `template<class U> U operator= (U u)`

Set current iterator's point.

- *u* New current point.

Static Public Member Functions

- `std::string name ()`

Return the name of the type.

Protected Member Functions

- `void goto_begin_ ()`

Set current point to the first iterator's point.

- `void goto_end_ ()`

Set current point to the last iterator's point.

- `bool is_at_end_ () const`

Test if iterator's current point is the last one.

- `void goto_next_ ()`

Go to the next iterator's point.

Friends

- `class abstract::iter< exact_type >`
- `class abstract::iter3d< exact_type >`

7.144.1 Detailed Description

`template<class Exact> class oln::fwd_iter3d< Exact >`

Backward Iterator on image 3 dimension.

Allow iterable object (like image, window, ...) of 3 dimensions forward traversing.

See also:

[iter](#)

Definition at line 55 of file `fwd_iter3d.hh`.

7.144.2 Member Typedef Documentation

7.144.2.1 `template<class Exact> typedef iter_traits<exact_type>::point_type oln::fwd_iter3d<Exact>::point_type`

The associate image's type of point.

Warning:

Prefer the macros `oln_point_type(Pointable)` and `oln_point_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented from `oln::abstract::iter< Exact >`.

Definition at line 74 of file `fwd_iter3d.hh`.

7.144.3 Member Function Documentation

7.144.3.1 `template<class Exact> void oln::fwd_iter3d< Exact >::goto_begin_ () [inline, protected]`

Set current point to the first iterator's point.

Set current point of iterator to the first iterator's point.

Definition at line 114 of file `fwd_iter3d.hh`.

```

115     {
116         this->p_.slice() = this->p_.row() = this->p_.col() = 0;
117     }
```

7.144.3.2 `template<class Exact> void oln::fwd_iter3d< Exact >::goto_end_ () [inline, protected]`

Set current point to the last iterator's point.

Set current point of iterator to the last iterator's point.

Definition at line 125 of file `fwd_iter3d.hh`.

```

126     {
127         this->p_.slice() = this->nslices_;
128     }
```

7.144.3.3 `template<class Exact> bool oln::fwd_iter3d< Exact >::is_at_end_ () const [inline, protected]`

Test if iterator's current point is the last one.

Returns:

True if current point is the last one.

Definition at line 135 of file `fwd_iter3d.hh`.

```
136     {  
137         return this->p_.slice() == this->nslices_  
138     }
```

The documentation for this class was generated from the following file:

- fwd_iter3d.hh

7.145 oln::convol::fast::internal::gaussian_< dim > Struct Template Reference

Compute the gaussian filter.

```
#include <fast_gaussian.hxx>
```

7.145.1 Detailed Description

template<unsigned dim> struct oln::convol::fast::internal::gaussian_< dim >

Compute the gaussian filter.

Look at specialization for details.

Definition at line 169 of file fast_gaussian.hxx.

The documentation for this struct was generated from the following file:

- fast_gaussian.hxx

7.146 `oln::convol::fast::internal::gaussian_< 1 >` Struct Template Reference

```
#include <fast_gaussian.hxx>
```

Static Public Member Functions

- `template<class I, class F> void doit (abstract::image_with_dim< 1, I > &img, const F &coef)`

7.146.1 Detailed Description

`template<> struct oln::convol::fast::internal::gaussian_< 1 >`

Gaussian filter for 1D images

[gaussian_](#) specialization for 1D images.

Parameters:

I Exact type of the image.

F Type of coefficients.

- `img` Image to process.
- `coef` Coefficients to use.

Definition at line 183 of file `fast_gaussian.hxx`.

The documentation for this struct was generated from the following file:

- `fast_gaussian.hxx`

7.147 oln::convol::fast::internal::gaussian_< 2 > Struct Template Reference

```
#include <fast_gaussian.hxx>
```

Static Public Member Functions

- `template<class I, class F> void doit (abstract::image_with_dim< 2, I > &img, const F &coef)`

7.147.1 Detailed Description

`template<> struct oln::convol::fast::internal::gaussian_< 2 >`

Gaussian filter for 2D images

[gaussian_](#) specialization for 2D images.

Parameters:

I Exact type of the image.

F Type of coefficients.

- `img` Image to process.
- `coef` Coefficients to use.

Definition at line 211 of file `fast_gaussian.hxx`.

The documentation for this struct was generated from the following file:

- `fast_gaussian.hxx`

7.148 `oln::convol::fast::internal::gaussian_< 3 >` Struct Template Reference

```
#include <fast_gaussian.hxx>
```

Static Public Member Functions

- `template<class I, class F> void doit (abstract::image_with_dim< 3, I > &img, const F &coef)`

7.148.1 Detailed Description

`template<> struct oln::convol::fast::internal::gaussian_< 3 >`

Gaussian filter for 3D images

[gaussian_](#) specialization for 3D images.

Parameters:

I Exact type of the image.

F Type of coefficients.

- `img` Image to process.
- `coef` Coefficients to use.

Definition at line 246 of file `fast_gaussian.hxx`.

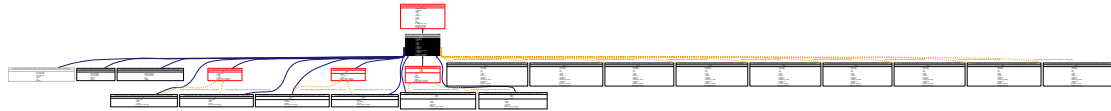
The documentation for this struct was generated from the following file:

- `fast_gaussian.hxx`

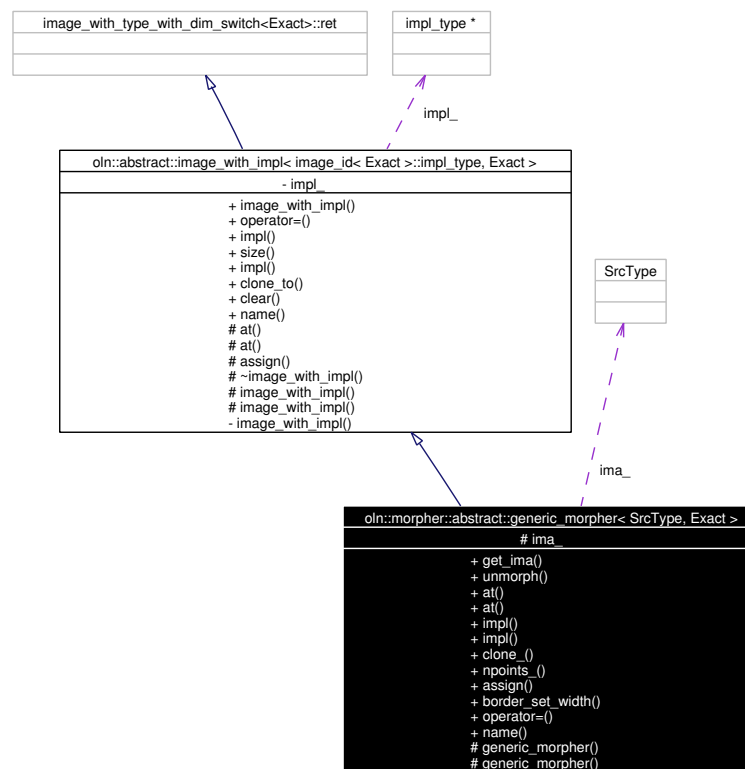
7.149 oln::morpher::abstract::generic_morpher< SrcType, Exact > Class Template Reference

```
#include <generic_morpher.hh>
```

Inheritance diagram for oln::morpher::abstract::generic_morpher< SrcType, Exact >:



Collaboration diagram for oln::morpher::abstract::generic_morpher< SrcType, Exact >:



Public Types

- typedef `generic_morpher< SrcType, Exact > self_type`
The self type.
- typedef `Exact exact_type`
The exact type of the morpher.
- typedef `image_traits< exact_type >::point_type point_type`
The morpher point type.

- typedef [image_traits](#)< [exact_type](#) >::dpoint_type [dpoint_type](#)
The morpher dpoint type.
- typedef [image_traits](#)< [exact_type](#) >::iter_type [iter_type](#)
The morpher iterator type.
- typedef [image_traits](#)< [exact_type](#) >::fwd_iter_type [fwd_iter_type](#)
The morpher forward iterator type.
- typedef [image_traits](#)< [exact_type](#) >::bkd_iter_type [bkd_iter_type](#)
The morpher backward iterator type.
- typedef [image_traits](#)< [exact_type](#) >::value_type [value_type](#)
The morpher value type.
- typedef [image_traits](#)< [exact_type](#) >::size_type [size_type](#)
The morpher size type.
- typedef [image_traits](#)< [exact_type](#) >::impl_type [impl_type](#)
The morpher underlying implementation.
- typedef dest_type< [image_traits](#)< [exact_type](#) >::dim, [value_type](#) >::ret **DestType**
- typedef SrcType [src_self_type](#)
Type of the decorated image.
- typedef mlc::exact< SrcType >::ret::point_type [src_point_type](#)
The decorated image point type.
- typedef mlc::exact< SrcType >::ret::dpoint_type [src_dpoint_type](#)
The decorated image dpoint type.
- typedef mlc::exact< SrcType >::ret::iter_type [src_iter_type](#)
The decorated image iterator type.
- typedef mlc::exact< SrcType >::ret::fwd_iter_type [src_fwd_iter_type](#)
The decorated image forward iterator type.
- typedef mlc::exact< SrcType >::ret::bkd_iter_type [src_bkd_iter_type](#)
The decorated image backward iterator type.
- typedef mlc::exact< SrcType >::ret::value_type [src_value_type](#)
The decorated image value type.
- typedef mlc::exact< SrcType >::ret::size_type [src_size_type](#)
The decorated image size type.
- typedef mlc::exact< SrcType >::ret::impl_type [src_impl_type](#)
The decorated image underlying implementation.

- `typedef mlc::exact< SrcType >::ret src_exact_type`
Exact type of the decorated image.
- `typedef oln::abstract::image_with_impl< impl_type, exact_type > super_type`
The upper class.

Public Member Functions

- `const SrcType & get_ima () const`
Return the decorated image.
- `DestType unmorph () const`
Instantiate and return the image that the morpher is simulating.
- `const src_value_type at (const src_point_type &p) const`
- `src_value_type & at (const src_point_type &p)`
- `const src_impl_type * impl () const`
Default implementation of impl.
- `src_impl_type * impl ()`
Default implementation of impl.
- `src_exact_type clone_ () const`
- `size_t npoints_ ()`
- `src_exact_type & assign (src_exact_type &rhs)`
- `void border_set_width (oln::coord min_border, bool copy_border=false) const`
- `self_type & operator= (self_type &rhs)`

Static Public Member Functions

- `std::string name ()`

Protected Member Functions

- `generic_morpher (const SrcType &Ima)`
Construct an instance of `generic_morpher` by assigning Ima to Ima_.
- `generic_morpher ()`
Empty Constructor.

Protected Attributes

- `const SrcType & ima_`
The decorated image.

7.149.1 Detailed Description

template<class SrcType, class Exact> class **oln::morpher::abstract::generic_morpher**< SrcType, Exact >

The Abstract morpher class.

Define a default implementation for all the dispatched methods of the image hierarchy.

Parameters:

DestType Output type of the morpher.

SrcType Input type decorated.

Exact Exact type

Definition at line 107 of file generic_morpher.hh.

7.149.2 Member Function Documentation

7.149.2.1 **template**<class SrcType, class Exact> **src_exact_type&**
oln::morpher::abstract::generic_morpher< SrcType, Exact >::assign (**src_exact_type** &
rhs) [inline]

Default implementation of assign.

Assign *rhs* to the decorated image.

Definition at line 263 of file generic_morpher.hh.

```
264         {
265             return to_exact(ima_).assign(rhs);
266         }
```

7.149.2.2 **template**<class SrcType, class Exact> **src_value_type&**
oln::morpher::abstract::generic_morpher< SrcType, Exact >::at (const **src_point_type**
&*p*) [inline]

Default implementation of at.

Return a reference to the value stored at *p* in the decorated image.

Warning:

This method should not be called directly. Prefer operator[].

Definition at line 219 of file generic_morpher.hh.

```
220         {
221             return ima_[p];
222         }
```

7.149.2.3 `template<class SrcType, class Exact> const src_value_type
 oln::morpher::abstract::generic_morpher< SrcType, Exact >::at (const src_point_type
 & p) const [inline]`

Default implementation of at.

Return the value stored at *p* in the decorated image.

Warning:

This method should not be called directly. Prefer operator[].

Definition at line 206 of file generic_morpher.hh.

```
207     {
208         return ima_[p];
209     }
```

7.149.2.4 `template<class SrcType, class Exact> void oln::morpher::abstract::generic_morpher<
 SrcType, Exact >::border_set_width (oln::coord min_border, bool copy_border = false)
 const [inline]`

Default implementation of border_set_width.

Set the border width of the decorated image to *min_border*.

Definition at line 274 of file generic_morpher.hh.

```
276     {
277         return to_exact(ima_).border_set_width(min_border, copy_border);
278     }
```

7.149.2.5 `template<class SrcType, class Exact> src_exact_type oln::morpher::abstract::generic_
 morpher< SrcType, Exact >::clone_ () const [inline]`

Default implementation of clone_.

Return a copy of the decorated image.

Definition at line 243 of file generic_morpher.hh.

```
244     {
245         return to_exact(ima_).clone();
246     }
```

7.149.2.6 `template<class SrcType, class Exact> size_t oln::morpher::abstract::generic_morpher<
 SrcType, Exact >::npoints_ () [inline]`

Default implementation of npoints_.

Return the size of the decorated image.

Definition at line 253 of file generic_morpher.hh.

```
254     {
255         return to_exact(ima_).npoints();
256     }
```

7.149.2.7 `template<class SrcType, class Exact> self_type& oln::morpher::abstract::generic_morpher< SrcType, Exact >::operator= (self_type & rhs)` [inline]

Default implementation of the = operator.

Assign the decorated image to 'a r.

Reimplemented in [oln::morpher::slicing_morpher](#)< SrcType, Exact >.

Definition at line 285 of file generic_morpher.hh.

```
286         {  
287             return to_exact(*this).assign(rhs.exact());  
288         }
```

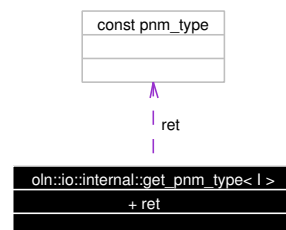
The documentation for this class was generated from the following file:

- generic_morpher.hh

7.150 oln::io::internal::get_pnm_type< I > Struct Template Reference

```
#include <pnm_common.hh>
```

Collaboration diagram for oln::io::internal::get_pnm_type< I >:



Public Types

- typedef mlc::bool_switch< mlc::bool_case< mlc::internal::wrap< typename mlc::internal::is_a< sizeof(mlc::form::get< ntg::unsigned_integer >)) >::check< typename ntg::type_traits< typename mlc::exact< I >::ret::value_type >::abstract_type, ntg::unsigned_integer > >::ret, get_it< PnmInteger >, mlc::bool_case< mlc::internal::wrap< typename mlc::internal::is_a< sizeof(mlc::form::get< ntg::binary >)) >::check< typename ntg::type_traits< typename mlc::exact< I >::ret::value_type >::abstract_type, ntg::binary > >::ret, get_it< PnmBinary >, mlc::bool_case< mlc::internal::wrap< typename mlc::internal::is_a< sizeof(mlc::form::get< ntg::vectorial >)) >::check< typename ntg::type_traits< typename mlc::exact< I >::ret::value_type >::abstract_type, ntg::vectorial > >::ret, get_it< PnmVectorial >, mlc::bool_case< true, get_it< PnmInvalid > > > > >::re tmp_type)

Static Public Attributes

- const pnm_type ret = tmp_type::ret

7.150.1 Detailed Description

template<class I> struct oln::io::internal::get_pnm_type< I >

A metaswitch that return the pnm type associated to an image type.

Todo

FIXME: this could be done by using labels images eg: read(binary_image_with_dim<2>& ima) { // ... }

Definition at line 78 of file pnm_common.hh.

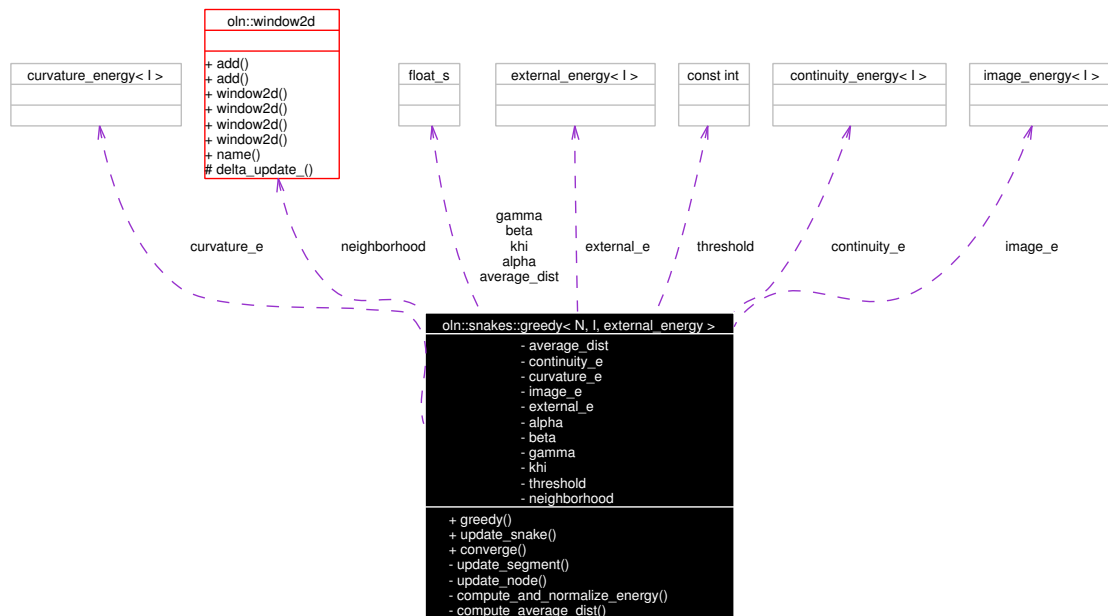
The documentation for this struct was generated from the following file:

- pnm_common.hh

7.151 oln::snakes::greedy< N, I, external_energy > Class Template Reference

```
#include <greedy.hh>
```

Collaboration diagram for oln::snakes::greedy< N, I, external_energy >:



Public Types

- typedef I **image_type**
- typedef I::point_type **point_type**
- typedef mlc::array2d< mlc::array2d_info< N, N >, ntg::float_s > **store_type**

Public Member Functions

- **greedy** (ntg::float_s alpha, ntg::float_s beta, ntg::float_s gamma, ntg::float_s khi)
- int **update_snake** (const I &gradient, **snake**< **greedy** > &s)
Asynchronous update for more efficient convergence.
- void **converge** (const I &gradient, **snake**< **greedy** > &s)

7.151.1 Detailed Description

```
template<int N, class I, template< typename > class external_energy = dummy_energy> class
oln::snakes::greedy< N, I, external_energy >
```

This class can be use as the algorithm of snake.

Parameters:

N is the size of the neighborhood.

Precondition:

N must be odd.

See also:

[snake](#)

Definition at line 47 of file greedy.hh.

7.151.2 Member Function Documentation

7.151.2.1 `template<int N, class I, template< typename > class external_energy> int
oln::snakes::greedy< N, I, external_energy >::update_snake (const I & gradient,
snake< greedy< N, I, external_energy > > & s) [inline]`

Asynchronous update for more efficient convergence.

This place is left void to make room for a future extension where a snake will be able to hold several segments.

Definition at line 53 of file greedy.hxx.

```
54     {  
57         average_dist = compute_average_dist(s.s);  
58         return update_segment(gradient, s.s);  
59     }
```

The documentation for this class was generated from the following files:

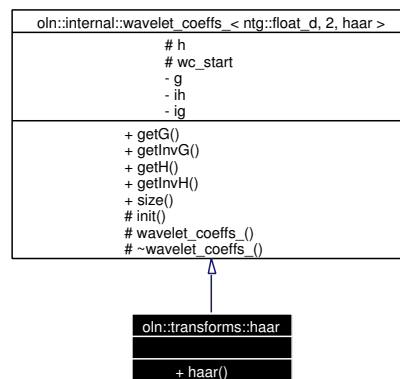
- greedy.hh
- greedy.hxx

7.152 oln::transforms::haar Struct Reference

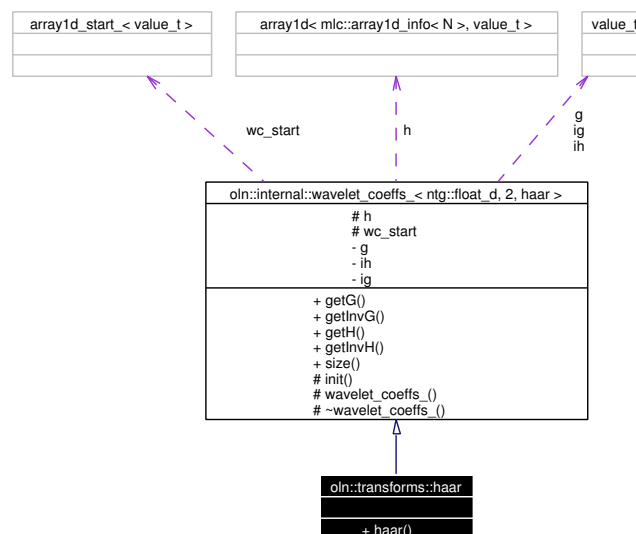
Haar wavelet coefficients.

```
#include <wavelet_coeffs.hh>
```

Inheritance diagram for oln::transforms::haar:



Collaboration diagram for oln::transforms::haar:



7.152.1 Detailed Description

Haar wavelet coefficients.

Definition at line 45 of file `wavelet_coeffs.hh`.

The documentation for this struct was generated from the following file:

- `wavelet_coeffs.hh`

7.153 oln::morpho::attr::height_type< T, Exact > Class Template Reference

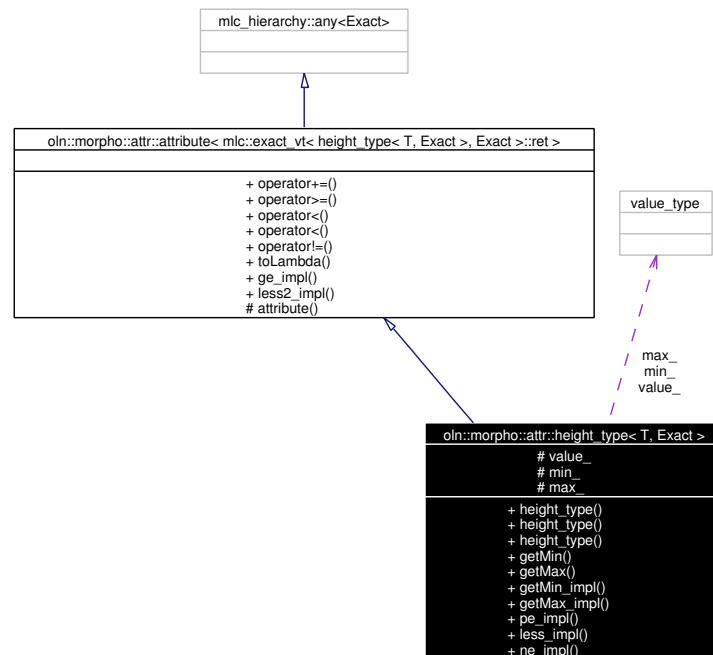
Attribute working on height between components.

```
#include <attributes.hh>
```

Inheritance diagram for oln::morpho::attr::height_type< T, Exact >:



Collaboration diagram for oln::morpho::attr::height_type< T, Exact >:



Public Types

- typedef `height_type< T, Exact >` **self_type**
- typedef `mlc::exact_vt< self_type, Exact >::ret` **exact_type**
- typedef `oln::morpho::attr::attr_traits< exact_type >::value_type` **value_type**
- typedef `oln::morpho::attr::attr_traits< exact_type >::env_type` **env_type**
- typedef `oln::morpho::attr::attr_traits< exact_type >::lambda_type` **lambda_type**

Public Member Functions

- `height_type ()`
Basic Ctor.
- `height_type (const lambda_type &lambda)`
Ctor from a lambda_type value.
- `template<class I> height_type (const abstract::image< I > &input, const typename mlc::exact< I >::ret::point_type &p, const env_type &)`
Ctor from a point and an image.
- `const value_type &getMin () const`
Accessor to min value.
- `const value_type &getMax () const`
Accessor to max value.
- `const value_type &getMin_impl () const`

Implementation of [getMin\(\)](#).

- const value_type & [getMax_impl](#) () const

Implementation of [getMax\(\)](#).

- void [pe_impl](#) (const height_type &rhs)

+= operator implementation.

- bool [less_impl](#) (const lambda_type &lambda) const

"<" operator implementation.

- bool [ne_impl](#) (const lambda_type &lambda) const

!= operator implementation.

Protected Attributes

- value_type [value_](#)

Current value.

- value_type [min_](#)

Current minimum.

- value_type [max_](#)

Current maximum.

7.153.1 Detailed Description

template<class T = unsigned, class Exact = mlc::final> class oln::morpho::attr::height_type< T, Exact >

Attribute working on height between components.

Definition at line 744 of file attributes.hh.

7.153.2 Constructor & Destructor Documentation

7.153.2.1 **template<class T = unsigned, class Exact = mlc::final> [oln::morpho::attr::height_type](#)< T, Exact >::[height_type](#) ()** `[inline]`

Basic Ctor.

Warning:

After this call, the object is only instantiated (not initialized).

Definition at line 757 of file attributes.hh.

```
758         {
759         };
```

7.153.3 Member Function Documentation

7.153.3.1 `template<class T = unsigned, class Exact = mlc::final> const value_type& oln::morpho::attr::height_type< T, Exact >::getMax () const [inline]`

Accessor to max value.

Virtual method.

See also:

[getMax_impl\(\)](#)

Definition at line 801 of file attributes.hh.

Referenced by `oln::morpho::attr::height_type< T, Exact >::pe_impl()`.

```
802         {
803             mlc_dispatch(getMax)();
804         };
```

7.153.3.2 `template<class T = unsigned, class Exact = mlc::final> const value_type& oln::morpho::attr::height_type< T, Exact >::getMax_impl () const [inline]`

Implementation of [getMax\(\)](#).

Override this method in order to provide a new version of [getMax\(\)](#).

Warning:

Do not call this method, use [getMax\(\)](#) instead.

Definition at line 828 of file attributes.hh.

References `oln::morpho::attr::height_type< T, Exact >::max_`.

```
829         {
830             return max_;
831         };
```

7.153.3.3 `template<class T = unsigned, class Exact = mlc::final> const value_type& oln::morpho::attr::height_type< T, Exact >::getMin () const [inline]`

Accessor to min value.

Virtual method.

See also:

[getMin_impl\(\)](#)

Definition at line 790 of file attributes.hh.

Referenced by `oln::morpho::attr::height_type< T, Exact >::pe_impl()`.

```
791         {
792             mlc_dispatch(getMin)();
793         };
```

7.153.3.4 `template<class T = unsigned, class Exact = mlc::final> const value_type&
 oln::morpho::attr::height_type< T, Exact >::getMin_impl () const [inline]`

Implementation of `getMin()`.

Override this method in order to provide a new version of `getMin()`.

Warning:

Do not call this method, use `getMin()` instead.

Definition at line 815 of file attributes.hh.

References `oln::morpho::attr::height_type< T, Exact >::min_`.

```
816      {
817          return min_;
818      };
```

7.153.3.5 `template<class T = unsigned, class Exact = mlc::final> bool
 oln::morpho::attr::height_type< T, Exact >::less_impl (const lambda_type & lambda)
 const [inline]`

"<" operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 854 of file attributes.hh.

```
855      {
856          return value_ < lambda;
857      };
```

7.153.3.6 `template<class T = unsigned, class Exact = mlc::final> bool
 oln::morpho::attr::height_type< T, Exact >::ne_impl (const lambda_type & lambda)
 const [inline]`

!= operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 866 of file attributes.hh.

```
867      {
868          return lambda != value_;
869      };
```

7.153.3.7 `template<class T = unsigned, class Exact = mlc::final> void
oln::morpho::attr::height_type< T, Exact >::pe_impl (const height_type< T, Exact > &
rhs) [inline]`

+= operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 840 of file attributes.hh.

References `oln::morpho::attr::height_type< T, Exact >::getMax()`, `oln::morpho::attr::height_type< T, Exact >::getMin()`, `oln::morpho::attr::height_type< T, Exact >::max_`, and `oln::morpho::attr::height_type< T, Exact >::min_`.

```
841      {  
842          min_ = ntg::min(min_, rhs.getMin());  
843          max_ = ntg::max(max_, rhs.getMax());  
844          value_ = max_ - min_;  
845      };
```

The documentation for this class was generated from the following file:

- attributes.hh

7.154 oln::utils::hist_traits< Exact > Struct Template Reference

Traits for [oln::utils::abstract::histogram](#) hierarchy.

```
#include <histogram.hh>
```

7.154.1 Detailed Description

template<class Exact> struct oln::utils::hist_traits< Exact >

Traits for [oln::utils::abstract::histogram](#) hierarchy.

Definition at line 97 of file histogram.hh.

The documentation for this struct was generated from the following file:

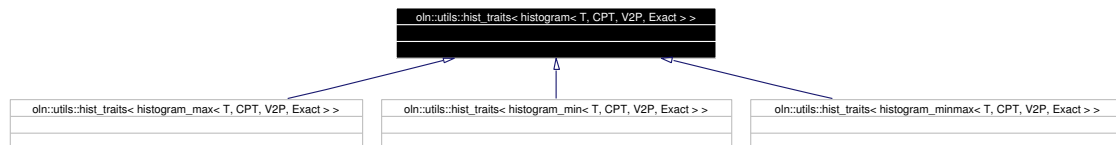
- histogram.hh

7.155 oln::utils::hist_traits< histogram< T, CPT, V2P, Exact > > Struct Template Reference

Traits for [oln::utils::abstract::histogram](#) hierarchy.

```
#include <histogram.hh>
```

Inheritance diagram for `oln::utils::hist_traits< histogram< T, CPT, V2P, Exact > >`:



Public Types

- typedef CPT [cpt_type](#)
Counter type.
- typedef T [value_type](#)
Value of the input image.
- typedef V2P [value_to_point_type](#)
- typedef `value_to_point_type::result_type` [point_type](#)
Point type of the internal image.
- typedef [dim_traits< dim, CPT >::img_type](#) [img_type](#)
Type of the internal image.
- enum { **dim** = `point_type::dim` }

7.155.1 Detailed Description

```
template<typename T, typename CPT, class V2P, class Exact> struct oln::utils::hist_traits<
histogram< T, CPT, V2P, Exact > >
```

Traits for [oln::utils::abstract::histogram](#) hierarchy.

Definition at line 160 of file `histogram.hh`.

7.155.2 Member Typedef Documentation

7.155.2.1 `template<typename T, typename CPT, class V2P, class Exact> typedef V2P`
[oln::utils::hist_traits< histogram< T, CPT, V2P, Exact > >::value_to_point_type](#)

Class that convert a value to a point

Definition at line 164 of file `histogram.hh`.

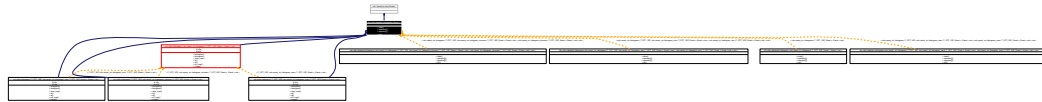
The documentation for this struct was generated from the following file:

- histogram.hh

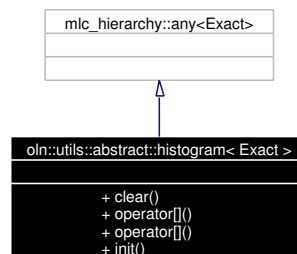
7.156 oln::utils::abstract::histogram< Exact > Class Template Reference

```
#include <histogram.hh>
```

Inheritance diagram for oln::utils::abstract::histogram< Exact >:



Collaboration diagram for oln::utils::abstract::histogram< Exact >:



Public Types

- typedef mlc::exact_vt< [histogram](#)< Exact >, Exact >::ret **exact_type**
- typedef [hist_traits](#)< exact_type >::cpt_type **cpt_type**
- typedef [hist_traits](#)< exact_type >::value_type **value_type**

Public Member Functions

- void [clear](#) ()
Put the number of occurrence of every value to zero.
- const [cpt_type](#) [operator\[\]](#) (const value_type &v) const
Read the number of occurrence of v.
- [cpt_type](#) & [operator\[\]](#) (const value_type &v)
Read or write the number of occurrence of v.
- template<class I> void [init](#) (const [oln::abstract::image](#)< I > &img)

7.156.1 Detailed Description

```
template<class Exact = mlc::final> class oln::utils::abstract::histogram< Exact >
```

Abstract base class for histogram.

See also:

oln::histogram

Definition at line 110 of file histogram.hh.

7.156.2 Member Function Documentation

7.156.2.1 `template<class Exact = mlc::final> template<class I> void
oln::utils::abstract::histogram< Exact >::init (const oln::abstract::image< I > &img)
[inline]`

Build the histogram of an image.

Attention:

The histogram is not cleared.

Definition at line 142 of file histogram.hh.

Referenced by oln::utils::histogram< T, CPT, V2P, mlc::exact_vt< histogram_minmax< T, CPT, V2P, Exact >, Exact >::ret >::histogram().

```
143         {  
144             return this->exact().init_impl(img.exact());  
145         }
```

The documentation for this class was generated from the following file:

- histogram.hh

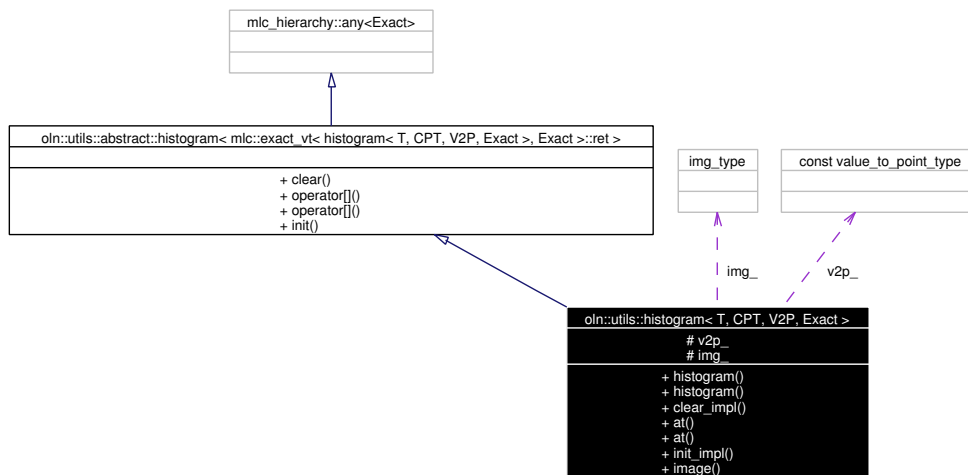
7.157 oln::utils::histogram< T, CPT, V2P, Exact > Class Template Reference

```
#include <histogram.hh>
```

Inheritance diagram for oln::utils::histogram< T, CPT, V2P, Exact >:



Collaboration diagram for oln::utils::histogram< T, CPT, V2P, Exact >:



Public Types

- typedef mlc::exact_vt< [histogram](#)< T, CPT, V2P, Exact >, Exact >::ret **exact_type**
- typedef [hist_traits](#)< exact_type >::value_type **value_type**
- typedef [hist_traits](#)< exact_type >::cpt_type **cpt_type**
- typedef [hist_traits](#)< exact_type >::value_to_point_type **value_to_point_type**
- typedef [hist_traits](#)< exact_type >::point_type **point_type**
- typedef [hist_traits](#)< exact_type >::img_type **img_type**
- enum { **dim** = hist_traits<exact_type>::dim }

Public Member Functions

- [histogram](#) (const value_to_point_type &c2p=value_to_point_type())
Empty histogram.
- template<class I> [histogram](#) (const [oln::abstract::image](#)< I > &input, const value_to_point_type &v2p=value_to_point_type())
This compute the histogram of an image.

- void `clear_impl()`
`clear()` should be called.
- const cpt_type `at` (const T &v) const
`operator[]` should be called.
- cpt_type & `at` (const value_type &v)
`operator[]` should be called.
- template<class I> void `init_impl` (const oln::abstract::image< I > &img)
`impl()` should be called.
- const img_type & `image` () const
Return the image of occurrence.

Protected Attributes

- const value_to_point_type `v2p_`
- img_type `img_`

7.157.1 Detailed Description

`template<typename T, typename CPT, class V2P, class Exact> class oln::utils::histogram< T, CPT, V2P, Exact >`

Histogram.

This histogram uses an image of unsigned to store the value. For example the histogram of an image<int__u8> will store the number of occurrences in an `image1d`; an image<rgb_8> will store the number of occurrences an `image3d` (because rgb_8 has 3 components).

Todo

FIXME: An image is inside the histogram. This is incorrect because it is not exactly an image (no border needed).

Parameters:

- T* Type of the image.
- CPT* Type used to count the occurrences (unsigned).
- V2P* Conversion class to convert a value T to a point.
- Exact* Exact type of the histogram.

See also:

oln::abstract::histogram

```
#include <oln/basics2d.hh>
#include <oln/utils/histogram.hh>
#include <ntg/all.hh>
#include <iostream>
```

```

int main()
{
    oln::image2d<ntg::rgb_8> in = oln::io::load(IMG_IN "lena.ppm");

    oln::utils::histogram<ntg::rgb_8> h(in);

    ntg::rgb_8 pink(215, 129, 113);

    // h[pink] = 14
    std::cout << "Number of occurrences of the color (213, 129, 135): "
                << h[pink] << std::endl;
}

```

Definition at line 214 of file histogram.hh.

7.157.2 Constructor & Destructor Documentation

7.157.2.1 `template<typename T, typename CPT, class V2P, class Exact> oln::utils::histogram<T, CPT, V2P, Exact >::histogram (const value_to_point_type & c2p = value_to_point_type()) [inline]`

Empty histogram.

Note:

The function `init(image)` should be used after this constructor.

Definition at line 233 of file histogram.hh.

```

233                                     :
234         v2p_(c2p), img_(internal::img_max_size<value_type>())
235         {
236             clear();
237         }

```

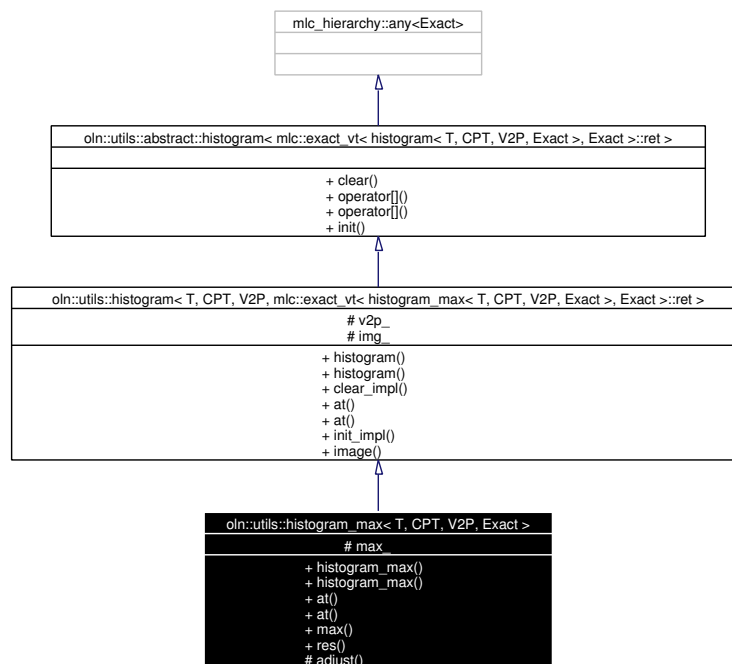
The documentation for this class was generated from the following file:

- histogram.hh

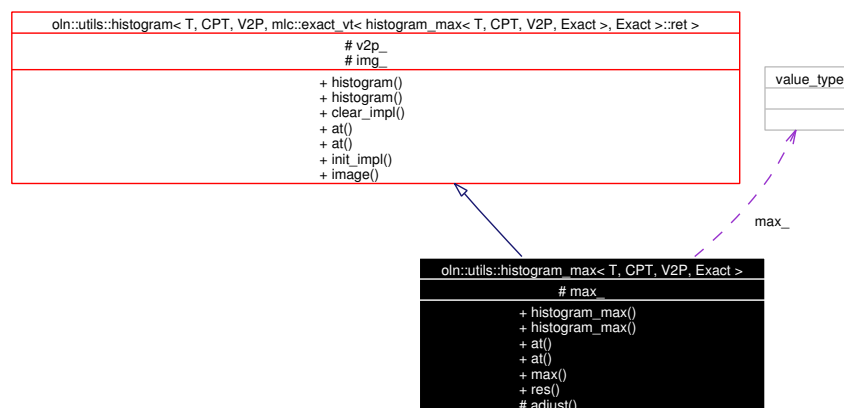
7.158 oln::utils::histogram_max< T, CPT, V2P, Exact > Class Template Reference

```
#include <histogram.hh>
```

Inheritance diagram for oln::utils::histogram_max< T, CPT, V2P, Exact >:



Collaboration diagram for oln::utils::histogram_max< T, CPT, V2P, Exact >:



Public Types

- typedef mlc::exact_vt< [histogram_max](#)< T, CPT, V2P, Exact >, Exact >::ret **exact_type**
- typedef [hist_traits](#)< exact_type >::value_type **value_type**

- typedef [hist_traits](#)< exact_type >::cpt_type **cpt_type**
- typedef [hist_traits](#)< exact_type >::value_to_point_type **value_to_point_type**
- typedef [hist_traits](#)< exact_type >::point_type **point_type**
- typedef [hist_traits](#)< exact_type >::img_type **img_type**
- typedef [histogram](#)< T, CPT, V2P, exact_type > **upper_type**
- enum { **dim** = [hist_traits](#)<exact_type>::dim }

Public Member Functions

- **histogram_max** (const value_to_point_type &v2p=value_to_point_type())
- template<class I> **histogram_max** (const [oln::abstract::image](#)< I > &input, const value_to_point_type &v2p=value_to_point_type())
- const cpt_type **at** (const value_type &i) const
operator[] should be called.
- cpt_type & **at** (const value_type &i)
operator[] should be called.
- value_type **max** ()
Quick function max.
- value_type **res** ()
Return the max.

Protected Member Functions

- void **adjust** (const value_type &idx)
Maintain the worst max.

Protected Attributes

- value_type **max_**
Index of the worst max element.

7.158.1 Detailed Description

template<typename T, typename CPT, class V2P, class Exact> class [oln::utils::histogram_max](#)< T, CPT, V2P, Exact >

Build the histogram and has quick max function.

See also:

[histogram_minmax](#)

Definition at line 568 of file histogram.hh.

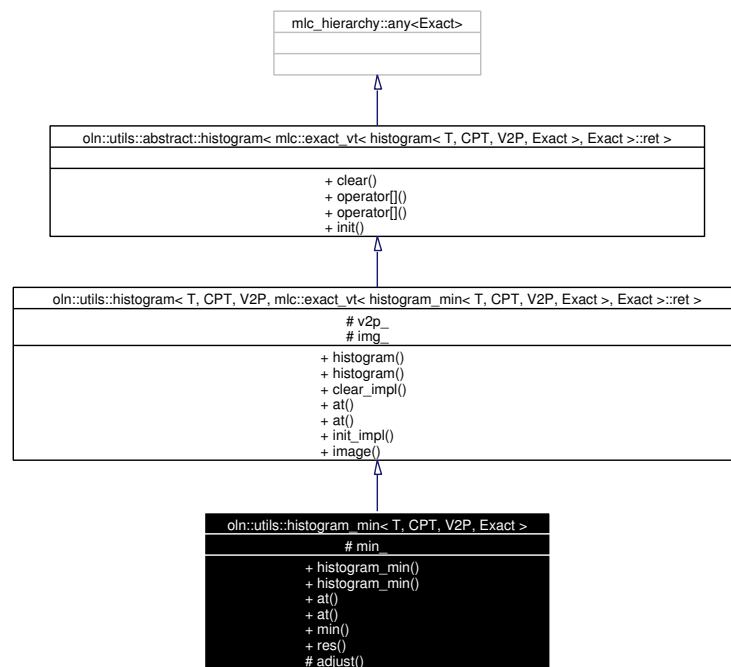
The documentation for this class was generated from the following file:

- histogram.hh

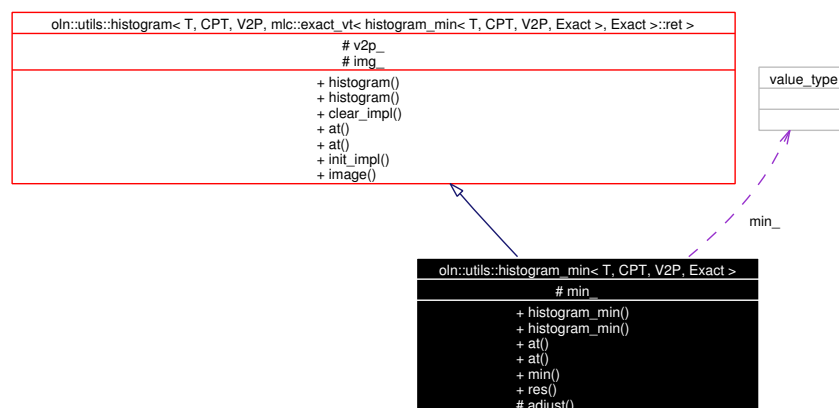
7.159 oln::utils::histogram_min< T, CPT, V2P, Exact > Class Template Reference

```
#include <histogram.hh>
```

Inheritance diagram for oln::utils::histogram_min< T, CPT, V2P, Exact >:



Collaboration diagram for oln::utils::histogram_min< T, CPT, V2P, Exact >:



Public Types

- typedef mlc::exact_vt< [histogram_min](#)< T, CPT, V2P, Exact >, Exact >::ret **exact_type**
- typedef [hist_traits](#)< exact_type >::value_type **value_type**

- typedef [hist_traits](#)< exact_type >::cpt_type **cpt_type**
- typedef [hist_traits](#)< exact_type >::value_to_point_type **value_to_point_type**
- typedef [hist_traits](#)< exact_type >::point_type **point_type**
- typedef [hist_traits](#)< exact_type >::img_type **img_type**
- typedef [histogram](#)< T, CPT, V2P, exact_type > **upper_type**
- enum { **dim** = hist_traits<exact_type>::dim }

Public Member Functions

- **histogram_min** (const value_to_point_type &v2p=value_to_point_type())
- template<class I> **histogram_min** (const [oln::abstract::image](#)< I > &input, const value_to_point_type &v2p=value_to_point_type())
- const cpt_type **at** (const value_type &i) const
operator[] should be called.
- cpt_type & **at** (const value_type &i)
operator[] should be called.
- value_type **min** ()
Quick function min.
- value_type **res** ()
Return the min.

Protected Member Functions

- void **adjust** (const value_type &idx)
Maintain the worst min.

Protected Attributes

- value_type **min_**
Index of the worst min element.

7.159.1 Detailed Description

template<typename T, typename CPT, class V2P, class Exact> class oln::utils::histogram_min< T, CPT, V2P, Exact >

Build the histogram and has quick min function.

See also:

[histogram_minmax](#)

Definition at line 477 of file histogram.hh.

The documentation for this class was generated from the following file:

- histogram.hh

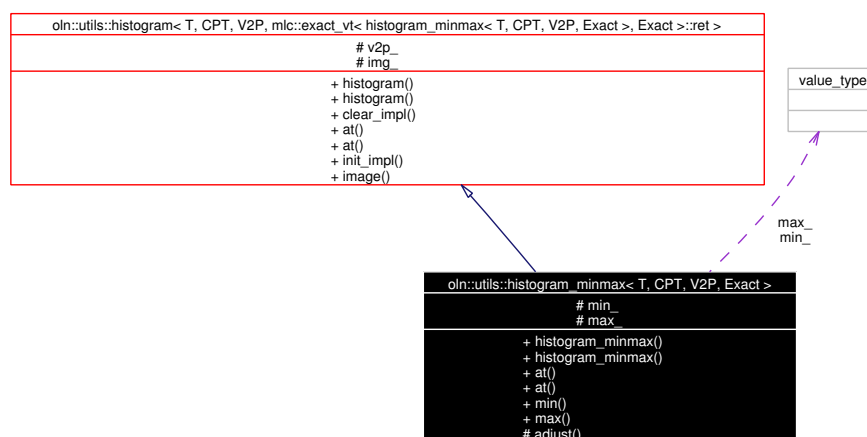
7.160 oln::utils::histogram_minmax< T, CPT, V2P, Exact > Class Template Reference

```
#include <histogram.hh>
```

Inheritance diagram for oln::utils::histogram_minmax< T, CPT, V2P, Exact >:



Collaboration diagram for oln::utils::histogram_minmax< T, CPT, V2P, Exact >:



Public Types

- typedef mlc::exact_vt< [histogram_minmax< T, CPT, V2P, Exact >](#), Exact >::ret exact_type

- typedef [hist_traits](#)< exact_type >::value_type **value_type**
- typedef [hist_traits](#)< exact_type >::cpt_type **cpt_type**
- typedef [hist_traits](#)< exact_type >::value_to_point_type **value_to_point_type**
- typedef [hist_traits](#)< exact_type >::point_type **point_type**
- typedef [hist_traits](#)< exact_type >::img_type **img_type**
- typedef [histogram](#)< T, CPT, V2P, exact_type > **upper_type**
- enum { **dim** = [hist_traits](#)<exact_type>::dim }

Public Member Functions

- **histogram_minmax** (const value_to_point_type &v2p=value_to_point_type())
- template<class I> **histogram_minmax** (const [oln::abstract::image](#)< I > &input, const value_to_point_type &v2p=value_to_point_type())
- const cpt_type **at** (const value_type &i) const
operator[] should be called.
- cpt_type & **at** (const value_type &i)
operator[] should be called.
- value_type **min** ()
Quick function min.
- value_type **max** ()
Quick function max.

Protected Member Functions

- void **adjust** (const value_type &idx)
Maintain the worst min and max.

Protected Attributes

- value_type **min_**
- value_type **max_**
Indices of min and max elements.

7.160.1 Detailed Description

template<typename T, typename CPT, class V2P, class Exact> class [oln::utils::histogram_minmax](#)< T, CPT, V2P, Exact >

Build the histogram and has quick min and max functions.

The idea behind the min- and max-specialized histogram is to maintain worst min and max bounds while updating histogram. It does not maintain `_exact_min` and `max` bounds, because this would involve some

costly computation when removing values from the histogram and maybe this time will be lost if the removed value is reinserted before `max()` or `min()` is called.

So it updates the `_worst_min` and `max` bounds whenever the histogram value are accessed, and delay the `_real_min` and `max` computation until `min()` or `max()` is called.

See also:

[histogram](#)
[histogram_min](#)
[histogram_max](#)

Definition at line 376 of file `histogram.hh`.

The documentation for this class was generated from the following file:

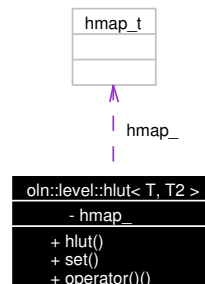
- `histogram.hh`

7.161 oln::level::hlut< T, T2 > Class Template Reference

Look up table "id" version.

```
#include <lut.hh>
```

Collaboration diagram for oln::level::hlut< T, T2 >:



Public Types

- typedef T2 **output_t**

Public Member Functions

- **hlut** & **set** (const T &val, const T2 &newval)
- const T2 **operator()** (const T &val) const

7.161.1 Detailed Description

```
template<class T, class T2 = T> class oln::level::hlut< T, T2 >
```

Look up table "id" version.

If the value has not been set, return id (see the example).

See also:

[hlut_def](#)

```

** hlut<int> l;
**
** l.set(16, 64);
**
** cout << l(16) << ", "    // print 64,
**      << l(51) << endl; // print 51
**

```

Definition at line 54 of file lut.hh.

7.161.2 Member Function Documentation

7.161.2.1 `template<class T, class T2 = T> const T2 oln::level::hlut< T, T2 >::operator() (const T & val) const` [inline]

Get the value corresponding to the key.

Return the key if the key has not been set.

Definition at line 81 of file lut.hh.

```
82     {
83         static typename hmap_t::const_iterator i;
84         i = hmap_.find(val);
85         return i != hmap_.end() ? i->second : val;
86     }
```

7.161.2.2 `template<class T, class T2 = T> hlut& oln::level::hlut< T, T2 >::set (const T & val, const T2 & newval)` [inline]

Register a value.

- val Key.
- newval Value corresponding to the key.

Definition at line 70 of file lut.hh.

```
71     {
72         hmap_[val] = newval;
73         return *this;
74     }
```

The documentation for this class was generated from the following file:

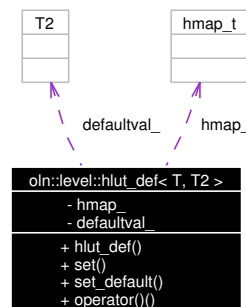
- lut.hh

7.162 oln::level::hlut_def< T, T2 > Class Template Reference

Look up table "default" version.

```
#include <lut.hh>
```

Collaboration diagram for oln::level::hlut_def< T, T2 >:



Public Types

- typedef T2 **output_t**

Public Member Functions

- [hlut_def](#) & [set](#) (const T &val, const T2 &newval)
- [hlut_def](#) & [set_default](#) (const T2 &defaultval)

Set a default value.

- const T2 [operator\(\)](#) (const T &val) const

7.162.1 Detailed Description

```
template<class T, class T2 = T> class oln::level::hlut_def< T, T2 >
```

Look up table "default" version.

If the value has not been set, return the default value.

Note:

The default value can be redefined.

See also:

[hlut_def](#)

```

** hlut_def<int> l;
**
** l.set(16, 64);
**
** cout << l(16) << ", " // print "64,"
**      << l(51) << endl; // print "0"
** l.set_default(42);
** cout << l(51) << endl; // print "42"
**

```

Definition at line 111 of file lut.hh.

7.162.2 Member Function Documentation

7.162.2.1 `template<class T, class T2 = T> const T2 oln::level::hlut_def< T, T2 >::operator() (const T & val) const` [inline]

Get the value corresponding to the key.

Return defaultval_ if the key has not been set.

Definition at line 148 of file lut.hh.

```
149     {  
150         static typename hmap_t::const_iterator i;  
151         i = hmap_.find(val);  
152         return i != hmap_.end() ? i->second : defaultval_;  
153     }
```

7.162.2.2 `template<class T, class T2 = T> hlut_def& oln::level::hlut_def< T, T2 >::set (const T & val, const T2 & newval)` [inline]

Register a value.

- val Key.
- newval Value corresponding to the key.

Definition at line 129 of file lut.hh.

Referenced by oln::morpho::internal_kill_cc_area().

```
130     {  
131         hmap_[val] = newval;  
132         return *this;  
133     }
```

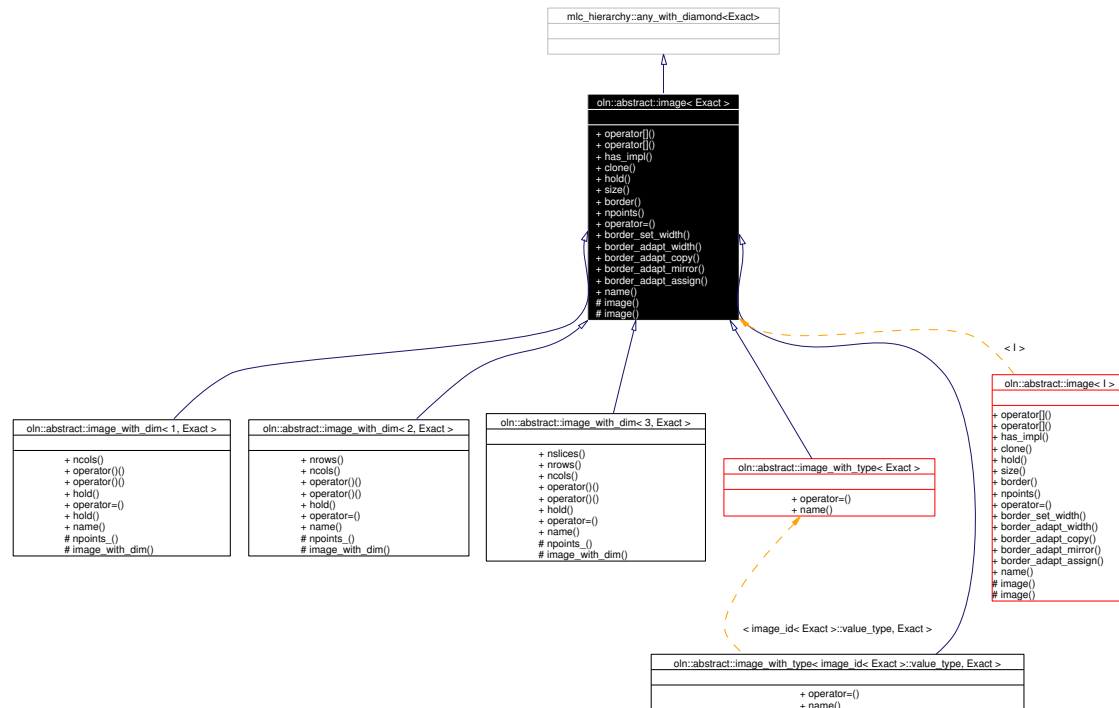
The documentation for this class was generated from the following file:

- lut.hh

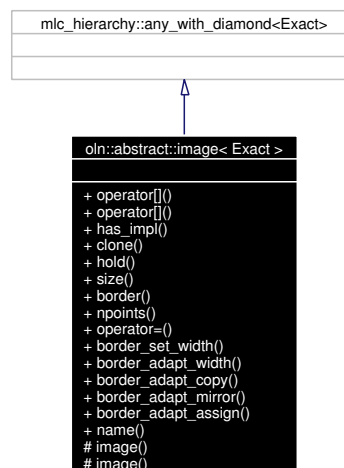
7.163 oln::abstract::image< Exact > Class Template Reference

```
#include <image.hh>
```

Inheritance diagram for oln::abstract::image< Exact >:



Collaboration diagram for oln::abstract::image< Exact >:



Public Types

- typedef `image_traits< Exact >::point_type point_type`

- typedef [image_traits](#)< Exact >::dpoint_type dpoint_type
- typedef [image_traits](#)< Exact >::iter_type iter_type
- typedef [image_traits](#)< Exact >::fwd_iter_type fwd_iter_type
- typedef [image_traits](#)< Exact >::bkd_iter_type bkd_iter_type
- typedef [image_traits](#)< Exact >::value_type value_type
- typedef [image_traits](#)< Exact >::size_type size_type
- typedef [image_traits](#)< Exact >::impl_type impl_type
- typedef [image](#)< Exact > **self_type**
- typedef Exact **exact_type**
- enum { **dim** = image_traits<Exact>::dim }

Public Member Functions

- const [value_type](#) [operator\[\]](#) (const [abstract::point](#)< [point_type](#) > &p) const
Return a reference to the value stored at p in the current image.
- [value_type](#) & [operator\[\]](#) (const [abstract::point](#)< [point_type](#) > &p)
Return the value stored stored at p in the current image.
- bool [has_impl](#) () const
Indicate if the image can be processed.
- [exact_type](#) [clone](#) () const
Clone the image, all the points are duplicated.
- bool [hold](#) (const [abstract::point](#)< [point_type](#) > &p) const
Test if the point p belong to the image.
- const [size_type](#) [size](#) () const
Return a reference to the image size.
- [coord](#) [border](#) () const
Return the value of the border width.
- [size_t](#) [npoints](#) () const
Return the total number of points in the current image.
- [exact_type](#) & [operator=](#) (self_type rhs)
Perform a shallow copy from rhs to the current image, the points are not duplicated but shared between the two images.
- void [border_set_width](#) ([coord](#) new_border, bool copy_border=false) const
Set the width of the border to perform operations on the image sides.
- void [border_adapt_width](#) ([coord](#) min_border, bool copy_border=false) const
Adapt the border if min_border is less or equal to the current border value.
- void [border_adapt_copy](#) ([coord](#) min_border) const
The border points will have the same value as the nearer real point of the image.

- void [border_adapt_mirror](#) (coord min_border) const

The border points value are set according to the value of the points on the image sides.

- void [border_adapt_assign](#) (coord min_border, value_type val) const

The border points value will be equal to the val parameters.

Static Public Member Functions

- std::string [name](#) ()

Protected Member Functions

- [image](#) (self_type &rhs)

7.163.1 Detailed Description

`template<class Exact> class oln::abstract::image< Exact >`

The image class whom derives all other image classes.

Definition at line 82 of file core/abstract/image.hh.

7.163.2 Member Typedef Documentation

7.163.2.1 `template<class Exact> typedef image_traits<Exact>::bkd_iter_type
oln::abstract::image< Exact >::bkd_iter_type`

Backward iterator type.

Reimplemented in [oln::abstract::image_with_dim](#)< 1, Exact >, [oln::abstract::image_with_dim](#)< 2, Exact >, and [oln::abstract::image_with_dim](#)< 3, Exact >.

Definition at line 106 of file core/abstract/image.hh.

7.163.2.2 `template<class Exact> typedef image_traits<Exact>::dpoint_type
oln::abstract::image< Exact >::dpoint_type`

Prefer the macro `oln_dpoint_type(I)` to retrieve the `dpoint_type` of an image.

See also:

[dpoint](#)

Reimplemented in [oln::abstract::image_with_dim](#)< 1, Exact >, [oln::abstract::image_with_dim](#)< 2, Exact >, and [oln::abstract::image_with_dim](#)< 3, Exact >.

Definition at line 92 of file core/abstract/image.hh.

7.163.2.3 `template<class Exact> typedef image_traits<Exact>::fwd_iter_type
oln::abstract::image< Exact >::fwd_iter_type`

Forward iterator type.

Reimplemented in [oln::abstract::image_with_dim](#)< 1, Exact >, [oln::abstract::image_with_dim](#)< 2, Exact >, and [oln::abstract::image_with_dim](#)< 3, Exact >.

Definition at line 104 of file core/abstract/image.hh.

7.163.2.4 `template<class Exact> typedef image_traits<Exact>::impl_type oln::abstract::image<
Exact >::impl_type`

Underlying implementation.

Definition at line 117 of file core/abstract/image.hh.

7.163.2.5 `template<class Exact> typedef image_traits<Exact>::iter_type oln::abstract::image<
Exact >::iter_type`

Prefer the macro `oln_iter_type(I)` to retrieve the `iter_type` of an image

See also:

[iter](#)

Reimplemented in [oln::abstract::image_with_dim](#)< 1, Exact >, [oln::abstract::image_with_dim](#)< 2, Exact >, and [oln::abstract::image_with_dim](#)< 3, Exact >.

Definition at line 98 of file core/abstract/image.hh.

7.163.2.6 `template<class Exact> typedef image_traits<Exact>::point_type
oln::abstract::image< Exact >::point_type`

Prefer the macro `oln_point_type(I)` to retrieve the `point_type` of an image.

See also:

[point](#)

Reimplemented in [oln::abstract::image_with_dim](#)< 1, Exact >, [oln::abstract::image_with_dim](#)< 2, Exact >, and [oln::abstract::image_with_dim](#)< 3, Exact >.

Definition at line 86 of file core/abstract/image.hh.

7.163.2.7 `template<class Exact> typedef image_traits<Exact>::size_type oln::abstract::image<
Exact >::size_type`

Indicate how the image size is handled.

See also:

[image_size](#)

Reimplemented in [oln::abstract::image_with_dim](#)< 1, Exact >, [oln::abstract::image_with_dim](#)< 2, Exact >, and [oln::abstract::image_with_dim](#)< 3, Exact >.

Definition at line 112 of file core/abstract/image.hh.

7.163.2.8 `template<class Exact> typedef image_traits<Exact>::value_type oln::abstract::image< Exact >::value_type`

Prefer the macro `oln_value_type(I)` to retrieve the `value_type` of an image.

Reimplemented in [oln::abstract::image_with_dim< 1, Exact >](#), [oln::abstract::image_with_dim< 2, Exact >](#), and [oln::abstract::image_with_dim< 3, Exact >](#).

Definition at line 108 of file `core/abstract/image.hh`.

7.163.3 Member Function Documentation

7.163.3.1 `template<class Exact> void oln::abstract::image< Exact >::border_adapt_assign (coord min_border, value_type val) const [inline]`

The border points value will be equal to the *val* parameters.

- *min_border* The new value of the border width.
- *val* The new value of the border points.

See also:

[image_size::border_](#)

Definition at line 321 of file `core/abstract/image.hh`.

```

322     {
323         border_adapt_width(min_border);
324         const_cast<impl_type *>(this->exact().impl())->border_assign(val);
325     }
```

7.163.3.2 `template<class Exact> void oln::abstract::image< Exact >::border_adapt_copy (coord min_border) const [inline]`

The border points will have the same value as the nearer real point of the image.

- *min_border* The new value of the border width.

See also:

[image_size::border_](#)

Definition at line 289 of file `core/abstract/image.hh`.

Referenced by `oln::convol::slow::convolve()`, `oln::morpho::dilation()`, `oln::morpho::erosion()`, `oln::morpho::fast_morpho()`, and `oln::morpho::sure::geodesic_dilation()`.

```

290     {
291         border_adapt_width(min_border);
292         const_cast<impl_type *>(this->exact().impl())->border_replicate();
293     }
```

7.163.3.3 `template<class Exact> void oln::abstract::image< Exact >::border_adapt_mirror(coord min_border) const [inline]`

The border points value are set according to the value of the points on the image sides.

- `min_border` The new value of the border width.

See also:

[image_size::border_](#)

Definition at line 304 of file `core/abstract/image.hh`.

```

305     {
306         border_adapt_width(min_border);
307         const_cast<impl_type *>(this->exact().impl())->border_mirror();
308     }
```

7.163.3.4 `template<class Exact> void oln::abstract::image< Exact >::border_adapt_width(coord min_border, bool copy_border = false) const [inline]`

Adapt the border if `min_border` is less or equal to the current border value.

- `min_border` The new value of the border width.
- `copy_border` Its default value is false.

Precondition:

`min_border >= 0`

See also:

[image_size::border_](#)

Definition at line 270 of file `core/abstract/image.hh`.

Referenced by `oln::abstract::image< I >::border_adapt_assign()`, `oln::abstract::image< I >::border_adapt_copy()`, `oln::abstract::image< I >::border_adapt_mirror()`, and `oln::convol::nagao_generalized()`.

```

272     {
273         precondition(min_border >= 0);
274         if (border() >= min_border)
275             return; // Don't shrink.
276
277         this->exact().border_set_width(min_border, copy_border);
278     }
```

7.163.3.5 `template<class Exact> void oln::abstract::image< Exact >::border_set_width(coord new_border, bool copy_border = false) const [inline]`

Set the width of the border to perform operations on the image sides.

- `new_border` The new value of the border width.

- `copy_border` Its default value is false.

Precondition:

`new_border >= 0`
`has_impl() == true`

See also:

[image_size::border_](#)

Definition at line 246 of file `core/abstract/image.hh`.

```

247     {
248         precondition(new_border >= 0);
249         precondition(has_impl());
250         if (border() == new_border)
251             return; // Nothing to do.
252
253         const_cast<impl_type *>(this->exact().impl())->border_reallocate_and_copy(new_border, copy_bor
254     }
```

7.163.3.6 `template<class Exact> exact_type oln::abstract::image< Exact >::clone () const` `[inline]`

Clone the image, all the points are duplicated.

Returns:

A real copy of the current image.

Definition at line 162 of file `core/abstract/image.hh`.

Referenced by `oln::morpho::fast_maxima_killer()`, `oln::morpho::fast_minima_killer()`, `oln::morpho::hybrid::geodesic_reconstruction_dilation()`, `oln::morpho::sequential::geodesic_reconstruction_dilation()`, `oln::morpho::sure::geodesic_reconstruction_dilation()`, `oln::morpho::n_dilation()`, and `oln::morpho::n_erosion()`.

```

163     {
164         return this->exact().clone();
165     }
```

7.163.3.7 `template<class Exact> bool oln::abstract::image< Exact >::has_impl () const` `[inline]`

Indicate if the image can be processed.

Returns:

True if the image can be processed, false otherwise.

Definition at line 150 of file `core/abstract/image.hh`.

Referenced by `oln::abstract::image< I >::border_set_width()`, `oln::abstract::image< I >::hold()`, and `oln::abstract::image< I >::size()`.

```

151     {
152         return this->exact().impl() != 0;
153     }
```

7.163.3.8 `template<class Exact> bool oln::abstract::image< Exact >::hold (const abstract::point< point_type > &p) const` [inline]

Test if the point p belong to the image.

Returns:

True if p belong to the image, false otherwise.

Definition at line 173 of file core/abstract/image.hh.

Referenced by `oln::level::frontp_connected_component()`, `oln::morpho::is_a_strict_maximum()`, `oln::morpho::is_a_strict_minimum()`, and `oln::morpho::watershed_seg_or()`.

```

174     {
175         assertion(has_impl());
176         return this->exact().impl()->hold(p.exact());
177     }

```

7.163.3.9 `template<class Exact> exact_type& oln::abstract::image< Exact >::operator=(self_type rhs)` [inline]

Perform a shallow copy from *rhs* to the current image, the points are not duplicated but shared between the two images.

See also:

[image::clone\(\)](#)

Reimplemented in `oln::abstract::image_with_dim< 1, Exact >`, `oln::abstract::image_with_dim< 2, Exact >`, `oln::abstract::image_with_dim< 3, Exact >`, `oln::abstract::image_with_type< T, Exact >`, `oln::abstract::data_type_image< Exact >`, `oln::abstract::data_type_image_with_dim< Dim, Exact >`, `oln::abstract::vectorial_image< Exact >`, `oln::abstract::non_vectorial_image< Exact >`, `oln::abstract::non_vectorial_image_with_dim< Dim, Exact >`, `oln::abstract::binary_image< Exact >`, `oln::abstract::binary_image_with_dim< Dim, Exact >`, `oln::abstract::integer_image< Exact >`, `oln::abstract::integer_image_with_dim< Dim, Exact >`, `oln::abstract::decimal_image< Exact >`, `oln::abstract::decimal_image_with_dim< Dim, Exact >`, `oln::abstract::image_with_type< image_id< Exact >::value_type, Exact >`, `oln::abstract::image_with_type< image_id< I >::value_type, I >`, `oln::abstract::data_type_image< I >`, and `oln::abstract::non_vectorial_image< I >`.

Definition at line 215 of file core/abstract/image.hh.

```

216     {
217         return this->exact().assign(rhs.exact());
218     }

```

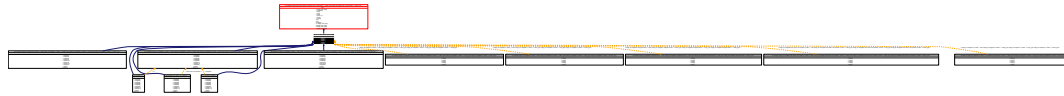
The documentation for this class was generated from the following file:

- core/abstract/image.hh

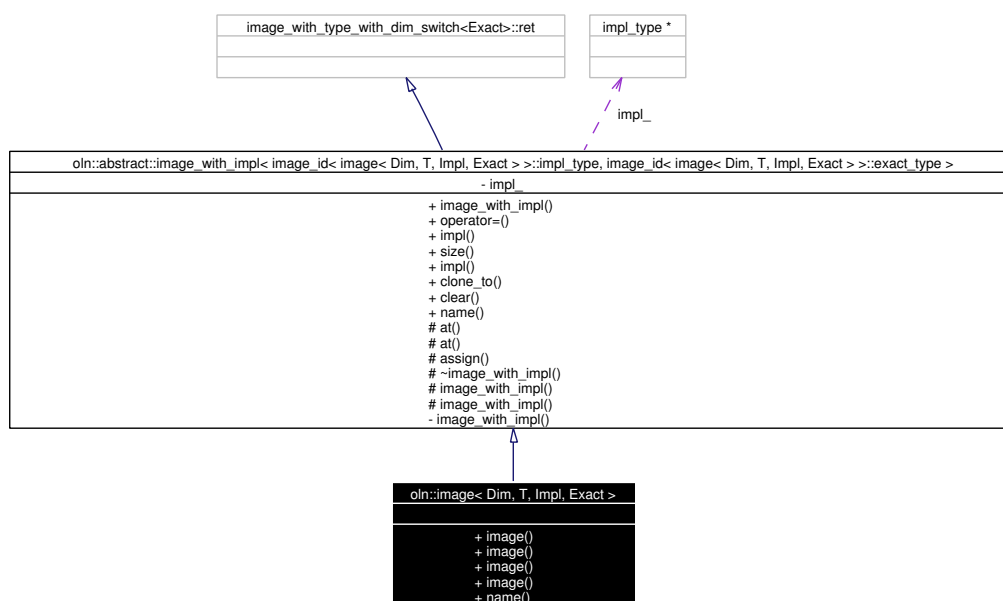
7.164 oln::image< Dim, T, Impl, Exact > Class Template Reference

```
#include <image.hh>
```

Inheritance diagram for oln::image< Dim, T, Impl, Exact >:



Collaboration diagram for oln::image< Dim, T, Impl, Exact >:



Public Types

- typedef mlc::exact_vt< [image](#)< Dim, T, Impl, Exact >, Exact >::ret **exact_type**
- typedef [image_traits](#)< exact_type >::point_type **point_type**
- typedef [image_traits](#)< exact_type >::dpoint_type **dpoint_type**
- typedef [image_traits](#)< exact_type >::iter_type **iter_type**
- typedef [image_traits](#)< exact_type >::fwd_iter_type **fwd_iter_type**
- typedef [image_traits](#)< exact_type >::bkd_iter_type **bkd_iter_type**
- typedef [image_traits](#)< exact_type >::value_type **value_type**
- typedef [image_traits](#)< exact_type >::size_type **size_type**
- typedef [image_traits](#)< exact_type >::impl_type **impl_type**
- typedef [image](#)< Dim, T, Impl, Exact > **self_type**
- typedef [abstract::image_with_impl](#)< Impl, exact_type > **super_type**

Public Member Functions

- [image](#) (self_type &rhs)

The new image is a shallow copy of rhs. The rhs points are not duplicated, they are shared by the new image.

- [image](#) ([impl_type](#) *i)
The image data pointer is set to i.
- [image](#) (const [size_type](#) &size)
Allocate memory according to the size value.

Static Public Member Functions

- `std::string name ()`

7.164.1 Detailed Description

`template<unsigned Dim, class T, class Impl, class Exact> class oln::image< Dim, T, Impl, Exact >`

Generic image class, all the classic image class (image with dim = N) will derive from it.

Definition at line 83 of file core/image.hh.

7.164.2 Member Typedef Documentation

7.164.2.1 `template<unsigned Dim, class T, class Impl, class Exact> typedef
image_traits<exact_type>::bkd_iter_type oln::image< Dim, T, Impl, Exact
>::bkd_iter_type`

Backward iterator type.

Reimplemented from [oln::abstract::image_with_impl](#)< Impl, Exact >.

Definition at line 113 of file core/image.hh.

7.164.2.2 `template<unsigned Dim, class T, class Impl, class Exact> typedef
image_traits<exact_type>::dpoint_type oln::image< Dim, T, Impl, Exact
>::dpoint_type`

Prefer the macro `oln_dpoint_type(I)` to retrieve the `dpoint_type` of an image.

See also:

[abstract::dpoint](#)

Definition at line 99 of file core/image.hh.

7.164.2.3 `template<unsigned Dim, class T, class Impl, class Exact> typedef
image_traits<exact_type>::fwd_iter_type oln::image< Dim, T, Impl, Exact
>::fwd_iter_type`

Forward iterator type.

Reimplemented from [oln::abstract::image_with_impl](#)< Impl, Exact >.

Definition at line 111 of file core/image.hh.

7.164.2.4 `template<unsigned Dim, class T, class Impl, class Exact> typedef
image_traits<exact_type>::impl_type oln::image< Dim, T, Impl, Exact >::impl_type`

Underlying implementation.

Reimplemented from [oln::abstract::image_with_impl< Impl, Exact >](#).

Reimplemented in [oln::image1d< T, Exact >](#), [oln::image2d< T, Exact >](#), [oln::image3d< T, Exact >](#), [oln::image2d< T >](#), [oln::image2d< unsigned >](#), and [oln::image2d< std::vector< oln::point2d > >](#).

Definition at line 121 of file core/image.hh.

7.164.2.5 `template<unsigned Dim, class T, class Impl, class Exact> typedef
image_traits<exact_type>::iter_type oln::image< Dim, T, Impl, Exact >::iter_type`

Prefer the macro `oln_iter_type(I)` to retrieve the `iter_type` of an image.

See also:

[abstract::iter](#)

Reimplemented from [oln::abstract::image_with_impl< Impl, Exact >](#).

Definition at line 105 of file core/image.hh.

7.164.2.6 `template<unsigned Dim, class T, class Impl, class Exact> typedef
image_traits<exact_type>::point_type oln::image< Dim, T, Impl, Exact >::point_type`

Prefer the macro `oln_point_type(I)` to retrieve the `point_type` of an image.

See also:

[abstract::point](#)

Reimplemented from [oln::abstract::image_with_impl< Impl, Exact >](#).

Definition at line 93 of file core/image.hh.

7.164.2.7 `template<unsigned Dim, class T, class Impl, class Exact> typedef
image_traits<exact_type>::size_type oln::image< Dim, T, Impl, Exact >::size_type`

Indicate how the image size is handled.

Reimplemented from [oln::abstract::image_with_impl< Impl, Exact >](#).

Definition at line 119 of file core/image.hh.

7.164.2.8 `template<unsigned Dim, class T, class Impl, class Exact> typedef
image_traits<exact_type>::value_type oln::image< Dim, T, Impl, Exact >::value_type`

Prefer the macro `oln_value_type(I)` to retrieve the `value_type` of an image.

Reimplemented from [oln::abstract::image_with_impl< Impl, Exact >](#).

Reimplemented in [oln::image1d< T, Exact >](#), [oln::image2d< T, Exact >](#), [oln::image3d< T, Exact >](#), [oln::image2d< T >](#), [oln::image2d< unsigned >](#), and [oln::image2d< std::vector< oln::point2d > >](#).

Definition at line 115 of file core/image.hh.

7.164.3 Constructor & Destructor Documentation

7.164.3.1 `template<unsigned Dim, class T, class Impl, class Exact> oln::image< Dim, T, Impl, Exact >::image (self_type & rhs) [inline]`

The new image is a shallow copy of *rhs*. The *rhs* points are not duplicated, they are shared by the new image.

See also:

[oln::abstract::image::clone\(\)](#)

Definition at line 139 of file core/image.hh.

```
139                                     : super_type(rhs)
140     {
141         mlc_init_static_hierarchy(Exact);
142     }
```

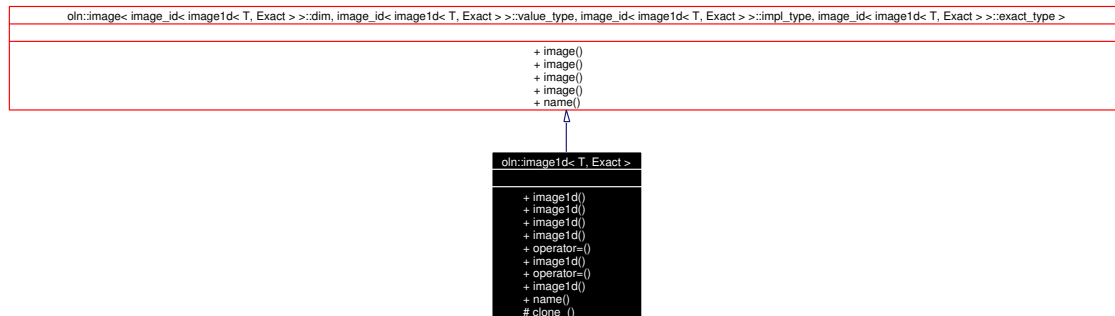
The documentation for this class was generated from the following file:

- core/image.hh

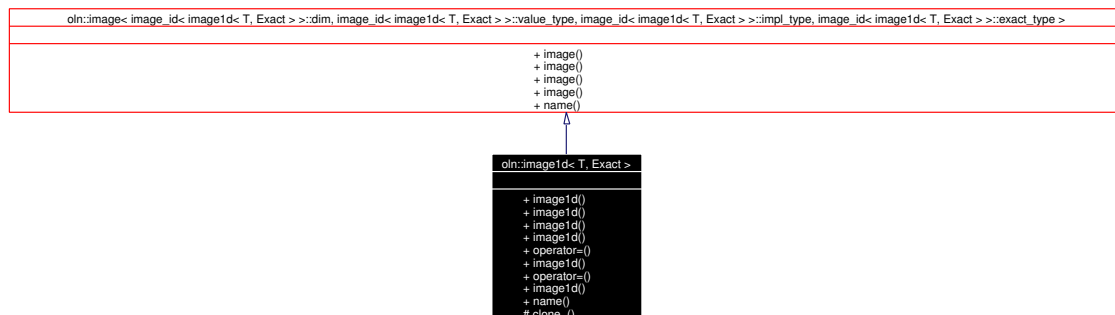
7.165 oln::image1d< T, Exact > Class Template Reference

```
#include <image1d.hh>
```

Inheritance diagram for oln::image1d< T, Exact >:



Collaboration diagram for oln::image1d< T, Exact >:



Public Types

- typedef `image1d< T, Exact >` **self_type**
- typedef `image_id< image1d< T, Exact >>::value_type` **value_type**
- typedef `image_id< image1d< T, Exact >>::exact_type` **exact_type**
- typedef `image_id< image1d< T, Exact >>::impl_type` **impl_type**
- typedef `image< image_id< image1d< T, Exact >>::dim, value_type, impl_type, exact_type >` **super_type**

Public Member Functions

- `image1d (coord ncols, coord border=2)`
Allocate memory to contain an `image1d` with `ncols` column plus a border width equal to 2 by default.
- `image1d (const image1d_size &size)`
Allocate memory to contain an `image1d` with a size equal to `size`.
- `image1d (self_type &rhs)`

Build a new [image1d](#) by performing a shallow copy of rhs, all the points will be shared between rhs and the current image.

- `exact_type & operator= (self_type rhs)`

Perform a shallow copy from rhs to the current image, the points are not duplicated but shared between the two images.

- `image1d (const io::internal::anything &r)`

Perform a shallow copy from r to the new image, the points are not duplicated, but shared between the two images.

- `image1d & operator= (const io::internal::anything &r)`

Perform a shallow copy from r to the current image, the points are not duplicated, but shared between the two images.

- `image1d (const self_type &rhs)`

Static Public Member Functions

- `std::string name ()`

Protected Member Functions

- `exact_type clone_ () const`

Return a deep copy of the current image.

Friends

- `class abstract::image< exact_type >`

7.165.1 Detailed Description

`template<class T, class Exact> class oln::image1d< T, Exact >`

To instantiate an [image1d](#) with `oln::rgb_8` as `value_type`, one can write:

```
oln::image1d<ntg::rgb_8> t;
```

Definition at line 91 of file `image1d.hh`.

7.165.2 Member Typedef Documentation

7.165.2.1 `template<class T, class Exact> typedef image_id<image1d<T, Exact> >::impl_type oln::image1d< T, Exact >::impl_type`

Underlying implementation.

Reimplemented from `oln::image< Dim, T, Impl, Exact >`.

Definition at line 103 of file `image1d.hh`.

7.165.2.2 `template<class T, class Exact> typedef image_id<image1d<T, Exact> >::value_type oln::image1d< T, Exact >::value_type`

Prefer the macro `oln_value_type(I)` to retrieve the `value_type` of an image.

Reimplemented from `oln::image< Dim, T, Impl, Exact >`.

Definition at line 101 of file `image1d.hh`.

7.165.3 Constructor & Destructor Documentation

7.165.3.1 `template<class T, class Exact> oln::image1d< T, Exact >::image1d (self_type & rhs) [inline]`

Build a new `image1d` by performing a shallow copy of *rhs*, all the points will be shared between *rhs* and the current image.

See also:

`abstract::image::clone()`

Definition at line 146 of file `image1d.hh`.

```
146                                     : // shallow copy
147     super_type(rhs)
148     { mlc_init_static_hierarchy(Exact); }
```

7.165.3.2 `template<class T, class Exact> oln::image1d< T, Exact >::image1d (const io::internal::anything & r) [inline]`

Perform a shallow copy from *r* to the new image, the points are not duplicated, but shared between the two images.

See also:

`abstract::image::clone()`

Definition at line 172 of file `image1d.hh`.

References `oln::io::internal::anything::assign()`.

```
173     : super_type()
174     {
175     mlc_init_static_hierarchy(Exact);
176     r.assign(*this);
177     }
```

7.165.4 Member Function Documentation

7.165.4.1 `template<class T, class Exact> image1d& oln::image1d< T, Exact >::operator= (const io::internal::anything & r) [inline]`

Perform a shallow copy from *r* to the current image, the points are not duplicated, but shared between the two images.

See also:

[abstract::image::clone\(\)](#)

Definition at line 187 of file image1d.hh.

References [oln::io::internal::anything::assign\(\)](#).

```
188     {  
189         return r.assign(*this);  
190     }
```

7.165.4.2 `template<class T, class Exact> exact_type& oln::image1d< T, Exact >::operator=(self_type rhs) [inline]`

Perform a shallow copy from *rhs* to the current image, the points are not duplicated but shared between the two images.

See also:

[abstract::image::clone\(\)](#)

Reimplemented from [oln::abstract::image_with_impl< Impl, Exact >](#).

Definition at line 158 of file image1d.hh.

```
159     {  
160         return this->exact().assign(rhs.exact());  
161     }
```

The documentation for this class was generated from the following file:

- image1d.hh

7.166 oln::image1d< T, Exact >::mute< U > Struct Template Reference

Define ret equal to image1d<U>.

```
#include <image1d.hh>
```

Public Types

- typedef [image1d](#)< U > ret

7.166.1 Detailed Description

template<class T, class Exact>template<class U> struct oln::image1d< T, Exact >::mute< U >

Define ret equal to image1d<U>.

Definition at line 204 of file image1d.hh.

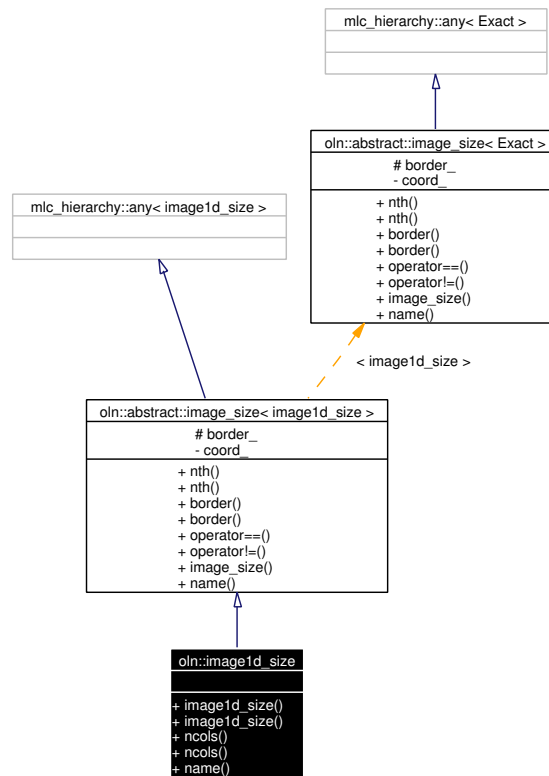
The documentation for this struct was generated from the following file:

- image1d.hh

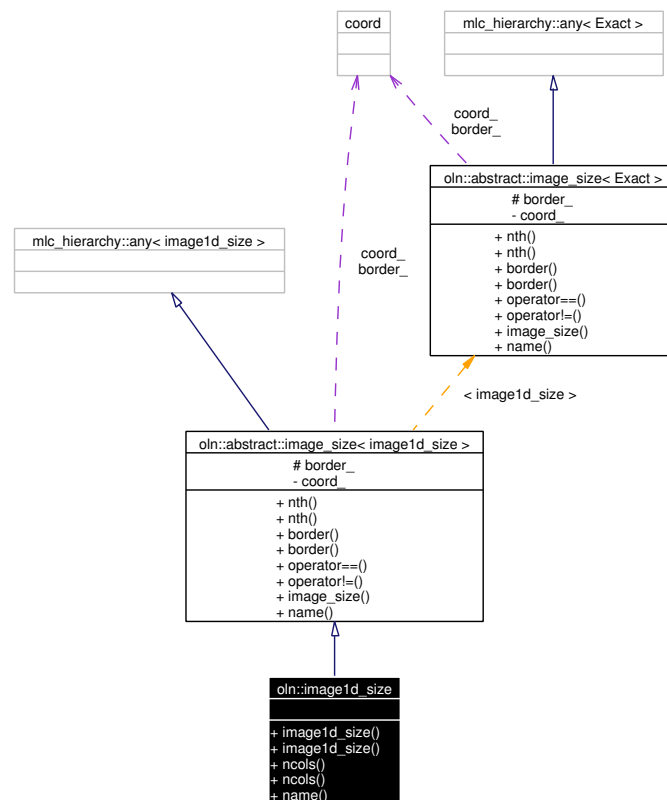
7.167 oln::image1d_size Struct Reference

```
#include <image1d_size.hh>
```

Inheritance diagram for oln::image1d_size:



Collaboration diagram for oln::image1d_size:



Public Member Functions

- `image1d_size` (`coord` ncols, `coord` border)

Image1d_size constructor.

- `coord` ncols () const

Return the number of columns in the image.

- `coord` & ncols ()

Return a reference to the number of columns in the image.

Static Public Member Functions

- `std::string` name ()

7.167.1 Detailed Description

Size_type for `image1d`.

Definition at line 55 of file `image1d_size.hh`.

7.167.2 Constructor & Destructor Documentation

7.167.2.1 oln::image1d_size::image1d_size (coord *ncols*, coord *border*) [inline]

Image1d_size constructor.

- *ncols* The number of columns in the image is set to *ncols*.
- *border* The border width of the image is set to *border*.

Definition at line 67 of file image1d_size.hh.

References oln::coord, and oln::abstract::image_size< image1d_size >::nth().

```
68     {  
69         nth(0) = ncols;  
70         border_ = border;  
71     }
```

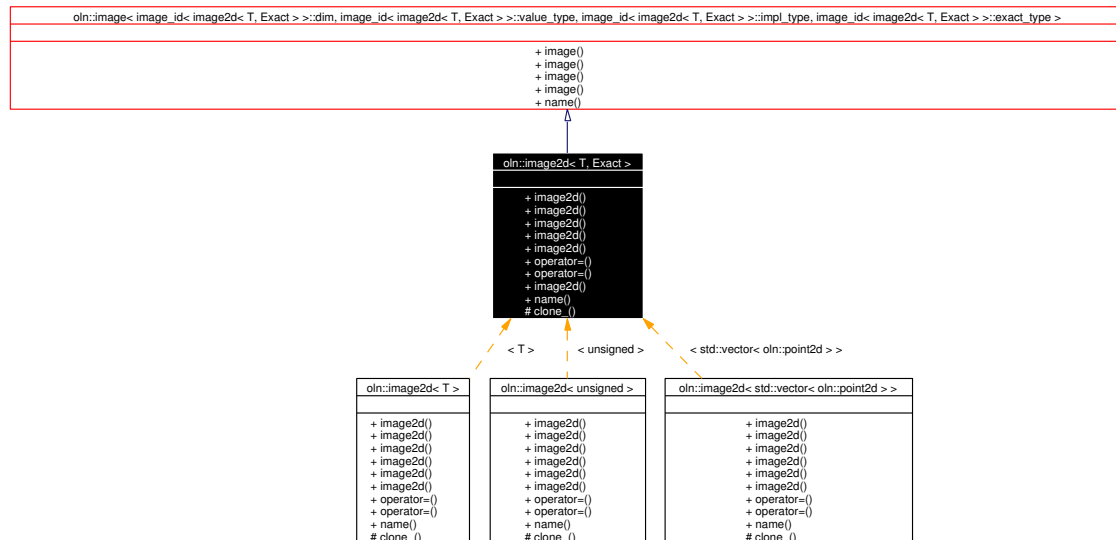
The documentation for this struct was generated from the following file:

- image1d_size.hh

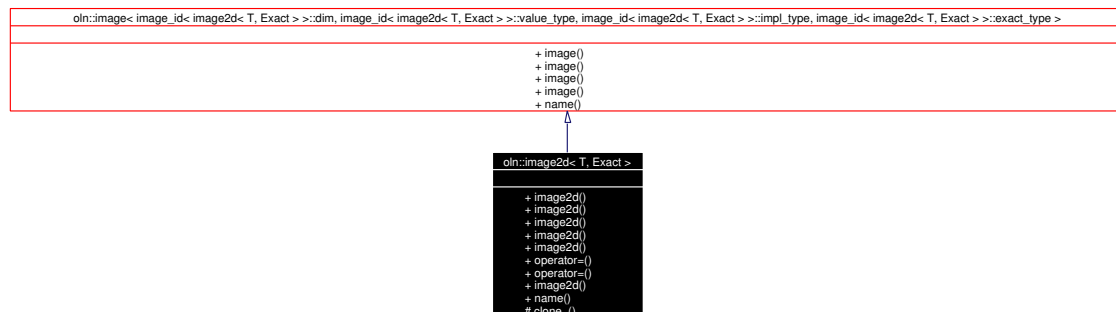
7.168 oln::image2d< T, Exact > Class Template Reference

```
#include <image2d.hh>
```

Inheritance diagram for oln::image2d< T, Exact >:



Collaboration diagram for oln::image2d< T, Exact >:



Public Types

- typedef `image2d< T, Exact > self_type`
- typedef `image_id< image2d< T, Exact > >::value_type value_type`
- typedef `image_id< image2d< T, Exact > >::exact_type exact_type`
- typedef `image_id< image2d< T, Exact > >::impl_type impl_type`
- typedef `oln::image< image_id< image2d< T, Exact > >::dim, value_type, impl_type, exact_type > super_type`

Public Member Functions

- `image2d` (`coord` nrow, `coord` ncol, `coord` border=2)

Allocate memory to contain an *image2d* with ncols column and nrows rows plus a border width equal to 2 by default.

- *image2d* (const *image2d_size* &size)

Allocate memory to contain an *image2d* with a size equal to size.

- *image2d* (self_type &rhs)

Build a new *image2d* by performing a shallow copy of rhs, the points are not duplicated, but shared between rhs and the new image.

- *image2d* (const io::internal::anything &r)

Perform a shallow copy from r to the new image, the points are not duplicated, but shared between the two images.

- *image2d* & operator= (const io::internal::anything &r)

Perform a shallow copy from rhs to the current image, the points are not duplicated, but shared between the two images.

- exact_type & operator= (self_type rhs)

Perform a shallow copy from r to the current image, the points are not duplicated but shared between the two images.

- *image2d* (const self_type &rhs)

Static Public Member Functions

- std::string name ()

Protected Member Functions

- exact_type clone_ () const

Return a deep copy of the current image.

Friends

- class abstract::image< exact_type >

7.168.1 Detailed Description

template<class T, class Exact> class oln::image2d< T, Exact >

To instantiate an *image2d* with oln::rgb_8 as value_type, one can write:

```
oln::image2d<ntg::rgb_8> t = load("img.ppm");
```

Definition at line 89 of file image2d.hh.

7.168.2 Member Typedef Documentation

7.168.2.1 `template<class T, class Exact> typedef image_id<image2d<T, Exact> >::impl_type
oln::image2d< T, Exact >::impl_type`

Underlying implementation.

Reimplemented from [oln::image](#)< Dim, T, Impl, Exact >.

Definition at line 101 of file `image2d.hh`.

7.168.2.2 `template<class T, class Exact> typedef image_id<image2d<T, Exact> >::value_type
oln::image2d< T, Exact >::value_type`

Prefer the macro `oln_value_type(I)` to retrieve the `value_type` of an image.

Reimplemented from [oln::image](#)< Dim, T, Impl, Exact >.

Definition at line 99 of file `image2d.hh`.

7.168.3 Constructor & Destructor Documentation

7.168.3.1 `template<class T, class Exact> oln::image2d< T, Exact >::image2d (self_type & rhs)
[inline]`

Build a new [image2d](#) by performing a shallow copy of *rhs*, the points are not duplicated, but shared between *rhs* and the new image.

See also:

[abstract::image::clone\(\)](#)

Definition at line 147 of file `image2d.hh`.

```
147                                     : // shallow copy
148     super_type(rhs)
149     {
150         mlc_init_static_hierarchy(Exact);
151     }
```

7.168.3.2 `template<class T, class Exact> oln::image2d< T, Exact >::image2d (const
io::internal::anything & r) [inline]`

Perform a shallow copy from *r* to the new image, the points are not duplicated, but shared between the two images.

See also:

[abstract::image::clone\(\)](#)

Definition at line 161 of file `image2d.hh`.

```
162     : super_type()
163     {
164         mlc_init_static_hierarchy(Exact);
165         r.assign(*this);
166     }
```

7.168.4 Member Function Documentation

7.168.4.1 `template<class T, class Exact> exact_type& oln::image2d< T, Exact >::operator=(self_type rhs) [inline]`

Perform a shallow copy from *r* to the current image, the points are not duplicated but shared between the two images.

See also:

[abstract::image::clone\(\)](#)

Reimplemented from [oln::abstract::image_with_impl< Impl, Exact >](#).

Definition at line 189 of file image2d.hh.

```
190     {
191         return this->exact().assign(rhs.exact());
192     }
```

7.168.4.2 `template<class T, class Exact> image2d& oln::image2d< T, Exact >::operator=(const io::internal::anything & r) [inline]`

Perform a shallow copy from *rhs* to the current image, the points are not duplicated, but shared between the two images.

See also:

[abstract::image::clone\(\)](#)

Definition at line 176 of file image2d.hh.

```
177     {
178         return r.assign(*this);
179     }
```

The documentation for this class was generated from the following file:

- image2d.hh

7.169 oln::image2d< T, Exact >::mute< U > Struct Template Reference

Define ret equal to image2d<U>.

```
#include <image2d.hh>
```

Public Types

- typedef [image2d](#)< U > ret

7.169.1 Detailed Description

template<class T, class Exact>template<class U> struct oln::image2d< T, Exact >::mute< U >

Define ret equal to image2d<U>.

Definition at line 206 of file image2d.hh.

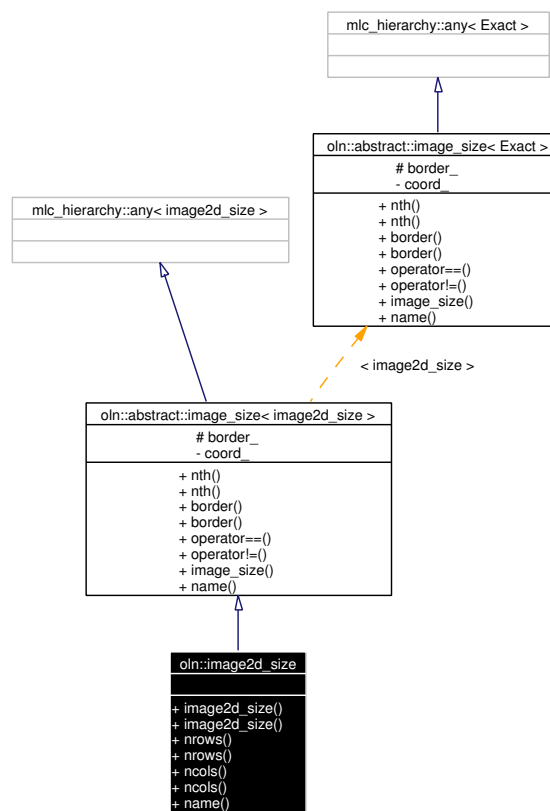
The documentation for this struct was generated from the following file:

- image2d.hh

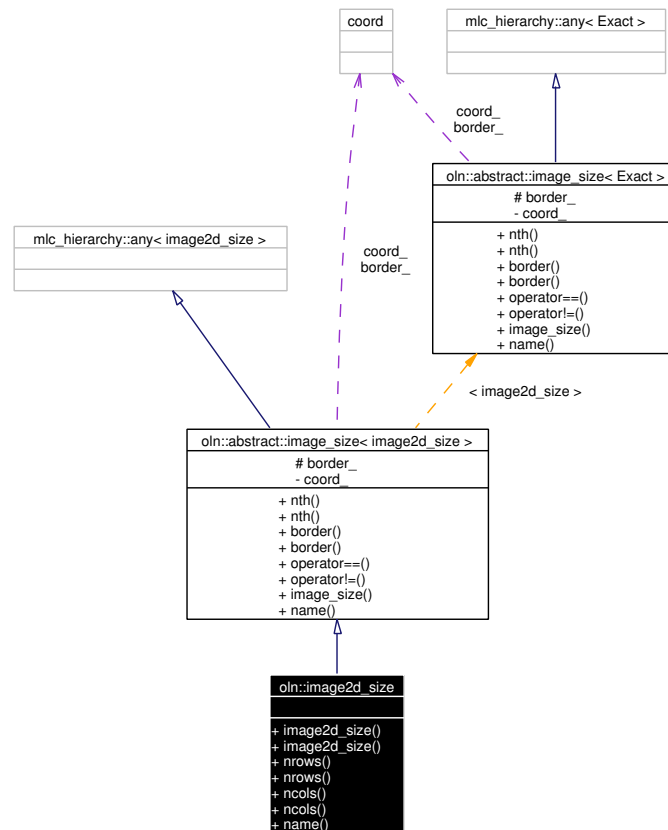
7.170 oln::image2d_size Struct Reference

```
#include <image2d_size.hh>
```

Inheritance diagram for oln::image2d_size:



Collaboration diagram for oln::image2d_size:



Public Member Functions

- `image2d_size` (`coord` `nrows`, `coord` `ncols`, `coord` `border`)

Image2d_size constructor.

- `coord` `nrows` () const

Return the number of rows in the image.

- `coord` & `nrows` ()

Return a reference to the number of rows in the image.

- `coord` `ncols` () const

Return the number of columns in the image.

- `coord` & `ncols` ()

Return a reference to the number of columns in the image.

Static Public Member Functions

- `std::string` `name` ()

7.170.1 Detailed Description

Size_type for [image2d](#).

Definition at line 56 of file image2d_size.hh.

7.170.2 Constructor & Destructor Documentation

7.170.2.1 oln::image2d_size::image2d_size (coord nrows, coord ncols, coord border) [inline]

Image2d_size constructor.

- **nrows** The number of rows in the image is set to *nrows*.
- **ncols** The number of columns in the image is set to *ncols*.
- **border** The border width of the image is set to *border*.

Definition at line 70 of file image2d_size.hh.

References [oln::coord](#), and [oln::abstract::image_size< image2d_size >::nth\(\)](#).

```
71     {  
72         nth(0) = nrows;  
73         nth(1) = ncols;  
74         border_ = border;  
75     }
```

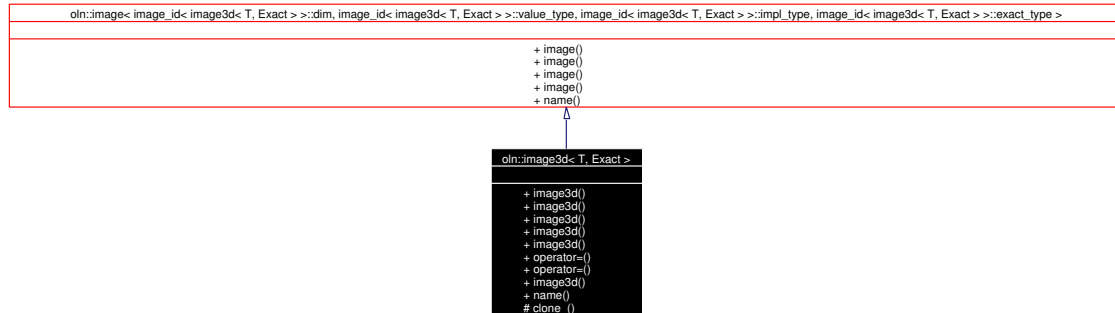
The documentation for this struct was generated from the following file:

- image2d_size.hh

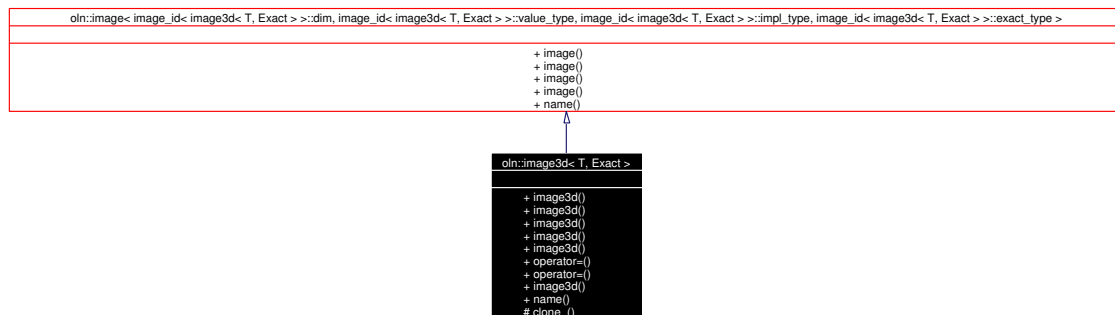
7.171 oln::image3d< T, Exact > Class Template Reference

```
#include <image3d.hh>
```

Inheritance diagram for oln::image3d< T, Exact >:



Collaboration diagram for oln::image3d< T, Exact >:



Public Types

- typedef `image3d< T, Exact >` **self_type**
- typedef `image_id< image3d< T, Exact > >::value_type` **value_type**
- typedef `image_id< image3d< T, Exact > >::exact_type` **exact_type**
- typedef `image_id< image3d< T, Exact > >::impl_type` **impl_type**
- typedef `image< image_id< image3d< T, Exact > >::dim, value_type, impl_type, exact_type >` **super_type**

Public Member Functions

- `image3d` (`coord` nslices, `coord` nrows, `coord` ncols, `coord` border=2)
Allocate memory to contain an `image3d` with ncols column, nrows rows, and nslices slices plus a border width equal to 2 by default.
- `image3d` (const `image3d_size` &size)
Allocate memory to contain an `image3d` with a size equal to size.

- [image3d](#) (self_type &rhs)
Build a new [image3d](#) by performing a shallow copy of rhs, the points are not duplicated, but shared between rhs and the new image.
- [image3d](#) (const [io::internal::anything](#) &r)
Perform a shallow copy from r to the new image, the points are not duplicated, but shared between the two images.
- [image3d](#) & [operator=](#) (const [io::internal::anything](#) &r)
Perform a shallow copy from rhs to the current image, the points are not duplicated, but shared between the two images.
- exact_type & [operator=](#) (self_type rhs)
Perform a shallow copy from r to the current image, the points are not duplicated but shared between the two images.
- [image3d](#) (const self_type &rhs)

Static Public Member Functions

- std::string [name](#) ()

Protected Member Functions

- exact_type [clone_](#) () const
Return a deep copy of the current image.

Friends

- class [abstract::image](#)< exact_type >

7.171.1 Detailed Description

template<class T, class Exact> class oln::image3d< T, Exact >

To instantiate an [image3d](#) with oln::rgb_8 as value_type, one can write:

```
oln::image3d<ntg::rgb_8> t;
```

Definition at line 89 of file image3d.hh.

7.171.2 Member Typedef Documentation

7.171.2.1 **template<class T, class Exact> typedef image_id<[image3d](#)<T, Exact> >::[impl_type](#) [oln::image3d](#)< T, Exact >::[impl_type](#)**

Underlying implementation.

Reimplemented from [oln::image](#)< Dim, T, Impl, Exact >.

Definition at line 101 of file image3d.hh.

7.171.2.2 `template<class T, class Exact> typedef image_id<image3d<T, Exact> >::value_type oln::image3d< T, Exact >::value_type`

Prefer the macro `oln_value_type(I)` to retrieve the `value_type` of an image.

Reimplemented from `oln::image< Dim, T, Impl, Exact >`.

Definition at line 99 of file `image3d.hh`.

7.171.3 Constructor & Destructor Documentation

7.171.3.1 `template<class T, class Exact> oln::image3d< T, Exact >::image3d (self_type & rhs) [inline]`

Build a new `image3d` by performing a shallow copy of *rhs*, the points are not duplicated, but shared between *rhs* and the new image.

See also:

`abstract::image::clone()`

Definition at line 146 of file `image3d.hh`.

```

146                                     :
147     super_type(rhs)
148     {
149         mlc_init_static_hierarchy(Exact);
150     }
```

7.171.3.2 `template<class T, class Exact> oln::image3d< T, Exact >::image3d (const io::internal::anything & r) [inline]`

Perform a shallow copy from *r* to the new image, the points are not duplicated, but shared between the two images.

See also:

`abstract::image::clone()`

Definition at line 160 of file `image3d.hh`.

References `oln::io::internal::anything::assign()`.

```

160                                     : super_type()
161     {
162         mlc_init_static_hierarchy(Exact);
163         r.assign(*this);
164     }
```

7.171.4 Member Function Documentation

7.171.4.1 `template<class T, class Exact> exact_type& oln::image3d< T, Exact >::operator= (self_type rhs) [inline]`

Perform a shallow copy from *r* to the current image, the points are not duplicated but shared between the two images.

See also:

[abstract::image::clone\(\)](#)

Reimplemented from [oln::abstract::image_with_impl< Impl, Exact >](#).

Definition at line 187 of file image3d.hh.

References [oln::abstract::image_with_impl< Impl, Exact >::assign\(\)](#).

```
188     {
189         return this->exact().assign(rhs.exact());
190     }
```

7.171.4.2 `template<class T, class Exact> image3d& oln::image3d< T, Exact >::operator= (const io::internal::anything & r)` [inline]

Perform a shallow copy from *rhs* to the current image, the points are not duplicated, but shared between the two images.

See also:

[abstract::image::clone\(\)](#)

Definition at line 174 of file image3d.hh.

References [oln::io::internal::anything::assign\(\)](#).

```
175     {
176         return r.assign(*this);
177     }
```

The documentation for this class was generated from the following file:

- image3d.hh

7.172 `oln::image3d< T, Exact >::mute< U >` Struct Template Reference

Define `ret` equal to `image3d<U>`.

```
#include <image3d.hh>
```

Public Types

- typedef `image3d< U >` `ret`

7.172.1 Detailed Description

`template<class T, class Exact>template<class U> struct oln::image3d< T, Exact >::mute< U >`

Define `ret` equal to `image3d<U>`.

Definition at line 204 of file `image3d.hh`.

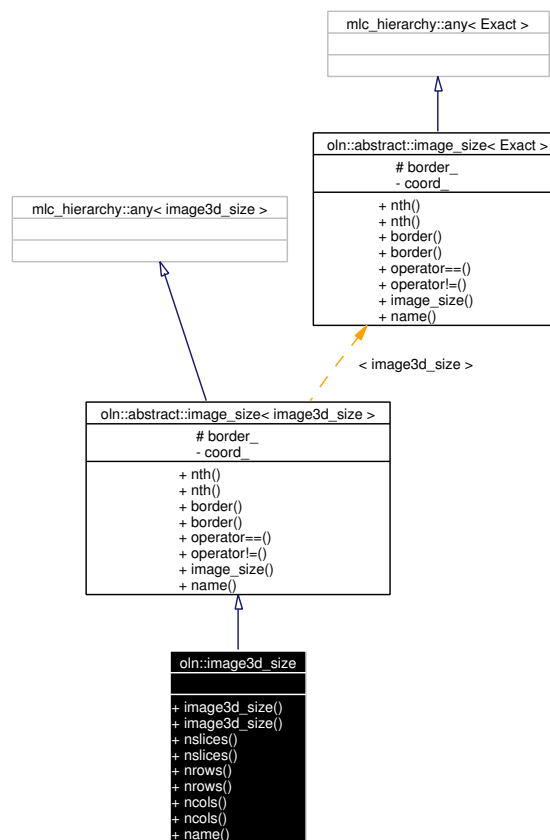
The documentation for this struct was generated from the following file:

- `image3d.hh`

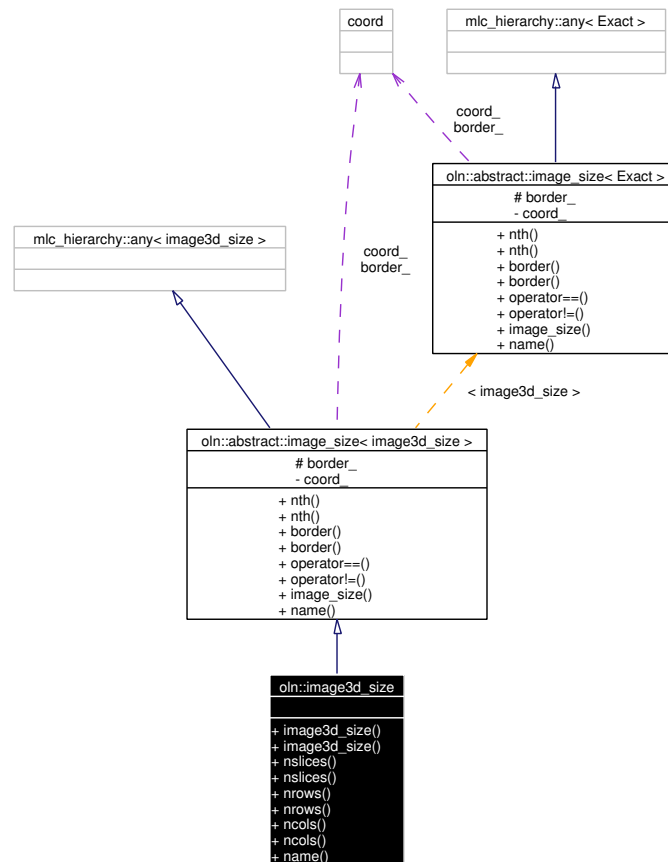
7.173 oln::image3d_size Struct Reference

```
#include <image3d_size.hh>
```

Inheritance diagram for oln::image3d_size:



Collaboration diagram for oln::image3d_size:



Public Member Functions

- **image3d_size** (**coord** nslices, **coord** nrows, **coord** ncols, **coord** border)
Image2d_size constructor.
- **coord nslices** () const
Return the number of slices in the image.
- **coord & nslices** ()
Return a reference to the number of slices in the image.
- **coord nrows** () const
Return the number of rows in the image.
- **coord & nrows** ()
Return a reference to the number of rows in the image.
- **coord ncols** () const
Return the number of columns in the image.
- **coord & ncols** ()
Return a reference to the number of columns in the image.

Static Public Member Functions

- `std::string name ()`

7.173.1 Detailed Description

Size_type for [image3d](#).

Definition at line 54 of file `image3d_size.hh`.

7.173.2 Constructor & Destructor Documentation

7.173.2.1 `oln::image3d_size::image3d_size (coord nslices, coord nrows, coord ncols, coord border)` [inline]

Image2d_size constructor.

- `nslices` The number of slices in the image is set to *nslices*
- `nrows` The number of rows in the image is set to *nrows*.
- `ncols` The number of columns in the image is set to *ncols*.
- `border` The border width of the image is set to *border*.

Definition at line 75 of file `image3d_size.hh`.

References `oln::coord`, and `oln::abstract::image_size< image3d_size >::nth()`.

```
76     {  
77         nth(0) = nslices;  
78         nth(1) = nrows;  
79         nth(2) = ncols;  
80         border_ = border;  
81     }
```

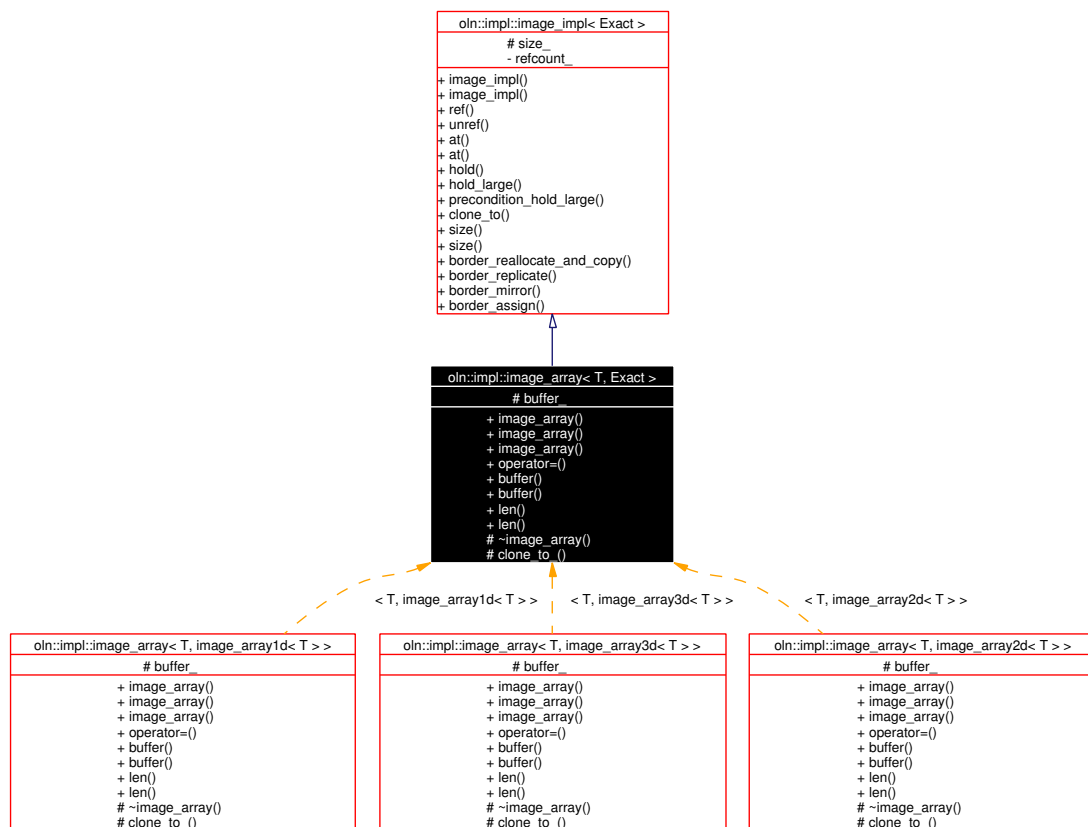
The documentation for this struct was generated from the following file:

- `image3d_size.hh`

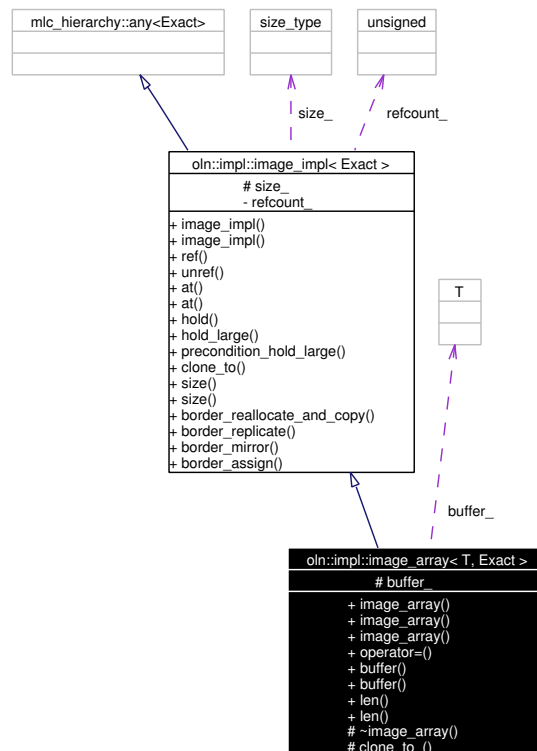
7.174 oln::impl::image_array< T, Exact > Class Template Reference

```
#include <image_array.hh>
```

Inheritance diagram for oln::impl::image_array< T, Exact >:



Collaboration diagram for oln::impl::image_array< T, Exact >:



Public Types

- `typedef impl_traits< Exact >::point_type point_type`
- `typedef impl_traits< Exact >::value_type value_type`
- `typedef impl_traits< Exact >::size_type size_type`
- `typedef Exact exact_type`
- `typedef image_impl< Exact > super_type`
- `typedef image_array< T, Exact > self_type`
- `enum { dim = impl_traits<Exact>::dim }`

Public Member Functions

- `image_array (const size_type &s)`
Constructor that allocates `buffer_` to be an array of `s` `value_type`.
- `image_array (const self_type &)`
- `void operator= (const self_type &)`
- `const T * buffer () const`
Return a constant pointer to the data array.
- `T * buffer ()`
Return a point to the data array.
- `size_t len () const`

Return the length of the data array.

- `size_t len` (const `size_type` &`s`) const

Return the length of the data array.

Protected Member Functions

- void `clone_to_` (`exact_type` *`output_data`) const

Perform a deep copy of the current data array. This copy is pointed to by `output_data`.

Protected Attributes

- `T * buffer_`

Friends

- class `image_impl`< `Exact` >

7.174.1 Detailed Description

`template<class T, class Exact> class oln::impl::image_array< T, Exact >`

Array implementation of image data.

Definition at line 95 of file `image_array.hh`.

7.174.2 Constructor & Destructor Documentation

7.174.2.1 `template<class T, class Exact> oln::impl::image_array< T, Exact >::image_array`
(const `size_type` &`s`) [`inline`]

Constructor that allocates `buffer_` to be an array of `s` *value_type*.

Precondition:

`s > 0`

Definition at line 119 of file `image_array.hh`.

```

119                                     : super_type(s), buffer_(0)
120     {
121         allocate_data_(buffer_, len(s));
122     }
```

7.174.3 Member Function Documentation

7.174.3.1 `template<class T, class Exact> T* oln::impl::image_array< T, Exact >::buffer` ()
[`inline`]

Return a point to the data array.

Invariant:`buffer_ != 0`

Definition at line 150 of file `image_array.hh`.

```

151     {
152         invariant(buffer_ != 0);
153         return buffer_;
154     }

```

7.174.3.2 `template<class T, class Exact> const T* oln::impl::image_array< T, Exact >::buffer () const` [inline]

Return a constant pointer to the data array.

Invariant:`buffer_ != 0`

Definition at line 138 of file `image_array.hh`.

```

139     {
140         invariant(buffer_ != 0);
141         return buffer_;
142     }

```

7.174.3.3 `template<class T, class Exact> void oln::impl::image_array< T, Exact >::clone_to_(exact_type * output_data) const` [inline, protected]

Perform a deep copy of the current data array. This copy is pointed to by *output_data*.

Precondition:

```

output_data != 0
output_data->len() == len()

```

Definition at line 188 of file `image_array.hh`.

```

189     {
190         precondition(output_data != 0);
191         precondition(output_data->len() == len());
192         memcpy(output_data->buffer(),
193                buffer_,
194                len() * sizeof(T));
195     }

```

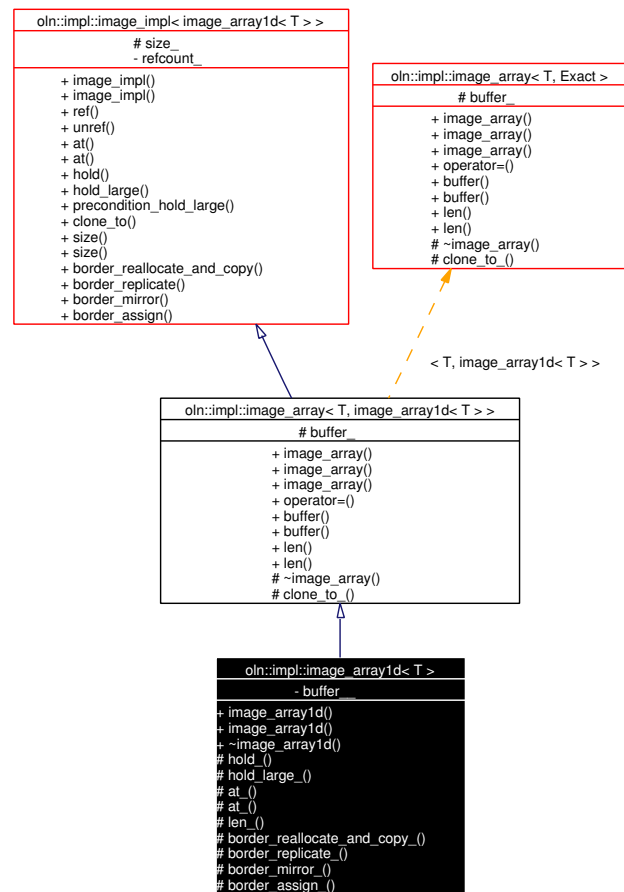
The documentation for this class was generated from the following file:

- `image_array.hh`

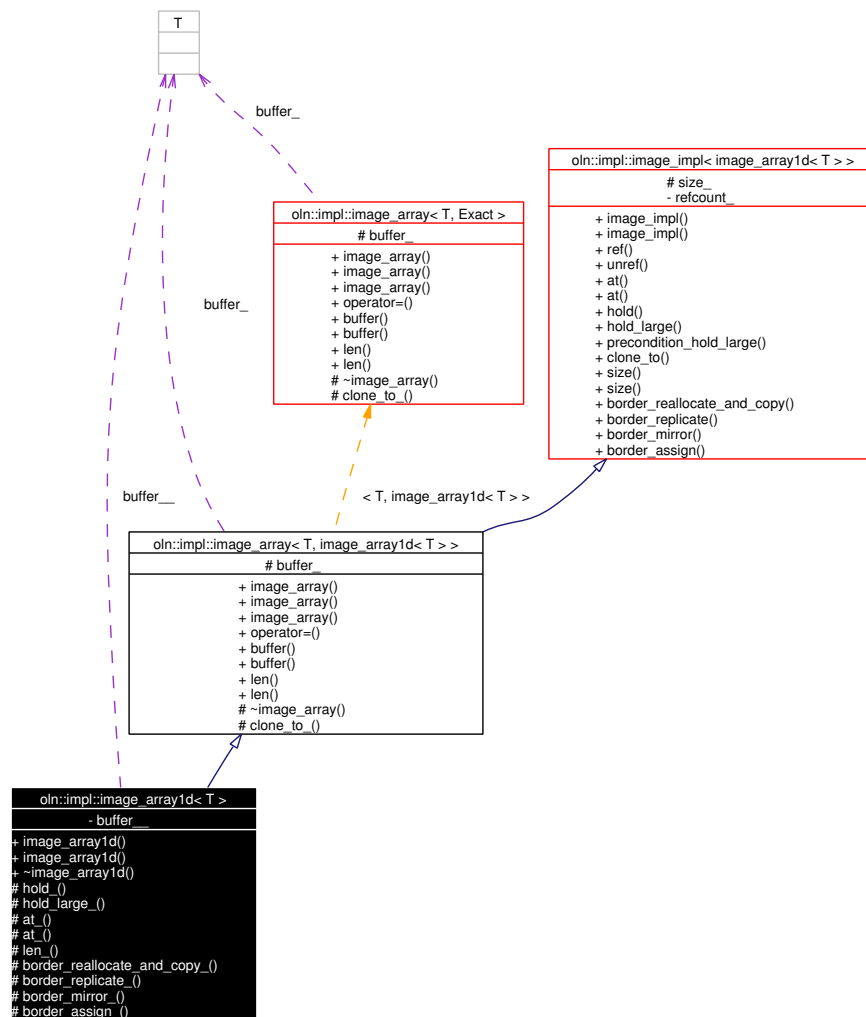
7.175 oln::impl::image_array1d< T > Class Template Reference

```
#include <image_array1d.hh>
```

Inheritance diagram for oln::impl::image_array1d< T >:



Collaboration diagram for oln::impl::image_array1d< T >:



Public Types

- typedef [image_array1d< T >](#) **self_type**
- typedef [image_array1d< T >](#) **exact_type**
- typedef impl_traits< exact_type >::point_type **point_type**
- typedef impl_traits< exact_type >::value_type **value_type**
- typedef impl_traits< exact_type >::size_type **size_type**
- typedef [image_array< T, image_array1d< T > >](#) **super_type**

Public Member Functions

- **image_array1d** (const size_type &s)

Protected Member Functions

- bool [hold_](#) (const point_type &p) const

Return true if `p` belongs to the image.

- `bool hold_large_ (const point_type &p) const`
Return true if `p` belongs to the image or the image border.
- `value_type & at_ (const point_type &p)`
Return a reference to the value stored at `p`.
- `value_type & at_ (coord col)`
Return a reference to the value stored at `col`.
- `size_t len_ (const size_type &s) const`
Return the total size of the data array.
- `void border_reallocate_and_copy_ (coord new_border, bool copy_border)`
Reallocate the border regarding to the value of `new_border`.
- `void border_replicate_ (void)`
The border points are all set to the value of the closest image point.
- `void border_mirror_ (void)`
The border points are set by mirroring the image edges.
- `void border_assign_ (value_type val)`
The border points are set to `val`.

Friends

- `class image_impl< image_array1d< T > >`
- `class image_array< T, image_array1d< T > >`

7.175.1 Detailed Description

`template<class T> class oln::impl::image_array1d< T >`

Data array implementation for `image1d`

Definition at line 78 of file `image_array1d.hh`.

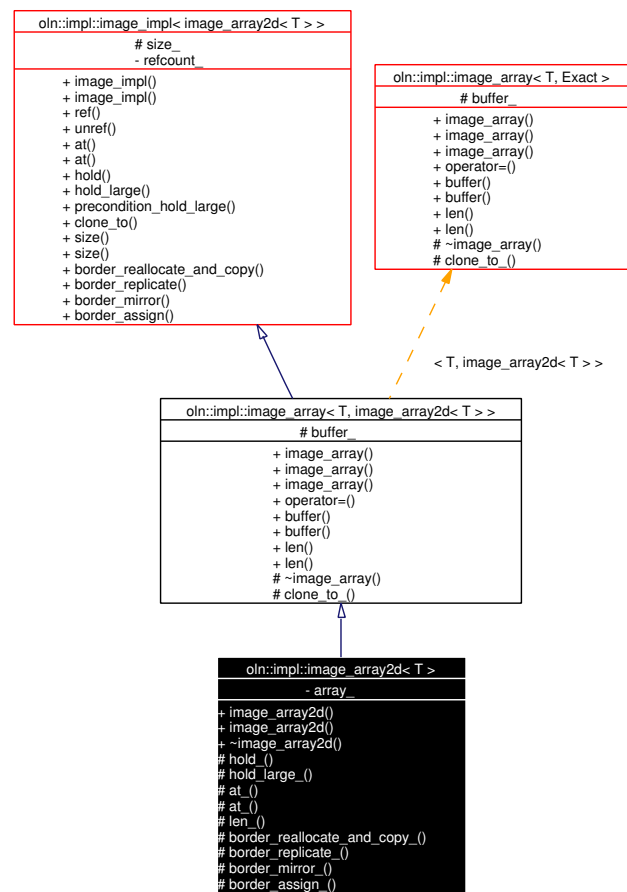
The documentation for this class was generated from the following file:

- `image_array1d.hh`

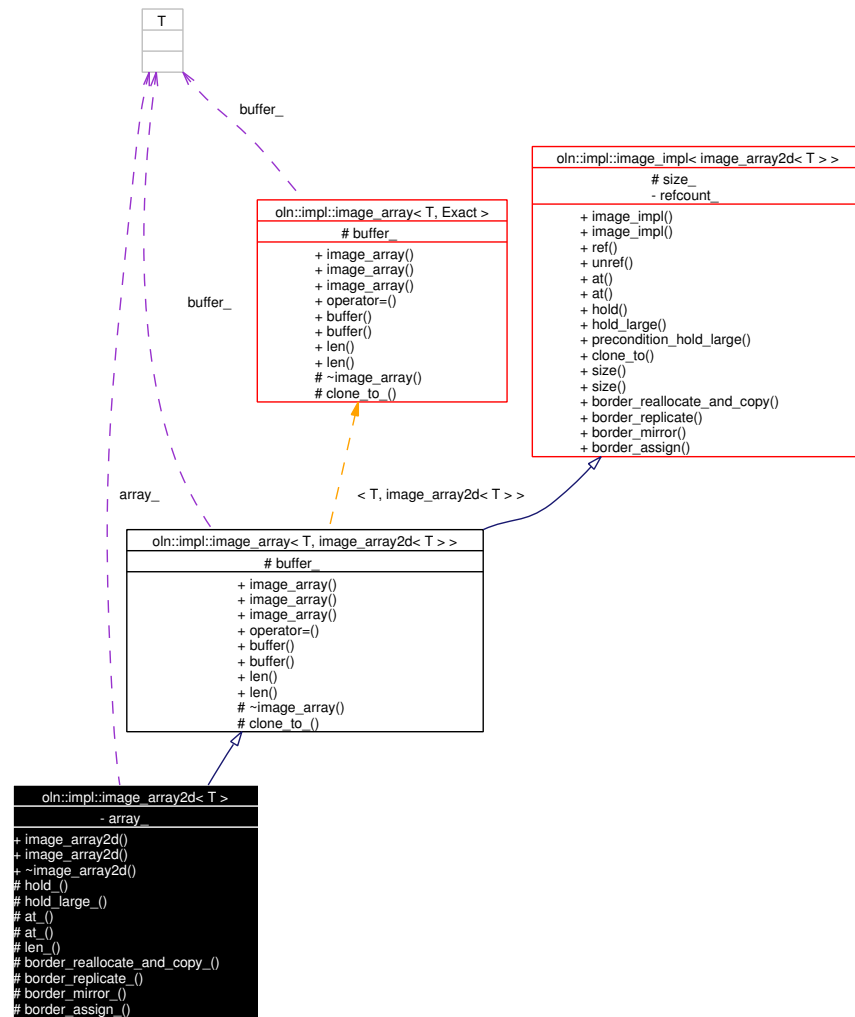
7.176 oln::impl::image_array2d< T > Class Template Reference

```
#include <image_array2d.hh>
```

Inheritance diagram for oln::impl::image_array2d< T >:



Collaboration diagram for oln::impl::image_array2d< T >:



Public Types

- typedef `image_array2d< T >` **self_type**
- typedef `image_array2d< T >` **exact_type**
- typedef `impl_traits< exact_type >::point_type` **point_type**
- typedef `impl_traits< exact_type >::value_type` **value_type**
- typedef `impl_traits< exact_type >::size_type` **size_type**
- typedef `image_array< T, image_array2d< T > >` **super_type**

Public Member Functions

- `image_array2d` (const size_type &s)

Protected Member Functions

- bool `hold_` (const point_type &p) const

Return true if p belongs to the image.

- bool [hold_large_](#) (const point_type &p) const
Return true if p belongs to the image or the image border.
- value_type & [at_](#) (const point_type &p)
Return a reference to the value stored at p.
- value_type & [at_](#) (coord row, coord col)
Return a reference to the value stored at row and col.
- size_t [len_](#) (const size_type &s) const
Return the total size of the data array.
- void [border_reallocate_and_copy_](#) (coord new_border, bool copy_border)
Reallocate the border regarding to the value of new_border.
- void [border_replicate_](#) (void)
The border points are all set to the value of the closest image point.
- void [border_mirror_](#) (void)
The border points are set by mirroring the image edges.
- void [border_assign_](#) (value_type val)
The border points are set to val.

Friends

- class **image_impl**< **image_array2d**< T > >
- class **image_array**< T, **image_array2d**< T > >

7.176.1 Detailed Description

template<class T> **class** oln::impl::image_array2d< T >

Data array implementation for [image2d](#)

Definition at line 99 of file image_array2d.hh.

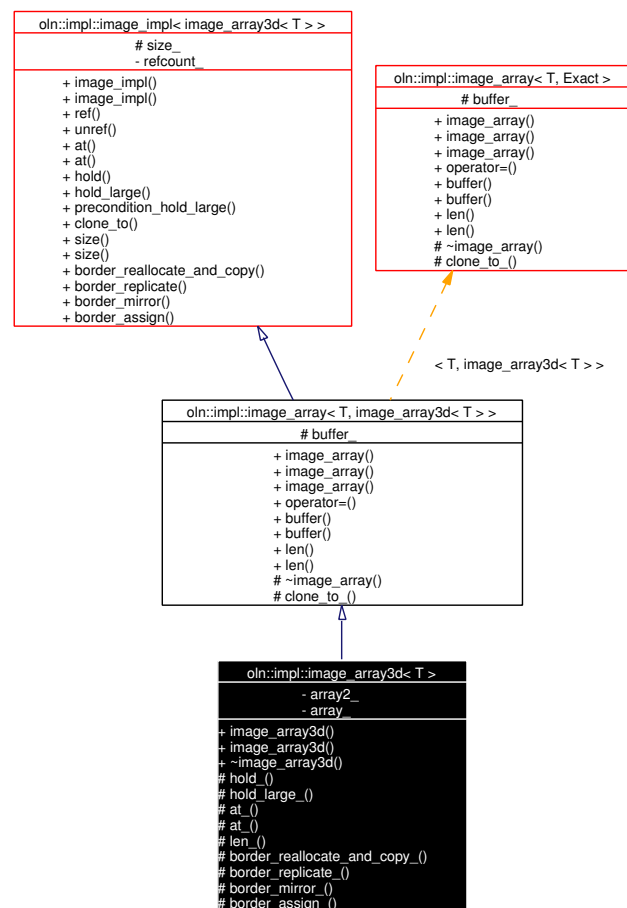
The documentation for this class was generated from the following file:

- image_array2d.hh

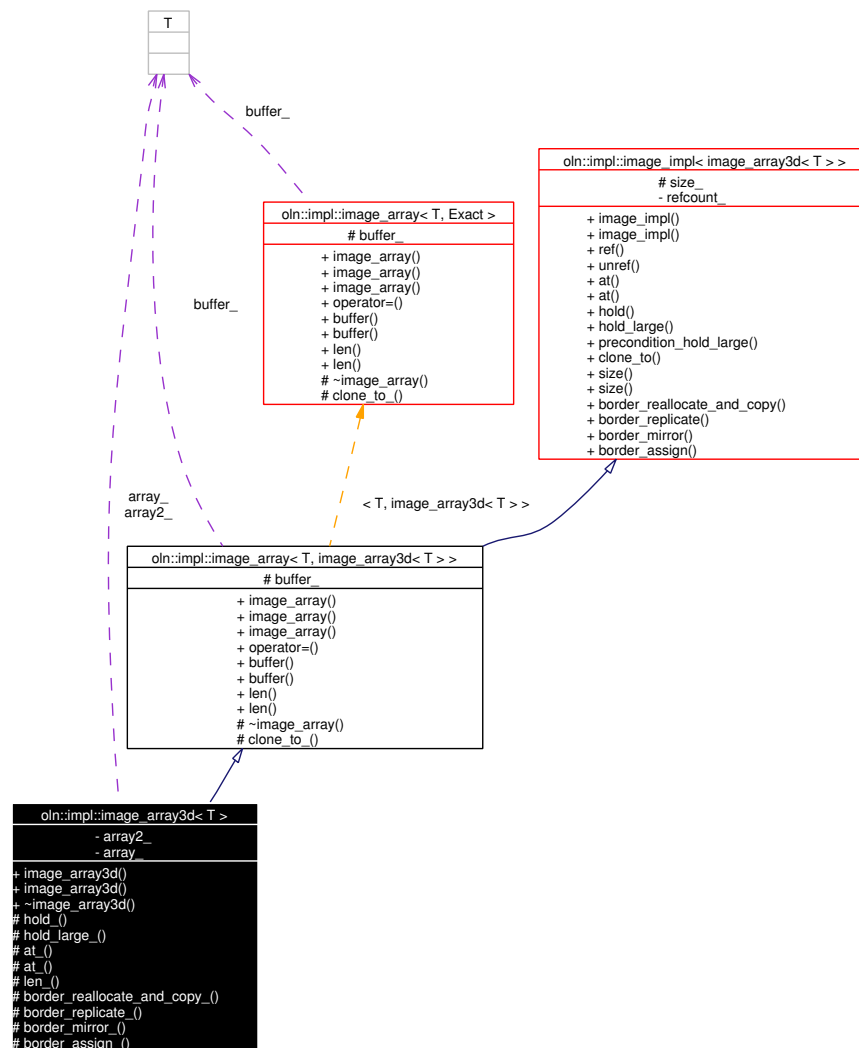
7.177 oln::impl::image_array3d< T > Class Template Reference

```
#include <image_array3d.hh>
```

Inheritance diagram for oln::impl::image_array3d< T >:



Collaboration diagram for oln::impl::image_array3d< T >:



Public Types

- typedef `image_array3d< T >` **self_type**
- typedef `image_array3d< T >` **exact_type**
- typedef `impl_traits< exact_type >::point_type` **point_type**
- typedef `impl_traits< exact_type >::value_type` **value_type**
- typedef `impl_traits< exact_type >::size_type` **size_type**
- typedef `image_array< T, image_array3d< T > >` **super_type**

Public Member Functions

- `image_array3d` (const `size_type` &s)

Protected Member Functions

- bool `hold_` (const `point_type` &p) const

Return true if p belongs to the image.

- bool [hold_large_](#) (const point_type &p) const
Return true if p belongs to the image or the image border.
- value_type & [at_](#) (const point_type &p)
Return a reference to the value stored at p.
- value_type & [at_](#) (coord slice, coord row, coord col)
Return a reference to the value stored at slice, row and col.
- size_t [len_](#) (const size_type &s) const
Return the total size of the data array.
- void [border_reallocate_and_copy_](#) (coord new_border, bool copy_border)
Reallocate the border regarding to the value of new_border.
- void [border_replicate_](#) (void)
The border points are all set to the value of the closest image point.
- void [border_mirror_](#) (void)
The border points are set by mirroring the image edges.
- void [border_assign_](#) (value_type val)
The border points are set to val.

Friends

- class **image_impl**< **image_array3d**< T > >
- class **image_array**< T, **image_array3d**< T > >

7.177.1 Detailed Description

template<class T> class oln::impl::image_array3d< T >

Data array implementation for [image3d](#)

Definition at line 113 of file image_array3d.hh.

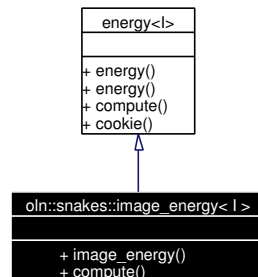
The documentation for this class was generated from the following file:

- image_array3d.hh

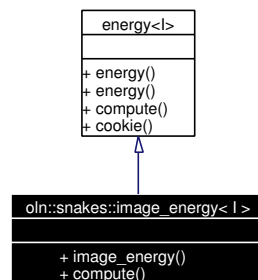
7.178 oln::snakes::image_energy< I > Class Template Reference

```
#include <energies.hh>
```

Inheritance diagram for oln::snakes::image_energy< I >:



Collaboration diagram for oln::snakes::image_energy< I >:



Public Types

- `typedef I image_type`

Public Member Functions

- `image_energy` (void *)

Static Public Member Functions

- `ntg::float_s compute` (const I &gradient, const `node< I >` &, const `node< I >` ¤t, const `node< I >` &)

7.178.1 Detailed Description

```
template<class I> class oln::snakes::image_energy< I >
```

Energy of the gradient.

The snake should follow the edge of the object. The higher the gradient is, the less the energy is.

Definition at line 133 of file energies.hh.

7.178.2 Member Function Documentation

7.178.2.1 `template<class I> ntg::float_s oln::snakes::image_energy< I >::compute (const I & gradient, const node< I > &, const node< I > & current, const node< I > &)`
[inline, static]

Return the energy.

The first arg is the gradient of the image; the 3 nodes are the previous, the current and the next node.

Reimplemented from [oln::snakes::energy](#)< I >.

Definition at line 67 of file energies.hxx.

```
71     {  
72         // FIXME: Add magic trick: if there is very little gradient difference,  
73         // don't pay too much attention to it.  
74         // If max_gradient < min_gradient + 5: max_gradient = min_gradient + 5  
75         return ntg_sup_val(oln_value_type(I)) - gradient[current];  
76     }
```

The documentation for this class was generated from the following files:

- energies.hh
- energies.hxx

7.179 oln::image_id< image1d< T, Exact > > Struct Template Reference

```
#include <image1d.hh>
```

Public Types

- typedef T **value_type**
- typedef mlc::exact_vt< [image1d](#)< T, Exact >, Exact >::ret **exact_type**
- typedef [impl::image_array1d](#)< T > **impl_type**
- typedef [point1d](#) **point_type**
- typedef [image1d_size](#) **size_type**
- enum { **dim** = 1 }

7.179.1 Detailed Description

template<class T, class Exact> struct oln::image_id< image1d< T, Exact > >

Helper class used by [image_traits](#) to retrieve the typedef associated to an image.

Definition at line 54 of file image1d.hh.

The documentation for this struct was generated from the following file:

- image1d.hh

7.180 oln::image_id< image2d< T, Exact > > Struct Template Reference

```
#include <image2d.hh>
```

Public Types

- typedef T **value_type**
- typedef mlc::exact_vt< [image2d](#)< T, Exact >, Exact >::ret **exact_type**
- typedef [impl::image_array2d](#)< T > **impl_type**
- typedef [point2d](#) **point_type**
- typedef [image2d_size](#) **size_type**
- enum { **dim** = 2 }

7.180.1 Detailed Description

template<class T, class Exact> struct oln::image_id< image2d< T, Exact > >

Helper class used by [image_traits](#) to retrieve the typedef associated to an image.

Definition at line 54 of file image2d.hh.

The documentation for this struct was generated from the following file:

- image2d.hh

7.181 oln::image_id< image3d< T, Exact > > Struct Template Reference

```
#include <image3d.hh>
```

Public Types

- typedef T **value_type**
- typedef mlc::exact_vt< [image3d](#)< T, Exact >, Exact >::**ret exact_type**
- typedef [impl::image_array3d](#)< T > **impl_type**
- typedef [point3d](#) **point_type**
- typedef [image3d_size](#) **size_type**
- enum { **dim** = 3 }

7.181.1 Detailed Description

template<class T, class Exact> struct oln::image_id< image3d< T, Exact > >

Helper class used by [image_traits](#) to retrieve the typedef associated to an image.

Definition at line 54 of file image3d.hh.

The documentation for this struct was generated from the following file:

- image3d.hh

7.182 oln::image_id< image< Dim, T, Impl, Exact > > Struct Template Reference

```
#include <image.hh>
```

Public Types

- typedef T **value_type**
- typedef Impl **impl_type**
- typedef mlc::exact_vt< [image](#)< Dim, T, Impl, Exact >, Exact >::ret **exact_type**
- enum { **dim** = Dim }

7.182.1 Detailed Description

template<unsigned Dim, class T, class Impl, class Exact> struct oln::image_id< image< Dim, T, Impl, Exact > >

Helper class used by [image_traits](#) to retrieve the typedef associated to an image.

Definition at line 51 of file core/image.hh.

The documentation for this struct was generated from the following file:

- core/image.hh

7.183 oln::image_id< morpher::color_morpher< I, Exact > > Struct Template Reference

Retrieve types and dimension of the color_morpher.

```
#include <color_morpher.hh>
```

Public Types

- typedef mlc::exact< I >::ret::impl_type [impl_type](#)
- typedef [ntg::type_traits](#)< typename mlc::exact< I >::ret::value_type >::comp_type [value_type](#)
- typedef mlc::exact_vt< [morpher::color_morpher](#)< I, Exact >, Exact >::ret exact_type
- typedef mlc::exact< I >::ret::point_type [point_type](#)
- typedef mlc::exact< I >::ret::iter_type [iter_type](#)
- enum { **dim** = I::dim }

7.183.1 Detailed Description

```
template<class I, class Exact> struct oln::image_id< morpher::color_morpher< I, Exact > >
```

Retrieve types and dimension of the color_morpher.

Parameters:

I The type of the decorated image.

Exact The exact type of the object.

Definition at line 51 of file color_morpher.hh.

7.183.2 Member Typedef Documentation

7.183.2.1 `template<class I, class Exact> typedef mlc::exact_vt<morpher::color_morpher<I, Exact>, Exact>::ret oln::image_id< morpher::color_morpher< I, Exact > >::exact_type`

<The value type of the decorated image. Here the component type of the source image value_type is retrieved.

Definition at line 63 of file color_morpher.hh.

7.183.2.2 `template<class I, class Exact> typedef mlc::exact< I >::ret::impl_type oln::image_id< morpher::color_morpher< I, Exact > >::impl_type`

<The Image dimension.

Definition at line 55 of file color_morpher.hh.

7.183.2.3 `template<class I, class Exact> typedef mlc::exact< I >::ret::point_type oln::image_id< morpher::color_morpher< I, Exact > >::point_type`

<Retrieve the exact type of the image. It depends on the value of Exact.

Definition at line 68 of file color_morpher.hh.

7.183.2.4 `template<class I, class Exact> typedef ntg::type_traits< typename mlc::exact< I >::ret::value_type >::comp_type oln::image_id< morpher::color_morpher< I, Exact > >::value_type`

<Underlying implementation.

Definition at line 57 of file color_morpher.hh.

The documentation for this struct was generated from the following file:

- color_morpher.hh

7.184 oln::image_id< morpher::iter_morpher< SrcType, IterType, Exact > > Struct Template Reference

Informations about the iter morpher.

```
#include <iter_morpher.hh>
```

Public Types

- typedef mlc::exact< SrcType >::ret::impl_type [impl_type](#)
Underlying implementation.
- typedef mlc::exact< SrcType >::ret::value_type [value_type](#)
The value type of the decorated image.
- typedef mlc::exact_vt< morpher::iter_morpher< SrcType, IterType, Exact >, Exact >::ret [exact_type](#)
Retrieve the exact type of the image.
- typedef mlc::exact< SrcType >::ret::point_type **point_type**
- typedef mlc::exact< SrcType >::ret::dpoint_type **dpoint_type**
- typedef mlc::exact< SrcType >::ret::size_type **size_type**
- typedef IterType **iter_type**
- enum { **dim** = SrcType::dim }

7.184.1 Detailed Description

```
template<class SrcType, class IterType, class Exact> struct oln::image_id< morpher::iter_morpher< SrcType, IterType, Exact > >
```

Informations about the iter morpher.

Definition at line 44 of file iter_morpher.hh.

The documentation for this struct was generated from the following file:

- iter_morpher.hh

7.185 `oln::image_id< morpher::piece_morpher< SrcType, Exact > >` Struct Template Reference

Informations about the piece morpher.

```
#include <piece_morpher.hh>
```

Public Types

- `typedef mlc::exact< SrcType >::ret::impl_type impl_type`
Underlying implementation.
- `typedef mlc::exact< SrcType >::ret::value_type value_type`
The value type of the decorated image.
- `typedef mlc::exact_vt< morpher::piece_morpher< SrcType, Exact >, Exact >::ret exact_type`
Retrieve the exact type of the image.
- `typedef mlc::exact< SrcType >::ret::point_type point_type`
- `typedef mlc::exact< SrcType >::ret::dpoint_type dpoint_type`
- `typedef mlc::exact< SrcType >::ret::size_type size_type`
- `typedef mlc::exact< SrcType >::ret::iter_type iter_type`
- `enum { dim = SrcType::dim }`

7.185.1 Detailed Description

```
template<class SrcType, class Exact> struct oln::image_id< morpher::piece_morpher< SrcType, Exact > >
```

Informations about the piece morpher.

Definition at line 57 of file `piece_morpher.hh`.

The documentation for this struct was generated from the following file:

- `piece_morpher.hh`

7.186 oln::image_id< morpher::slicing_morpher< const SrcType, Exact > > Struct Template Reference

Informations about the const slicing morpher.

```
#include <slicing_morpher.hh>
```

Public Types

- typedef mlc::exact_vt< [morpher::slicing_morpher](#)< SrcType, Exact >, Exact >::ret exact_type
Retrieve the exact type of the image.
- typedef [dim_traits](#)< dim, typename image_id< SrcType >::value_type, exact_type >::img_type **img_type**
- typedef image_id< img_type >::value_type **value_type**
- typedef [image_traits](#)< img_type >::point_type **point_type**
- typedef [image_traits](#)< img_type >::size_type **size_type**
- typedef [image_traits](#)< img_type >::impl_type **impl_type**
- enum { **dim** = SrcType::dim - 1 }

7.186.1 Detailed Description

template<class SrcType, class Exact> struct oln::image_id< morpher::slicing_morpher< const SrcType, Exact > >

Informations about the const slicing morpher.

Definition at line 100 of file slicing_morpher.hh.

The documentation for this struct was generated from the following file:

- slicing_morpher.hh

7.187 oln::image_id< morpher::slicing_morpher< SrcType, Exact > > Struct Template Reference

Informations about the slicing morpher.

```
#include <slicing_morpher.hh>
```

Public Types

- typedef mlc::exact_vt< [morpher::slicing_morpher](#)< SrcType, Exact >, Exact >::ret exact_type
Retrieve the exact type of the image.
- typedef [dim_traits](#)< dim, typename image_id< SrcType >::value_type, exact_type >::img_type **img_type**
- typedef image_id< img_type >::value_type **value_type**
- typedef image_id< img_type >::point_type **point_type**
- typedef image_id< img_type >::size_type **size_type**
- typedef image_id< img_type >::impl_type **impl_type**
- enum { **dim** = SrcType::dim - 1 }

7.187.1 Detailed Description

```
template<class SrcType, class Exact> struct oln::image_id< morpher::slicing_morpher< SrcType, Exact > >
```

Informations about the slicing morpher.

Definition at line 82 of file slicing_morpher.hh.

The documentation for this struct was generated from the following file:

- slicing_morpher.hh

7.188 oln::image_id< morpher::super_piece_morpher< SrcType, Exact > > Struct Template Reference

Informations about the super piece morpher.

```
#include <piece_morpher.hh>
```

Public Types

- typedef mlc::exact_vt< [morpher::super_piece_morpher](#)< SrcType, Exact >, Exact >::ret exact_type

Retrieve the exact type of the image.

7.188.1 Detailed Description

```
template<class SrcType, class Exact> struct oln::image_id< morpher::super_piece_morpher< SrcType, Exact > >
```

Informations about the super piece morpher.

Definition at line 46 of file piece_morpher.hh.

The documentation for this struct was generated from the following file:

- piece_morpher.hh

7.189 oln::image_id< morpher::super_slicing_morpher< const SrcType, Exact > > Struct Template Reference

Informations about the const super slicing morpher.

```
#include <slicing_morpher.hh>
```

Public Types

- typedef mlc::exact_vt< morpher::super_slicing_morpher< SrcType, Exact >, Exact >::ret exact_type
Retrieve the exact type of the image.
- typedef dim_traits< dim, typename image_id< SrcType >::value_type, exact_type >::img_type img_type
- typedef image_traits< img_type >::size_type size_type
- typedef image_traits< img_type >::impl_type impl_type
- enum { dim = SrcType::dim - 1 }

7.189.1 Detailed Description

```
template<class SrcType, class Exact> struct oln::image_id< morpher::super_slicing_morpher<
const SrcType, Exact > >
```

Informations about the const super slicing morpher.

Definition at line 64 of file slicing_morpher.hh.

The documentation for this struct was generated from the following file:

- slicing_morpher.hh

7.190 oln::image_id< morpher::super_slicing_morpher< SrcType, Exact > > Struct Template Reference

Informations about the super slicing morpher.

```
#include <slicing_morpher.hh>
```

Public Types

- typedef mlc::exact_vt< [morpher::super_slicing_morpher](#)< SrcType, Exact >, Exact >::ret_exact_type
Retrieve the exact type of the image.
- typedef [dim_traits](#)< dim, typename image_id< SrcType >::value_type, exact_type >::img_type **img_type**
- typedef [image_traits](#)< img_type >::size_type **size_type**
- typedef [image_traits](#)< img_type >::impl_type **impl_type**
- enum { **dim** = SrcType::dim - 1 }

7.190.1 Detailed Description

```
template<class SrcType, class Exact> struct oln::image_id< morpher::super_slicing_morpher< SrcType, Exact > >
```

Informations about the super slicing morpher.

Definition at line 46 of file slicing_morpher.hh.

The documentation for this struct was generated from the following file:

- slicing_morpher.hh

7.191 oln::image_id< oln::morpher::subq_morpher< SrcType, N, Exact > > Struct Template Reference

```
#include <subq_morpher.hh>
```

Public Types

- typedef mlc::exact< SrcType >::ret::impl_type [impl_type](#)
- typedef [oln::morpher::color_mute](#)< typename mlc::exact< SrcType >::ret::value_type, N >::ret value_type
- typedef mlc::exact_vt< [oln::morpher::subq_morpher](#)< SrcType, N, Exact >, Exact >::ret exact_type
- typedef mlc::exact< SrcType >::ret::point_type [point_type](#)
- enum { **dim** = SrcType::dim }

7.191.1 Detailed Description

```
template<class SrcType, unsigned N, class Exact> struct oln::image_id< oln::morpher::subq_morpher< SrcType, N, Exact > >
```

Retrieve types and dimension of the subq_morpher.

Parameters:

SrcType Input type decorated.

N The new number of bits by component.

Exact The exact type of the morpher.

Definition at line 86 of file subq_morpher.hh.

7.191.2 Member Typedef Documentation

7.191.2.1 `template<class SrcType, unsigned N, class Exact> typedef mlc::exact_vt<oln::morpher::subq_morpher<SrcType, N, Exact>, Exact>::ret oln::image_id<oln::morpher::subq_morpher< SrcType, N, Exact > >::exact_type`

<The modified value type.

Definition at line 95 of file subq_morpher.hh.

7.191.2.2 `template<class SrcType, unsigned N, class Exact> typedef mlc::exact< SrcType >::ret::impl_type oln::image_id< oln::morpher::subq_morpher< SrcType, N, Exact > >::impl_type`

<The image dimension.

Definition at line 90 of file subq_morpher.hh.

7.191.2.3 `template<class SrcType, unsigned N, class Exact> typedef oln::morpher::color_mute<typename mlc::exact< SrcType >::ret::value_type, N>::ret oln::image_id< oln::morpher::subq_morpher< SrcType, N, Exact > >::value_type`

<The underlying implementation.

Definition at line 92 of file subq_morpher.hh.

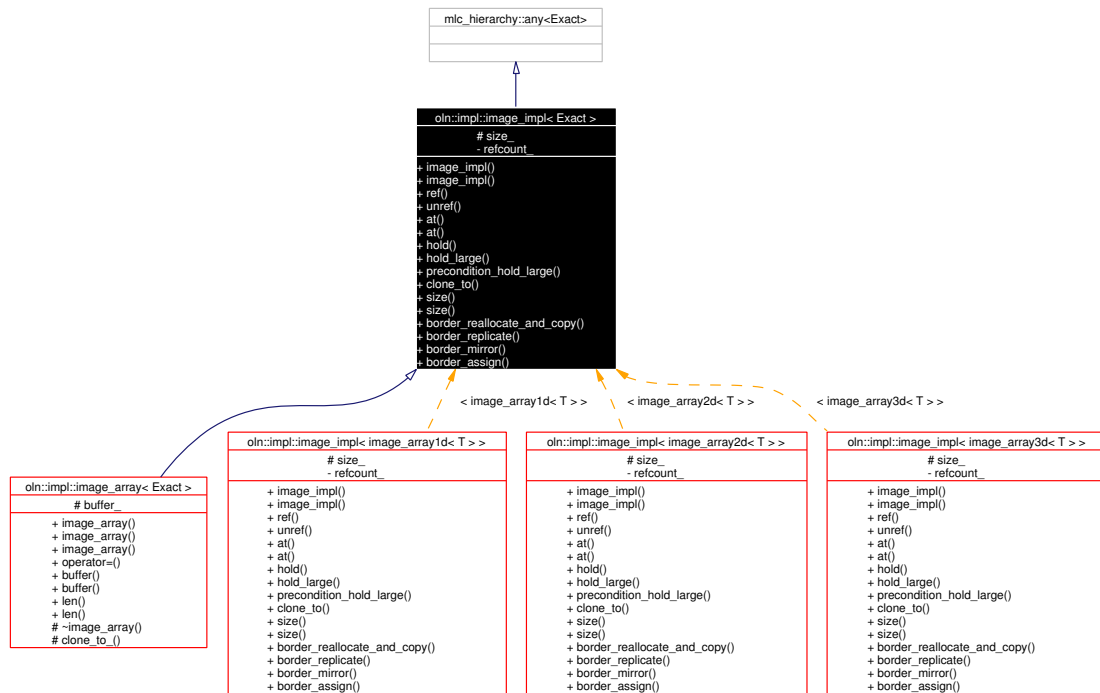
The documentation for this struct was generated from the following file:

- subq_morpher.hh

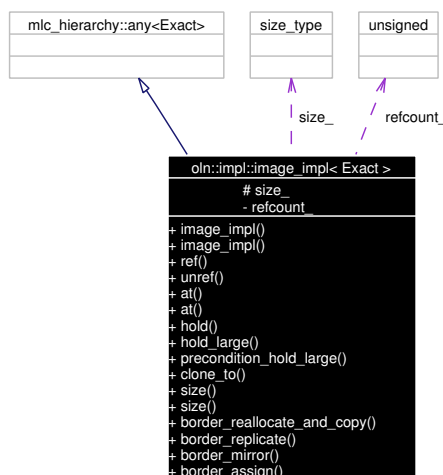
7.192 oln::impl::image_impl< Exact > Class Template Reference

```
#include <image_impl.hh>
```

Inheritance diagram for oln::impl::image_impl< Exact >:



Collaboration diagram for oln::impl::image_impl< Exact >:



Public Types

- `typedef impl_traits< Exact >::point_type point_type`

- typedef impl_traits< Exact >::value_type **value_type**
- typedef impl_traits< Exact >::size_type **size_type**
- typedef Exact **exact_type**

Public Member Functions

- **image_impl** (const size_type s)
- void **ref** () const
Notice that there is a new reference to the object.
- void **unref** () const
Notice that there a reference to the object has disappeared. When there is no reference left, the object is deleted.
- const value_type & **at** (const point_type &p) const
Return a constant reference to the value stored at p.
- value_type & **at** (const point_type &p)
Return a reference to the value stored at p.
- bool **hold** (const point_type &p) const
Return true if p belongs to the image.
- bool **hold_large** (const point_type &p) const
Return true if p belongs to the image or the image border.
- void **precondition_hold_large** (const point_type &p) const
Use the function for debugging purpose.
- void **clone_to** (exact_type *output_data) const
Perform a deep copy from the data array to output_data.
- const size_type & **size** () const
Return the number of point in the image.
- size_type & **size** ()
- void **border_reallocate_and_copy** (coord new_border, bool copy_border)
Reallocate the border regarding to the value of new_border.
- void **border_replicate** (void)
The border points are all set to the value of the closest image point.
- void **border_mirror** (void)
The border points are set by mirroring the image edges.
- void **border_assign** (value_type val)
The border points are set to val.

Protected Attributes

- size_type **size_**

7.192.1 Detailed Description

template<class Exact> class oln::impl::image_impl< Exact >

Data array implementation for image

Definition at line 71 of file image_impl.hh.

The documentation for this class was generated from the following file:

- image_impl.hh

7.193 oln::io::internal::image_reader< F, I > Struct Template Reference

Read an image of type reader_id.

```
#include <image_base.hh>
```

Static Public Member Functions

- const std::string & [name](#) ()
Do nothing because of the type of reader.
- bool [knows_ext](#) (const std::string &)
Do nothing because of the type of reader.
- bool [read](#) (std::istream &, I &)

7.193.1 Detailed Description

```
template<reader_id F, class I> struct oln::io::internal::image_reader< F, I >
```

Read an image of type reader_id.

Parameters:

F The reader identifier.

In fact, do nothing because of the type of reader. Used when an errors occurs on others types of reader.

See also:

reader_id

Definition at line 63 of file image_base.hh.

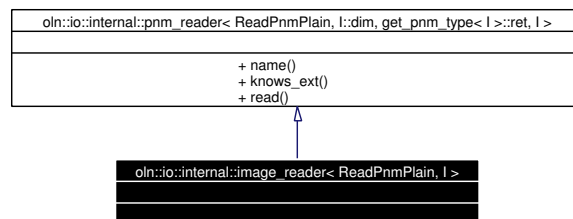
The documentation for this struct was generated from the following file:

- image_base.hh

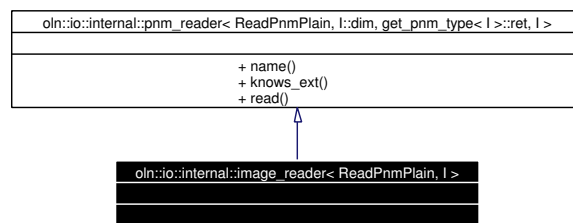
7.194 oln::io::internal::image_reader< ReadPnmPlain, I > Struct Template Reference

```
#include <pnm_read.hh>
```

Inheritance diagram for oln::io::internal::image_reader< ReadPnmPlain, I >:



Collaboration diagram for oln::io::internal::image_reader< ReadPnmPlain, I >:



7.194.1 Detailed Description

template<class I> struct oln::io::internal::image_reader< ReadPnmPlain, I >

Specialized version for ReadPnmPlain

Definition at line 86 of file pnm_read.hh.

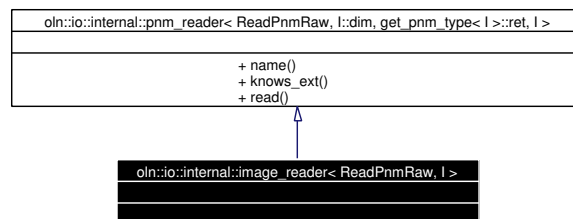
The documentation for this struct was generated from the following file:

- pnm_read.hh

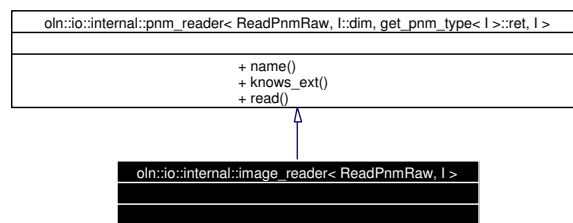
7.195 oln::io::internal::image_reader< ReadPnmRaw, I > Struct Template Reference

```
#include <pnm_read.hh>
```

Inheritance diagram for oln::io::internal::image_reader< ReadPnmRaw, I >:



Collaboration diagram for oln::io::internal::image_reader< ReadPnmRaw, I >:



7.195.1 Detailed Description

template<class I> struct oln::io::internal::image_reader< ReadPnmRaw, I >

Specialized version for ReadPnmRaw

Definition at line 96 of file pnm_read.hh.

The documentation for this struct was generated from the following file:

- pnm_read.hh

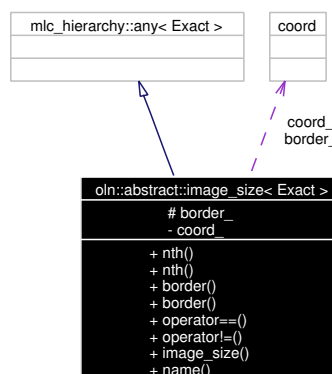
7.196 oln::abstract::image_size< Exact > Struct Template Reference

```
#include <image_size.hh>
```

Inheritance diagram for oln::abstract::image_size< Exact >:



Collaboration diagram for oln::abstract::image_size< Exact >:



Public Types

- enum { **dim** = image_size_traits<Exact>::dim }

Public Member Functions

- [coord nth](#) (unsigned n) const
Return the number of coordinates in the nth section of the image.
- [coord & nth](#) (unsigned n)
Return a reference to the number of coordinates in the nth section of the image.
- [coord border](#) () const
Return the value border width of the current image.
- [coord & border](#) ()
Return a reference to the border width of the current image.
- template<class S> bool [operator==](#) (const [image_size](#)< S > &size) const
Test if two images have compatible size.
- template<class S> bool [operator!=](#) (const [image_size](#)< S > &size) const
Test if two images do not have compatible size.

Static Public Member Functions

- std::string [name](#) ()

Protected Attributes

- [coord border_](#)

7.196.1 Detailed Description

template<class Exact> struct oln::abstract::image_size< Exact >

The class that defines the image size according to its dimension.

Definition at line 76 of file image_size.hh.

7.196.2 Member Function Documentation

7.196.2.1 template<class Exact> template<class S> bool [olin::abstract::image_size](#)< Exact >::operator!= (const [image_size](#)< S > &size) const [inline]

Test if two images do not have compatible size.

Returns:

False if the two images have compatible size, true otherwise.

Definition at line 144 of file image_size.hh.

```

145     {
146         for (unsigned i = 0; i < dim; ++i)
147             if (coord_[i] != size.coord_[i])
148                 return true;
149         return false;
150     }

```

7.196.2.2 `template<class Exact> template<class S> bool oln::abstract::image_size< Exact >::operator==(const image_size< S > & size) const` `[inline]`

Test if two images have compatible size.

Returns:

True if the two images have compatible size, false otherwise.

Definition at line 129 of file image_size.hh.

```

130     {
131         for (unsigned i = 0; i < dim; ++i)
132             if (coord_[i] != size.coord_[i])
133                 return false;
134         return true;
135     }

```

7.196.3 Member Data Documentation

7.196.3.1 `template<class Exact> coord oln::abstract::image_size< Exact >::border_` `[protected]`

`border_` represents the width of the image border such a mechanism allow algorithm to perform operation on the points at the edge of the image as if they were normally surrounded by points

Definition at line 172 of file image_size.hh.

The documentation for this struct was generated from the following file:

- image_size.hh

7.197 oln::image_size_traits< abstract::image_size< Exact > > Struct Template Reference

```
#include <image_size.hh>
```

7.197.1 Detailed Description

template<class Exact> struct oln::image_size_traits< abstract::image_size< Exact > >

A class specialized for each image type which gives the dimension of the template parameter.

Definition at line 61 of file image_size.hh.

The documentation for this struct was generated from the following file:

- image_size.hh

7.198 oln::image_size_traits< image1d_size > Struct Template Reference

```
#include <image1d_size.hh>
```

Public Types

- enum { **dim** = 1 }

7.198.1 Detailed Description

template<> struct oln::image_size_traits< image1d_size >

The specialized version for [image1d_size](#).

Definition at line 45 of file image1d_size.hh.

The documentation for this struct was generated from the following file:

- image1d_size.hh

7.199 oln::image_size_traits< image2d_size > Struct Template Reference

```
#include <image2d_size.hh>
```

Public Types

- enum { **dim** = 2 }

7.199.1 Detailed Description

template<> struct oln::image_size_traits< image2d_size >

The specialized version for [image2d_size](#).

Definition at line 45 of file image2d_size.hh.

The documentation for this struct was generated from the following file:

- image2d_size.hh

7.200 oln::image_size_traits< image3d_size > Struct Template Reference

```
#include <image3d_size.hh>
```

Public Types

- enum { **dim** = 3 }

7.200.1 Detailed Description

template<> struct oln::image_size_traits< image3d_size >

The specialized version for [image3d_size](#).

Definition at line 44 of file image3d_size.hh.

The documentation for this struct was generated from the following file:

- image3d_size.hh

7.201 oln::image_traits Class Reference

```
#include <image.hh>
```

Inheritance diagram for oln::image_traits:



7.201.1 Detailed Description

A helping structure to find the exact_type of a given class.

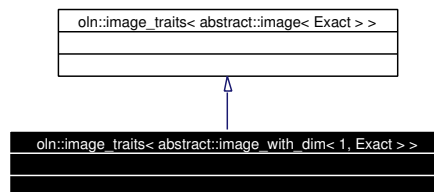
The documentation for this class was generated from the following file:

- core/abstract/image.hh

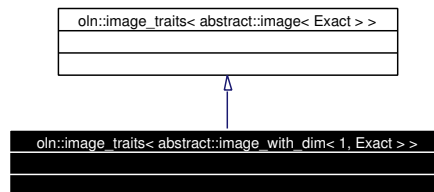
7.202 oln::image_traits< abstract::image_with_dim< 1, Exact > > Struct Template Reference

```
#include <image_with_dim.hh>
```

Inheritance diagram for oln::image_traits< abstract::image_with_dim< 1, Exact > >:



Collaboration diagram for oln::image_traits< abstract::image_with_dim< 1, Exact > >:



Public Types

- typedef [point1d](#) [point_type](#)
- typedef [dpoint1d](#) [dpoint_type](#)
- typedef [fwd_iter1d](#)< [mlc::final](#) > [iter_type](#)
- typedef [fwd_iter1d](#)< [mlc::final](#) > [fwd_iter_type](#)
- typedef [bkd_iter1d](#)< [mlc::final](#) > [bkd_iter_type](#)
- typedef [image1d_size](#) [size_type](#)
- enum { **dim** = 1 }

7.202.1 Detailed Description

```
template<class Exact> struct oln::image_traits< abstract::image_with_dim< 1, Exact > >
```

The specialized version for 1d image.

Definition at line 78 of file image_with_dim.hh.

7.202.2 Member Typedef Documentation

7.202.2.1 template<class Exact> typedef [bkd_iter1d](#)<[mlc::final](#)> [oln::image_traits](#)<
abstract::image_with_dim< 1, Exact > >::bkd_iter_type

Image1d bkd_iter_type is bkd_iter1d<mlc::final>.

Definition at line 93 of file image_with_dim.hh.

7.202.2.2 `template<class Exact> typedef dpoint1d oln::image_traits<
abstract::image_with_dim< 1, Exact > >::dpoint_type`

Image1d dpoint_type is [dpoint1d](#).

Definition at line 87 of file image_with_dim.hh.

7.202.2.3 `template<class Exact> typedef fwd_iter1d<mlc::final> oln::image_traits<
abstract::image_with_dim< 1, Exact > >::fwd_iter_type`

Image1d fwd_iter_type is fwd_iter1d<mlc::final>.

Definition at line 91 of file image_with_dim.hh.

7.202.2.4 `template<class Exact> typedef fwd_iter1d<mlc::final> oln::image_traits<
abstract::image_with_dim< 1, Exact > >::iter_type`

Image1d iter_type is fwd_iter1d<mlc::final>.

Definition at line 89 of file image_with_dim.hh.

7.202.2.5 `template<class Exact> typedef point1d oln::image_traits< abstract::image_with_dim<
1, Exact > >::point_type`

Image1d point_type is [point1d](#).

Definition at line 85 of file image_with_dim.hh.

7.202.2.6 `template<class Exact> typedef image1d_size oln::image_traits<
abstract::image_with_dim< 1, Exact > >::size_type`

Image1d size_type is [image1d_size](#).

Definition at line 95 of file image_with_dim.hh.

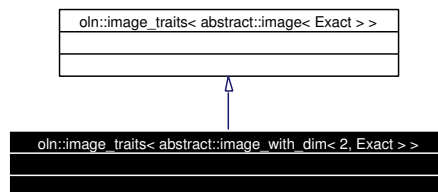
The documentation for this struct was generated from the following file:

- image_with_dim.hh

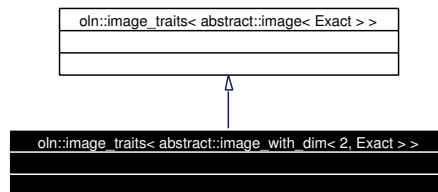
7.203 oln::image_traits< abstract::image_with_dim< 2, Exact > > Struct Template Reference

```
#include <image_with_dim.hh>
```

Inheritance diagram for oln::image_traits< abstract::image_with_dim< 2, Exact > >:



Collaboration diagram for oln::image_traits< abstract::image_with_dim< 2, Exact > >:



Public Types

- typedef [point2d](#) [point_type](#)
- typedef [dpoint2d](#) [dpoint_type](#)
- typedef [fwd_iter2d](#)< mlc::final > [iter_type](#)
- typedef [fwd_iter2d](#)< mlc::final > [fwd_iter_type](#)
- typedef [bkd_iter2d](#)< mlc::final > [bkd_iter_type](#)
- typedef [image2d_size](#) [size_type](#)
- enum { **dim** = 2 }

7.203.1 Detailed Description

```
template<class Exact> struct oln::image_traits< abstract::image_with_dim< 2, Exact > >
```

The specialized version for 2d image.

Definition at line 106 of file image_with_dim.hh.

7.203.2 Member Typedef Documentation

7.203.2.1 template<class Exact> typedef [bkd_iter2d](#)<mlc::final> [oln::image_traits](#)<
abstract::image_with_dim< 2, Exact > >::bkd_iter_type

Image2d bkd_iter_type is bkd_iter2d<mlc::final>.

Definition at line 120 of file image_with_dim.hh.

7.203.2.2 `template<class Exact> typedef dpoint2d oln::image_traits<
abstract::image_with_dim< 2, Exact > >::dpoint_type`

Image2d dpoint_type is [dpoint2d](#).

Definition at line 114 of file image_with_dim.hh.

7.203.2.3 `template<class Exact> typedef fwd_iter2d<mlc::final> oln::image_traits<
abstract::image_with_dim< 2, Exact > >::fwd_iter_type`

Image2d fwd_iter_type is fwd_iter2d<mlc::final>.

Definition at line 118 of file image_with_dim.hh.

7.203.2.4 `template<class Exact> typedef fwd_iter2d<mlc::final> oln::image_traits<
abstract::image_with_dim< 2, Exact > >::iter_type`

Image2d iter_type is fwd_iter2d<mlc::final>.

Definition at line 116 of file image_with_dim.hh.

7.203.2.5 `template<class Exact> typedef point2d oln::image_traits< abstract::image_with_dim<
2, Exact > >::point_type`

Imaged2 point_type is [point2d](#).

Definition at line 112 of file image_with_dim.hh.

7.203.2.6 `template<class Exact> typedef image2d_size oln::image_traits<
abstract::image_with_dim< 2, Exact > >::size_type`

Image2d size_type is [image2d_size](#).

Definition at line 122 of file image_with_dim.hh.

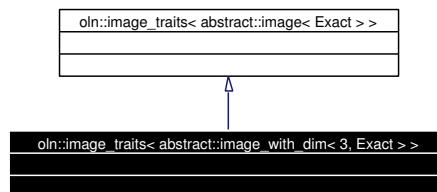
The documentation for this struct was generated from the following file:

- image_with_dim.hh

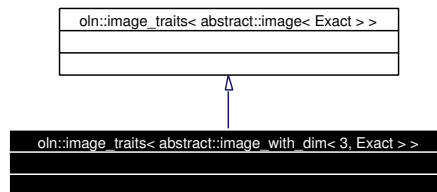
7.204 oln::image_traits< abstract::image_with_dim< 3, Exact > > Struct Template Reference

```
#include <image_with_dim.hh>
```

Inheritance diagram for oln::image_traits< abstract::image_with_dim< 3, Exact > >:



Collaboration diagram for oln::image_traits< abstract::image_with_dim< 3, Exact > >:



Public Types

- typedef [point3d](#) [point_type](#)
- typedef [dpoint3d](#) [dpoint_type](#)
- typedef [fwd_iter3d](#)< [mlc::final](#) > [iter_type](#)
- typedef [fwd_iter3d](#)< [mlc::final](#) > [fwd_iter_type](#)
- typedef [bkd_iter3d](#)< [mlc::final](#) > [bkd_iter_type](#)
- typedef [image3d_size](#) [size_type](#)
- enum { **dim** = 3 }

7.204.1 Detailed Description

```
template<class Exact> struct oln::image_traits< abstract::image_with_dim< 3, Exact > >
```

The specialized version for 3d image.

Definition at line 131 of file image_with_dim.hh.

7.204.2 Member Typedef Documentation

7.204.2.1 template<class Exact> typedef [bkd_iter3d](#)<[mlc::final](#)> [oln::image_traits](#)<
abstract::image_with_dim< 3, Exact > >::bkd_iter_type

Image3d bkd_iter_type is bkd_iter3d<mlc::final>.

Definition at line 145 of file image_with_dim.hh.

7.204.2.2 `template<class Exact> typedef dpoint3d oln::image_traits<
abstract::image_with_dim< 3, Exact > >::dpoint_type`

Image3d dpoint_type is [dpoint3d](#).

Definition at line 139 of file image_with_dim.hh.

7.204.2.3 `template<class Exact> typedef fwd_iter3d<mlc::final> oln::image_traits<
abstract::image_with_dim< 3, Exact > >::fwd_iter_type`

Image3d fwd_iter_type is fwd_iter3d<mlc::final>.

Definition at line 143 of file image_with_dim.hh.

7.204.2.4 `template<class Exact> typedef fwd_iter3d<mlc::final> oln::image_traits<
abstract::image_with_dim< 3, Exact > >::iter_type`

Image3d iter_type is fwd_iter3d<mlc::final>.

Definition at line 141 of file image_with_dim.hh.

7.204.2.5 `template<class Exact> typedef point3d oln::image_traits< abstract::image_with_dim<
3, Exact > >::point_type`

Image3d point_type is [point3d](#).

Definition at line 137 of file image_with_dim.hh.

7.204.2.6 `template<class Exact> typedef image3d_size oln::image_traits<
abstract::image_with_dim< 3, Exact > >::size_type`

Image3d size_type is [image3d_size](#).

Definition at line 147 of file image_with_dim.hh.

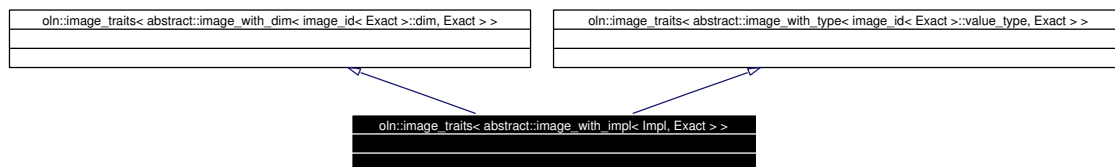
The documentation for this struct was generated from the following file:

- image_with_dim.hh

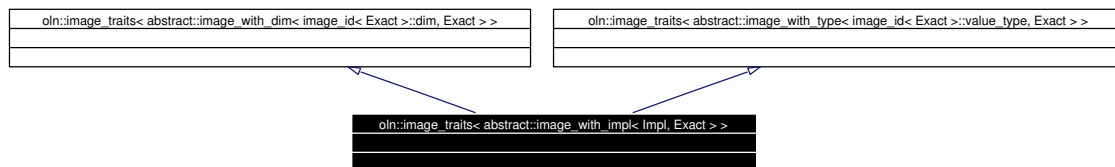
7.205 oln::image_traits< abstract::image_with_impl< Impl, Exact > > Struct Template Reference

```
#include <image_with_impl.hh>
```

Inheritance diagram for oln::image_traits< abstract::image_with_impl< Impl, Exact > >:



Collaboration diagram for oln::image_traits< abstract::image_with_impl< Impl, Exact > >:



Public Types

- typedef Impl **impl_type**

7.205.1 Detailed Description

template<class Impl, class Exact> struct oln::image_traits< abstract::image_with_impl< Impl, Exact > >

The specialized version for image_with_impl, give the impl_type of an image.

Definition at line 52 of file image_with_impl.hh.

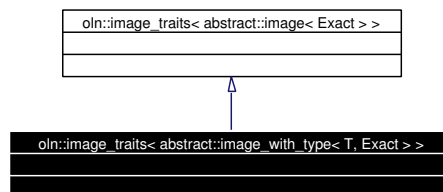
The documentation for this struct was generated from the following file:

- image_with_impl.hh

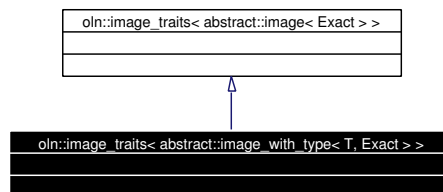
7.206 oln::image_traits< abstract::image_with_type< T, Exact > > Struct Template Reference

```
#include <image_with_type.hh>
```

Inheritance diagram for oln::image_traits< abstract::image_with_type< T, Exact > >:



Collaboration diagram for oln::image_traits< abstract::image_with_type< T, Exact > >:



Public Types

- typedef T value_type

7.206.1 Detailed Description

```
template<class T, class Exact> struct oln::image_traits< abstract::image_with_type< T, Exact > >
```

The specialized version for image_with_type

Warning:

This class may change, prefer the macro oln_value_type(I) to retrieve the value_type of an image.

Definition at line 55 of file image_with_type.hh.

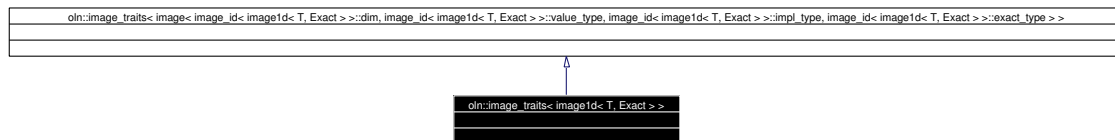
The documentation for this struct was generated from the following file:

- image_with_type.hh

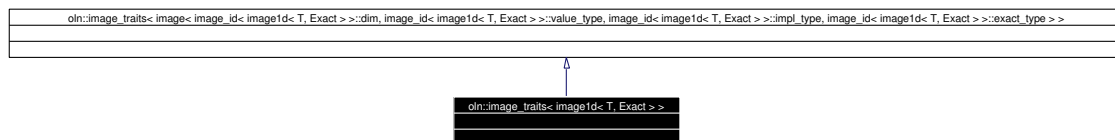
7.207 oln::image_traits< image1d< T, Exact > > Struct Template Reference

```
#include <image1d.hh>
```

Inheritance diagram for oln::image_traits< image1d< T, Exact > >:



Collaboration diagram for oln::image_traits< image1d< T, Exact > >:



7.207.1 Detailed Description

```
template<class T, class Exact> struct oln::image_traits< image1d< T, Exact > >
```

Helper class usefull to retrieve all the type relative to an image.

Definition at line 71 of file image1d.hh.

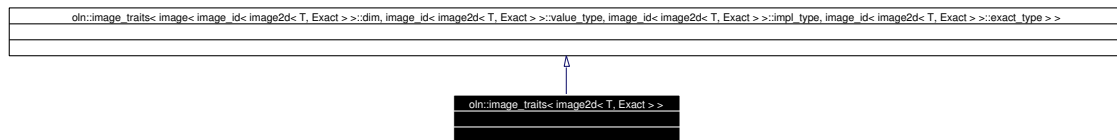
The documentation for this struct was generated from the following file:

- image1d.hh

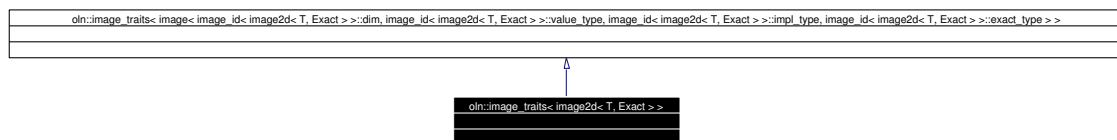
7.208 oln::image_traits< image2d< T, Exact > > Struct Template Reference

```
#include <image2d.hh>
```

Inheritance diagram for oln::image_traits< image2d< T, Exact > >:



Collaboration diagram for oln::image_traits< image2d< T, Exact > >:



7.208.1 Detailed Description

```
template<class T, class Exact> struct oln::image_traits< image2d< T, Exact > >
```

Helper class usefull to retrieve all the type relative to an image.

Definition at line 71 of file image2d.hh.

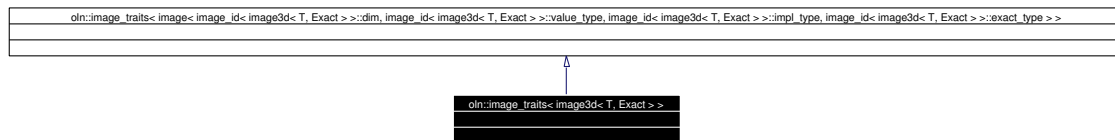
The documentation for this struct was generated from the following file:

- image2d.hh

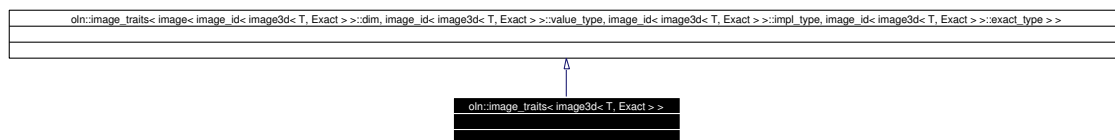
7.209 oln::image_traits< image3d< T, Exact > > Struct Template Reference

```
#include <image3d.hh>
```

Inheritance diagram for oln::image_traits< image3d< T, Exact > >:



Collaboration diagram for oln::image_traits< image3d< T, Exact > >:



7.209.1 Detailed Description

template<class T, class Exact> struct oln::image_traits< image3d< T, Exact > >

Helper class usefull to retrieve all the type relative to an image.

Definition at line 71 of file image3d.hh.

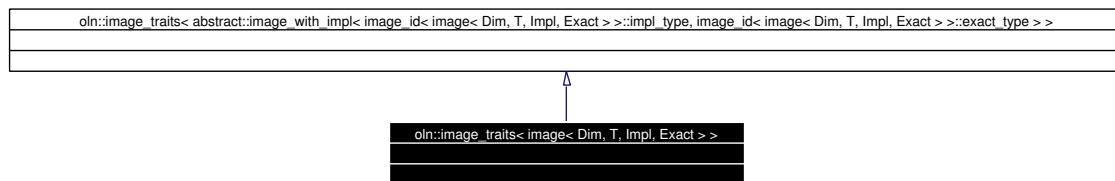
The documentation for this struct was generated from the following file:

- image3d.hh

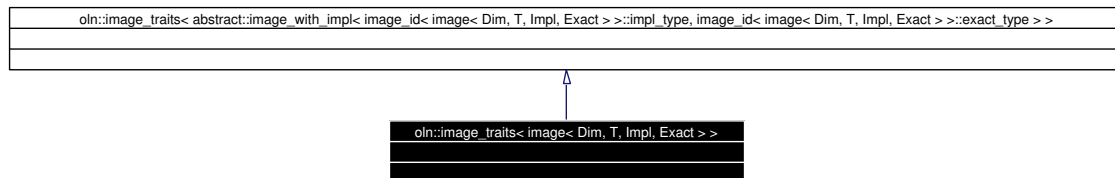
7.210 oln::image_traits< image< Dim, T, Impl, Exact > > Struct Template Reference

```
#include <image.hh>
```

Inheritance diagram for oln::image_traits< image< Dim, T, Impl, Exact > >:



Collaboration diagram for oln::image_traits< image< Dim, T, Impl, Exact > >:



7.210.1 Detailed Description

template<unsigned Dim, class T, class Impl, class Exact> struct oln::image_traits< image< Dim, T, Impl, Exact > >

Helper class usefull to retrieve all the type relative to an image.

Definition at line 66 of file core/image.hh.

The documentation for this struct was generated from the following file:

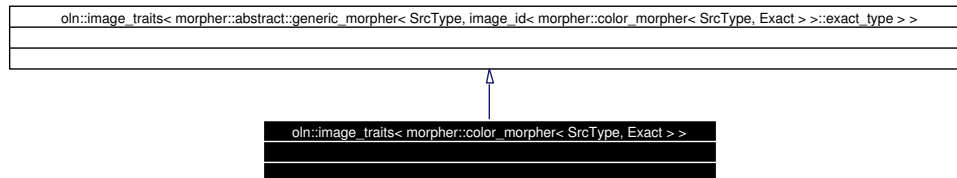
- core/image.hh

7.211 oln::image_traits< morpher::color_morpher< SrcType, Exact > > Struct Template Reference

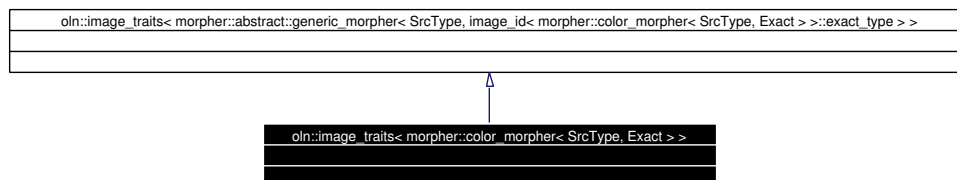
Specialized version for color_morpher.

```
#include <color_morpher.hh>
```

Inheritance diagram for oln::image_traits< morpher::color_morpher< SrcType, Exact > >:



Collaboration diagram for oln::image_traits< morpher::color_morpher< SrcType, Exact > >:



7.211.1 Detailed Description

template<class SrcType, class Exact> struct oln::image_traits< morpher::color_morpher< SrcType, Exact > >

Specialized version for color_morpher.

Parameters:

SrcType The type of the decorated image.

Exact The exact type of the object.

Definition at line 80 of file color_morpher.hh.

The documentation for this struct was generated from the following file:

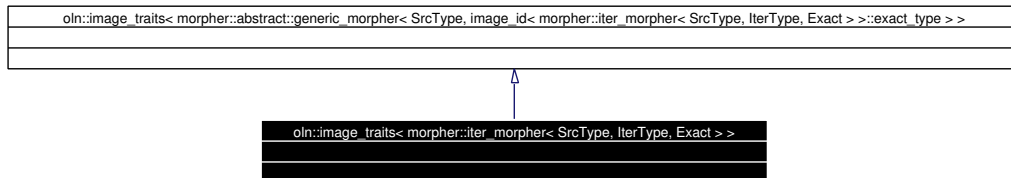
- color_morpher.hh

7.212 oln::image_traits< morpher::iter_morpher< SrcType, IterType, Exact > > Struct Template Reference

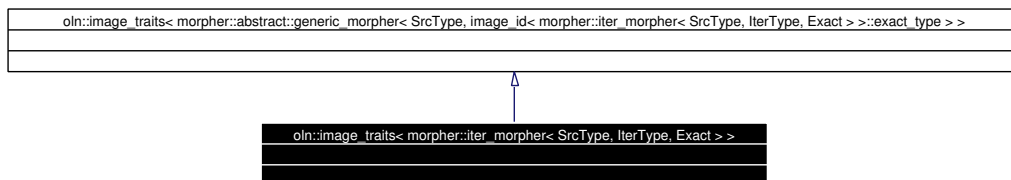
Traits for iter morpher.

```
#include <iter_morpher.hh>
```

Inheritance diagram for oln::image_traits< morpher::iter_morpher< SrcType, IterType, Exact > >:



Collaboration diagram for oln::image_traits< morpher::iter_morpher< SrcType, IterType, Exact > >:



Public Types

- typedef IterType **iter_type**

7.212.1 Detailed Description

```
template<class SrcType, class IterType, class Exact> struct oln::image_traits< morpher::iter_morpher< SrcType, IterType, Exact > >
```

Traits for iter morpher.

Definition at line 64 of file iter_morpher.hh.

The documentation for this struct was generated from the following file:

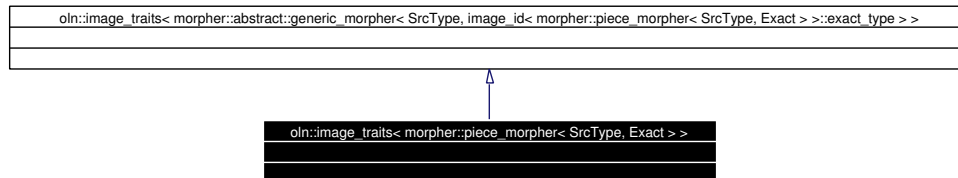
- iter_morpher.hh

7.213 oln::image_traits< morpher::piece_morpher< SrcType, Exact > > Struct Template Reference

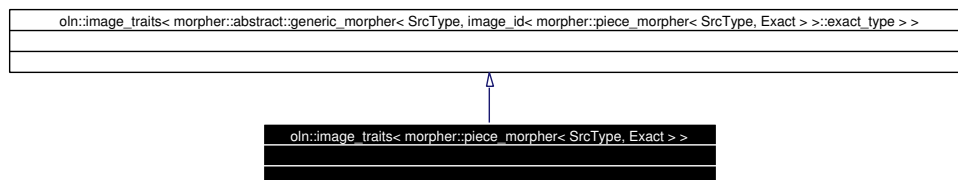
Traits for piece morpher.

```
#include <piece_morpher.hh>
```

Inheritance diagram for oln::image_traits< morpher::piece_morpher< SrcType, Exact > >:



Collaboration diagram for oln::image_traits< morpher::piece_morpher< SrcType, Exact > >:



7.213.1 Detailed Description

template<class SrcType, class Exact> struct oln::image_traits< morpher::piece_morpher< SrcType, Exact > >

Traits for piece morpher.

Definition at line 75 of file piece_morpher.hh.

The documentation for this struct was generated from the following file:

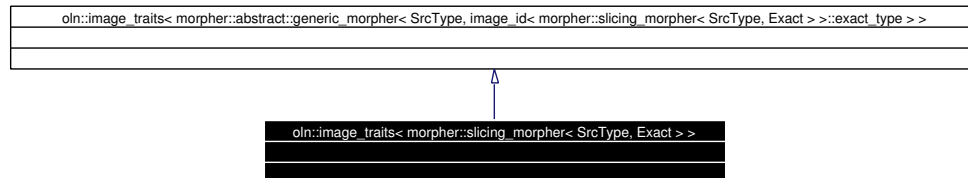
- piece_morpher.hh

7.214 oln::image_traits< morpher::slicing_morpher< SrcType, Exact > > Struct Template Reference

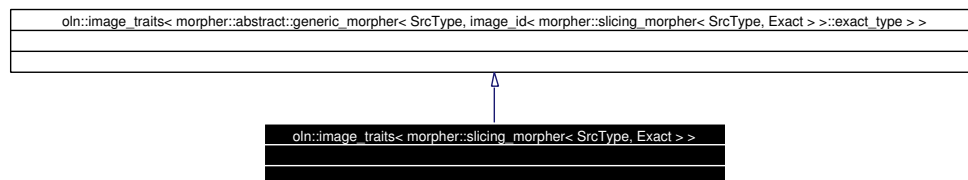
Traits for slicing morpher.

```
#include <slicing_morpher.hh>
```

Inheritance diagram for oln::image_traits< morpher::slicing_morpher< SrcType, Exact > >:



Collaboration diagram for oln::image_traits< morpher::slicing_morpher< SrcType, Exact > >:



7.214.1 Detailed Description

template<class SrcType, class Exact> struct oln::image_traits< morpher::slicing_morpher< SrcType, Exact > >

Traits for slicing morpher.

Definition at line 118 of file slicing_morpher.hh.

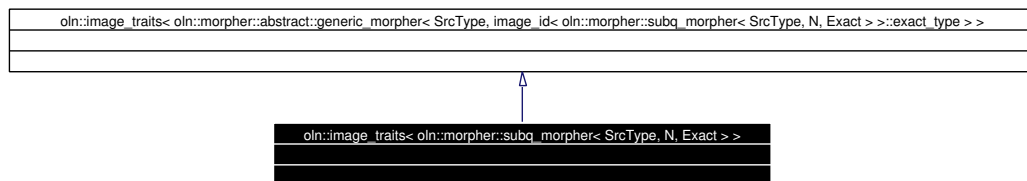
The documentation for this struct was generated from the following file:

- slicing_morpher.hh

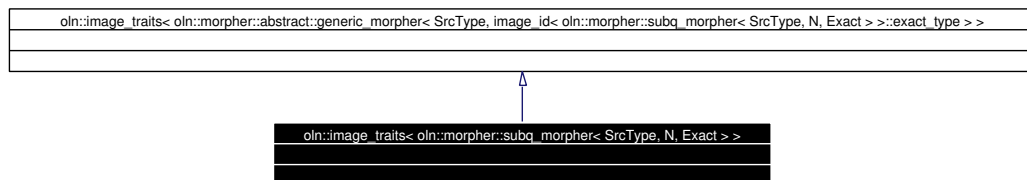
7.215 oln::image_traits< oln::morpher::subq_morpher< SrcType, N, Exact > > Struct Template Reference

```
#include <subq_morpher.hh>
```

Inheritance diagram for oln::image_traits< oln::morpher::subq_morpher< SrcType, N, Exact > >:



Collaboration diagram for oln::image_traits< oln::morpher::subq_morpher< SrcType, N, Exact > >:



7.215.1 Detailed Description

```
template<class SrcType, unsigned N, class Exact> struct oln::image_traits< oln::morpher::subq_morpher< SrcType, N, Exact > >
```

Specialized version for subq_morpher.

Parameters:

- SrcType* Input type decorated.
- N* The new number of bits by components.
- Exact* The exact type of the morpher.

Definition at line 109 of file subq_morpher.hh.

The documentation for this struct was generated from the following file:

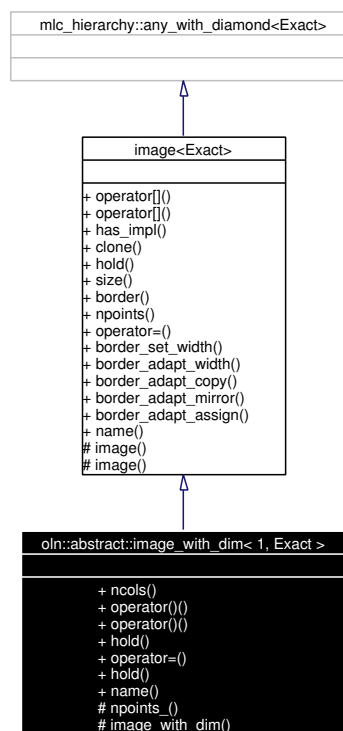
- subq_morpher.hh

7.216 oln::abstract::image_with_dim< 1, Exact > Class Template Reference

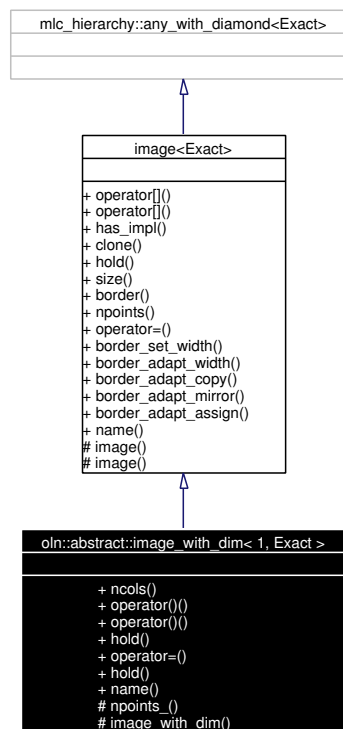
The specialized version for [image1d](#).

```
#include <image_with_dim.hh>
```

Inheritance diagram for oln::abstract::image_with_dim< 1, Exact >:



Collaboration diagram for oln::abstract::image_with_dim< 1, Exact >:



Public Types

- typedef `image_traits< Exact >::point_type` `point_type`
- typedef `image_traits< Exact >::point_type` `dpoint_type`
- typedef `image_traits< Exact >::iter_type` `iter_type`
- typedef `image_traits< Exact >::fwd_iter_type` `fwd_iter_type`
- typedef `image_traits< Exact >::bkd_iter_type` `bkd_iter_type`
- typedef `image_traits< Exact >::value_type` `value_type`
- typedef `image_traits< Exact >::size_type` `size_type`
- typedef `image< Exact >` `super_type`
- typedef `image_with_dim< 1, Exact >` `self_type`
- typedef `Exact` `exact_type`

Public Member Functions

- `coord` `ncols` () const
Return the number of columns in the current image.
- const `value_type` `operator`() (coord col) const
Return the value stored at col coordinate in the current image.
- `value_type` & `operator`() (coord col)
Return a reference to the value stored at col coordinate in the current image.
- bool `hold` (coord col) const

Test if a point belongs to the current image.

- `exact_type & operator= (self_type rhs)`

Perform a shallow copy from rhs to the current image, the points are not duplicated but shared between the two images.

- `bool hold (const abstract::point< point_type > &p) const`

Test if the point p belong to the image.

Static Public Member Functions

- `std::string name ()`

Protected Member Functions

- `size_t npoints_ () const`

Return the total number of points in the current image.

- `image_with_dim ()`

Friends

- `class image< exact_type >`

7.216.1 Detailed Description

`template<class Exact> class oln::abstract::image_with_dim< 1, Exact >`

The specialized version for `image1d`.

Definition at line 156 of file `image_with_dim.hh`.

7.216.2 Member Typedef Documentation

7.216.2.1 `template<class Exact> typedef image_traits<Exact>::bkd_iter_type oln::abstract::image_with_dim< 1, Exact >::bkd_iter_type`

Backward iterator type.

Reimplemented from `oln::abstract::image< Exact >`.

Definition at line 180 of file `image_with_dim.hh`.

7.216.2.2 `template<class Exact> typedef image_traits<Exact>::point_type oln::abstract::image_with_dim< 1, Exact >::dpoint_type`

Prefer the macro `oln_dpoint_type(I)` to retrieve the `dpoint_type` of an image.

See also:

[oln::dpoint1d](#)

Reimplemented from [oln::abstract::image< Exact >](#).

Definition at line 166 of file image_with_dim.hh.

7.216.2.3 `template<class Exact> typedef image_traits<Exact>::fwd_iter_type
oln::abstract::image_with_dim< 1, Exact >::fwd_iter_type`

Forward iterator type.

Reimplemented from [oln::abstract::image< Exact >](#).

Definition at line 178 of file image_with_dim.hh.

7.216.2.4 `template<class Exact> typedef image_traits<Exact>::iter_type
oln::abstract::image_with_dim< 1, Exact >::iter_type`

Prefer the macro `oln_iter_type(I)` to retrieve the `iter_type` of an image.

See also:

[iter1d](#)

Reimplemented from [oln::abstract::image< Exact >](#).

Definition at line 172 of file image_with_dim.hh.

7.216.2.5 `template<class Exact> typedef image_traits<Exact>::point_type
oln::abstract::image_with_dim< 1, Exact >::point_type`

Prefer the macro `oln_point_type(I)` to retrieve the `point_type` of an image.

See also:

[oln::point1d](#)

Reimplemented from [oln::abstract::image< Exact >](#).

Definition at line 160 of file image_with_dim.hh.

7.216.2.6 `template<class Exact> typedef image_traits<Exact>::size_type
oln::abstract::image_with_dim< 1, Exact >::size_type`

Indicate how the image size is handled.

See also:

[oln::image1d_size](#)

Reimplemented from [oln::abstract::image< Exact >](#).

Definition at line 186 of file image_with_dim.hh.

7.216.2.7 `template<class Exact> typedef image_traits<Exact>::value_type oln::abstract::image_with_dim< 1, Exact >::value_type`

Prefer the macro `oln_value_type(I)` to retrieve the `value_type` of an image.

Reimplemented from [oln::abstract::image< Exact >](#).

Definition at line 182 of file `image_with_dim.hh`.

7.216.3 Member Function Documentation

7.216.3.1 `template<class Exact> bool oln::abstract::image< Exact >::hold (const abstract::point< point_type > &p) const` [`inline`]

Test if the point p belong to the image.

Returns:

True if p belong to the image, false otherwise.

Definition at line 173 of file `core/abstract/image.hh`.

```

174     {
175         assertion(has_impl());
176         return this->exact().impl()->hold(p.exact());
177     }
```

7.216.3.2 `template<class Exact> bool oln::abstract::image_with_dim< 1, Exact >::hold (coord col) const` [`inline`]

Test if a point belongs to the current image.

- `col` Column coordinate of the point.

Returns:

True if the point belongs to the image, false otherwise.

Definition at line 236 of file `image_with_dim.hh`.

References `oln::coord`.

```

237     {
238         return hold(point_type(col));
239     }
```

7.216.3.3 `template<class Exact> exact_type& oln::abstract::image_with_dim< 1, Exact >::operator= (self_type rhs)` [`inline`]

Perform a shallow copy from *rhs* to the current image, the points are not duplicated but shared between the two images.

See also:

[image::clone\(\)](#)

Reimplemented from [oln::abstract::image< Exact >](#).

Definition at line 249 of file image_with_dim.hh.

```
250     {  
251         return this->exact().assign(rhs.exact());  
252     }
```

The documentation for this class was generated from the following file:

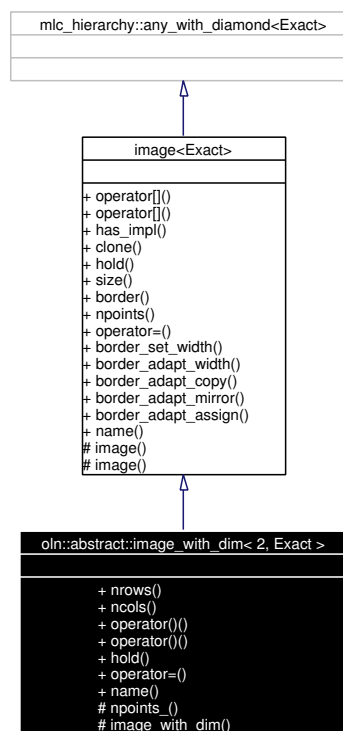
- image_with_dim.hh

7.217 oln::abstract::image_with_dim< 2, Exact > Class Template Reference

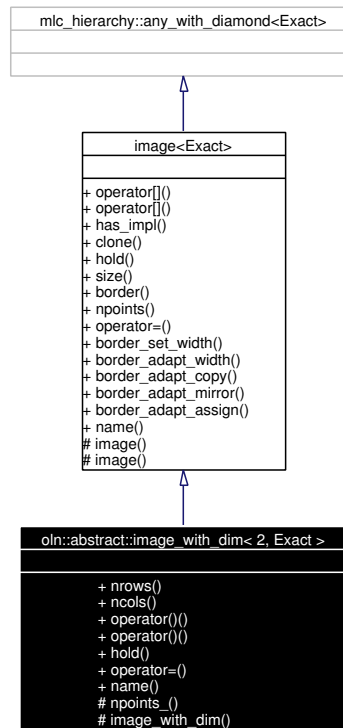
The specialized version for [image2d](#).

```
#include <image_with_dim.hh>
```

Inheritance diagram for oln::abstract::image_with_dim< 2, Exact >:



Collaboration diagram for oln::abstract::image_with_dim< 2, Exact >:



Public Types

- typedef `image_traits< Exact >::point_type` `point_type`
- typedef `image_traits< Exact >::point_type` `dpoint_type`
- typedef `image_traits< Exact >::iter_type` `iter_type`
- typedef `image_traits< Exact >::fwd_iter_type` `fwd_iter_type`
- typedef `image_traits< Exact >::bkd_iter_type` `bkd_iter_type`
- typedef `image_traits< Exact >::value_type` `value_type`
- typedef `image_traits< Exact >::size_type` `size_type`
- typedef `image< Exact >` `super_type`
- typedef `image_with_dim< 2, Exact >` `self_type`
- typedef `Exact` `exact_type`

Public Member Functions

- `coord` `nrows` () const
Return the number of rows in the current image.
- `coord` `ncols` () const
Return the number of columns in the current image.
- const `value_type` `operator`() (`coord` row, `coord` col) const
Return the value stored at row, col coordinates on the current image.
- `value_type` & `operator`() (`coord` row, `coord` col)

Return a reference to the value stored at row, col coordinates on the current image.

- bool [hold](#) (coord row, coord col) const

Test if a point belongs to the current image.

- exact_type & [operator=](#) (self_type rhs)

Perform a shallow copy from rhs to the current image, the points are not duplicated but shared between the two image.

Static Public Member Functions

- std::string [name](#) ()

Protected Member Functions

- size_t [npoints_](#) () const

Return the total number of points in the current image.

- [image_with_dim](#) ()

Friends

- class [image](#)< exact_type >

7.217.1 Detailed Description

`template<class Exact> class oln::abstract::image_with_dim< 2, Exact >`

The specialized version for [image2d](#).

Definition at line 281 of file `image_with_dim.hh`.

7.217.2 Member Typedef Documentation

7.217.2.1 `template<class Exact> typedef image_traits<Exact>::bkd_iter_type oln::abstract::image_with_dim< 2, Exact >::bkd_iter_type`

Backward iterator type.

Reimplemented from [oln::abstract::image](#)< Exact >.

Definition at line 306 of file `image_with_dim.hh`.

7.217.2.2 `template<class Exact> typedef image_traits<Exact>::point_type oln::abstract::image_with_dim< 2, Exact >::dpoint_type`

Prefer the macro `oln_dpoint_type(I)` to retrieve the `dpoint_type` of an image.

See also:

[oln::dpoint2d](#)

Reimplemented from [oln::abstract::image< Exact >](#).

Definition at line 292 of file image_with_dim.hh.

7.217.2.3 `template<class Exact> typedef image_traits<Exact>::fwd_iter_type
oln::abstract::image_with_dim< 2, Exact >::fwd_iter_type`

Forward iterator type.

Reimplemented from [oln::abstract::image< Exact >](#).

Definition at line 304 of file image_with_dim.hh.

7.217.2.4 `template<class Exact> typedef image_traits<Exact>::iter_type
oln::abstract::image_with_dim< 2, Exact >::iter_type`

Prefer the macro `oln_iter_type(I)` to retrieve the `iter_type` of an image.

See also:

[iter2d](#)

Reimplemented from [oln::abstract::image< Exact >](#).

Definition at line 298 of file image_with_dim.hh.

7.217.2.5 `template<class Exact> typedef image_traits<Exact>::point_type
oln::abstract::image_with_dim< 2, Exact >::point_type`

Prefer the macro `oln_point_type(I)` to retrieve the `point_type` of an image.

See also:

[oln::point2d](#)

Reimplemented from [oln::abstract::image< Exact >](#).

Definition at line 286 of file image_with_dim.hh.

7.217.2.6 `template<class Exact> typedef image_traits<Exact>::size_type
oln::abstract::image_with_dim< 2, Exact >::size_type`

Indicate how the image size is handled.

See also:

[oln::image2d_size](#)

Reimplemented from [oln::abstract::image< Exact >](#).

Definition at line 312 of file image_with_dim.hh.

7.217.2.7 `template<class Exact> typedef image_traits<Exact>::value_type oln::abstract::image_with_dim< 2, Exact >::value_type`

Prefer the macro `oln_value_type(I)` to retrieve the `value_type` of an image.

Reimplemented from `oln::abstract::image< Exact >`.

Definition at line 308 of file `image_with_dim.hh`.

7.217.3 Member Function Documentation

7.217.3.1 `template<class Exact> bool oln::abstract::image_with_dim< 2, Exact >::hold (coord row, coord col) const [inline]`

Test if a point belongs to the current image.

- row Row coordinate of the point.
- col Column coordinate of the point.

Returns:

True if the point belongs to the image, false otherwise.

Definition at line 372 of file `image_with_dim.hh`.

References `oln::coord`.

```

373     {
374         return hold(point_type(row, col));
375     }
```

7.217.3.2 `template<class Exact> exact_type& oln::abstract::image_with_dim< 2, Exact >::operator= (self_type rhs) [inline]`

Perform a shallow copy from *rhs* to the current image, the points are not duplicated but shared between the two image.

See also:

`image::clone()`

Reimplemented from `oln::abstract::image< Exact >`.

Definition at line 386 of file `image_with_dim.hh`.

```

387     {
388         return this->exact().assign(rhs.exact());
389     }
```

The documentation for this class was generated from the following file:

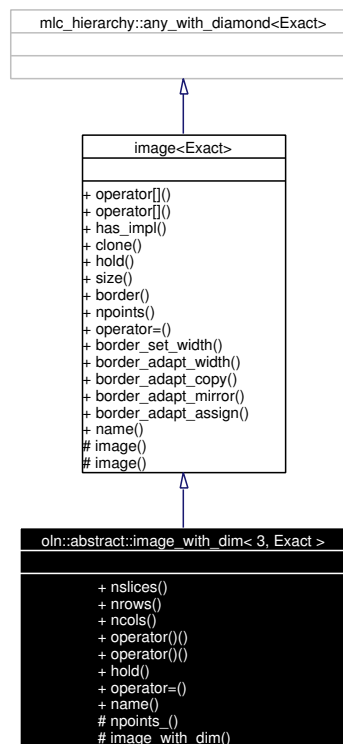
- `image_with_dim.hh`

7.218 oln::abstract::image_with_dim< 3, Exact > Class Template Reference

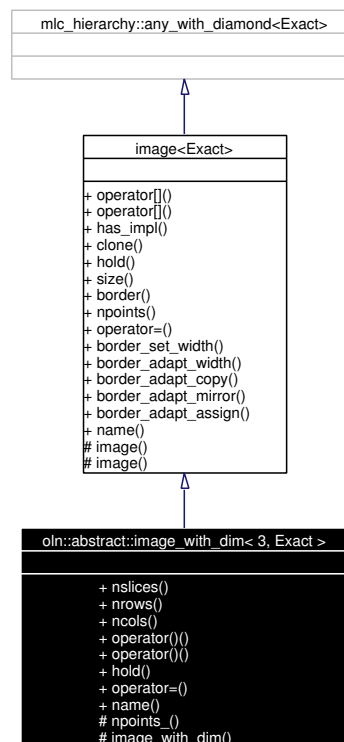
The specialized version for [image3d](#).

```
#include <image_with_dim.hh>
```

Inheritance diagram for oln::abstract::image_with_dim< 3, Exact >:



Collaboration diagram for oln::abstract::image_with_dim< 3, Exact >:



Public Types

- typedef `image_traits< Exact >::point_type` `point_type`
- typedef `image_traits< Exact >::point_type` `dpoint_type`
- typedef `image_traits< Exact >::iter_type` `iter_type`
- typedef `image_traits< Exact >::fwd_iter_type` `fwd_iter_type`
- typedef `image_traits< Exact >::bkd_iter_type` `bkd_iter_type`
- typedef `image_traits< Exact >::value_type` `value_type`
- typedef `image_traits< Exact >::size_type` `size_type`
- typedef `image< Exact >` **super_type**
- typedef `image_with_dim< 3, Exact >` **self_type**
- typedef `Exact` **exact_type**

Public Member Functions

- `coord nslices ()` const
Return the number of slices in the current image.
- `coord nrows ()` const
Return the number of rows in the current image.
- `coord ncols ()` const
Return the number of columns in the image.
- const `value_type operator()` (`coord` slice, `coord` row, `coord` col) const

Return the value stored at slice, row and col coordinates on the current image.

- `value_type & operator()` (`coord` slice, `coord` row, `coord` col)

Return a reference to the value stored at slice, row and col coordinates on the current image.

- `bool hold` (`coord` slice, `coord` row, `coord` col) `const`

Test if a point belongs to the current image.

- `exact_type & operator=` (`self_type` rhs)

Perform a shallow copy from rhs to the current image, the points are not duplicated but shared between the two image.

Static Public Member Functions

- `std::string name` ()

Protected Member Functions

- `size_t npoints_` () `const`

Return the total number of points in the current image.

- `image_with_dim` ()

Friends

- `class image< exact_type >`

7.218.1 Detailed Description

`template<class Exact> class oln::abstract::image_with_dim< 3, Exact >`

The specialized version for `image3d`.

Definition at line 420 of file `image_with_dim.hh`.

7.218.2 Member Typedef Documentation

7.218.2.1 `template<class Exact> typedef image_traits<Exact>::bkd_iter_type oln::abstract::image_with_dim< 3, Exact >::bkd_iter_type`

Backward iterator type.

Reimplemented from `oln::abstract::image< Exact >`.

Definition at line 444 of file `image_with_dim.hh`.

7.218.2.2 `template<class Exact> typedef image_traits<Exact>::point_type
oln::abstract::image_with_dim< 3, Exact >::dpoint_type`

Prefer the macro `oln_dpoint_type(I)` to retrieve the `dpoint_type` of an image.

See also:

[oln::dpoint3d](#)

Reimplemented from [oln::abstract::image< Exact >](#).

Definition at line 430 of file `image_with_dim.hh`.

7.218.2.3 `template<class Exact> typedef image_traits<Exact>::fwd_iter_type
oln::abstract::image_with_dim< 3, Exact >::fwd_iter_type`

Forward iterator type.

Reimplemented from [oln::abstract::image< Exact >](#).

Definition at line 442 of file `image_with_dim.hh`.

7.218.2.4 `template<class Exact> typedef image_traits<Exact>::iter_type
oln::abstract::image_with_dim< 3, Exact >::iter_type`

Prefer the macro `oln_iter_type(I)` to retrieve the `iter_type` of an image.

See also:

[iter3d](#)

Reimplemented from [oln::abstract::image< Exact >](#).

Definition at line 436 of file `image_with_dim.hh`.

7.218.2.5 `template<class Exact> typedef image_traits<Exact>::point_type
oln::abstract::image_with_dim< 3, Exact >::point_type`

Prefer the macro `oln_point_type(I)` to retrieve the `point_type` of an image.

See also:

[oln::point3d](#)

Reimplemented from [oln::abstract::image< Exact >](#).

Definition at line 424 of file `image_with_dim.hh`.

7.218.2.6 `template<class Exact> typedef image_traits<Exact>::size_type
oln::abstract::image_with_dim< 3, Exact >::size_type`

Indicate how the image size is handled.

See also:

[oln::image3d_size](#)

Reimplemented from [oln::abstract::image< Exact >](#).

Definition at line 450 of file `image_with_dim.hh`.

7.218.2.7 `template<class Exact> typedef image_traits<Exact>::value_type oln::abstract::image_with_dim< 3, Exact >::value_type`

Prefer the macro `oln_value_type(I)` to retrieve the `value_type` of an image.

Reimplemented from [oln::abstract::image< Exact >](#).

Definition at line 446 of file `image_with_dim.hh`.

7.218.3 Member Function Documentation

7.218.3.1 `template<class Exact> bool oln::abstract::image_with_dim< 3, Exact >::hold (coord slice, coord row, coord col) const [inline]`

Test if a point belongs to the current image.

- slice Slice coordinate of the point.
- row Row coordinate of the point.
- col Column coordinate of the point.

Returns:

True if the point belongs to the image, false otherwise.

Definition at line 523 of file `image_with_dim.hh`.

References [oln::coord](#).

```
524     {
525         return hold(point_type(slice, row, col));
526     }
```

7.218.3.2 `template<class Exact> exact_type& oln::abstract::image_with_dim< 3, Exact >::operator= (self_type rhs) [inline]`

Perform a shallow copy from *rhs* to the current image, the points are not duplicated but shared between the two image.

See also:

[image::clone\(\)](#)

Reimplemented from [oln::abstract::image< Exact >](#).

Definition at line 537 of file `image_with_dim.hh`.

```
538     {
539         return this->exact().assign(rhs.exact());
540     }
```

The documentation for this class was generated from the following file:

- `image_with_dim.hh`

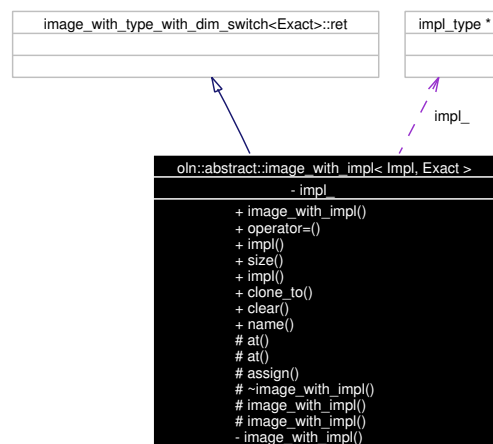
7.219 oln::abstract::image_with_impl< Impl, Exact > Class Template Reference

```
#include <image_with_impl.hh>
```

Inheritance diagram for oln::abstract::image_with_impl< Impl, Exact >:



Collaboration diagram for oln::abstract::image_with_impl< Impl, Exact >:



Public Types

- typedef [image_traits< Exact >::point_type](#) [point_type](#)
- typedef [image_traits< Exact >::iter_type](#) [iter_type](#)
- typedef [image_traits< Exact >::fwd_iter_type](#) [fwd_iter_type](#)
- typedef [image_traits< Exact >::bkwd_iter_type](#) [bkwd_iter_type](#)
- typedef [image_traits< Exact >::value_type](#) [value_type](#)
- typedef [image_traits< Exact >::size_type](#) [size_type](#)
- typedef [image_traits< Exact >::impl_type](#) [impl_type](#)
- typedef [image_with_impl< Impl, Exact >](#) **self_type**
- typedef Exact **exact_type**
- typedef [image_with_type_with_dim_switch< Exact >::ret](#) **super_type**

Public Member Functions

- [image_with_impl](#) (self_type &rhs)
- exact_type & [operator=](#) (self_type rhs)

Perform a shallow copy from rhs to the current image, the points will not be duplicated but shared between the two images.

- const [impl_type](#) * [impl](#) () const

Return the core data of the image.

- `const size_type & size () const`
Get the size of the image.
- `impl_type * impl ()`
Return the core data of the image.
- `void clone_to (impl_type *output_data) const`
Create a copy of the current image data, at the output_data address.
- `void clear ()`
Free the image data.

Static Public Member Functions

- `std::string name ()`

Protected Member Functions

- `const value_type at (const point_type &p) const`
Return a reference to the value stored at p in the current image.
- `value_type & at (const point_type &p)`
Return the value stored at p in the current image.
- `exact_type & assign (exact_type rhs)`
Perform a shallow copy from rhs to the current image, the points will not be duplicated but shared between the two images.
- `image_with_impl ()`
Empty constructor for image_with_impl, set impl_ to 0.
- `image_with_impl (impl_type *impl)`
Assign impl to this->impl_.

Friends

- `class image< exact_type >`
- `class image_with_dim< image_id< Exact >::dim, exact_type >`
- `class image_with_type< typename image_id< Exact >::value_type, Exact >`

7.219.1 Detailed Description

`template<class Impl, class Exact> class oln::abstract::image_with_impl< Impl, Exact >`

This class contains all the implementation relative methods.

Definition at line 69 of file image_with_impl.hh.

7.219.2 Member Typedef Documentation

7.219.2.1 `template<class Impl, class Exact> typedef image_traits<Exact>::bkd_iter_type` `oln::abstract::image_with_impl< Impl, Exact >::bkd_iter_type`

Backward iterator type.

Reimplemented in `oln::image< Dim, T, Impl, Exact >`, `oln::morpher::abstract::generic_morpher< SrcType, Exact >`, `oln::image< image_id< image2d< T, Exact > >::dim, image_id< image2d< T, Exact > >::value_type, image_id< image2d< T, Exact > >::impl_type, image_id< image2d< T, Exact > >::exact_type >`, `oln::image< image_id< image3d< T, Exact > >::dim, image_id< image3d< T, Exact > >::value_type, image_id< image3d< T, Exact > >::impl_type, image_id< image3d< T, Exact > >::exact_type >`, `oln::image< image_id< image2d< unsigned, Exact > >::dim, image_id< image2d< unsigned, Exact > >::value_type, image_id< image2d< unsigned, Exact > >::impl_type, image_id< image2d< unsigned, Exact > >::exact_type >`, `oln::image< image_id< image2d< std::vector< oln::point2d >, Exact > >::dim, image_id< image2d< std::vector< oln::point2d >, Exact > >::value_type, image_id< image2d< std::vector< oln::point2d >, Exact > >::impl_type, image_id< image2d< std::vector< oln::point2d >, Exact > >::exact_type >`, `oln::image< image_id< image1d< T, Exact > >::dim, image_id< image1d< T, Exact > >::value_type, image_id< image1d< T, Exact > >::impl_type, image_id< image1d< T, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< const SrcType, image_id< iter_morpher< const SrcType, IterType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< SrcType, oln::image_id< subq_morpher< SrcType, N, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< SrcType, image_id< color_morpher< SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< const , image_id< slicing_morpher< const SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< const , image_id< color_morpher< const SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< SrcType, image_id< piece_morpher< SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< const , image_id< piece_morpher< const SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< SrcType, image_id< iter_morpher< SrcType, IterType, Exact > >::exact_type >`, and `oln::morpher::abstract::generic_morpher< SrcType, image_id< slicing_morpher< SrcType, Exact > >::exact_type >`.

Definition at line 89 of file `image_with_impl.hh`.

7.219.2.2 `template<class Impl, class Exact> typedef image_traits<Exact>::fwd_iter_type` `oln::abstract::image_with_impl< Impl, Exact >::fwd_iter_type`

Forward iterator type.

Reimplemented in `oln::image< Dim, T, Impl, Exact >`, `oln::morpher::abstract::generic_morpher< SrcType, Exact >`, `oln::image< image_id< image2d< T, Exact > >::dim, image_id< image2d< T, Exact > >::value_type, image_id< image2d< T, Exact > >::impl_type, image_id< image2d< T, Exact > >::exact_type >`, `oln::image< image_id< image3d< T, Exact > >::dim, image_id< image3d< T, Exact > >::value_type, image_id< image3d< T, Exact > >::impl_type, image_id< image3d< T, Exact > >::exact_type >`, `oln::image< image_id< image2d< unsigned, Exact > >::dim, image_id< image2d< unsigned, Exact > >::value_type, image_id< image2d< unsigned, Exact > >::impl_type, image_id< image2d< unsigned, Exact > >::exact_type >`, `oln::image< image_id< image2d< std::vector< oln::point2d >, Exact > >::dim, image_id< image2d< std::vector< oln::point2d >, Exact > >::value_type, image_id< image2d< std::vector< oln::point2d >, Exact > >::impl_type, image_id< image2d< std::vector< oln::point2d >, Exact > >::exact_type >`, `oln::image< image_id< image1d< T, Exact > >::dim, image_id< image1d< T, Exact > >::value_type, image_id< image1d< T, Exact > >::impl_type, image_id< image1d< T, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< const SrcType, image_id< iter_morpher< const SrcType,`

```
IterType, Exact > >::exact_type >, oln::morpher::abstract::generic_morpher< SrcType, oln::image_id<
subq_morpher< SrcType, N, Exact > >::exact_type >, oln::morpher::abstract::generic_morpher< Src-
Type, image_id< color_morpher< SrcType, Exact > >::exact_type >, oln::morpher::abstract::generic_-
morpher< const , image_id< slicing_morpher< const SrcType, Exact > >::exact_type >,
oln::morpher::abstract::generic_morpher< const , image_id< color_morpher< const SrcType, Exact
> >::exact_type >, oln::morpher::abstract::generic_morpher< SrcType, image_id< piece_morpher<
SrcType, Exact > >::exact_type >, oln::morpher::abstract::generic_morpher< const , image_-
id< piece_morpher< const SrcType, Exact > >::exact_type >, oln::morpher::abstract::generic_-
morpher< SrcType, image_id< iter_morpher< SrcType, IterType, Exact > >::exact_type >, and
oln::morpher::abstract::generic_morpher< SrcType, image_id< slicing_morpher< SrcType, Exact >
>::exact_type >.
```

Definition at line 87 of file image_with_impl.hh.

7.219.2.3 `template<class Impl, class Exact> typedef image_traits<Exact>::impl_type` `oln::abstract::image_with_impl< Impl, Exact >::impl_type`

Underlying implementation.

Reimplemented in `oln::image< Dim, T, Impl, Exact >`, `oln::image1d< T, Exact >`, `oln::image2d< T, Exact >`, `oln::image3d< T, Exact >`, `oln::morpher::super_color_morpher< SrcType, Exact >`, `oln::morpher::abstract::generic_morpher< SrcType, Exact >`, `oln::morpher::super_slicing_morpher< SrcType, Exact >`, `oln::morpher::subq_morpher< SrcType, N, Exact >`, `oln::image< image_id< image2d< T, Exact > >::dim, image_id< image2d< T, Exact > >::value_type, image_id< image2d< T, Exact > >::impl_type, image_id< image2d< T, Exact > >::exact_type >`, `oln::image< image_id< image3d< T, Exact > >::dim, image_id< image3d< T, Exact > >::value_type, image_id< image3d< T, Exact > >::impl_type, image_id< image3d< T, Exact > >::exact_type >`, `oln::image< image_id< image2d< unsigned, Exact > >::dim, image_id< image2d< unsigned, Exact > >::value_type, image_id< image2d< unsigned, Exact > >::impl_type, image_id< image2d< unsigned, Exact > >::exact_type >`, `oln::image< image_id< image2d< std::vector< oln::point2d >, Exact > >::dim, image_id< image2d< std::vector< oln::point2d >, Exact > >::value_type, image_id< image2d< std::vector< oln::point2d >, Exact > >::impl_type, image_id< image2d< std::vector< oln::point2d >, Exact > >::exact_type >`, `oln::image< image_id< image1d< T, Exact > >::dim, image_id< image1d< T, Exact > >::value_type, image_id< image1d< T, Exact > >::impl_type, image_id< image1d< T, Exact > >::exact_type >`, `oln::image2d< T >`, `oln::image2d< unsigned >`, `oln::image2d< std::vector< oln::point2d > >`, `oln::morpher::super_color_morpher< SrcType, image_id< color_morpher< SrcType, Exact > >::exact_type >`, `oln::morpher::super_color_morpher< const SrcType, image_id< color_morpher< const SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< const SrcType, image_id< iter_morpher< const SrcType, IterType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< SrcType, oln::image_id< subq_morpher< SrcType, N, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< SrcType, image_id< color_morpher< SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< const , image_id< slicing_morpher< const SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< const , image_id< color_morpher< const SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< SrcType, image_id< piece_morpher< SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< const , image_id< piece_morpher< const SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< SrcType, image_id< iter_morpher< SrcType, IterType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< SrcType, image_id< slicing_morpher< SrcType, Exact > >::exact_type >`, `oln::morpher::super_slicing_morpher< const SrcType, image_id< slicing_morpher< const SrcType, Exact > >::exact_type >`, and `oln::morpher::super_slicing_morpher< SrcType, image_id< slicing_morpher< SrcType, Exact > >::exact_type >`.

Definition at line 100 of file image_with_impl.hh.

7.219.2.4 `template<class Impl, class Exact> typedef image_traits<Exact>::iter_type` `oln::abstract::image_with_impl< Impl, Exact >::iter_type`

Prefer the macro `oln_iter_type(I)` to retrieve the `iter_type` of an image.

See also:

[iter](#)

Reimplemented in `oln::image< Dim, T, Impl, Exact >`, `oln::morpher::color_morpher< SrcType, Exact >`, `oln::morpher::color_morpher< const SrcType, Exact >`, `oln::morpher::abstract::generic_morpher< SrcType, Exact >`, `oln::morpher::iter_morpher< const SrcType, IterType, Exact >`, `oln::image< image_id< image2d< T, Exact > >::dim, image_id< image2d< T, Exact > >::value_type, image_id< image2d< T, Exact > >::impl_type, image_id< image2d< T, Exact > >::exact_type >`, `oln::image< image_id< image3d< T, Exact > >::dim, image_id< image3d< T, Exact > >::value_type, image_id< image3d< T, Exact > >::impl_type, image_id< image3d< T, Exact > >::exact_type >`, `oln::image< image_id< image2d< unsigned, Exact > >::dim, image_id< image2d< unsigned, Exact > >::value_type, image_id< image2d< unsigned, Exact > >::impl_type, image_id< image2d< unsigned, Exact > >::exact_type >`, `oln::image< image_id< image2d< std::vector< oln::point2d >, Exact > >::dim, image_id< image2d< std::vector< oln::point2d >, Exact > >::value_type, image_id< image2d< std::vector< oln::point2d >, Exact > >::impl_type, image_id< image2d< std::vector< oln::point2d >, Exact > >::exact_type >`, `oln::image< image_id< image1d< T, Exact > >::dim, image_id< image1d< T, Exact > >::value_type, image_id< image1d< T, Exact > >::impl_type, image_id< image1d< T, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< const SrcType, image_id< iter_morpher< const SrcType, IterType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< SrcType, oln::image_id< subq_morpher< SrcType, N, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< SrcType, image_id< color_morpher< SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< const, image_id< slicing_morpher< const SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< const, image_id< color_morpher< const SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< SrcType, image_id< piece_morpher< SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< const, image_id< piece_morpher< const SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< SrcType, image_id< iter_morpher< SrcType, IterType, Exact > >::exact_type >`, and `oln::morpher::abstract::generic_morpher< SrcType, image_id< slicing_morpher< SrcType, Exact > >::exact_type >`.

Definition at line 81 of file `image_with_impl.hh`.

7.219.2.5 `template<class Impl, class Exact> typedef image_traits<Exact>::point_type` `oln::abstract::image_with_impl< Impl, Exact >::point_type`

Prefer the macro `oln_point_type(I)` to retrieve the `point_type` of an image.

See also:

[point](#)

Reimplemented in `oln::image< Dim, T, Impl, Exact >`, `oln::morpher::color_morpher< SrcType, Exact >`, `oln::morpher::color_morpher< const SrcType, Exact >`, `oln::morpher::abstract::generic_morpher< SrcType, Exact >`, `oln::morpher::iter_morpher< const SrcType, IterType, Exact >`, `oln::morpher::piece_morpher< SrcType, Exact >`, `oln::morpher::piece_morpher< const SrcType, Exact >`, `oln::morpher::slicing_morpher< SrcType, Exact >`, `oln::morpher::slicing_morpher< const SrcType, Exact >`, `oln::morpher::subq_morpher< SrcType, N, Exact >`, `oln::image< image_id< image2d< T, Exact > >::dim, image_id< image2d< T, Exact > >::value_type, image_id< image2d< T, Exact > >::impl_type, image_id< image2d< T, Exact > >::exact_type >`, `oln::image< image_id< image3d<`

```
T, Exact > >::dim, image_id< image3d< T, Exact > >::value_type, image_id< image3d< T, Exact >
>::impl_type, image_id< image3d< T, Exact > >::exact_type >, oln::image< image_id< image2d<
unsigned, Exact > >::dim, image_id< image2d< unsigned, Exact > >::value_type, image_id<
image2d< unsigned, Exact > >::impl_type, image_id< image2d< unsigned, Exact > >::exact_type >,
oln::image< image_id< image2d< std::vector< oln::point2d >, Exact > >::dim, image_id< image2d<
std::vector< oln::point2d >, Exact > >::value_type, image_id< image2d< std::vector< oln::point2d >,
Exact > >::impl_type, image_id< image2d< std::vector< oln::point2d >, Exact > >::exact_type >,
oln::image< image_id< image1d< T, Exact > >::dim, image_id< image1d< T, Exact > >::value_
type, image_id< image1d< T, Exact > >::impl_type, image_id< image1d< T, Exact > >::exact_type
>, oln::morpher::abstract::generic_morpher< const SrcType, image_id< iter_morpher< const SrcType,
IterType, Exact > >::exact_type >, oln::morpher::abstract::generic_morpher< SrcType, oln::image_id<
subq_morpher< SrcType, N, Exact > >::exact_type >, oln::morpher::abstract::generic_morpher< Src
Type, image_id< color_morpher< SrcType, Exact > >::exact_type >, oln::morpher::abstract::generic_
morpher< const , image_id< slicing_morpher< const SrcType, Exact > >::exact_type >,
oln::morpher::abstract::generic_morpher< const , image_id< color_morpher< const SrcType, Exact
> >::exact_type >, oln::morpher::abstract::generic_morpher< SrcType, image_id< piece_morpher<
SrcType, Exact > >::exact_type >, oln::morpher::abstract::generic_morpher< const , image_
id< piece_morpher< const SrcType, Exact > >::exact_type >, oln::morpher::abstract::generic_
morpher< SrcType, image_id< iter_morpher< SrcType, IterType, Exact > >::exact_type >, and
oln::morpher::abstract::generic_morpher< SrcType, image_id< slicing_morpher< SrcType, Exact >
>::exact_type >.
```

Definition at line 75 of file image_with_impl.hh.

7.219.2.6 `template<class Impl, class Exact> typedef image_traits<Exact>::size_type oln::abstract::image_with_impl< Impl, Exact >::size_type`

Indicate how the image size is handled.

See also:

[image_size](#)

Reimplemented in `oln::image< Dim, T, Impl, Exact >`, `oln::morpher::abstract::generic_morpher< SrcType, Exact >`, `oln::morpher::super_piece_morpher< SrcType, Exact >`, `oln::morpher::piece_morpher< SrcType, Exact >`, `oln::morpher::piece_morpher< const SrcType, Exact >`, `oln::morpher::super_slicing_morpher< SrcType, Exact >`, `oln::image< image_id< image2d< T, Exact > >::dim, image_id< image2d< T, Exact > >::value_type, image_id< image2d< T, Exact > >::impl_type, image_id< image2d< T, Exact > >::exact_type >`, `oln::image< image_id< image3d< T, Exact > >::dim, image_id< image3d< T, Exact > >::value_type, image_id< image3d< T, Exact > >::impl_type, image_id< image3d< T, Exact > >::exact_type >`, `oln::image< image_id< image2d< unsigned, Exact > >::dim, image_id< image2d< unsigned, Exact > >::value_type, image_id< image2d< unsigned, Exact > >::impl_type, image_id< image2d< unsigned, Exact > >::exact_type >`, `oln::image< image_id< image2d< std::vector< oln::point2d >, Exact > >::dim, image_id< image2d< std::vector< oln::point2d >, Exact > >::value_type, image_id< image2d< std::vector< oln::point2d >, Exact > >::impl_type, image_id< image2d< std::vector< oln::point2d >, Exact > >::exact_type >`, `oln::image< image_id< image1d< T, Exact > >::dim, image_id< image1d< T, Exact > >::value_type, image_id< image1d< T, Exact > >::impl_type, image_id< image1d< T, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< const SrcType, image_id< iter_morpher< const SrcType, IterType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< SrcType, oln::image_id< subq_morpher< SrcType, N, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< SrcType, image_id< color_morpher< SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< const , image_id< slicing_morpher< const SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< const , image_id< color_morpher< const SrcType, Exact`

```
> >::exact_type >, oln::morpher::abstract::generic_morpher< SrcType, image_id< piece_morpher<
SrcType, Exact > >::exact_type >, oln::morpher::abstract::generic_morpher< const , image_
id< piece_morpher< const SrcType, Exact > >::exact_type >, oln::morpher::abstract::generic_
morpher< SrcType, image_id< iter_morpher< SrcType, IterType, Exact > >::exact_type >,
oln::morpher::abstract::generic_morpher< SrcType, image_id< slicing_morpher< SrcType, Exact >
>::exact_type >, oln::morpher::super_piece_morpher< const SrcType, image_id< piece_morpher< const
SrcType, Exact > >::exact_type >, oln::morpher::super_piece_morpher< SrcType, image_id< piece_
morpher< SrcType, Exact > >::exact_type >, oln::morpher::super_slicing_morpher< const SrcType,
image_id< slicing_morpher< const SrcType, Exact > >::exact_type >, and oln::morpher::super_slicing_
morpher< SrcType, image_id< slicing_morpher< SrcType, Exact > >::exact_type >.
```

Definition at line 95 of file image_with_impl.hh.

7.219.2.7 `template<class Impl, class Exact> typedef image_traits<Exact>::value_type` `oln::abstract::image_with_impl< Impl, Exact >::value_type`

Prefer the macro `oln_value_type(I)` to retrieve the `value_type` of an image.

Reimplemented in `oln::image< Dim, T, Impl, Exact >`, `oln::image1d< T, Exact >`, `oln::image2d< T, Exact >`, `oln::image3d< T, Exact >`, `oln::morpher::color_morpher< SrcType, Exact >`, `oln::morpher::color_`
`morpher< const SrcType, Exact >`, `oln::morpher::abstract::generic_morpher< SrcType, Exact >`,
`oln::morpher::iter_morpher< const SrcType, IterType, Exact >`, `oln::morpher::piece_morpher< Src`
`Type, Exact >`, `oln::morpher::piece_morpher< const SrcType, Exact >`, `oln::morpher::slicing_morpher<`
`SrcType, Exact >`, `oln::morpher::slicing_morpher< const SrcType, Exact >`, `oln::morpher::subq_`
`morpher< SrcType, N, Exact >`, `oln::image< image_id< image2d< T, Exact > >::dim, image_`
`id< image2d< T, Exact > >::value_type, image_id< image2d< T, Exact > >::impl_type, image_`
`id< image2d< T, Exact > >::exact_type >`, `oln::image< image_id< image3d< T, Exact > >::dim,`
`image_id< image3d< T, Exact > >::value_type, image_id< image3d< T, Exact > >::impl_type,`
`image_id< image3d< T, Exact > >::exact_type >`, `oln::image< image_id< image2d< unsigned, Ex`
`act > >::dim, image_id< image2d< unsigned, Exact > >::value_type, image_id< image2d< un`
`signed, Exact > >::impl_type, image_id< image2d< unsigned, Exact > >::exact_type >`, `oln::image<`
`image_id< image2d< std::vector< oln::point2d >, Exact > >::dim, image_id< image2d< std::vector<`
`oln::point2d >, Exact > >::value_type, image_id< image2d< std::vector< oln::point2d >, Ex`
`act > >::impl_type, image_id< image2d< std::vector< oln::point2d >, Exact > >::exact_type >`,
`oln::image< image_id< image1d< T, Exact > >::dim, image_id< image1d< T, Exact > >::value_`
`type, image_id< image1d< T, Exact > >::impl_type, image_id< image1d< T, Exact > >::exact_`
`type >`, `oln::image2d< T >`, `oln::image2d< unsigned >`, `oln::image2d< std::vector< oln::point2d >`
`>`, `oln::morpher::abstract::generic_morpher< const SrcType, image_id< iter_morpher< const SrcType,`
`IterType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< SrcType, oln::image_id<`
`subq_morpher< SrcType, N, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< Src`
`Type, image_id< color_morpher< SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_`
`morpher< const , image_id< slicing_morpher< const SrcType, Exact > >::exact_type >`,
`oln::morpher::abstract::generic_morpher< const , image_id< color_morpher< const SrcType, Exact`
`> >::exact_type >`, `oln::morpher::abstract::generic_morpher< SrcType, image_id< piece_morpher<`
`SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_morpher< const , image_`
`id< piece_morpher< const SrcType, Exact > >::exact_type >`, `oln::morpher::abstract::generic_`
`morpher< SrcType, image_id< iter_morpher< SrcType, IterType, Exact > >::exact_type >`, and
`oln::morpher::abstract::generic_morpher< SrcType, image_id< slicing_morpher< SrcType, Exact >`
`>::exact_type >.`

Definition at line 91 of file image_with_impl.hh.

7.219.3 Constructor & Destructor Documentation

7.219.3.1 `template<class Impl, class Exact> oln::abstract::image_with_impl< Impl, Exact >::image_with_impl (self_type & rhs) [inline]`

/brief Construct a new image by performing a shallow copy, the points will not be duplicated but shared between *rhs* and the current image.

See also:

[image::clone\(\)](#)

Definition at line 119 of file `image_with_impl.hh`.

```

120         : super_type(rhs)
121     {
122         assertion(rhs.has_impl());
123         impl_ = rhs.impl();
124         impl_->ref();
125     }
```

7.219.4 Member Function Documentation

7.219.4.1 `template<class Impl, class Exact> exact_type& oln::abstract::image_with_impl< Impl, Exact >::assign (exact_type rhs) [inline, protected]`

Perform a shallow copy from *rhs* to the current image, the points will not be duplicated but shared between the two images.

See also:

[image::clone\(\)](#)

Definition at line 240 of file `image_with_impl.hh`.

Referenced by `oln::image3d< T, Exact >::operator=()`, and `oln::image2d< std::vector< oln::point2d > >::operator=()`.

```

241     {
242         assertion(rhs.impl() != 0);
243         if ( &rhs == this )
244             return this->exact();
245         if (this->impl() != 0)
246             this->impl()->unref();
247         this->impl_ = rhs.impl();
248         this->impl()->ref();
249         return this->exact();
250     }
```

7.219.4.2 `template<class Impl, class Exact> value_type& oln::abstract::image_with_impl< Impl, Exact >::at (const point_type & p) [inline, protected]`

Return the value stored at *p* in the current image.

Warning:

This method must not be overloaded, unlike `operator[]`.

See also:

[image::operator\[\]\(\)](#)

Reimplemented in [oln::morpher::color_morpher< SrcType, Exact >](#), [oln::morpher::piece_morpher< SrcType, Exact >](#), and [oln::morpher::slicing_morpher< SrcType, Exact >](#).

Definition at line 227 of file image_with_impl.hh.

```
228     {
229         return impl_->at(p);
230     }
```

7.219.4.3 `template<class Impl, class Exact> const value_type oln::abstract::image_with_impl< Impl, Exact >::at (const point_type & p) const` [inline, protected]

Return a reference to the value stored at p in the current image.

Warning:

This method must not be overloaded, unlike `operator[]`.

See also:

[image::operator\[\]\(\)](#)

Reimplemented in [oln::morpher::color_morpher< SrcType, Exact >](#), [oln::morpher::color_morpher< const SrcType, Exact >](#), [oln::morpher::iter_morpher< const SrcType, IterType, Exact >](#), [oln::morpher::piece_morpher< SrcType, Exact >](#), [oln::morpher::piece_morpher< const SrcType, Exact >](#), [oln::morpher::slicing_morpher< SrcType, Exact >](#), [oln::morpher::slicing_morpher< const SrcType, Exact >](#), and [oln::morpher::subq_morpher< SrcType, N, Exact >](#).

Definition at line 211 of file image_with_impl.hh.

```
212     {
213         return impl_->at(p);
214     }
```

7.219.4.4 `template<class Impl, class Exact> exact_type& oln::abstract::image_with_impl< Impl, Exact >::operator= (self_type rhs)` [inline]

Perform a shallow copy from `rhs` to the current image, the points will not be duplicated but shared between the two images.

See also:

[image::clone\(\)](#)

Reimplemented in [oln::image1d< T, Exact >](#), [oln::image2d< T, Exact >](#), [oln::image3d< T, Exact >](#), [oln::image2d< T >](#), [oln::image2d< unsigned >](#), and [oln::image2d< std::vector< oln::point2d > >](#).

Definition at line 135 of file image_with_impl.hh.

```
136     {
137         return this->exact().assign(rhs.exact());
138     }
```

7.219.4.5 `template<class Impl, class Exact> const size_type& oln::abstract::image_with_impl<Impl, Exact >::size () const` `[inline]`

Get the size of the image.

Returns:

The size.

Reimplemented in [oln::morpher::super_piece_morpher< SrcType, Exact >](#), [oln::morpher::super_slicing_morpher< SrcType, Exact >](#), [oln::morpher::super_piece_morpher< const SrcType, image_id< piece_morpher< const SrcType, Exact > >::exact_type >](#), [oln::morpher::super_piece_morpher< SrcType, image_id< piece_morpher< SrcType, Exact > >::exact_type >](#), [oln::morpher::super_slicing_morpher< const SrcType, image_id< slicing_morpher< const SrcType, Exact > >::exact_type >](#), and [oln::morpher::super_slicing_morpher< SrcType, image_id< slicing_morpher< SrcType, Exact > >::exact_type >](#).

Definition at line 153 of file [image_with_impl.hh](#).

Referenced by [oln::topo::dmap< T, T2 >::compute\(\)](#), and [oln::topo::dmap< T, T2 >::to_image\(\)](#).

```

154     {
155         assertion(has_impl());
156         return this->exact().impl()->size();
157     }

```

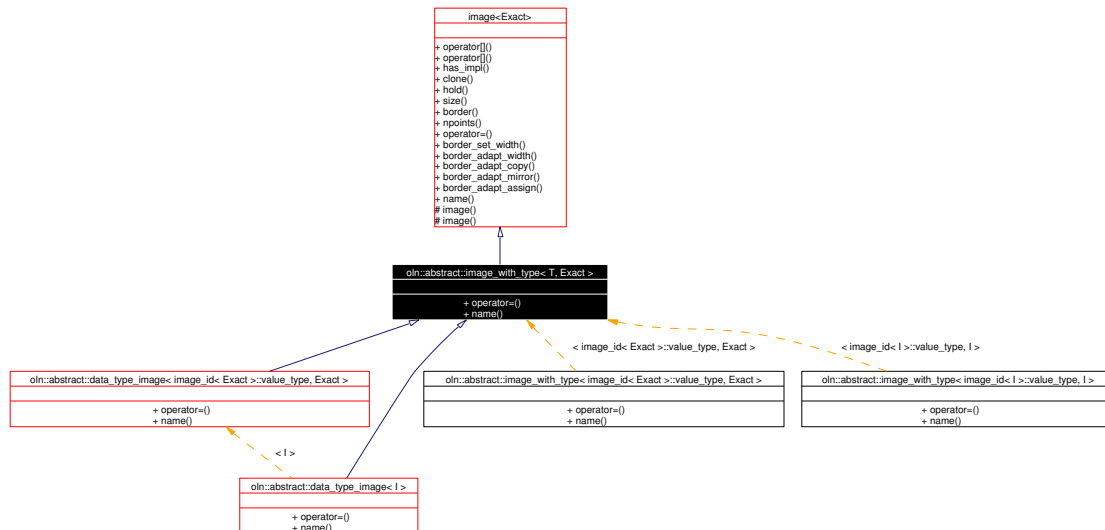
The documentation for this class was generated from the following file:

- [image_with_impl.hh](#)

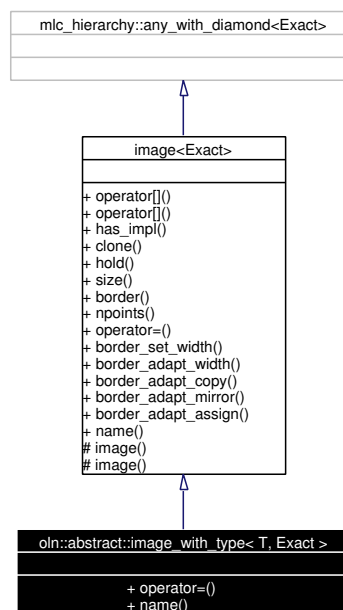
7.220 oln::abstract::image_with_type< T, Exact > Class Template Reference

```
#include <image_with_type.hh>
```

Inheritance diagram for oln::abstract::image_with_type< T, Exact >:



Collaboration diagram for oln::abstract::image_with_type< T, Exact >:



Public Types

- typedef `image<Exact>` `super_type`

- typedef [image_with_type](#)< T, Exact > **self_type**
- typedef Exact **exact_type**

Public Member Functions

- **exact_type** & [operator=](#) (self_type rhs)
Shallow copy from rhs to the current image, the points will not be duplicated but shared between the two images.

Static Public Member Functions

- std::string **name** ()

7.220.1 Detailed Description

template<class T, class Exact> **class** **oln::abstract::image_with_type**< T, Exact >

The template parameter T gives the value_type of the image.

Definition at line 71 of file image_with_type.hh.

7.220.2 Member Typedef Documentation

7.220.2.1 **template**<class T, class Exact> typedef [image](#)<Exact> **oln::abstract::image_with_type**< T, Exact >::[super_type](#)

The base class whom derives [image_with_type](#).

Definition at line 75 of file image_with_type.hh.

7.220.3 Member Function Documentation

7.220.3.1 **template**<class T, class Exact> **exact_type**& **oln::abstract::image_with_type**< T, Exact >::[operator=](#) (self_type rhs) [inline]

Shallow copy from *rhs* to the current image, the points will not be duplicated but shared between the two images.

See also:

[image::clone\(\)](#)

Reimplemented from [oln::abstract::image](#)< Exact >.

Reimplemented in [oln::abstract::data_type_image](#)< Exact >, [oln::abstract::data_type_image_with_dim](#)< Dim, Exact >, [oln::abstract::vectorial_image](#)< Exact >, [oln::abstract::non_vectorial_image](#)< Exact >, [oln::abstract::non_vectorial_image_with_dim](#)< Dim, Exact >, [oln::abstract::binary_image](#)< Exact >, [oln::abstract::binary_image_with_dim](#)< Dim, Exact >, [oln::abstract::integer_image](#)< Exact >, [oln::abstract::integer_image_with_dim](#)< Dim, Exact >, [oln::abstract::decimal_image](#)< Exact >, [oln::abstract::decimal_image_with_dim](#)< Dim, Exact >, [oln::abstract::data_type_image](#)< I >, and [oln::abstract::non_vectorial_image](#)< I >.

Definition at line 87 of file image_with_type.hh.

```
88     {  
89         return this->exact().assign(rhs.exact());  
90     }
```

The documentation for this class was generated from the following file:

- image_with_type.hh

7.221 `oln::abstract::image_with_type_with_dim_switch< Exact >` Struct Template Reference

```
#include <image_with_type_with_dim.hh>
```

Public Types

- `typedef image_id< Exact >::value_type T`
- `typedef mlc::bool_switch_< mlc::bool_case_< mlc::internal::wrap< typename mlc::internal::is_a_< sizeof(mlc::form::get< ntg::binary >)) >::check< typename ntg::type_traits< T >::abstract_type, ntg::binary > >::ret, binary_image_with_dim< Dim, Exact >, mlc::bool_case_< mlc::internal::wrap< typename mlc::internal::is_a_< sizeof(mlc::form::get< ntg::integer >)) >::check< typename ntg::type_traits< T >::abstract_type, ntg::integer > >::ret, integer_image_with_dim< Dim, Exact >, mlc::bool_case_< mlc::internal::wrap< typename mlc::internal::is_a_< sizeof(mlc::form::get< ntg::decimal >)) >::check< typename ntg::type_traits< T >::abstract_type, ntg::decimal > >::ret, decimal_image_with_dim< Dim, Exact >, mlc::bool_case_< mlc::internal::wrap< typename mlc::internal::is_a_< sizeof(mlc::form::get< ntg::vectorial >)) >::check< typename ntg::type_traits< T >::abstract_type, ntg::vectorial > >::ret, vectorial_image_with_dim< Dim, Exact >, mlc::bool_case_< typename mlc::internal::is_a_< sizeof(mlc::form::get< ntg::non_vectorial >)) >::check< typename ntg::type_traits< T >::abstract_type, ntg::non_vectorial > >::ret, non_vectorial_image_with_dim< Dim, Exact >, mlc::bool_case_< true, data_type_image_with_dim< Dim, Exact > > > > > >::re ret)`
- `enum { Dim = image_id<Exact>::dim }`

7.221.1 Detailed Description

```
template<class Exact> struct oln::abstract::image_with_type_with_dim_switch< Exact >
```

A metaswitch that return the right type of an image regarding its dimension and value_type.

Definition at line 272 of file `image_with_type_with_dim.hh`.

7.221.2 Member Typedef Documentation

7.221.2.1 `template<class Exact> typedef mlc::bool_switch_< mlc::bool_case_< <mlc::internal::wrap<typename mlc::internal::is_a_< sizeof(mlc::form::get< ntg::binary >)) >::check< typename ntg::type_traits< T >::abstract_type , ntg::binary > >::ret, binary_image_with_dim<Dim, Exact>, mlc::bool_case_<mlc::internal::wrap<typename mlc::internal::is_a_< sizeof(mlc::form::get< ntg::integer >)) >::check< typename ntg::type_traits< T >::abstract_type , ntg::integer > >::ret, integer_image_with_dim<Dim, Exact>, mlc::bool_case_<mlc::internal::wrap<typename mlc::internal::is_a_< sizeof(mlc::form::get< ntg::decimal >)) >::check< typename ntg::type_traits< T >::abstract_type , ntg::decimal > >::ret, decimal_image_with_dim<Dim, Exact>, mlc::bool_case_<mlc::internal::wrap<typename mlc::internal::is_a_< sizeof(mlc::form::get< ntg::vectorial >)) >::check< typename ntg::type_traits< T >::abstract_type , ntg::vectorial > >::ret, vectorial_image_with_dim<Dim, Exact>, mlc::bool_case_<mlc::internal::wrap<typename mlc::internal::is_a_< sizeof(mlc::form::get< ntg::non_vectorial >)) >::check< typename ntg::type_traits< T >::abstract_type , ntg::non_vectorial > >::ret, non_vectorial_image_with_dim<Dim, Exact>, mlc::bool_case_<true, data_type_image_with_dim<Dim, Exact> > > > > > >::re oln::abstract::image_with_type_with_dim_switch< Exact >::ret)`

Translated in pseudo code :

switch (T)

case ntg::binary : ret = binary_image_with_dim<Dim, Exact>

case ntg::integer : ret = integer_image_with_dim<Dim, Exact>

case ntg::decimal : ret = decimal_image_with_dim<Dim, Exact>

case ntg::vectorial : ret = vectorial_image_with_dim<Dim, Exact>

case ntg::non_vectorial : ret = non_vectorial_image_with_dim<Dim, Exact>

default : ret = data_type_image_with_dim<Dim, Exact>

Definition at line 300 of file image_with_type_with_dim.hh.

The documentation for this struct was generated from the following file:

- image_with_type_with_dim.hh

7.222 oln::io::internal::image_writer< F, I > Struct Template Reference

Write an image of type writer_id.

```
#include <image_base.hh>
```

Static Public Member Functions

- const std::string & [name](#) ()
Do nothing because of the type of writer.
- bool [knows_ext](#) (const std::string &)
Do nothing because of the type of writer.
- bool **write** (std::ostream &, const I &)

7.222.1 Detailed Description

```
template<writer_id F, class I> struct oln::io::internal::image_writer< F, I >
```

Write an image of type writer_id.

Parameters:

F The writer identifier.

In fact, do nothing because of the type of writer. Used when an errors occurs on others types of writer.

See also:

writer_id

Definition at line 111 of file image_base.hh.

The documentation for this struct was generated from the following file:

- image_base.hh

7.223 oln::utils::internal::img_max_size< T > Struct Template Reference

Return the size of the space needed to explore the type T.

```
#include <histogram.hh>
```

Public Types

- typedef [image1d_size](#) **size_type**
- typedef T **comp_type**

Public Member Functions

- [image1d_size](#) **operator()** ()

7.223.1 Detailed Description

```
template<typename T> struct oln::utils::internal::img_max_size< T >
```

Return the size of the space needed to explore the type T.

Definition at line 57 of file histogram.hh.

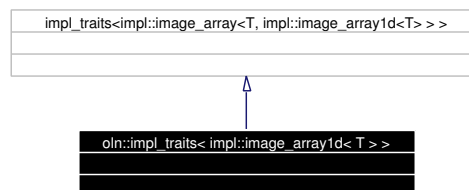
The documentation for this struct was generated from the following file:

- histogram.hh

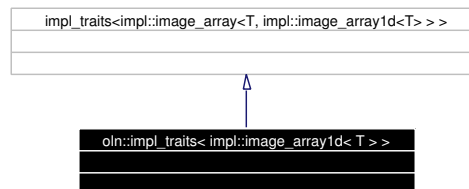
7.224 oln::impl_traits< impl::image_array1d< T > > Struct Template Reference

```
#include <image_array1d.hh>
```

Inheritance diagram for oln::impl_traits< impl::image_array1d< T > >:



Collaboration diagram for oln::impl_traits< impl::image_array1d< T > >:



Public Types

- typedef [point1d](#) **point_type**
- typedef [image1d_size](#) **size_type**
- typedef T **value_type**
- enum { **dim** = 1 }

7.224.1 Detailed Description

```
template<class T> struct oln::impl_traits< impl::image_array1d< T > >
```

Specialized version for impl::image_array1d<T>. Retrieve associated types.

Definition at line 61 of file image_array1d.hh.

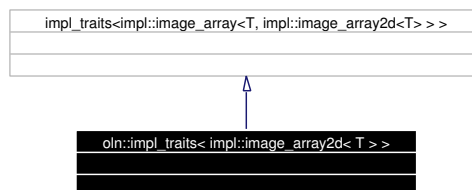
The documentation for this struct was generated from the following file:

- image_array1d.hh

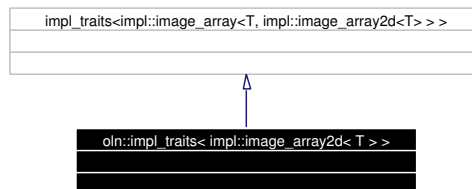
7.225 oln::impl_traits< impl::image_array2d< T > > Struct Template Reference

```
#include <image_array2d.hh>
```

Inheritance diagram for oln::impl_traits< impl::image_array2d< T > >:



Collaboration diagram for oln::impl_traits< impl::image_array2d< T > >:



Public Types

- typedef [point2d](#) **point_type**
- typedef [image2d_size](#) **size_type**
- typedef T **value_type**
- enum { **dim** = 2 }

7.225.1 Detailed Description

template<class T> struct oln::impl_traits< impl::image_array2d< T > >

Specialized version for impl::image_array2d<T>. Retrieve associated types.

Definition at line 81 of file image_array2d.hh.

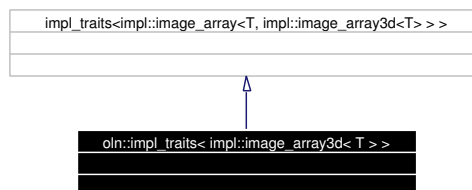
The documentation for this struct was generated from the following file:

- image_array2d.hh

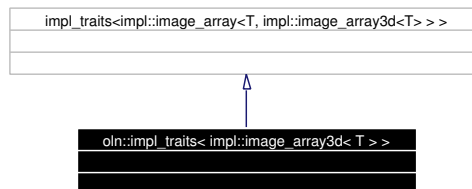
7.226 oln::impl_traits< impl::image_array3d< T > > Struct Template Reference

```
#include <image_array3d.hh>
```

Inheritance diagram for oln::impl_traits< impl::image_array3d< T > >:



Collaboration diagram for oln::impl_traits< impl::image_array3d< T > >:



Public Types

- typedef [point3d](#) **point_type**
- typedef [image3d_size](#) **size_type**
- typedef T **value_type**
- enum { **dim** = 3 }

7.226.1 Detailed Description

```
template<class T> struct oln::impl_traits< impl::image_array3d< T > >
```

Specialized version for impl::image_array3d<T>. Retrieve associated types.

Definition at line 96 of file image_array3d.hh.

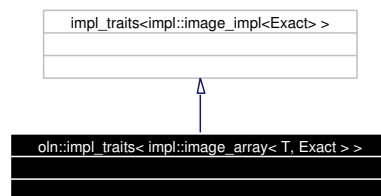
The documentation for this struct was generated from the following file:

- image_array3d.hh

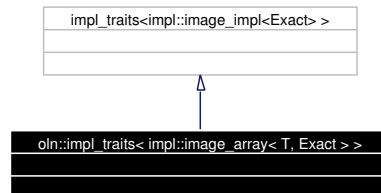
7.227 oln::impl_traits< impl::image_array< T, Exact > > Struct Template Reference

```
#include <image_array.hh>
```

Inheritance diagram for oln::impl_traits< impl::image_array< T, Exact > >:



Collaboration diagram for oln::impl_traits< impl::image_array< T, Exact > >:



7.227.1 Detailed Description

template<class T, class Exact> struct oln::impl_traits< impl::image_array< T, Exact > >

The specialized version for impl::image_array<T, Exact>

Definition at line 80 of file image_array.hh.

The documentation for this struct was generated from the following file:

- image_array.hh

7.228 **oln::impl_traits< impl::image_impl< Exact > > Struct Template Reference**

```
#include <image_impl.hh>
```

7.228.1 Detailed Description

template<class Exact> struct oln::impl_traits< impl::image_impl< Exact > >

Specialized version for impl::image_impl<Exact>. Retrieve associated types.

Definition at line 58 of file image_impl.hh.

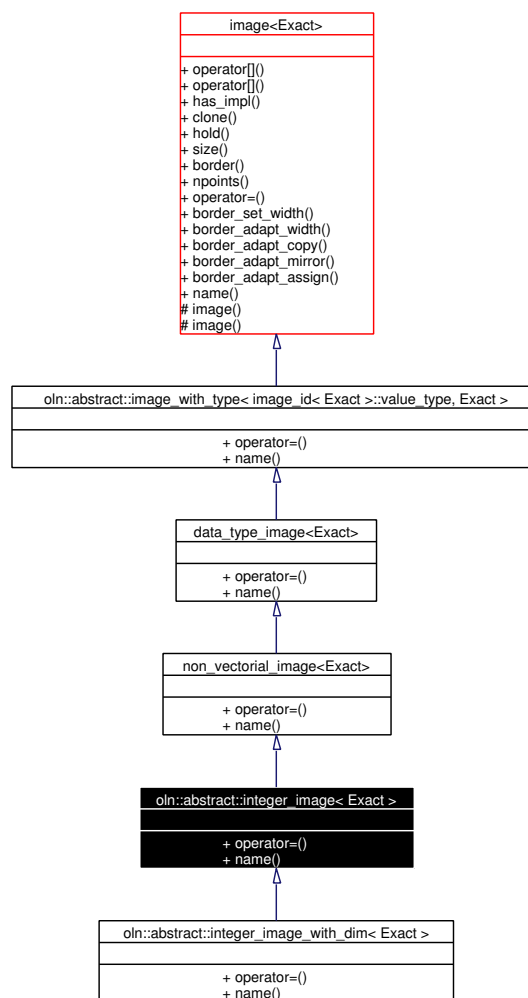
The documentation for this struct was generated from the following file:

- image_impl.hh

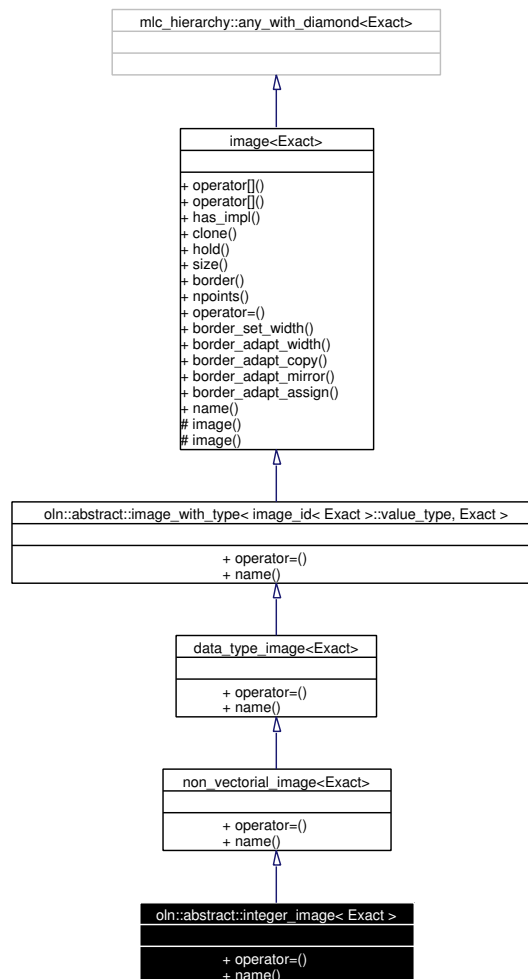
7.229 oln::abstract::integer_image< Exact > Class Template Reference

```
#include <image_with_type_with_dim.hh>
```

Inheritance diagram for oln::abstract::integer_image< Exact >:



Collaboration diagram for oln::abstract::integer_image< Exact >:



Public Types

- typedef [integer_image](#)< Exact > **self_type**
- typedef Exact **exact_type**

Public Member Functions

- exact_type & [operator=](#) (self_type rhs)

Perform a shallow copy between rhs and the current point, the points will not be duplicated but shared between the two images.

Static Public Member Functions

- std::string **name** ()

7.229.1 Detailed Description

template<class Exact> class oln::abstract::integer_image< Exact >

This class, when used in a function declaration, forces the function to only accept integer images.

Definition at line 237 of file image_with_type_with_dim.hh.

7.229.2 Member Function Documentation

7.229.2.1 **template<class Exact> exact_type& oln::abstract::integer_image< Exact >::operator=**
(self_type rhs) [inline]

Perform a shallow copy between *rhs* and the current point, the points will not be duplicated but shared between the two images.

See also:

[image::clone\(\)](#)

Reimplemented from [oln::abstract::non_vectorial_image< Exact >](#).

Reimplemented in [oln::abstract::integer_image_with_dim< Dim, Exact >](#).

Definition at line 237 of file image_with_type_with_dim.hh.

273 {

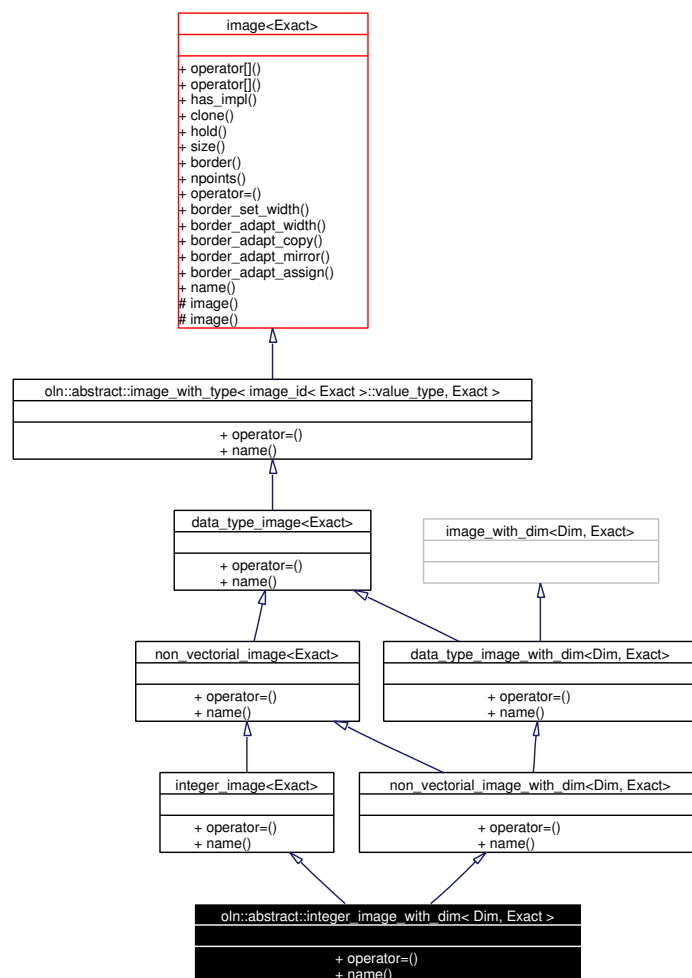
The documentation for this class was generated from the following file:

- image_with_type_with_dim.hh

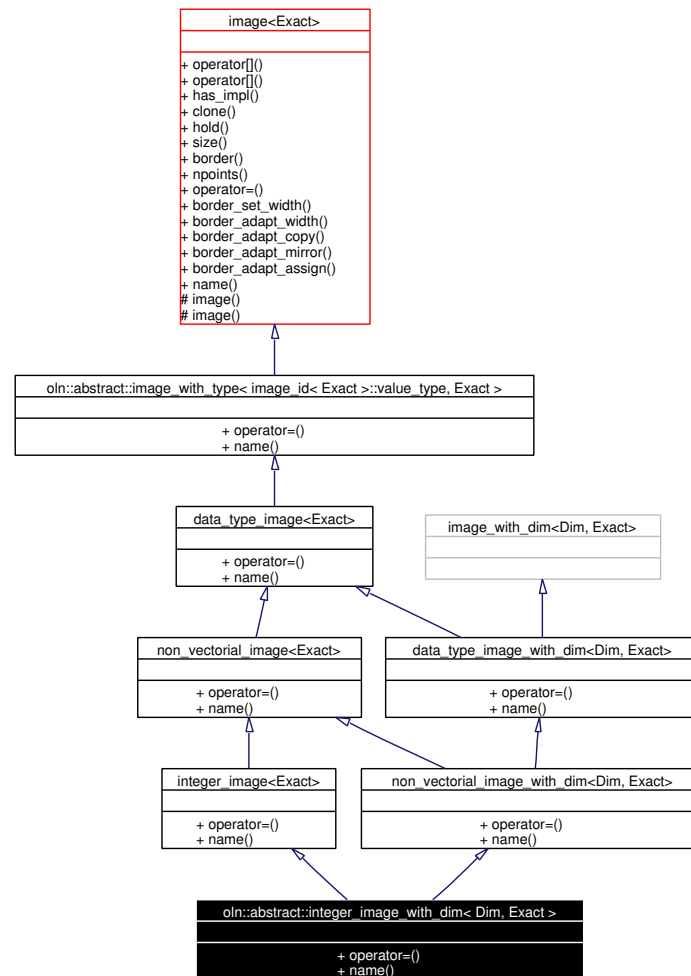
7.230 oln::abstract::integer_image_with_dim< Dim, Exact > Class Template Reference

```
#include <image_with_type_with_dim.hh>
```

Inheritance diagram for oln::abstract::integer_image_with_dim< Dim, Exact >:



Collaboration diagram for oln::abstract::integer_image_with_dim< Dim, Exact >:



Public Types

- typedef `integer_image_with_dim< Dim, Exact > self_type`
- typedef `Exact exact_type`

Public Member Functions

- `exact_type & operator= (self_type rhs)`

Perform a shallow copy between rhs and the current point, the points will not be duplicated but shared between the two images.

Static Public Member Functions

- `std::string name ()`

7.230.1 Detailed Description

`template<unsigned Dim, class Exact> class oln::abstract::integer_image_with_dim< Dim, Exact >`

This class, when used in a function declaration, forces the function to only accept integer images with a dimension of 'Dim'.

Definition at line 237 of file `image_with_type_with_dim.hh`.

7.230.2 Member Function Documentation

7.230.2.1 `template<unsigned Dim, class Exact> exact_type& oln::abstract::integer_image_with_dim< Dim, Exact >::operator= (self_type rhs)`
`[inline]`

Perform a shallow copy between *rhs* and the current point, the points will not be duplicated but shared between the two images.

See also:

[image::clone\(\)](#)

Reimplemented from `oln::abstract::non_vectorial_image_with_dim< Dim, Exact >`.

Definition at line 237 of file `image_with_type_with_dim.hh`.

273 {

The documentation for this class was generated from the following file:

- `image_with_type_with_dim.hh`

7.231 oln::morpho::attr::integral_type< T, Exact > Class Template Reference

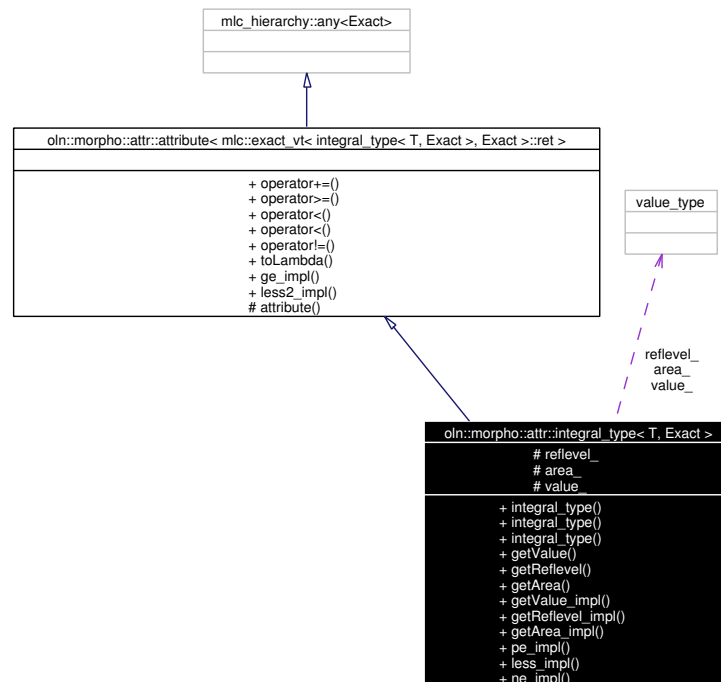
Integral attribute.

```
#include <attributes.hh>
```

Inheritance diagram for oln::morpho::attr::integral_type< T, Exact >:



Collaboration diagram for oln::morpho::attr::integral_type< T, Exact >:



Public Types

- typedef `integral_type< T, Exact >` `self_type`
- typedef `mlc::exact_vt< self_type, Exact >::ret exact_type`
- typedef `oln::morpho::attr::attr_traits< exact_type >::value_type value_type`
- typedef `oln::morpho::attr::attr_traits< exact_type >::env_type env_type`
- typedef `oln::morpho::attr::attr_traits< exact_type >::lambda_type lambda_type`

Public Member Functions

- `integral_type ()`
Basic Ctor.
- `integral_type (const lambda_type &lambda)`
Ctor from a lambda_type value.
- `template<class I> integral_type (const abstract::image< I > &input, const typename mlc::exact< I >::ret::point_type &p, const env_type &)`
Ctor from a point and an image.
- `const value_type &getValue () const`
Accessor to value_.
- `const value_type &getReflevel () const`
Accessor to the reference level.
- `const value_type &getArea () const`

Accessor to the current area.

- const value_type & [getValue_impl](#) () const
Implementation of [getValue\(\)](#).
- const value_type & [getReflevel_impl](#) () const
Implementation of [getReflevel\(\)](#).
- const value_type & [getArea_impl](#) () const
Implementation of [getArea\(\)](#).
- void [pe_impl](#) (const self_type &rhs)
+= operator implementation.
- bool [less_impl](#) (const lambda_type &lambda) const
"<" operator implementation.
- bool [ne_impl](#) (const lambda_type &lambda) const
!= operator implementation.

Protected Attributes

- value_type [reflevel_](#)
Reference level.
- value_type [area_](#)
Current area.
- value_type [value_](#)
Current value (deduced from area and level).

7.231.1 Detailed Description

template<class T = unsigned, class Exact = mlc::final> class oln::morpho::attr::integral_type< T, Exact >

Integral attribute.

It is equivalent to volume in 2D, and weight in 3D.

Definition at line 358 of file attributes.hh.

7.231.2 Member Typedef Documentation

7.231.2.1 **template**<class T = unsigned, class Exact = mlc::final> typedef [integral_type](#)<T, Exact> [oln::morpho::attr::integral_type](#)< T, Exact >::[self_type](#)

Self type of the class.

Reimplemented from [oln::morpho::attr::attribute< Exact >](#).

Definition at line 362 of file attributes.hh.

7.231.3 Constructor & Destructor Documentation

7.231.3.1 `template<class T = unsigned, class Exact = mlc::final> oln::morpho::attr::integral_type< T, Exact >::integral_type () [inline]`

Basic Ctor.

Warning:

After this call, the object is only instantiated (not initialized).

Definition at line 371 of file attributes.hh.

```
372         {
373     };
```

7.231.4 Member Function Documentation

7.231.4.1 `template<class T = unsigned, class Exact = mlc::final> const value_type& oln::morpho::attr::integral_type< T, Exact >::getArea () const [inline]`

Accessor to the current area.

See also:

[getArea_impl\(\)](#)

Definition at line 422 of file attributes.hh.

```
423         {
424             mlc_dispatch(getArea)();
425     };
```

7.231.4.2 `template<class T = unsigned, class Exact = mlc::final> const value_type& oln::morpho::attr::integral_type< T, Exact >::getArea_impl () const [inline]`

Implementation of [getArea\(\)](#).

Override this method in order to provide a new version of [getArea\(\)](#).

Warning:

Do not call this method, use [getArea\(\)](#) instead.

Definition at line 462 of file attributes.hh.

References [oln::morpho::attr::integral_type< T, Exact >::area_](#).

```
463         {
464             return area_;
465     };
```

7.231.4.3 `template<class T = unsigned, class Exact = mlc::final> const value_type&
oln::morpho::attr::integral_type< T, Exact >::getReflevel () const [inline]`

Accessor to the reference level.

See also:

[getReflevel_impl\(\)](#)

Definition at line 412 of file attributes.hh.

```
413         {  
414             mlc_dispatch(getReflevel)();  
415         };
```

7.231.4.4 `template<class T = unsigned, class Exact = mlc::final> const value_type&
oln::morpho::attr::integral_type< T, Exact >::getReflevel_impl () const [inline]`

Implementation of [getReflevel\(\)](#).

Override this method in order to provide a new version of [getReflevel\(\)](#).

Warning:

Do not call this method, use [getReflevel\(\)](#) instead.

Definition at line 449 of file attributes.hh.

References `oln::morpho::attr::integral_type< T, Exact >::reflevel_`.

```
450         {  
451             return reflevel_;  
452         };
```

7.231.4.5 `template<class T = unsigned, class Exact = mlc::final> const value_type&
oln::morpho::attr::integral_type< T, Exact >::getValue () const [inline]`

Accessor to value_.

Virtual method.

See also:

[getValue_impl\(\)](#)

Definition at line 402 of file attributes.hh.

```
403         {  
404             mlc_dispatch(getValue)();  
405         };
```

7.231.4.6 `template<class T = unsigned, class Exact = mlc::final> const value_type&
oln::morpho::attr::integral_type< T, Exact >::getValue_impl () const [inline]`

Implementation of `getValue()`.

Override this method in order to provide a new version of `getValue()`.

Warning:

Do not call this method, use `getValue()` instead.

Definition at line 436 of file attributes.hh.

```
437         {
438             return value_;
439         };
```

7.231.4.7 `template<class T = unsigned, class Exact = mlc::final> bool
oln::morpho::attr::integral_type< T, Exact >::less_impl (const lambda_type & lambda)
const [inline]`

"<" operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 487 of file attributes.hh.

```
488         {
489             return value_ < lambda;
490         };
```

7.231.4.8 `template<class T = unsigned, class Exact = mlc::final> bool
oln::morpho::attr::integral_type< T, Exact >::ne_impl (const lambda_type & lambda)
const [inline]`

!= operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 499 of file attributes.hh.

```
500         {
501             return lambda != value_;
502         };
```


7.231.4.9 `template<class T = unsigned, class Exact = mlc::final> void
 oln::morpho::attr::integral_type< T, Exact >::pe_impl (const self_type & rhs)
 [inline]`

`+=` operator implementation.

This is an implementation of the `+=` operator. Override this method to provide a new implementation of this operator.

Warning:

This method **SHOULDN'T** directly be called.

Definition at line 474 of file `attributes.hh`.

References `oln::morpho::attr::integral_type< T, Exact >::area_`, and `oln::morpho::attr::integral_type< T, Exact >::reflevel_`.

```
475         {  
476             value_ += rhs.getValue() + area_ * tools::diffabs(reflevel_, rhs.getReflevel());  
477             area_ += rhs.getArea();  
478         };
```

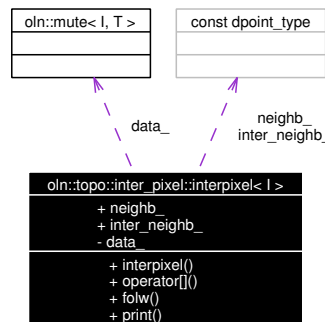
The documentation for this class was generated from the following file:

- `attributes.hh`

7.232 `oln::topo::inter_pixel::interpixel< I >` Class Template Reference

```
#include <inter-pixel.hh>
```

Collaboration diagram for `oln::topo::inter_pixel::interpixel< I >`:



Public Types

- typedef `mlc::exact< I >::ret::dpoint_type` **dpoint_type**
- typedef `mlc::exact< I >::ret::point_type` **point_type**
- typedef `oln::topo::inter_pixel::node< I >` **node_type**
- typedef `oln::topo::inter_pixel::fwd_dir_iter< I::dim >` **fwd_dir_iter_type**
- typedef `oln::topo::inter_pixel::bkd_dir_iter< I::dim >` **bkd_dir_iter_type**
- typedef `oln::topo::inter_pixel::internal::dir_traits< I::dim >::ret` **dir_type**
- typedef `oln::topo::inter_pixel::internal::dir_traits< I::dim >` **dir_traits_type**
- typedef `std::pair< typename mlc::exact< I >::ret::point_type, typename oln::topo::inter_pixel::internal::dir_traits< I::dim >::ret >` **head_type**
- typedef `oln::mute< I, oln::topo::inter_pixel::node< I >::ret` **inter_pixel_type**

Public Member Functions

- **interpixel** (const I &img)
Construct an inter pixel map of the image img.
- const **node_type** **operator[]** (const point_type &p) const
- head_type **folw** (const head_type &in) const
- std::ostream & **print** (std::ostream &ostr) const
Print the inter pixel map.

Static Public Attributes

- const dpoint_type **neighb_** [4]
- const dpoint_type **inter_neighb_** [4]

7.232.1 Detailed Description

template<class I> class oln::topo::inter_pixel::interpixel< I >

Inter pixel class.

This example give the node of the black connected component (bottom left).

```
#include <ntg/int.hh>
#include <oln/basics2d.hh>
#include <oln/topo/inter-pixel/inter-pixel.hh>
#include <iostream>
using namespace oln::topo::inter_pixel;

int main()
{
    typedef oln::image2d<ntg::int_u8> img_type;
    img_type in = oln::load(IMG_IN "test-cmap.pgm");
    interpixel<oln::image2d<ntg::int_u8> > ip(in);
    std::cout << ip << std::endl;
    // Print:
    // (5,0): east north south
    // (5,5): north west south
    // (7,5): east north west
    // (10,9): east north west
    // (10,11): north west south
    // (11,14): north west south
}
```

Todo

FIXME: Test the output values in the tests.

Definition at line 75 of file inter-pixel.hh.

7.232.2 Member Function Documentation

7.232.2.1 template<class I> head_type oln::topo::inter_pixel::interpixel< I >::folw (const head_type & in) const [inline]

Todo

FIXME: add doc.

Precondition:

precondition(data_[in.first].get(in.second) == true)

Definition at line 120 of file inter-pixel.hh.

```
121     {
122         precondition(data_[in.first].get(in.second) == true);
123     }
124     head_type out = in;
125
126     do
127     {
128         out.first += inter_neighb[out.second];
129
130         if (out.first == in.first || data_[out.first].rank() > 2)
131         {
132             out.second = dir_traits_type::opposite(out.second);
```

```

133             break;
134         }
135
136         dir_type next = dir_traits_type::next(out.second);
137         dir_type prev = dir_traits_type::prev(out.second);
138
139         out.second = data_[out.first].get(next) ? next :
140             data_[out.first].get(prev) ? prev : out.second;
141     }
142     while (out.first != in.first);
143
144     return out;
145 }

```

7.232.2.2]

template<class I> const [node_type oln::topo::inter_pixel::interpixel](#)< I >::operator[] (const point_type & p) const [inline]

Todo

FIXME: add doc.

Definition at line 110 of file inter-pixel.hh.

```

111     {
112         return data_[p];
113     }

```

7.232.3 Member Data Documentation

7.232.3.1 template<class I> const [mlc::exact](#)< I >::ret::dpoint_type
[oln::topo::inter_pixel::interpixel](#)< I >::inter_neighb_ [static]

Initial value:

```

{typename mlc::exact< I >::ret::dpoint_type(0,1),
                                     typename mlc::exact< I >::ret::dpoint_type(
                                     typename mlc::exact< I >::ret::dpoint_type(
                                     typename mlc::exact< I >::ret::dpoint_type(

```

Definition at line 198 of file inter-pixel.hh.

7.232.3.2 template<class I> const [mlc::exact](#)< I >::ret::dpoint_type
[oln::topo::inter_pixel::interpixel](#)< I >::neighb_ [static]

Initial value:

```

{typename mlc::exact< I >::ret::dpoint_type(0,0),
                                     typename mlc::exact< I >::ret::dpoint_type(-1,0),
                                     typename mlc::exact< I >::ret::dpoint_type(-1,-1),
                                     typename mlc::exact< I >::ret::dpoint_type(0,-1)}

```

Definition at line 192 of file inter-pixel.hh.

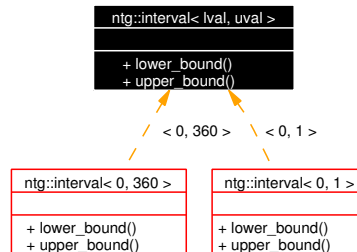
The documentation for this class was generated from the following file:

- inter-pixel.hh

7.233 ntg::interval< lval, uval > Struct Template Reference

```
#include <color.hh>
```

Inheritance diagram for ntg::interval< lval, uval >:



Static Public Member Functions

- `int lower_bound ()`
- `int upper_bound ()`

7.233.1 Detailed Description

`template<int lval, int uval> struct ntg::interval< lval, uval >`

Helper function to complete color_system (by inheritance).

Definition at line 223 of file color/color.hh.

The documentation for this struct was generated from the following file:

- color/color.hh

- `bool operator!= (const abstract::point< point_type > &p) const`
Compare with the current iterator's point.
 - *p The iterator's point to be compared to the current point.*
- `point_type operator+ (const abstract::dpoint< dpoint_type > &dp) const`
Sum a move point to the current point.
 - *p The move point.*
- `point_type operator- (const abstract::dpoint< dpoint_type > &dp) const`
Minor a move to the current point.
 - *p The move.*
- `operator point_type () const`
Cast to exact point type.
- `point_type cur () const`
Syntax improvement.
- `mlc::begin_type operator= (mlc::begin_type b)`
Set current point to the first iterator's point.
- `mlc::end_type operator= (mlc::end_type e)`
Set current point to the last iterator's point.
- `bool operator== (mlc::end_type) const`
Compare current point and last point.
- `void operator++ ()`
Go to the next iterator's point.
- `bool operator!= (mlc::end_type e) const`
Compare current point and last point.

Static Public Member Functions

- `std::string name ()`
Return the name of the type.

Protected Member Functions

- `iter ()`
Constructor.

Protected Attributes

- [point_type p_](#)

The iterator's current point.

7.234.1 Detailed Description

template<class Exact> struct oln::abstract::iter< Exact >

Iterator.

Allow iterable object (like image, window, ...) traversing.

Warning:

To know the type of iterator you need for an iterable object, use the macro `oln_iter_type(Iterable)` or `oln_iter_type_(Iterable)` (same without 'typename' keyword) rather than `Iterable::iter_type`.

Simple use of iterators:

```
#include <oln/basics2d.hh>
#include <ntg/all.hh>
#include <iostream>
#include <assert.h>
using namespace oln;
using namespace ntg;
int main(int argc, char **argv)
{
    image2d<bin> image1 = load(IMG_IN "se.pbm");
    assert(image1.has_impl());
    oln_iter_type_(image2d<bin>) i(image1);
    for_all(i)
    {
        std::cout << image1[i] << std::endl;
    }
}
```

This code is equivalent to the previous one but DEPRECATED (prefer to use the `for_all` macro):

```
#include <oln/basics2d.hh>
#include <ntg/all.hh>
#include <iostream>
#include <assert.h>
using namespace oln;
using namespace ntg;
int main(int argc, char **argv)
{
    image2d<bin> image1 = load(IMG_IN "se.pbm");
    assert(image1.has_impl());
    for (int row = 0; row < image1.nrows(); ++row)
        for (int col = 0; col < image1.ncols(); ++col)
            std::cout << image1(row, col) << std::endl;
}
```

You can use the same iterator on several image if they have the same size.

```
#include <oln/basics2d.hh>
#include <ntg/all.hh>
#include <iostream>
```



```

#include <assert.h>
using namespace oln;
using namespace ntg;
int main(int argc, char **argv)
{
    image2d<bin> image1 = load(IMG_IN "se.pbm");
    image2d<bin> image2 = load(IMG_IN "se.pbm");
    assert(image1.has_impl());
    assert(image2.has_impl());
    oln_iter_type_(image2d<bin>) i(image1);
    for_all(i)
    {
        std::cout << "image1:" << image1[i] << std::endl;
        std::cout << "image2:" << image2[i] << std::endl;
    }
    return 0;
}

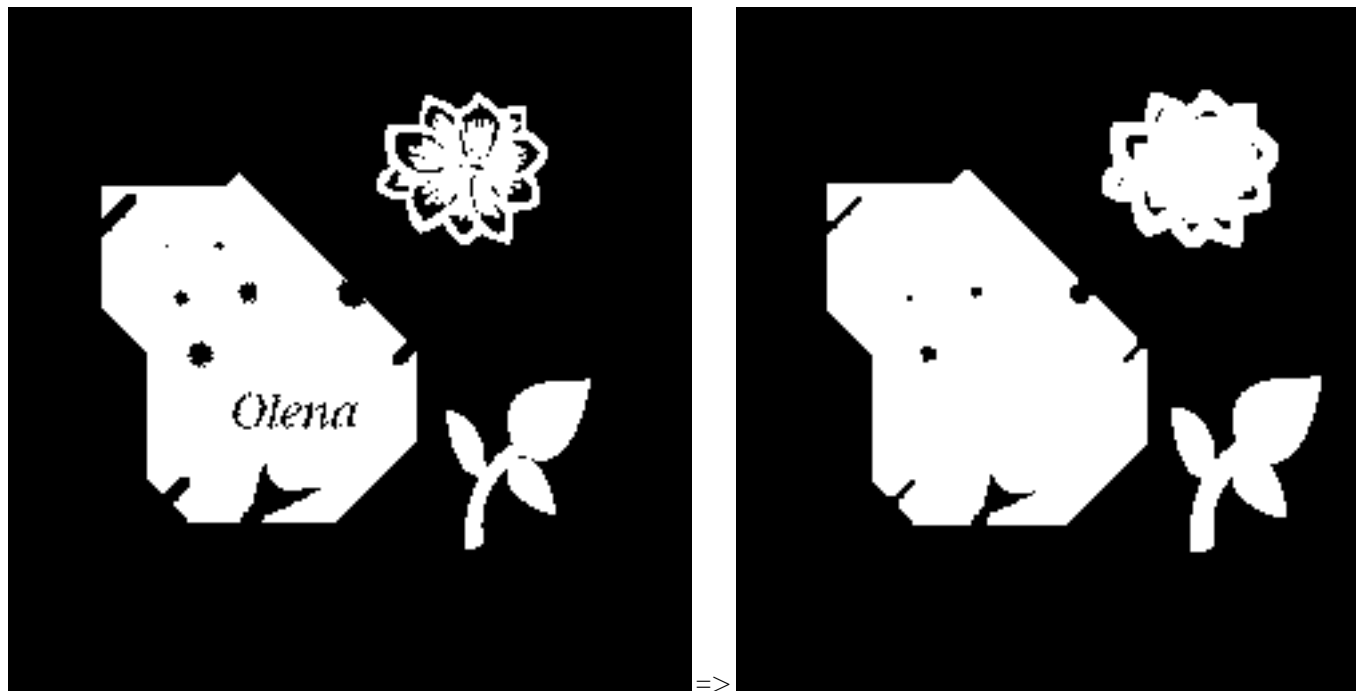
```

You can iterate not only image but windows. This example make a binary dilatation to show you how to use them:

```

#include <oln/basics2d.hh>
#include <ntg/all.hh>
#include <iostream>
#include <assert.h>
using namespace oln;
using namespace ntg;
int main(int argc, char **argv)
{
    image2d<bin> image1 = load(IMG_IN "object.pbm");
    assert(image1.has_impl());
    image2d<bin> image1_out(image1.size());
    oln_iter_type_(image2d<bin>) i(image1);
    for_all(i)
    {
        image1_out[i] = image1[i];
        if (!image1[i])
        {
            window2d win = win_c8_only();
            oln_iter_type_(window2d) j(win);
            bool change_color = false;
            for_all(j)
            {
                if (image1[i + j])
                    change_color = true;
            }
            image1_out[i] = change_color;
        }
    }
    save(image1_out, IMG_OUT "oln_abstract_iter.pbm");
    return 0;
}

```



Definition at line 179 of file iter.hh.

7.234.2 Member Typedef Documentation

7.234.2.1 `template<class Exact> typedef iter_traits<Exact>::dpoint_type oln::abstract::iter<Exact>::dpoint_type`

The associate image's type of dpoint (move point).

Warning:

Prefer the macros `oln_dpoint_type(Pointable)` and `oln_dpoint_type_(Pointable)` (the same without the 'typename' keyword)

Definition at line 192 of file iter.hh.

7.234.2.2 `template<class Exact> typedef iter_traits<Exact>::point_type oln::abstract::iter<Exact>::point_type`

The associate image's type of point.

Warning:

Prefer the macros `oln_point_type(Pointable)` and `oln_point_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented in `oln::bkd_iter1d< Exact >`, `oln::bkd_iter2d< Exact >`, `oln::bkd_iter3d< Exact >`, `oln::fwd_iter1d< Exact >`, `oln::fwd_iter2d< Exact >`, and `oln::fwd_iter3d< Exact >`.

Definition at line 186 of file iter.hh.

7.234.3 Constructor & Destructor Documentation

7.234.3.1 `template<class Exact> oln::abstract::iter< Exact >::iter () [inline, protected]`

Constructor.

Do nothing, used only by sub-classes

Definition at line 377 of file iter.hh.

```
378      {}
```

7.234.4 Member Function Documentation

7.234.4.1 `template<class Exact> point_type oln::abstract::iter< Exact >::cur () const [inline]`

Syntax improvement.

It's convenient to type 'it.cur()' instead of '(point)it' when necessary.

Definition at line 280 of file iter.hh.

```
281      {
282          return *this;
283      }
```

7.234.4.2 `template<class Exact> oln::abstract::iter< Exact >::operator point_type () const [inline]`

Cast to exact point type.

Returns:

The exact point type.

Return the exact point type by calling sub-classes methods.

Definition at line 268 of file iter.hh.

```
269      {
270          return this->exact().to_point();
271      }
```

7.234.4.3 `template<class Exact> bool oln::abstract::iter< Exact >::operator!= (mlc::end_type e) const [inline]`

Compare current point and last point.

Returns:

True if they are different.

Compare current point with last iterator's point.

Definition at line 356 of file iter.hh.

```

357     {
358         return ! this->operator==(e);
359     }

```

7.234.4.4 `template<class Exact> bool oln::abstract::iter< Exact >::operator!=(const abstract::point< point_type > &p) const` [inline]

Compare with the current iterator's point.

- p The iterator's point to be compared to the current point.

Compare the current iterator's point with p (his argument). If they are different, return true.

Definition at line 227 of file iter.hh.

```

228     {
229         return p_ != p.exact();
230     }

```

7.234.4.5 `template<class Exact> point_type oln::abstract::iter< Exact >::operator+ (const abstract::dpoint< dpoint_type > &dp) const` [inline]

Sum a move point to the current point.

- p The move point.

Returns:

The sum.

Precondition:

Instance != end.

Sum the current iterator's point and the move coordinates.

Definition at line 241 of file iter.hh.

```

242     {
243         precondition(*this != end);
244         return p_ + dp.exact();
245     }

```

7.234.4.6 `template<class Exact> void oln::abstract::iter< Exact >::operator++ ()` [inline]

Go to the next iterator's point.

Precondition:

Instance != end.

Definition at line 333 of file iter.hh.

```

334     {
335         precondition(*this != end);
336         this->exact().goto_next_();
337     }

```

7.234.4.7 `template<class Exact> point_type oln::abstract::iter< Exact >::operator- (const abstract::dpoint< dpoint_type > & dp) const [inline]`

Minor a move to the current point.

- p The move.

Returns:

The minoration.

Precondition:

Instance != end.

Minor the current point and the move coordinates.

Definition at line 256 of file iter.hh.

```

257     {
258         precondition(*this != end);
259         return p_ - dp.exact();
260     }
```

7.234.4.8 `template<class Exact> mlc::end_type oln::abstract::iter< Exact >::operator= (mlc::end_type e) [inline]`

Set current point to the last iterator's point.

Set current point of iterator to the last iterator's point.

Definition at line 310 of file iter.hh.

```

311     {
312         this->exact().goto_end();
313         return e;
314     }
```

7.234.4.9 `template<class Exact> mlc::begin_type oln::abstract::iter< Exact >::operator= (mlc::begin_type b) [inline]`

Set current point to the first iterator's point.

Set current point of iterator to the first iterator's point.

Definition at line 298 of file iter.hh.

```

299     {
300         this->exact().goto_begin();
301         return b;
302     }
```

7.234.4.10 `template<class Exact> bool oln::abstract::iter< Exact >::operator==(mlc::end_type) const [inline]`

Compare current point and last point.

Returns:

True if they are the same.

Compare current point with last iterator's point.

Definition at line 323 of file iter.hh.

```
324     {
325         return this->exact().is_at_end();
326     }
```

7.234.4.11 `template<class Exact> bool oln::abstract::iter< Exact >::operator==(const abstract::point< point_type > &p) const [inline]`

Compare with the current iterator's point.

- p The iterator's point to be compared to the current point.

Compare the current iterator's point with p (his argument). If successful, return true.

Definition at line 214 of file iter.hh.

Referenced by `oln::abstract::iter< mlc::exact_vt< bkd_iter2d< Exact >, Exact >::ret >::operator!=(())`.

```
215     {
216         return p_ == p.exact();
217     }
```

7.234.4.12 `template<class Exact> const point_type& oln::abstract::iter< Exact >::point_ref() const [inline]`

Accessor to current iterator's point.

Just return the current point of the iterator which is traversing an image.

Definition at line 201 of file iter.hh.

```
202     {
203         return p_;
204     }
```

The documentation for this struct was generated from the following file:

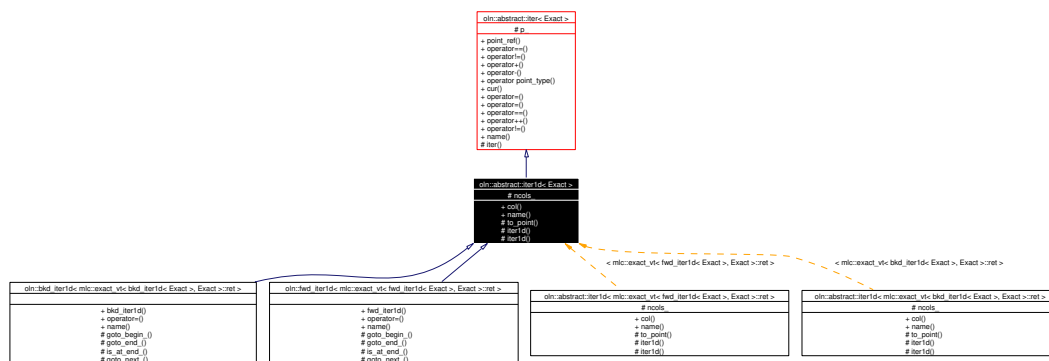
- iter.hh

7.235 oln::abstract::iter1d< Exact > Class Template Reference

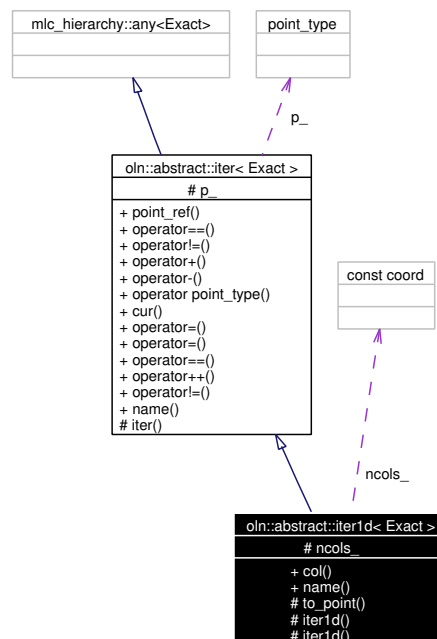
Iterator on image of 1 dimension.

```
#include <iter1d.hh>
```

Inheritance diagram for oln::abstract::iter1d< Exact >:



Collaboration diagram for oln::abstract::iter1d< Exact >:



Public Types

- typedef `iter< Exact >` `super_type`

The exact type of the object.

Public Member Functions

- `coord col ()` const

Get the coordinates of iterator's current point.

Static Public Member Functions

- `std::string name ()`

Return the name of the type.

Protected Member Functions

- `point1d to_point ()` const

Get the current point viewed by the iterator.

- `iter1d ()`

Constructor.

- `iter1d (const image1d_size &size)`

Constructor

– *size The size of the image to iterate.*

Protected Attributes

- const `coord ncols_`

The number of columns of the image you are iterating.

Friends

- class `iter< Exact >`

7.235.1 Detailed Description

`template<class Exact> class oln::abstract::iter1d< Exact >`

Iterator on image of 1 dimension.

Allow iterable object (like image, window, ...) of 1 dimension traversing.

See also:

[iter](#)

Definition at line 70 of file `iter1d.hh`.

7.235.2 Constructor & Destructor Documentation

7.235.2.1 `template<class Exact> oln::abstract::iter1d< Exact >::iter1d (const image1d_size & size) [inline, protected]`

Constructor

- size The size of the image to iterate.

Precondition:

size.ncols() > 0.

Construct an iterator (1d) on an image (1d).

Definition at line 128 of file iter1d.hh.

```

128                                     :
129     super_type(), ncols_(size.ncols())
130     {
131         precondition(size.ncols() > 0);
132         this->exact().goto_begin();
133     }
    };

```

7.235.3 Member Function Documentation

7.235.3.1 `template<class Exact> coord oln::abstract::iter1d< Exact >::col () const [inline]`

Get the coordinates of iterator's current point.

On this kind of image, all point are on the same line. So you are able to get the column number by this way (and never the line number).

Definition at line 86 of file iter1d.hh.

```

87     {
88         return this->p_.col();
89     }

```

7.235.3.2 `template<class Exact> point1d oln::abstract::iter1d< Exact >::to_point () const [inline, protected]`

Get the current point viewed by the iterator.

Returns:

The point (1 dimension) viewed by the iterator.

Precondition:

Instance != end.

Definition at line 107 of file iter1d.hh.

```
108     {
109         precondition(*this != end);
110         invariant(this->p_.col() >= 0 &&
111                 this->p_.col() < ncols_);
112         return this->p_;
113     }
```

The documentation for this class was generated from the following file:

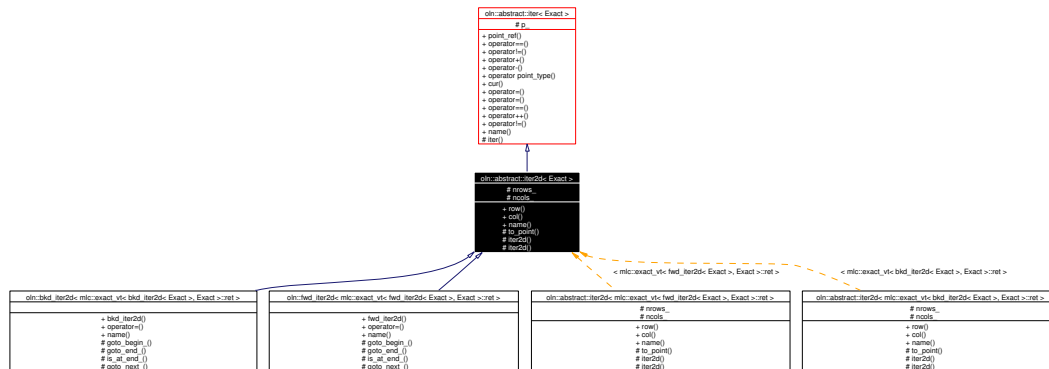
- iter1d.hh

7.236 oln::abstract::iter2d< Exact > Class Template Reference

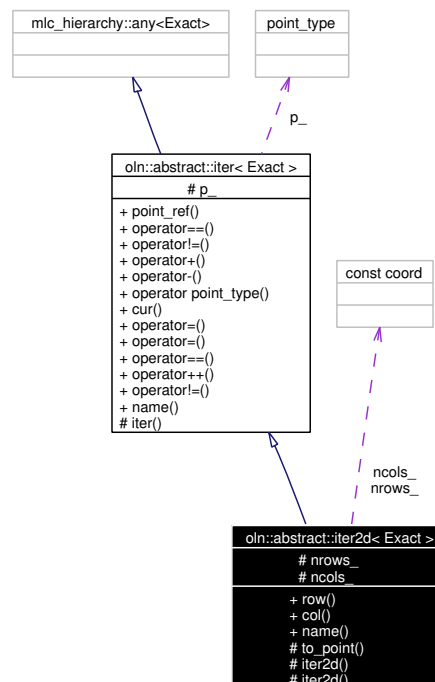
Iterator on image of 2 dimensions.

```
#include <iter2d.hh>
```

Inheritance diagram for oln::abstract::iter2d< Exact >:



Collaboration diagram for oln::abstract::iter2d< Exact >:



Public Types

- typedef [iter< Exact >](#) [super_type](#)

The exact type of the object.

Public Member Functions

- `coord row () const`
Get the coordinates (rows) of iterator's current point.
- `coord col () const`
Get the coordinates (columns) of iterator's current point.

Static Public Member Functions

- `std::string name ()`
Return the name of the type.

Protected Member Functions

- `point2d to_point () const`
Get the current point viewed by the iterator.
- `iter2d ()`
Constructor.
- `iter2d (const image2d_size &size)`
Construct an iterator (2d) on an image (2d).
 - *size The size of the image to iterate.*

Protected Attributes

- `const coord nrows_`
The number of rows of the image you are iterating.
- `const coord ncols_`
The number of columns of the image you are iterating.

Friends

- `class iter< Exact >`

7.236.1 Detailed Description

`template<class Exact> class oln::abstract::iter2d< Exact >`

Iterator on image of 2 dimensions.

Allow iterable object (like image, window, ...) of 2 dimensions traversing.

See also:

[iter](#)

Definition at line 70 of file iter2d.hh.

7.236.2 Constructor & Destructor Documentation

7.236.2.1 `template<class Exact> oln::abstract::iter2d< Exact >::iter2d (const image2d_size & size) [inline, protected]`

Construct an iterator (2d) on an inamge (2d).

- size The size of the image to iterate.

Precondition:

size.ncols() > 0.

size.nrows() > 0.

Definition at line 143 of file iter2d.hh.

```

143                                     :
144     super_type(),
145     nrows_(size.nrows()),
146     ncols_(size.ncols())
147     {
148     precondition(size.nrows() > 0 &&
149                 size.ncols() > 0);
150     this->exact().goto_begin();
151     }
    };

```

7.236.3 Member Function Documentation

7.236.3.1 `template<class Exact> coord oln::abstract::iter2d< Exact >::col () const [inline]`

Get the coordinates (columns) of iterator's current point.

Returns:

The column number.

On this kind of image (i.e. 2 dimensions), you are able to get the column number and the row number.

Definition at line 99 of file iter2d.hh.

```

100     {
101     return this->p_.col();
102     }

```

7.236.3.2 `template<class Exact> coord oln::abstract::iter2d< Exact >::row () const [inline]`

Get the coordinates (rows) of iterator's current point.

Returns:

The row number.

On this kind of image (i.e. 2 dimensions), you are able to get the column number and the row number.

Definition at line 86 of file iter2d.hh.

```
87     {
88         return this->p_.row();
89     }
```

7.236.3.3 `template<class Exact> point2d oln::abstract::iter2d< Exact >::to_point () const`
[inline, protected]

Get the current point viewed by the iterator.

Returns:

The point (2 dimensions) viewed by the iterator.

Precondition:

Instance != end.

Definition at line 121 of file iter2d.hh.

```
122     {
123         precondition(*this != end);
124         invariant(this->p_.row() >= 0 &&
125                 (this->p_.row() < nrows_ || this->p_.row() == nrows_) &&
126                 this->p_.col() >= 0 &&
127                 this->p_.col() < ncols_);
128         return this->p_;
129     }
```

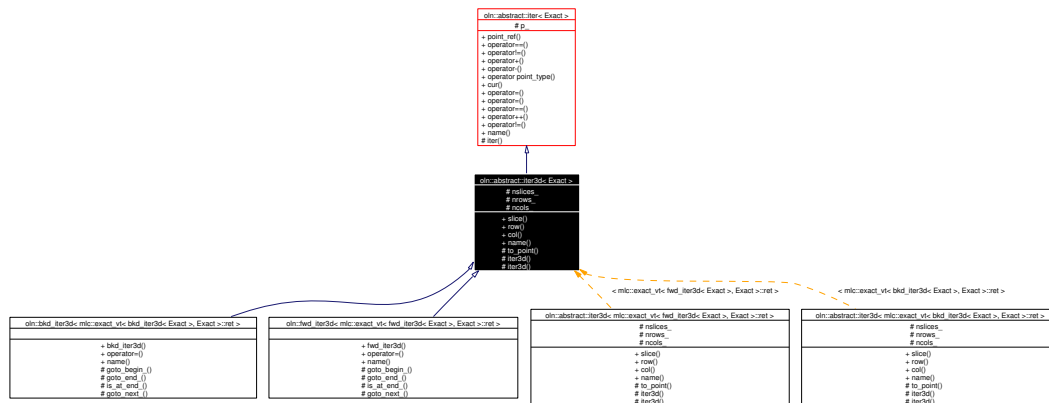
The documentation for this class was generated from the following file:

- iter2d.hh

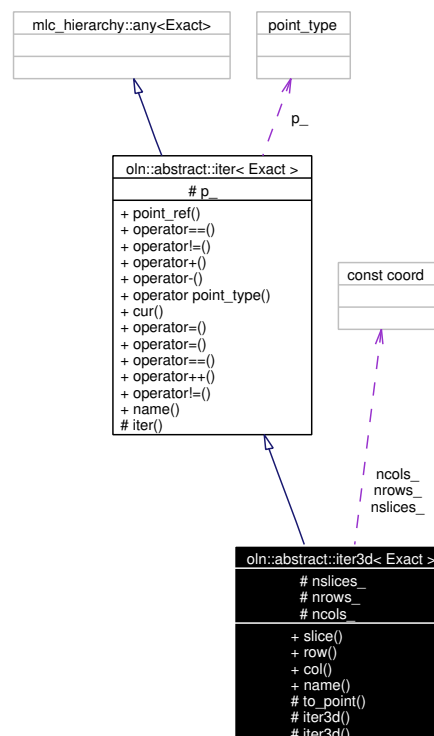
7.237 oln::abstract::iter3d< Exact > Class Template Reference

```
#include <iter3d.hh>
```

Inheritance diagram for oln::abstract::iter3d< Exact >:



Collaboration diagram for oln::abstract::iter3d< Exact >:



Public Types

- typedef `iter< Exact >` `super_type`

The exact type of the object.

Public Member Functions

- `coord slice () const`
Get the coordinates (slice) of iterator's current point.
- `coord row () const`
Get the coordinates (row) of iterator's current point.
- `coord col () const`
Get the coordinates (col) of iterator's current point.

Static Public Member Functions

- `std::string name ()`
Return the name of the type.

Protected Member Functions

- `point3d to_point () const`
Get the current point viewed by the iterator.
- `iter3d ()`
Constructor.
- `iter3d (const image3d_size &size)`
Construct an iterator (3d) on an image (3d).
 - *size The size of the image to iterate.*

Protected Attributes

- `const coord nslices_`
The number of slices of the image you are iterating.
- `const coord nrows_`
The number of rows of the image you are iterating.
- `const coord ncols_`
The number of columns of the image you are iterating.

Friends

- `class iter< Exact >`

7.237.1 Detailed Description

template<class Exact> class oln::abstract::iter3d< Exact >

Iterator on image of 3 dimensions.

Allow iterable object (like image, window, ...) of 3 dimensions traversing.

See also:

[iter](#)

Definition at line 70 of file iter3d.hh.

7.237.2 Constructor & Destructor Documentation

7.237.2.1 template<class Exact> oln::abstract::iter3d< Exact >::iter3d (const image3d_size & size) [inline, protected]

Construct an iterator (3d) on an image (3d).

- size The size of the image to iterate.

Precondition:

size.ncols() > 0.

size.nrows() > 0.

size.nsllices() > 0.

Definition at line 160 of file iter3d.hh.

```

160                                     :
161         super_type(),
162         nslices_(size.nsllices()),
163         nrows_(size.nrows()),
164         ncols_(size.ncols())
165     {
166         precondition(size.nsllices() > 0
167                     && size.nrows() > 0
168                     && size.ncols() > 0);
169         this->exact().goto_begin();
170     }
    };

```

7.237.3 Member Function Documentation

7.237.3.1 template<class Exact> coord oln::abstract::iter3d< Exact >::col () const [inline]

Get the coordinates (col) of iterator's current point.

Returns:

The col number.

On this kind of image (i.e. 3 dimensions), you are able to get the column number, the row number and the slice number.

Definition at line 112 of file iter3d.hh.

```
113     {  
114         return this->p_.col();  
115     }
```

7.237.3.2 `template<class Exact> coord oln::abstract::iter3d< Exact >::row () const` [inline]

Get the coordinates (row) of iterator's current point.

Returns:

The row number.

On this kind of image (i.e. 3 dimensions), you are able to get the column number, the row number and the slice number.

Definition at line 99 of file iter3d.hh.

```
100     {  
101         return this->p_.row();  
102     }
```

7.237.3.3 `template<class Exact> coord oln::abstract::iter3d< Exact >::slice () const` [inline]

Get the coordinates (slice) of iterator's current point.

Returns:

The slice number.

On this kind of image (i.e. 3 dimensions), you are able to get the column number, the row number and the slice number.

Definition at line 86 of file iter3d.hh.

```
87     {  
88         return this->p_.slice();  
89     }
```

7.237.3.4 `template<class Exact> point3d oln::abstract::iter3d< Exact >::to_point () const` [inline, protected]

Get the current point viewed by the iterator.

Returns:

The point (3 dimensions) viewed by the iterator.

Precondition:

Instance != end.

Definition at line 135 of file iter3d.hh.

```
136     {
137         precondition(*this != end);
138         invariant(this->p_.slice() >=0
139                 && this->p_.slice() < nslices_
140                 && this->p_.row() >= 0
141                 && this->p_.row() < nrows_
142                 && this->p_.col() >= 0
143                 && this->p_.col() < ncols_);
144         return this->p_;
145     }
```

The documentation for this class was generated from the following file:

- iter3d.hh

7.238 oln::morpher::iter_morpher< const SrcType, IterType, Exact > Struct Template Reference

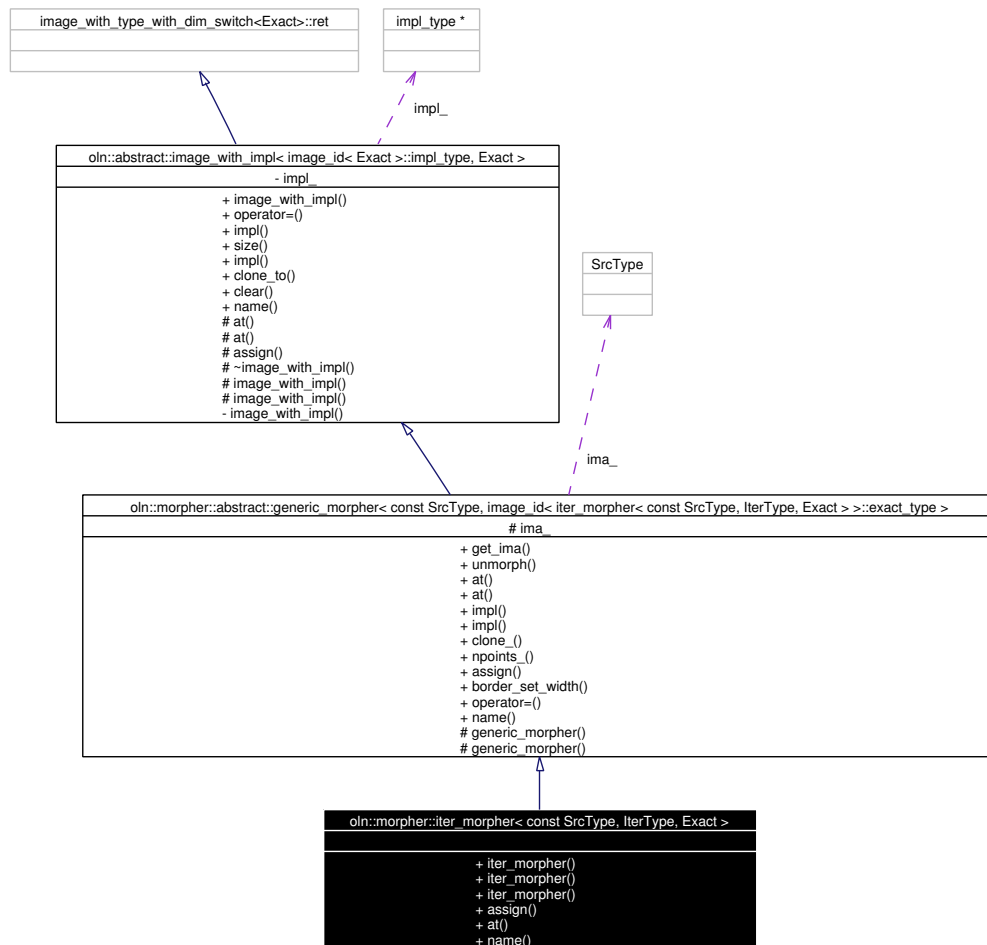
The specialized version for 'const' declared images.

```
#include <iter_morpher.hh>
```

Inheritance diagram for oln::morpher::iter_morpher< const SrcType, IterType, Exact >:



Collaboration diagram for oln::morpher::iter_morpher< const SrcType, IterType, Exact >:



Public Types

- typedef `iter_morpher< const SrcType, IterType, Exact >` [self_type](#)
The *self* type.
- typedef `image_id< self_type >::exact_type exact_type`
The *exact* type of the morpher.
- typedef `abstract::generic_morpher< const SrcType, exact_type >` [super_type](#)
The *upper* class.
- typedef `image_id< exact_type >::iter_type iter_type`
The *morpher* iterator type.
- typedef `image_id< exact_type >::value_type value_type`
The *morpher* value type.
- typedef `image_id< exact_type >::point_type point_type`
The *morpher* point type.

Public Member Functions

- `iter_morpher` (const SrcType &ima)
Construct the iter morpher with an image ima.
- `iter_morpher` (const `self_type` &r)
Construct the iter morpher with another iter morpher.
- `iter_morpher` ()
Empty constructor.
- `self_type` & `assign` (`self_type` &rhs)
- const `value_type` at (const `point_type` &p) const
Return the stored value at the point.
– *p The point.*

Static Public Member Functions

- std::string `name` ()
Useful to debug.

7.238.1 Detailed Description

template<class SrcType, class IterType, class Exact> struct oln::morpher::iter_morpher< const SrcType, IterType, Exact >

The specialized version for 'const' declared images.

Definition at line 162 of file `iter_morpher.hh`.

7.238.2 Member Function Documentation

7.238.2.1 **template<class SrcType, class IterType, class Exact> `self_type` & oln::morpher::iter_morpher< const SrcType, IterType, Exact >::assign (`self_type` & rhs) [inline]**

Perform a shallow copy from the decorated image of *rhs* to the current decorated image. The points will be shared by the two images.

Definition at line 202 of file `iter_morpher.hh`.

```

203     {
204         oln_iter_type(SrcType)  it(rhs);
205
206         for_all(it)
207             this->at(it) = rhs[it];
208         return this->exact();
209     }
```

7.238.2.2 `template<class SrcType, class IterType, class Exact> const value_type
 oln::morpher::iter_morpher< const SrcType, IterType, Exact >::at (const point_type &
 p) const` `[inline]`

Return the stored value at the point.

- p The point.

Returns:

The stored value.

Reimplemented from `oln::abstract::image_with_impl< Impl, Exact >`.

Definition at line 217 of file `iter_morpher.hh`.

```
218     {
219         return this->ima_[p];
220     }
```

7.238.2.3 `template<class SrcType, class IterType, class Exact> oln::morpher::iter_morpher<
 const SrcType, IterType, Exact >::iter_morpher ()` `[inline]`

Empty constructor.

Needed by `mlc_hierarchy::any_with_diamond`.

Definition at line 195 of file `iter_morpher.hh`.

```
195 {}
```

The documentation for this struct was generated from the following file:

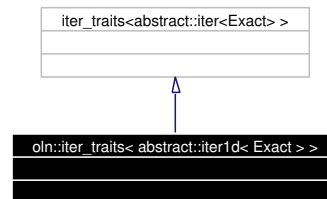
- `iter_morpher.hh`

7.239 oln::iter_traits< abstract::iter1d< Exact > > Struct Template Reference

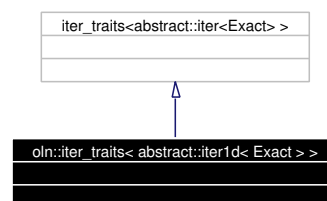
Traits for iter::iter1d.

```
#include <iter1d.hh>
```

Inheritance diagram for oln::iter_traits< abstract::iter1d< Exact > >:



Collaboration diagram for oln::iter_traits< abstract::iter1d< Exact > >:



Public Types

- typedef [point1d](#) [point_type](#)
The type of point of the image.
- typedef [dpoint1d](#) [dpoint_type](#)
The type of dpoint of the image.
- enum { **dim** = 1 }

7.239.1 Detailed Description

```
template<class Exact> struct oln::iter_traits< abstract::iter1d< Exact > >
```

Traits for iter::iter1d.

Definition at line 52 of file iter1d.hh.

The documentation for this struct was generated from the following file:

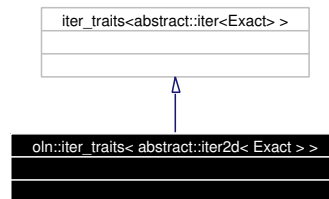
- iter1d.hh

7.240 oln::iter_traits< abstract::iter2d< Exact > > Struct Template Reference

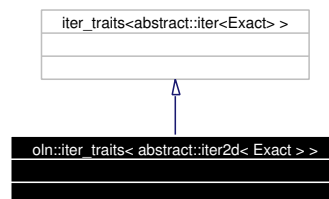
Traits for [abstract::iter2d](#).

```
#include <iter2d.hh>
```

Inheritance diagram for oln::iter_traits< abstract::iter2d< Exact > >:



Collaboration diagram for oln::iter_traits< abstract::iter2d< Exact > >:



Public Types

- typedef [point2d](#) [point_type](#)
The type of point of the image.
- typedef [dpoint2d](#) [dpoint_type](#)
The type of dpoint of the image.
- enum { **dim** = 2 }

7.240.1 Detailed Description

```
template<class Exact> struct oln::iter_traits< abstract::iter2d< Exact > >
```

Traits for [abstract::iter2d](#).

Definition at line 52 of file iter2d.hh.

The documentation for this struct was generated from the following file:

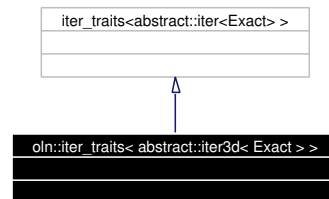
- iter2d.hh

7.241 oln::iter_traits< abstract::iter3d< Exact > > Struct Template Reference

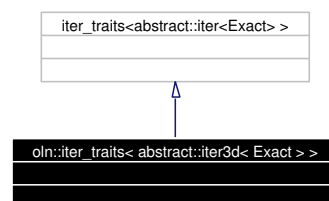
Traits for [abstract::iter3d](#).

```
#include <iter3d.hh>
```

Inheritance diagram for oln::iter_traits< abstract::iter3d< Exact > >:



Collaboration diagram for oln::iter_traits< abstract::iter3d< Exact > >:



Public Types

- typedef [point3d](#) [point_type](#)
The type of point of the image.
- typedef [dpoint3d](#) [dpoint_type](#)
The type of dpoint of the image.
- enum { **dim** = 3 }

7.241.1 Detailed Description

```
template<class Exact> struct oln::iter_traits< abstract::iter3d< Exact > >
```

Traits for [abstract::iter3d](#).

Definition at line 52 of file iter3d.hh.

The documentation for this struct was generated from the following file:

- iter3d.hh

7.242 oln::iter_traits< abstract::iter< Exact > > Struct Template Reference

Traits for [abstract::iter](#).

```
#include <iter.hh>
```

7.242.1 Detailed Description

```
template<class Exact> struct oln::iter_traits< abstract::iter< Exact > >
```

Traits for [abstract::iter](#).

Definition at line 53 of file iter.hh.

The documentation for this struct was generated from the following file:

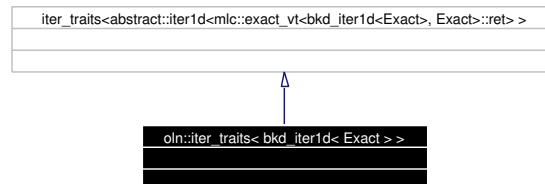
- iter.hh

7.243 oln::iter_traits< bkd_iter1d< Exact > > Struct Template Reference

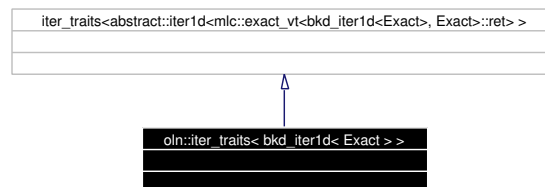
Traits for [bkd_iter1d](#).

```
#include <bkd_iter1d.hh>
```

Inheritance diagram for oln::iter_traits< bkd_iter1d< Exact > >:



Collaboration diagram for oln::iter_traits< bkd_iter1d< Exact > >:



Public Types

- typedef [point1d](#) `point_type`
- typedef [dpoint1d](#) `dpoint_type`

7.243.1 Detailed Description

```
template<class Exact> struct oln::iter_traits< bkd_iter1d< Exact > >
```

Traits for [bkd_iter1d](#).

Definition at line 42 of file `bkd_iter1d.hh`.

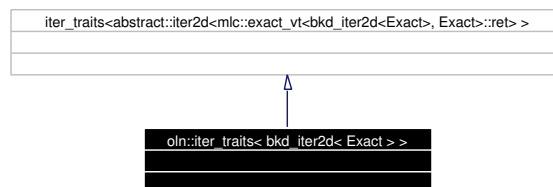
The documentation for this struct was generated from the following file:

- `bkd_iter1d.hh`

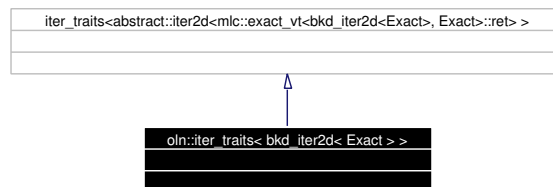
7.244 oln::iter_traits< bkd_iter2d< Exact > > Struct Template Reference

```
#include <bkd_iter2d.hh>
```

Inheritance diagram for oln::iter_traits< bkd_iter2d< Exact > >:



Collaboration diagram for oln::iter_traits< bkd_iter2d< Exact > >:



Public Types

- typedef [point2d](#) point_type
- typedef [dpoint2d](#) dpoint_type

7.244.1 Detailed Description

```
template<class Exact> struct oln::iter_traits< bkd_iter2d< Exact > >
```

Traits for [bkd_iter2d](#)

Definition at line 42 of file bkd_iter2d.hh.

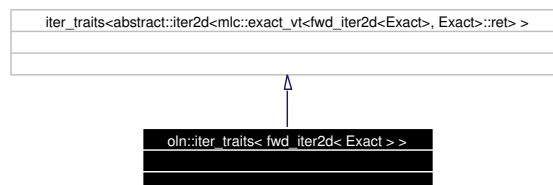
The documentation for this struct was generated from the following file:

- bkd_iter2d.hh

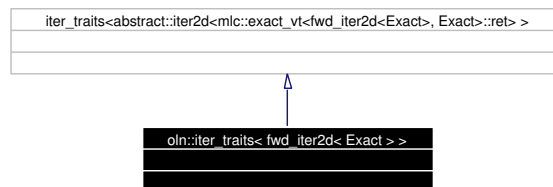
7.245 oln::iter_traits< fwd_iter2d< Exact > > Struct Template Reference

```
#include <fwd_iter2d.hh>
```

Inheritance diagram for oln::iter_traits< fwd_iter2d< Exact > >:



Collaboration diagram for oln::iter_traits< fwd_iter2d< Exact > >:



Public Types

- typedef [point2d](#) point_type
- typedef [dpoint2d](#) dpoint_type

7.245.1 Detailed Description

```
template<class Exact> struct oln::iter_traits< fwd_iter2d< Exact > >
```

Traits for [fwd_iter2d](#)

Definition at line 42 of file fwd_iter2d.hh.

The documentation for this struct was generated from the following file:

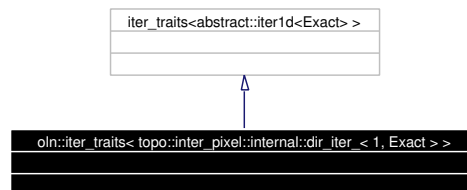
- fwd_iter2d.hh

7.246 oln::iter_traits< topo::inter_pixel::internal::dir_iter_< 1, Exact > > Struct Template Reference

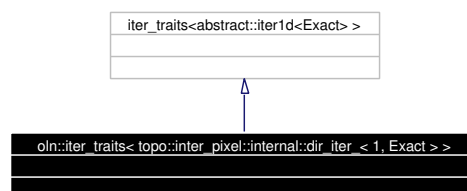
Traits for iterator for 1D directions.

```
#include <dir-iter.hh>
```

Inheritance diagram for oln::iter_traits< topo::inter_pixel::internal::dir_iter_< 1, Exact > >:



Collaboration diagram for oln::iter_traits< topo::inter_pixel::internal::dir_iter_< 1, Exact > >:



Public Types

- typedef [abstract::iter1d](#)< Exact > **super_type**
- typedef [point1d](#) **point_type**
- typedef [dpoint1d](#) **dpoint_type**

7.246.1 Detailed Description

template<class Exact> struct oln::iter_traits< topo::inter_pixel::internal::dir_iter_< 1, Exact > >

Traits for iterator for 1D directions.

Definition at line 52 of file dir-iter.hh.

The documentation for this struct was generated from the following file:

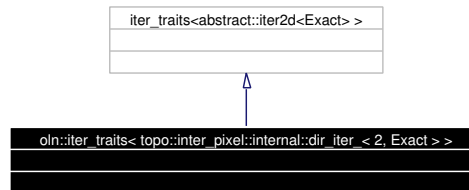
- dir-iter.hh

7.247 oln::iter_traits< topo::inter_pixel::internal::dir_iter_< 2, Exact > > Struct Template Reference

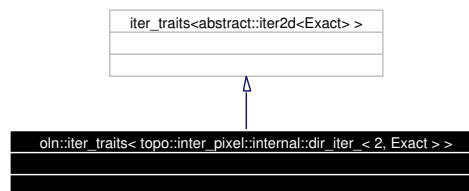
Traits for iterator for 2D directions.

```
#include <dir-iter.hh>
```

Inheritance diagram for oln::iter_traits< topo::inter_pixel::internal::dir_iter_< 2, Exact > >:



Collaboration diagram for oln::iter_traits< topo::inter_pixel::internal::dir_iter_< 2, Exact > >:



Public Types

- typedef [abstract::iter2d](#)< Exact > **super_type**
- typedef [point2d](#) **point_type**
- typedef [dpoint2d](#) **dpoint_type**

7.247.1 Detailed Description

```
template<class Exact> struct oln::iter_traits< topo::inter_pixel::internal::dir_iter_< 2, Exact > >
```

Traits for iterator for 2D directions.

Definition at line 62 of file dir-iter.hh.

The documentation for this struct was generated from the following file:

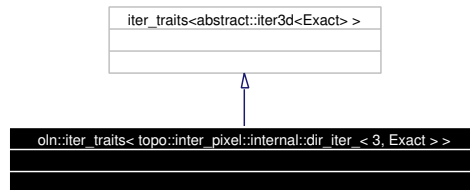
- dir-iter.hh

7.248 `oln::iter_traits< topo::inter_pixel::internal::dir_iter_< 3, Exact > >` Struct Template Reference

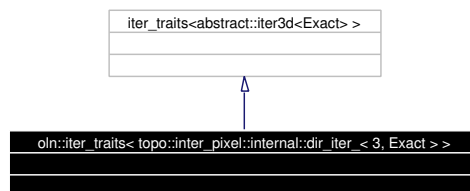
Traits for iterator for 3D directions.

```
#include <dir-iter.hh>
```

Inheritance diagram for `oln::iter_traits< topo::inter_pixel::internal::dir_iter_< 3, Exact > >`:



Collaboration diagram for `oln::iter_traits< topo::inter_pixel::internal::dir_iter_< 3, Exact > >`:



Public Types

- typedef `abstract::iter3d< Exact >` `super_type`
- typedef `point3d` `point_type`
- typedef `dpoint3d` `dpoint_type`

7.248.1 Detailed Description

`template<class Exact> struct oln::iter_traits< topo::inter_pixel::internal::dir_iter_< 3, Exact > >`

Traits for iterator for 3D directions.

Definition at line 72 of file `dir-iter.hh`.

The documentation for this struct was generated from the following file:

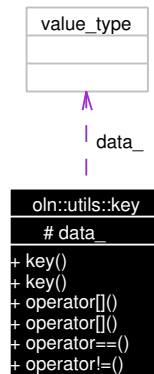
- `dir-iter.hh`

7.249 oln::utils::key Class Reference

16 bytes key

```
#include <key.hh>
```

Collaboration diagram for oln::utils::key:



Public Types

- typedef ntg::int_u8 [value_type](#)
Used data type.

Public Member Functions

- [key](#) (const std::vector< [value_type](#) > &data)
Constructor used to initialize the key.
- [key](#) (const [value_type](#) *data)
Constructor used to initialize the key.
- [value_type](#) & [operator](#)[] (unsigned i)
[] operator.
- const [value_type](#) & [operator](#)[] (unsigned i) const
[] operator.
- bool [operator](#)== (const [key](#) &k)
Check equality between two keys.
- bool [operator](#)!= (const [key](#) &k)
Check whether two key are different.

Protected Attributes

- [value_type data_](#) [16]

Internal data.

Friends

- `std::ostream & operator<< (std::ostream &stream, const key &k)`

Write of representation of the key on a stream.

7.249.1 Detailed Description

16 bytes key

Object to have a nice representation of a [MD5](#) result.

Definition at line 41 of file key.hh.

7.249.2 Constructor & Destructor Documentation

7.249.2.1 `oln::utils::key::key (const std::vector< value_type > &data) [inline, explicit]`

Constructor used to initialize the key.

Precondition:

There must at least 16 elements in data.

- data Input data.

Definition at line 150 of file key.hh.

7.249.2.2 `oln::utils::key::key (const value_type *data) [inline, explicit]`

Constructor used to initialize the key.

Precondition:

There must at least 16 elements in data.

- data Input data.

Definition at line 158 of file key.hh.

7.249.3 Member Function Documentation

7.249.3.1 `bool oln::utils::key::operator!= (const key &k) [inline]`

Check whether two key are different.

- k Key to be compared with.

Definition at line 192 of file key.hh.

7.249.3.2 `bool oln::utils::key::operator==(const key & k)` [inline]

Check equality between two keys.

- *k* Key to be compared with.

Definition at line 182 of file key.hh.

7.249.3.3 `[]`

`const key::value_type & oln::utils::key::operator[] (unsigned i) const` [inline]
`[] operator.`

Returns:

the *i*th byte of the key.

- *i* Index of the byte wanted.

This is the const version of the operator.

Definition at line 174 of file key.hh.

7.249.3.4 `[]`

`key::value_type & oln::utils::key::operator[] (unsigned i)` [inline]
`[] operator.`

Returns:

the *i*th byte of the key.

- *i* Index of the byte wanted.

This is the non const version of the operator.

Definition at line 166 of file key.hh.

7.249.4 Friends And Related Function Documentation

7.249.4.1 `std::ostream& operator<< (std::ostream & stream, const key & k)` [friend]

Write of representation of the key on a stream.

stream Stream to put the key on. *k* Key to represent.

Definition at line 104 of file key.hh.

```

105     {
106         stream << "{";
107         for (unsigned i = 0; i < 15; ++i)
108             stream << "0x" << std::hex << k[i] << ", ";
109         stream << "0x" << std::hex << k[15] << "}";
110         return stream;
111     }
```

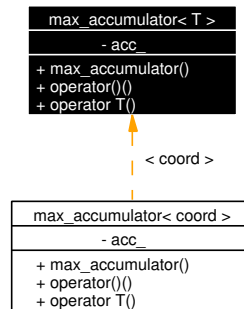
The documentation for this class was generated from the following file:

- key.hh

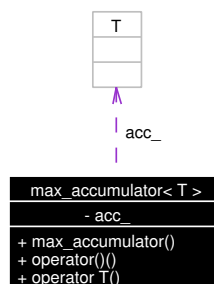
7.250 max_accumulator< T > Struct Template Reference

```
#include <accum.hh>
```

Inheritance diagram for max_accumulator< T >:



Collaboration diagram for max_accumulator< T >:



Public Member Functions

- **max_accumulator** (T t)
- void **operator()** (T t)
- **operator T** () const

7.250.1 Detailed Description

```
template<class T> struct max_accumulator< T >
```

This is a *functor*. It saves the maximum T value that has been passed as an argument of its `operator()` method. To retrieve the value saved, just use the `max_accumulator` as a T instance.

Definition at line 41 of file `accum.hh`.

The documentation for this struct was generated from the following file:

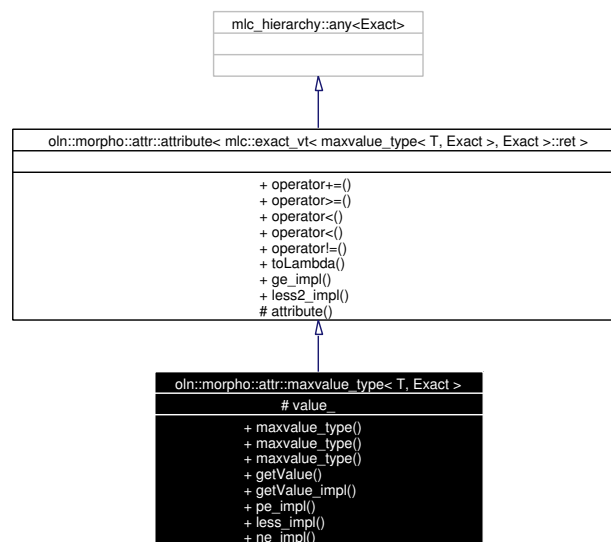
- `accum.hh`

7.251 oln::morpho::attr::maxvalue_type< T, Exact > Class Template Reference

Max value attribute.

```
#include <attributes.hh>
```

Inheritance diagram for oln::morpho::attr::maxvalue_type< T, Exact >:



Collaboration diagram for oln::morpho::attr::maxvalue_type< T, Exact >:



Public Types

- typedef `maxvalue_type`< T, Exact > `self_type`
Self type of the class.
- typedef `mlc::exact_vt`< `self_type`, Exact >::ret `exact_type`
- typedef `oln::morpho::attr::attr_traits`< `exact_type` >::value_type `value_type`
- typedef `oln::morpho::attr::attr_traits`< `exact_type` >::env_type `env_type`
- typedef `oln::morpho::attr::attr_traits`< `exact_type` >::lambda_type `lambda_type`

Public Member Functions

- `maxvalue_type` ()
Basic Ctor.
- `maxvalue_type` (const `lambda_type` &lambda)
Ctor from a `lambda_type` value.
- template<class I> `maxvalue_type` (const `abstract::image`< I > &input, const typename `mlc::exact`< I >::ret::point_type &p, const env_type &)
Ctor from a point and an image.
- const value_type & `getValue` () const
Accessor to `value_`.
- const value_type & `getValue_impl` () const
Implementation of `getValue()`.
- void `pe_impl` (const `maxvalue_type` &rhs)
`+=` operator implementation.
- bool `less_impl` (const `lambda_type` &lambda) const
`"<"` operator implementation.
- bool `ne_impl` (const `lambda_type` &lambda) const
`!=` operator implementation.

Protected Attributes

- value_type `value_`
Value of the attribute.

7.251.1 Detailed Description

template<class T = unsigned, class Exact = `mlc::final`> class `oln::morpho::attr::maxvalue_type`< T, Exact >

Max value attribute.

Parameters:*T* Data type.*Exact* The exact type.

Definition at line 887 of file attributes.hh.

7.251.2 Constructor & Destructor Documentation

7.251.2.1 `template<class T = unsigned, class Exact = mlc::final> oln::morpho::attr::maxvalue_type< T, Exact >::maxvalue_type () [inline]`

Basic Ctor.

Warning:

After this call, the object is only instantiated (not initialized).

Definition at line 900 of file attributes.hh.

```

901         {
902     };

```

7.251.2.2 `template<class T = unsigned, class Exact = mlc::final> oln::morpho::attr::maxvalue_type< T, Exact >::maxvalue_type (const lambda_type & lambda) [inline]`

Ctor from a lambda_type value.

- lambda Value of the attribute.

Definition at line 909 of file attributes.hh.

```

909                                     : value_(lambda)
910         {
911     };

```

7.251.2.3 `template<class T = unsigned, class Exact = mlc::final> template<class I> oln::morpho::attr::maxvalue_type< T, Exact >::maxvalue_type (const abstract::image< I > & input, const typename mlc::exact< I >::ret::point_type & p, const env_type &) [inline]`

Ctor from a point and an image.

Parameters:*I* Image exact type.

- input Input image.
- p Point to consider in the image.

Definition at line 922 of file attributes.hh.

```

924                                     :
925         value_(input[p])
926         {
927         };

```

7.251.3 Member Function Documentation

7.251.3.1 `template<class T = unsigned, class Exact = mlc::final> const value_type& oln::morpho::attr::maxvalue_type< T, Exact >::getValue () const [inline]`

Accessor to value_.

Virtual method.

See also:

[getValue_impl\(\)](#)

Definition at line 935 of file attributes.hh.

Referenced by `oln::morpho::attr::maxvalue_type< T, Exact >::pe_impl()`.

```

936         {
937             mlc_dispatch(getValue)();
938         };

```

7.251.3.2 `template<class T = unsigned, class Exact = mlc::final> const value_type& oln::morpho::attr::maxvalue_type< T, Exact >::getValue_impl () const [inline]`

Implementation of [getValue\(\)](#).

Override this method in order to provide a new version of [getValue\(\)](#).

Warning:

Do not call this method, use [getValue\(\)](#) instead.

Definition at line 948 of file attributes.hh.

```

949         {
950             return value_;
951         };

```

7.251.3.3 `template<class T = unsigned, class Exact = mlc::final> bool oln::morpho::attr::maxvalue_type< T, Exact >::less_impl (const lambda_type & lambda) const [inline]`

"<" operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 972 of file attributes.hh.

```

973         {
974             return value_ < lambda;
975         };

```

7.251.3.4 `template<class T = unsigned, class Exact = mlc::final> bool
 oln::morpho::attr::maxvalue_type< T, Exact >::ne_impl (const lambda_type & lambda)
 const [inline]`

!= operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 984 of file attributes.hh.

```

985         {
986             return lambda != value_;
987         };

```

7.251.3.5 `template<class T = unsigned, class Exact = mlc::final> void
 oln::morpho::attr::maxvalue_type< T, Exact >::pe_impl (const maxvalue_type< T,
 Exact > & rhs) [inline]`

+= operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 960 of file attributes.hh.

References oln::morpho::attr::maxvalue_type< T, Exact >::getValue().

```

961         {
962             value_ = ntg::max(value_, rhs.getValue());
963         };

```

The documentation for this class was generated from the following file:

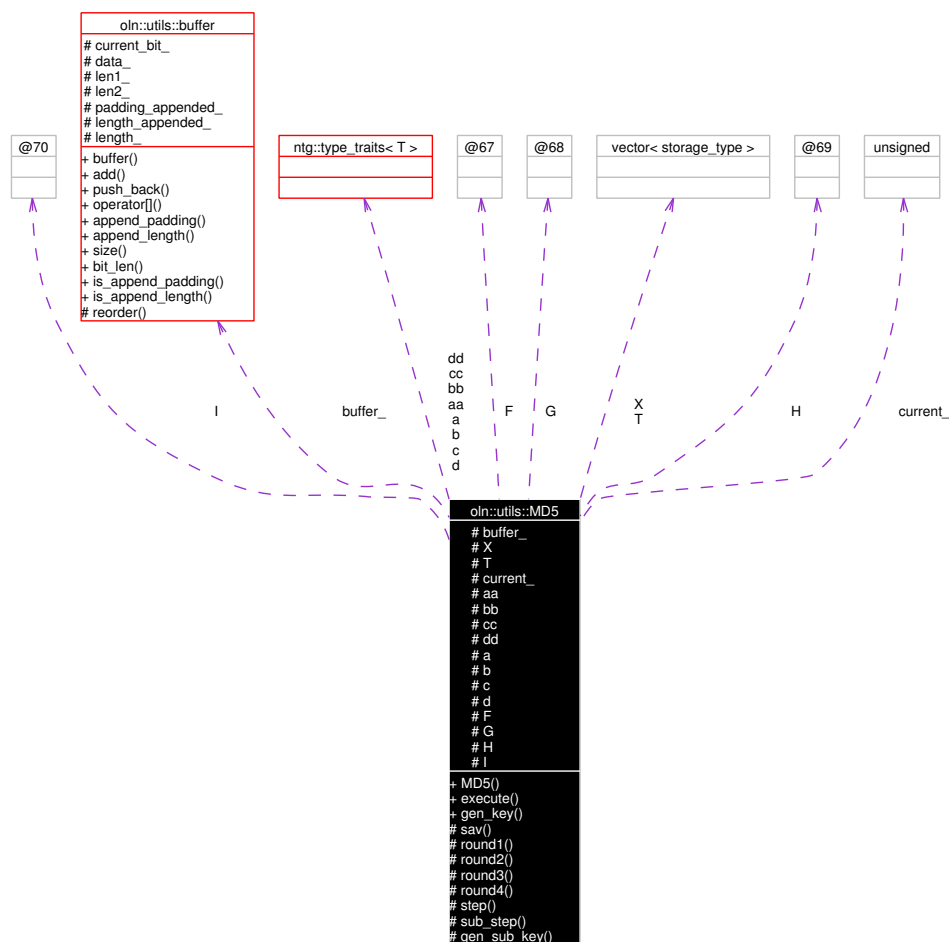
- attributes.hh

7.252 oln::utils::MD5 Class Reference

Class used to compute a [MD5](#) digest.

```
#include <md5.hh>
```

Collaboration diagram for oln::utils::MD5:



Public Types

- typedef `ntg::int_u32` [value_type](#)
Value to work on.
- typedef `ntg::type_traits< value_type >::storage_type` [storage_type](#)
Representation type of value type.

Public Member Functions

- [MD5](#) ([buffer](#) `b`)

Constructor.

- [key execute \(\)](#)
Compute the digest.
- [key gen_key \(\)](#)
Generate the key.

Protected Member Functions

- void [sav \(\)](#)
initialization of a [MD5](#) algorithm step.
- void [round1 \(\)](#)
Round 1 of the algorithm.
- void [round2 \(\)](#)
Round 2 of the algorithm.
- void [round3 \(\)](#)
Round 3 of the algorithm.
- void [round4 \(\)](#)
Round 4 of the algorithm.
- void [step \(\)](#)
A step of the algorithm.
- template<class Fun> void [sub_step](#) (const Fun &fun, [storage_type](#) &a, const [storage_type](#) &b, const [storage_type](#) &c, const [storage_type](#) &d, unsigned k, unsigned s, unsigned i)
Sub step to update a register value.
- void [gen_sub_key](#) (std::vector< [key::value_type](#) > &v, [storage_type](#) x, unsigned base)
Generate a sub part of the key.

Protected Attributes

- [buffer buffer_](#)
The buffer to process.
- std::vector< [storage_type](#) > [X](#)
Vector of 16 words.
- std::vector< [storage_type](#) > [T](#)
Vector of 64 words.
- unsigned [current_](#)

Current position in the buffer.

- [storage_type aa](#)
Save of a.
- [storage_type bb](#)
Save of b.
- [storage_type cc](#)
Save of c.
- [storage_type dd](#)
Save of d.
- [storage_type a](#)
A register.
- [storage_type b](#)
B register.
- [storage_type c](#)
C register.
- [storage_type d](#)
D register.
- struct {
 [storage_type operator\(\)](#) (const [storage_type](#) &x, const [storage_type](#) &y, const [storage_type](#) &z) const
} **F**

Functor for bit operations.
- struct {
 [storage_type operator\(\)](#) (const [storage_type](#) &x, const [storage_type](#) &y, const [storage_type](#) &z) const
} **G**

Functor for bit operations.
- struct {
 [storage_type operator\(\)](#) (const [storage_type](#) &x, const [storage_type](#) &y, const [storage_type](#) &z) const
} **H**

Functor for bit operations.
- struct {
 [storage_type operator\(\)](#) (const [storage_type](#) &x, const [storage_type](#) &y, const [storage_type](#) &z) const
} **I**

Functor for bit operations.

7.252.1 Detailed Description

Class used to compute a [MD5](#) digest.

Definition at line 44 of file md5.hh.

7.252.2 Constructor & Destructor Documentation

7.252.2.1 oln::utils::MD5::MD5 (buffer *b*) [inline, explicit]

Constructor.

Initialization from a buffer.

- The buffer to use.

Definition at line 234 of file md5.hh.

7.252.3 Member Function Documentation

7.252.3.1 key oln::utils::MD5::gen_key () [inline]

Generate the key.

Precondition:

To avoid meaningless result, you should call execute before.

Definition at line 465 of file md5.hh.

7.252.3.2 void oln::utils::MD5::gen_sub_key (std::vector< key::value_type > & *v*, storage_type *x*, unsigned *base*) [inline, protected]

Generate a sub part of the key.

- *v* Vector of values (output).
- *x* Data to put in *v*.
- *base* Where to put *x* in *v*.

Definition at line 452 of file md5.hh.

7.252.4 Member Data Documentation

7.252.4.1 struct { ... } oln::utils::MD5::F [protected]

Functor for bit operations.

$F(X,Y,Z) = XY \vee \text{not}(X) Z$

7.252.4.2 struct { ... } [oln::utils::MD5::G](#) [protected]

Functor for bit operations.

$G(X,Y,Z) = XZ \vee Y \text{ not}(Z)$

7.252.4.3 struct { ... } [oln::utils::MD5::H](#) [protected]

Functor for bit operations.

$H(X,Y,Z) = X \text{ xor } Y \text{ xor } Z$

7.252.4.4 struct { ... } [oln::utils::MD5::I](#) [protected]

Functor for bit operations.

$I(X,Y,Z) = Y \text{ xor } (X \vee \text{ not}(Z))$

The documentation for this class was generated from the following file:

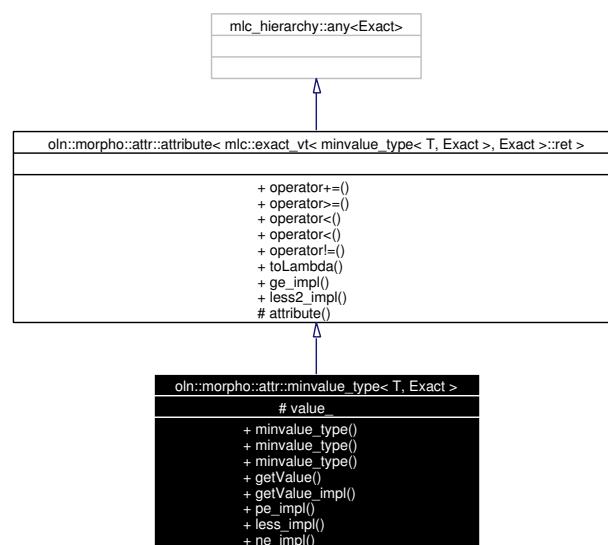
- md5.hh

7.253 oln::morpho::attr::minvalue_type< T, Exact > Class Template Reference

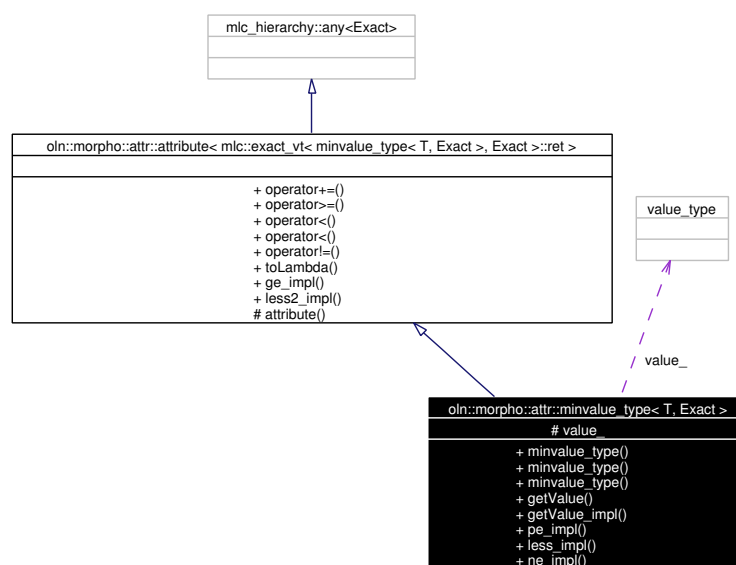
Min value attribute.

```
#include <attributes.hh>
```

Inheritance diagram for oln::morpho::attr::minvalue_type< T, Exact >:



Collaboration diagram for oln::morpho::attr::minvalue_type< T, Exact >:



Public Types

- typedef [minvalue_type](#)< T, Exact > **self_type**
- typedef `mlc::exact_vt`< self_type, Exact >::ret **exact_type**
- typedef `oln::morpho::attr::attr_traits`< exact_type >::value_type **value_type**
- typedef `oln::morpho::attr::attr_traits`< exact_type >::env_type **env_type**
- typedef `oln::morpho::attr::attr_traits`< exact_type >::lambda_type **lambda_type**

Public Member Functions

- [minvalue_type](#) ()
Basic Ctor.
- [minvalue_type](#) (const lambda_type &lambda)
Ctor from a lambda_type value.
- template<class I> [minvalue_type](#) (const [abstract::image](#)< I > &input, const typename `mlc::exact`< I >::ret::point_type &p, const env_type &)
Ctor from a point and an image.
- const value_type & [getValue](#) () const
Accessor to value_.
- const value_type & [getValue_impl](#) () const
Implementation of [getValue](#)().
- void [pe_impl](#) (const [minvalue_type](#) &rhs)
+= operator implementation.
- bool [less_impl](#) (const lambda_type &lambda) const
"<" operator implementation.
- bool [ne_impl](#) (const lambda_type &lambda) const
!= operator implementation.

Protected Attributes

- value_type [value_](#)
Value of the attribute.

7.253.1 Detailed Description

template<class T = unsigned, class Exact = `mlc::final`> class `oln::morpho::attr::minvalue_type`< T, Exact >

Min value attribute.

Parameters:*T* Data type.*Exact* The exact type.

Definition at line 1003 of file attributes.hh.

7.253.2 Constructor & Destructor Documentation

7.253.2.1 `template<class T = unsigned, class Exact = mlc::final> oln::morpho::attr::minvalue_type< T, Exact >::minvalue_type () [inline]`

Basic Ctor.

Warning:

After this call, the object is only instantiated (not initialized).

Definition at line 1016 of file attributes.hh.

```

1017         {
1018     };

```

7.253.2.2 `template<class T = unsigned, class Exact = mlc::final> oln::morpho::attr::minvalue_type< T, Exact >::minvalue_type (const lambda_type & lambda) [inline]`

Ctor from a lambda_type value.

- lambda Value of the attribute.

Definition at line 1025 of file attributes.hh.

```

1025                                     : value_(lambda)
1026         {
1027     };

```

7.253.2.3 `template<class T = unsigned, class Exact = mlc::final> template<class I> oln::morpho::attr::minvalue_type< T, Exact >::minvalue_type (const abstract::image< I > & input, const typename mlc::exact< I >::ret::point_type & p, const env_type &) [inline]`

Ctor from a point and an image.

Parameters:*I* Image exact type.

- input Input image.
- p Point to consider in the image.

Definition at line 1038 of file attributes.hh.

```

1040                                     :
1041         value_(input[p])
1042         {
1043         };

```

7.253.3 Member Function Documentation

7.253.3.1 `template<class T = unsigned, class Exact = mlc::final> const value_type& oln::morpho::attr::minvalue_type< T, Exact >::getValue () const [inline]`

Accessor to value_.

Virtual method.

See also:

[getValue_impl\(\)](#)

Definition at line 1051 of file attributes.hh.

Referenced by `oln::morpho::attr::minvalue_type< T, Exact >::pe_impl()`.

```

1052         {
1053             mlc_dispatch(getValue)();
1054         };

```

7.253.3.2 `template<class T = unsigned, class Exact = mlc::final> const value_type& oln::morpho::attr::minvalue_type< T, Exact >::getValue_impl () const [inline]`

Implementation of [getValue\(\)](#).

Override this method in order to provide a new version of [getValue\(\)](#).

Warning:

Do not call this method, use [getValue\(\)](#) instead.

Definition at line 1064 of file attributes.hh.

```

1065         {
1066             return value_;
1067         };

```

7.253.3.3 `template<class T = unsigned, class Exact = mlc::final> bool oln::morpho::attr::minvalue_type< T, Exact >::less_impl (const lambda_type & lambda) const [inline]`

"<" operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 1088 of file attributes.hh.

```
1089         {
1090             return value_ > lambda;
1091         };
```

7.253.3.4 `template<class T = unsigned, class Exact = mlc::final> bool
 oln::morpho::attr::minvalue_type< T, Exact >::ne_impl (const lambda_type & lambda)
 const [inline]`

!= operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 1100 of file attributes.hh.

```
1101         {
1102             return lambda != value_;
1103         };
```

7.253.3.5 `template<class T = unsigned, class Exact = mlc::final> void
 oln::morpho::attr::minvalue_type< T, Exact >::pe_impl (const minvalue_type< T,
 Exact > & rhs) [inline]`

+= operator implementation.

This is an implementation of the += operator. Override this method to provide a new implementation of this operator.

Warning:

This method SHOULDN'T directly be called.

Definition at line 1076 of file attributes.hh.

References oln::morpho::attr::minvalue_type< T, Exact >::getValue().

```
1077         {
1078             value_ = ntg::min(value_, rhs.getValue());
1079         };
```

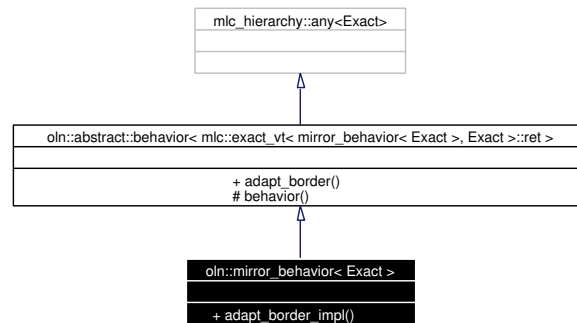
The documentation for this class was generated from the following file:

- attributes.hh

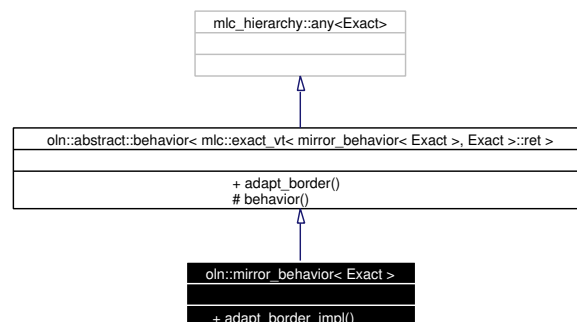
7.254 oln::mirror_behavior< Exact > Class Template Reference

```
#include <behavior.hh>
```

Inheritance diagram for oln::mirror_behavior< Exact >:



Collaboration diagram for oln::mirror_behavior< Exact >:



Public Types

- typedef [mirror_behavior< Exact > self_type](#)
- typedef `mlc::exact_vt< self_type, Exact >::ret` [exact_type](#)

Public Member Functions

- template<class I> void **adapt_border_impl** ([oln::abstract::image< I > &im](#), [coord](#) border_size) const

7.254.1 Detailed Description

```
template<class Exact = mlc::final> class oln::mirror_behavior< Exact >
```

Make the border be a mirror of the image.

See also:

[abstract::image_size::border_](#)

Definition at line 42 of file olena/oln/core/behavior.hh.

7.254.2 Member Typedef Documentation

7.254.2.1 `template<class Exact = mlc::final> typedef mlc::exact_vt< self_type , Exact >::ret
oln::mirror_behavior< Exact >::exact_type`

The exact type.

Reimplemented from [oln::abstract::behavior](#)< Exact >.

Definition at line 47 of file olena/oln/core/behavior.hh.

7.254.2.2 `template<class Exact = mlc::final> typedef mirror_behavior<Exact>
oln::mirror_behavior< Exact >::self_type`

The self type.

Reimplemented from [oln::abstract::behavior](#)< Exact >.

Definition at line 46 of file olena/oln/core/behavior.hh.

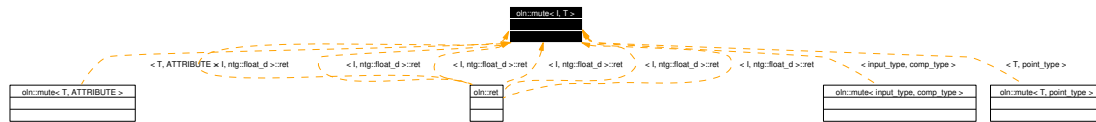
The documentation for this class was generated from the following file:

- olena/oln/core/behavior.hh

7.255 oln::mute< I, T > Struct Template Reference

```
#include <image.hh>
```

Inheritance diagram for oln::mute< I, T >:



Public Types

- typedef `mlc::exact< I >::ret::template mute< T >::ret` **ret**

7.255.1 Detailed Description

template<class I, class T = typename mlc::exact<I>::ret::value_type> struct oln::mute< I, T >

ret is the same type as *I* excepted the *value_type* which will be *T*.

Definition at line 177 of file `core/image.hh`.

The documentation for this struct was generated from the following file:

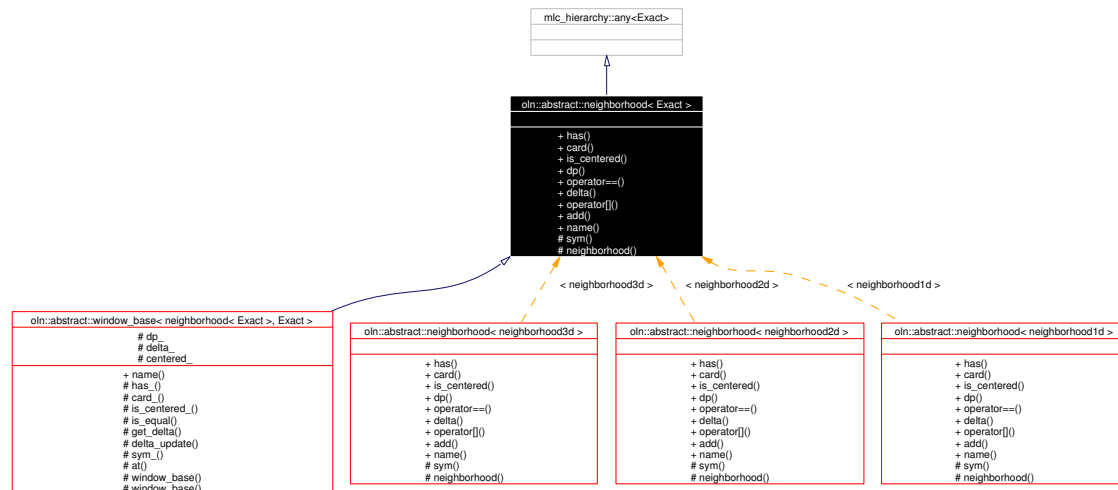
- `core/image.hh`

7.256 oln::abstract::neighborhood< Exact > Struct Template Reference

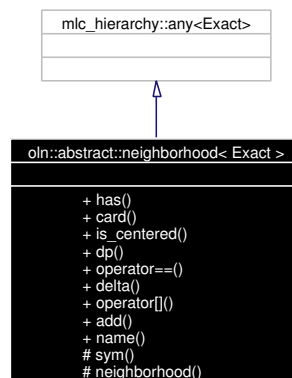
Neighborhood.

```
#include <neighborhood.hh>
```

Inheritance diagram for oln::abstract::neighborhood< Exact >:



Collaboration diagram for oln::abstract::neighborhood< Exact >:



Public Types

- typedef `Exact` [exact_type](#)
Set the exact type.
- typedef `neighborhood< Exact >` [self_type](#)
Set its type.
- typedef `struct_elt_traits< Exact >::iter_type` [iter_type](#)

The associate image's type of iterator.

- `typedef struct_elt_traits< Exact >::neighb_type neighb_type`
Set the neighborhood type.
- `typedef struct_elt_traits< Exact >::win_type win_type`
Set the window type.
- `typedef struct_elt_traits< Exact >::dpoint_type dpoint_type`
The associate image's type of dpoint (move point).
- `typedef struct_elt_traits< Exact >::abstract_type abstract_type`
Set the abstract type.
- `enum { dim = struct_elt_traits<Exact>::dim }`

Public Member Functions

- `bool has (const abstract::dpoint< dpoint_type > &dp) const`
Test if the set of points contains this one.
– *dp a dpoint (deplacement point).*
- `unsigned card () const`
Get the number of points.
- `bool is_centered () const`
Test if the neighborhood is centered.
- `const dpoint_type dp (unsigned i) const`
Get the nth element of the neighborhood.
– *i The nth.*
- `bool operator== (const self_type &win) const`
Compare two sets of structuring elements.
– *win The structuring elements to compare.*
- `coord delta () const`
Get the delta of the neighborhood.
- `const dpoint_type operator[] (unsigned i) const`
Get the nth element of the neighborhood.
– *i The nth.*
- `exact_type & add (const abstract::dpoint< dpoint_type > &dp)`
Add a point to the neighborhood.
– *dp The new point.*

Static Public Member Functions

- `std::string name ()`
Return the name of the type.

Protected Member Functions

- `void sym ()`
Set neighborhood to opposite.
- `neighborhood ()`
Do nothing, used only by sub-classes.

7.256.1 Detailed Description

`template<class Exact> struct oln::abstract::neighborhood< Exact >`

Neighborhood.

It looks like structuring elements but here, when you add an element, you add its opposite. This abstract class defines several virtual methods for his subclasses. Its goal is to deal with a set of displacement points.

Definition at line 65 of file neighborhood.hh.

7.256.2 Member Typedef Documentation

7.256.2.1 `template<class Exact> typedef struct_elt_traits<Exact>::dpoint_type
olin::abstract::neighborhood< Exact >::dpoint_type`

The associate image's type of dpoint (move point).

Warning:

Prefer the macros `olin_dpoint_type(Pointable)` and `olin_dpoint_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented in `olin::abstract::neighborhoodnd< Exact >`, `olin::neighborhood1d`, `olin::neighborhood2d`, `olin::neighborhood3d`, `olin::abstract::neighborhoodnd< neighborhood3d >`, `olin::abstract::neighborhoodnd< neighborhood2d >`, `olin::abstract::neighborhoodnd< neighborhood1d >`, `olin::abstract::window_base< neighborhood< neighborhood2d >`, `neighborhood2d >`, `olin::abstract::window_base< neighborhood< neighborhood3d >`, `neighborhood3d >`, `olin::abstract::window_base< neighborhood< neighborhood1d >`, `neighborhood1d >`, and `olin::abstract::window_base< neighborhood< Exact >`, `Exact >`.

Definition at line 87 of file neighborhood.hh.

7.256.2.2 `template<class Exact> typedef struct_elt_traits<Exact>::iter_type
olin::abstract::neighborhood< Exact >::iter_type`

The associate image's type of iterator.

Warning:

Prefer the macros `oln_iter_type(Iterable)` and `oln_iter_type_(Iterable)` (the same without the 'type-name' keyword)

Reimplemented in [oln::neighborhood1d](#), [oln::neighborhood2d](#), and [oln::neighborhood3d](#).

Definition at line 75 of file neighborhood.hh.

7.256.3 Member Function Documentation

7.256.3.1 `template<class Exact> exact_type& oln::abstract::neighborhood< Exact >::add (const abstract::dpoint< dpoint_type > & dp) [inline]`

Add a point to the neighborhood.

- `dp` The new point.

Add a new member to the neighborhood.

Definition at line 188 of file neighborhood.hh.

Referenced by `oln::convert::ng_to_cse()`, and `oln::convert::ng_to_se()`.

```

189     {
190         this->exact().add(dp.exact());
191         return this->exact().add(-dp.exact());
192     }
```

7.256.3.2 `template<class Exact> unsigned oln::abstract::neighborhood< Exact >::card () const [inline]`

Get the number of points.

Returns:

The number of points.

Definition at line 118 of file neighborhood.hh.

Referenced by `oln::inter()`, `oln::mk_win_from_neighb()`, and `oln::uni()`.

```

119     {
120         return this->exact().card_();
121     }
```

7.256.3.3 `template<class Exact> coord oln::abstract::neighborhood< Exact >::delta () const [inline]`

Get the delta of the neighborhood.

Returns:

Delta.

Delta is the biggest element of the neighborhood.

Definition at line 165 of file neighborhood.hh.

Referenced by oln::morpho::sure::geodesic_dilation(), oln::morpho::sure::geodesic_erosion(), oln::morpho::hybrid::geodesic_reconstruction_dilation(), oln::morpho::hybrid::geodesic_reconstruction_erosion(), oln::morpho::fast::tarjan::tarjan_set< T, ATTRIBUTE, Env >::get_comptute(), and oln::topo::tarjan::abstract::tarjan_with_attr< mlc::exact_vt< flat_zone< T, DestType, A, Exact >, Exact >::ret >::get_compute_impl().

```
166     {
167         return this->exact().get_delta();
168     }
```

7.256.3.4 template<class Exact> const dpoint_type oln::abstract::neighborhood< Exact >::dp (unsigned i) const [inline]

Get the nth element of the neighborhood.

- i The nth.

Returns:

The nth dpoint.

Definition at line 142 of file neighborhood.hh.

Referenced by oln::inter(), oln::mk_win_from_neighb(), and oln::uni().

```
143     {
144         return this->exact()[i];
145     }
```

7.256.3.5 template<class Exact> bool oln::abstract::neighborhood< Exact >::has (const abstract::dpoint< dpoint_type > & dp) const [inline]

Test if the set of points contains this one.

- dp a dpoint (displacement point).

Returns:

True if the set of points contains this dpoint.

Definition at line 108 of file neighborhood.hh.

Referenced by oln::inter().

```
109     {
110         return this->exact().has_(dp.exact());
111     }
```

7.256.3.6 `template<class Exact> bool oln::abstract::neighborhood< Exact >::is_centered (void) const [inline]`

Test if the neighborhood is centered.

Returns:

True if it's centered.

Neighborhood are centered when they contains at least one element.

Definition at line 131 of file neighborhood.hh.

```
132     {
133         return this->exact().is_centered_();
134     }
```

7.256.3.7 `template<class Exact> bool oln::abstract::neighborhood< Exact >::operator==(const self_type & win) const [inline]`

Compare two sets of structuring elements.

- win The structuring elements to compare.

Returns:

True if they are the same.

Definition at line 153 of file neighborhood.hh.

```
154     {
155         return this->exact().is_equal(win.exact());
156     }
```

7.256.3.8]

`template<class Exact> const dpoint_type oln::abstract::neighborhood< Exact >::operator[] (unsigned i) const [inline]`

Get the nth element of the neighborhood.

- i The nth.

Returns:

The nth dpoint.

Definition at line 176 of file neighborhood.hh.

```
177     {
178         return this->exact().at(i);
179     }
```

7.256.3.9 `template<class Exact> void oln::abstract::neighborhood< Exact >::sym ()`
[inline, protected]

Set neighborhood to opposite.

Each point of neighborhood is assigned to its opposite.

Definition at line 211 of file neighborhood.hh.

```
212     {  
213         this->exact().sym_();  
214     }
```

The documentation for this struct was generated from the following file:

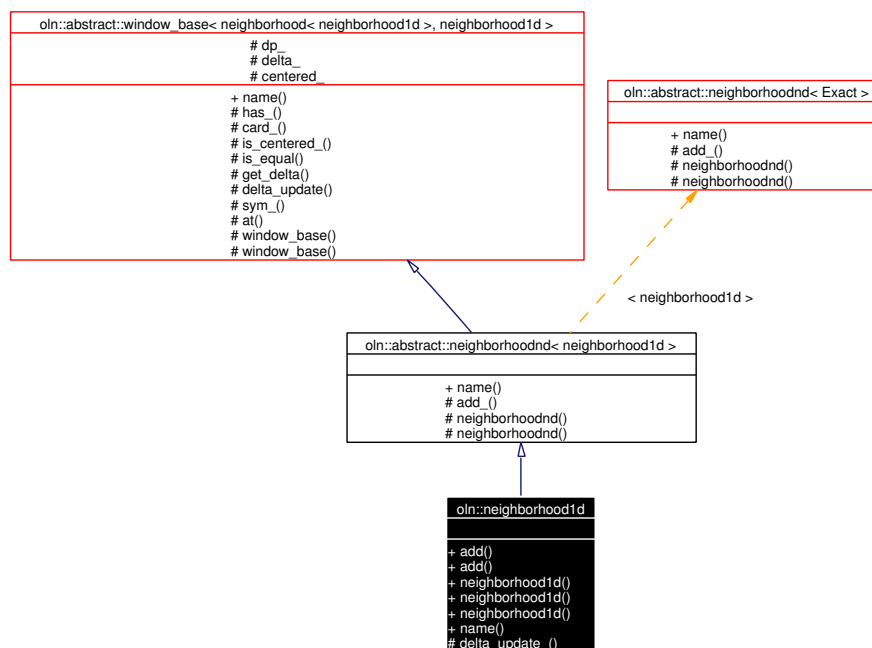
- neighborhood.hh

7.257 oln::neighborhood1d Class Reference

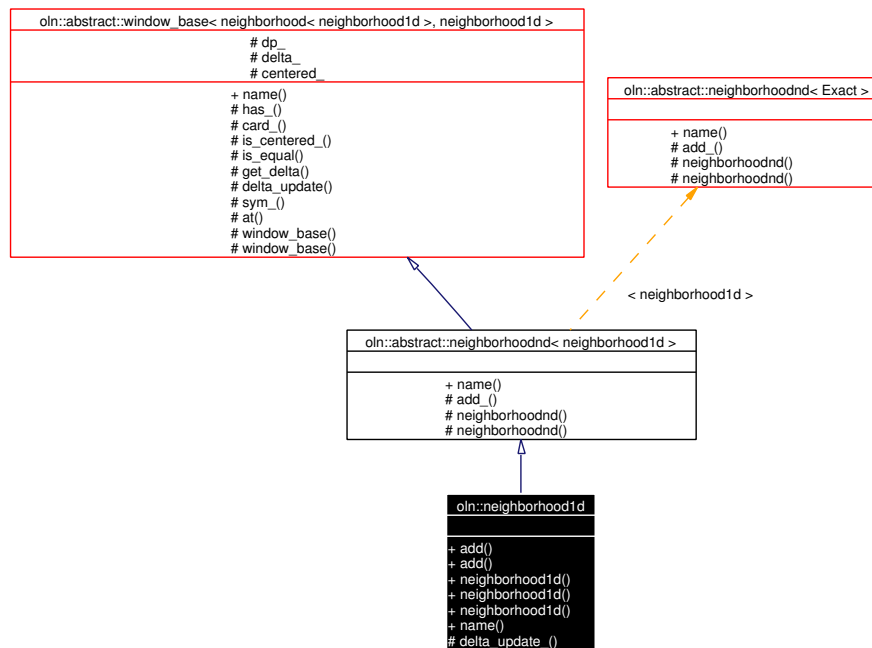
Neighborhood 1 dimension.

```
#include <neighborhood1d.hh>
```

Inheritance diagram for oln::neighborhood1d:



Collaboration diagram for oln::neighborhood1d:



Public Types

- typedef `abstract::neighborhoodnd< neighborhood1d >` `super_type`
Super type.
- typedef `neighborhood1d` `self_type`
Self type.
- typedef `struct_elt_traits< self_type >::iter_type` `iter_type`
The associate image's type of iterator (move point).
- typedef `struct_elt_traits< self_type >::neighb_type` `neighb_type`
Set the neighborhood type.
- typedef `struct_elt_traits< self_type >::dpoint_type` `dpoint_type`
The associate image's type of dpoint (move point).

Public Member Functions

- `neighborhood1d & add (const dpoint_type &dp)`
Add a dpoint (move point) to the neighborhood.
– *dp* The new point.
- `neighborhood1d & add (coord col)`
Add a point by coordinates to the neighborhood.
– *col* The coordinate of the new point (1 dimension).

- `neighborhood1d()`
Construct a neighborhood of 1 dimension.
- `neighborhood1d(unsigned size)`
Construct a neighborhood of 1 dimension.
 - *size Reserve 'size' elements for the neighborhood.*
- `neighborhood1d(unsigned n, const coord crd[])`
Construct a neighborhood of 1 dimension.
 - *n Add 'n' elements to the neighborhood.*
 - *crd Coordinates of the 'n' elements.*

Static Public Member Functions

- `std::string name()`
Return the name of the type.

Protected Member Functions

- `coord delta_update_(const dpoint_type &dp)`
Update delta.
 - *dp a move point.*

Friends

- `class abstract::window_base< abstract::neighborhood< neighborhood1d >, neighborhood1d >`

7.257.1 Detailed Description

Neighborhood 1 dimension.

It looks like structuring elements but here, when you add an element, you add its opposite. Points (dpoint) have 1 dimension.

Definition at line 64 of file neighborhood1d.hh.

7.257.2 Member Typedef Documentation

7.257.2.1 `typedef struct_elt_traits< self_type >::dpoint_type oln::neighborhood1d::dpoint_type`

The associate image's type of dpoint (move point).

Warning:

Prefer the macros `oln_dpoint_type(Pointable)` and `oln_dpoint_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented from [oln::abstract::neighborhoodnd< neighborhood1d >](#).

Definition at line 86 of file neighborhood1d.hh.

Referenced by add(), and neighborhood1d().

7.257.2.2 typedef struct_elt_traits< self_type >::iter_type oln::neighborhood1d::iter_type

The associate image's type of iterator (move point).

Warning:

Prefer the macros oln_iter_type(Pointable) and oln_iter_type_(Pointable) (the same without the 'type-name' keyword)

Reimplemented from [oln::abstract::neighborhood< neighborhood1d >](#).

Definition at line 78 of file neighborhood1d.hh.

7.257.3 Member Function Documentation

7.257.3.1 neighborhood1d& oln::neighborhood1d::add (coord col) [inline]

Add a point by coordinates to the neighborhood.

- col The coordinate of the new point (1 dimension).

Add a new member by its coordinates to the neighborhood. The coordinates are only the column number because the neighborhood has 1 dimension.

Definition at line 113 of file neighborhood1d.hh.

References add(), oln::coord, and dpoint_type.

```

114     {
115         return this->add(dpoint_type(col));
116     }
```

7.257.3.2 neighborhood1d& oln::neighborhood1d::add (const dpoint_type & dp) [inline]

Add a dpoint (move point) to the neighborhood.

- dp The new point.

Add a new member to the neighborhood. This point must be of 1 dimension.

Definition at line 98 of file neighborhood1d.hh.

Referenced by add(), oln::mk_neighb_segment(), and neighborhood1d().

```

99     {
100         this->exact().add_(dp);
101         return this->exact().add_(-dp);
102     }
```

7.257.3.3 `coord oln::neighborhood1d::delta_update_ (const dpoint_type & dp)` [inline, protected]

Update delta.

- *dp* a move point.

Returns:

Delta.

If the point is the biggest element of the neighborhood, then this point is assigned to delta.

Definition at line 160 of file neighborhood1d.hh.

References `oln::abstract::window_base< neighborhood< neighborhood1d >, neighborhood1d >::delta_`.

```
161     {  
162         delta_(abs(dp.col()));  
163         return delta_;  
164     }
```

The documentation for this class was generated from the following file:

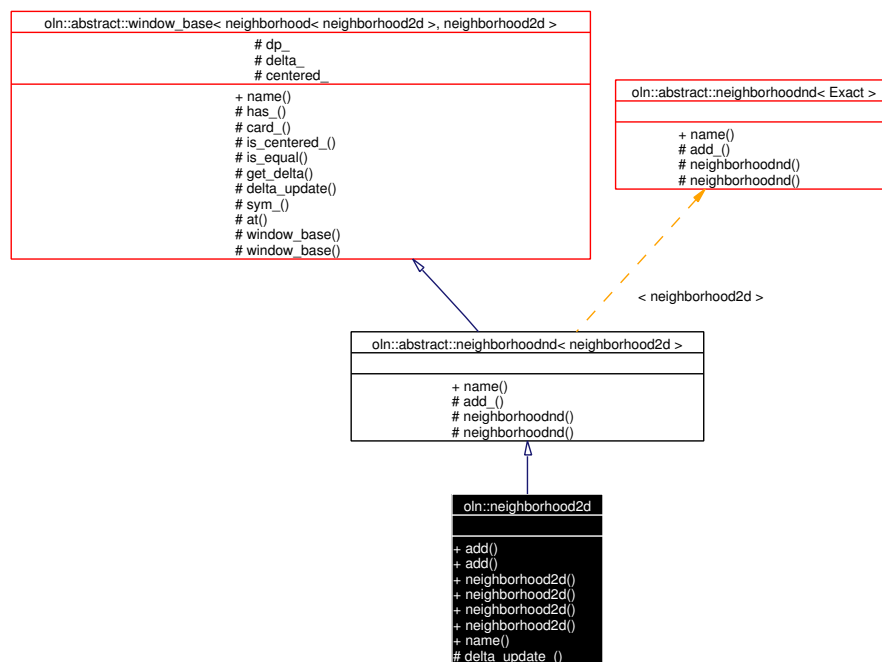
- neighborhood1d.hh

7.258 oln::neighborhood2d Class Reference

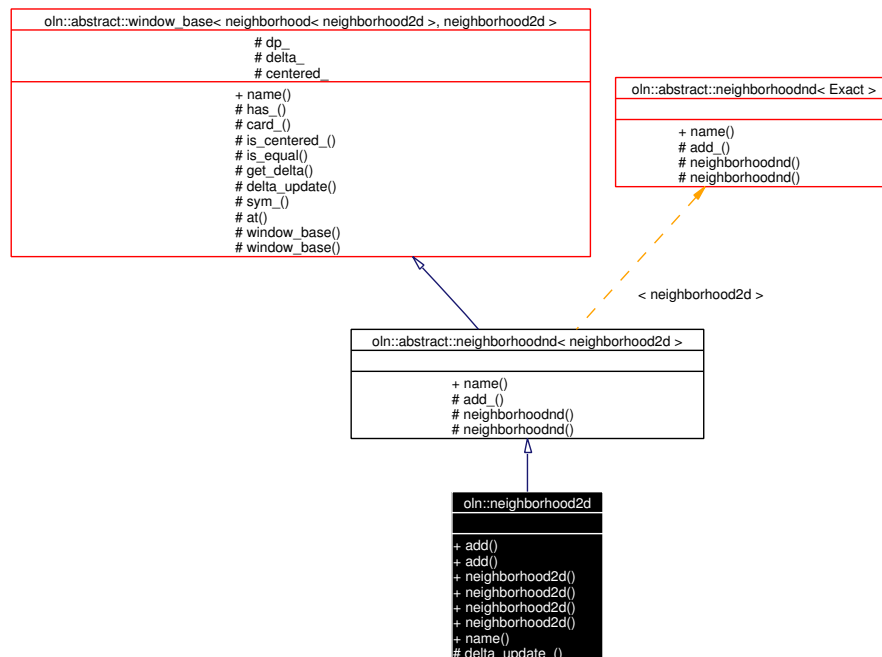
Neighborhood 2 dimensions.

```
#include <neighborhood2d.hh>
```

Inheritance diagram for oln::neighborhood2d:



Collaboration diagram for oln::neighborhood2d:



Public Types

- typedef `abstract::neighborhoodnd< neighborhood2d > super_type`
Super type.
- typedef `neighborhood2d self_type`
Self type.
- typedef struct_elt_traits< `self_type` >::iter_type iter_type
The associate image's type of iterator.
- typedef struct_elt_traits< `self_type` >::neighb_type neighb_type
Set the neighborhood type.
- typedef struct_elt_traits< `self_type` >::dpoint_type dpoint_type
The associate image's type of dpoint (move point).

Public Member Functions

- `neighborhood2d & add (const dpoint_type &dp)`
Add a dpoint (move point) to the neighborhood.
– *dp* The new point.
- `neighborhood2d & add (coord row, coord col)`
Add a point by coordinates to the neighborhood.
– *row* The coordinates of the new point.

- *col* The coordinates of the new point.
- [neighborhood2d](#) ()
Construct a neighborhood of 2 dimensions.
- [neighborhood2d](#) (unsigned size)
Construct a neighborhood of 2 dimensions.
 - *size* Reserve 'size' elements for the neighborhood.
- [neighborhood2d](#) (unsigned n, const [coord](#) crd[])
Construct a neighborhood of 2 dimensions.
 - *n* Add 'n' elements to the neighborhood.
 - *crd* Coordinates of the 'n' elements.
- [neighborhood2d](#) (const [io::internal::anything](#) &r)

Static Public Member Functions

- `std::string` [name](#) ()
Return the name of the type.

Protected Member Functions

- [coord](#) [delta_update_](#) (const [dpoint_type](#) &dp)
Update delta.
 - *dp* a displacement point.

Friends

- class [abstract::window_base](#)< [abstract::neighborhood](#)< [neighborhood2d](#) >, [neighborhood2d](#) >

7.258.1 Detailed Description

Neighborhood 2 dimensions.

It looks like structuring elements but here, when you add an element, you add its opposite. Points have 2 dimensions.

Definition at line 65 of file `neighborhood2d.hh`.

7.258.2 Member Typedef Documentation

7.258.2.1 `typedef struct_elt_traits< self_type >::dpoint_type oln::neighborhood2d::dpoint_type`

The associate image's type of dpoint (move point).

Warning:

Prefer the macros `oln_dpoint_type(Pointable)` and `oln_dpoint_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented from [oln::abstract::neighborhoodnd< neighborhood2d >](#).

Definition at line 88 of file `neighborhood2d.hh`.

Referenced by `add()`, and `neighborhood2d()`.

7.258.2.2 typedef struct_elt_traits< self_type >::iter_type oln::neighborhood2d::iter_type

The associate image's type of iterator.

Warning:

Prefer the macros `oln_iter_type(Iterable)` and `oln_iter_type_(Iterable)` (the same without the 'typename' keyword)

Reimplemented from [oln::abstract::neighborhood< neighborhood2d >](#).

Definition at line 79 of file `neighborhood2d.hh`.

7.258.3 Member Function Documentation**7.258.3.1 neighborhood2d& oln::neighborhood2d::add (coord row, coord col) [inline]**

Add a point by coordinates to the neighborhood.

- row The coordinates of the new point.
- col The coordinates of the new point.

Add a new member by its coordinates to the neighborhood. The coordinates have 2 dimensions.

Definition at line 115 of file `neighborhood2d.hh`.

References `add()`, `oln::coord`, and `dpoint_type`.

```

116     {
117         return this->add(dpoint_type(row, col));
118     }
```

7.258.3.2 neighborhood2d& oln::neighborhood2d::add (const dpoint_type & dp) [inline]

Add a dpoint (move point) to the neighborhood.

- dp The new point.

Add a new member to the neighborhood. This point must be of 2 dimensions.

Definition at line 100 of file `neighborhood2d.hh`.

Referenced by `add()`, `oln::mk_neighb_rectangle()`, and `neighborhood2d()`.

```

101     {
102         this->exact().add_(dp);
103         return this->exact().add_(-dp);
104     }
```


7.258.3.3 `coord` `oln::neighborhood2d::delta_update_ (const dpoint_type & dp)` [`inline`, `protected`]

Update delta.

- `dp` a displacement point.

Returns:

Delta.

If the point is the biggest element of the neighborhood, then this point is assigned to delta.

Definition at line 168 of file `neighborhood2d.hh`.

References `oln::abstract::window_base< neighborhood< neighborhood2d >, neighborhood2d >::delta_`.

```
169     {  
170         delta_(abs(dp.row()));  
171         delta_(abs(dp.col()));  
172         return delta_;  
173     }
```

The documentation for this class was generated from the following file:

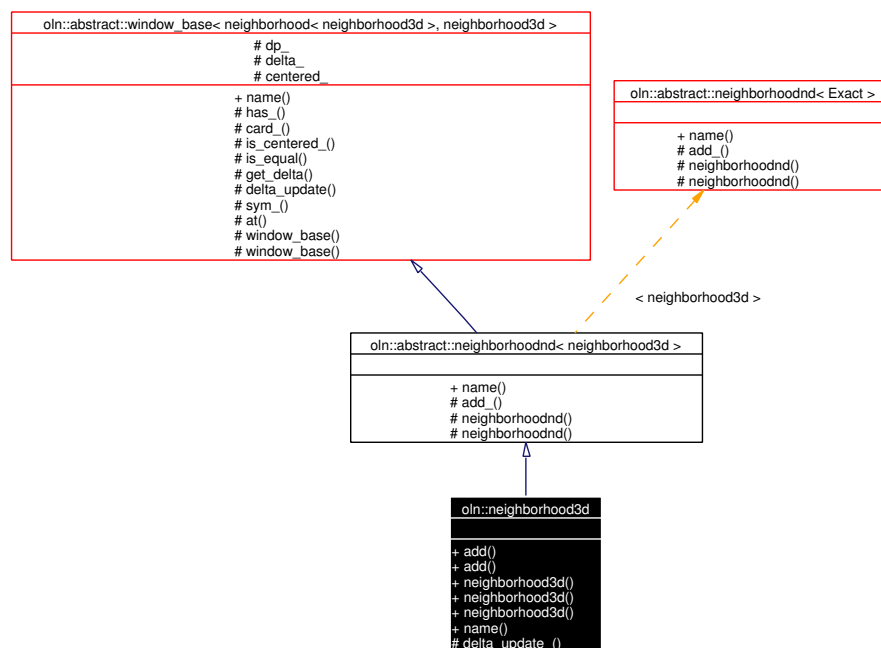
- `neighborhood2d.hh`

7.259 oln::neighborhood3d Class Reference

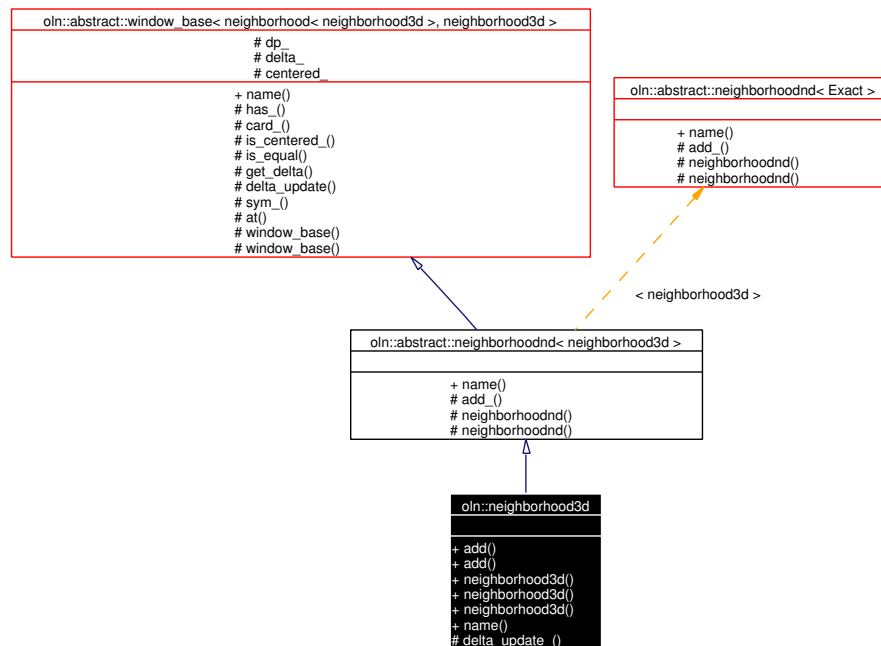
Neighborhood 3 dimensions.

```
#include <neighborhood3d.hh>
```

Inheritance diagram for oln::neighborhood3d:



Collaboration diagram for oln::neighborhood3d:



Public Types

- typedef `abstract::neighborhoodnd< neighborhood3d > super_type`
Super type.
- typedef `neighborhood3d self_type`
Self type.
- typedef `struct_elt_traits< self_type >::iter_type iter_type`
The associate image's type of iterator.
- typedef `struct_elt_traits< self_type >::neighb_type neighb_type`
Set the neighborhood type.
- typedef `struct_elt_traits< self_type >::dpoint_type dpoint_type`
The associate image's type of dpoint (move point).

Public Member Functions

- `neighborhood3d & add (const dpoint_type &dp)`
Add a dpoint (move point) to the neighborhood.
– *dp* The new point.
- `neighborhood3d & add (coord slice, coord row, coord col)`
Add a point by coordinates to the neighborhood.
– *slice* The coordinates of the new point.

- *row* The coordinates of the new point.
- *col* The coordinates of the new point.
- `neighborhood3d ()`
Construct a neighborhood of 3 dimensions.
- `neighborhood3d (unsigned size)`
Construct a neighborhood of 3 dimensions.
 - *size* Reserve 'size' elements for the neighborhood.
- `neighborhood3d (unsigned n, const coord crd[])`
Construct a neighborhood of 3 dimension.
 - *n* Add 'n' elements to the neighborhood.
 - *crd* Coordinates of the 'n' elements.

Static Public Member Functions

- `std::string name ()`
Return the name of the type.

Protected Member Functions

- `coord delta_update_ (const dpoint_type &dp)`
Update delta.
 - *dp* a displacement point.

Friends

- `class abstract::window_base< abstract::neighborhood< neighborhood3d >, neighborhood3d >`

7.259.1 Detailed Description

Neighborhood 3 dimensions.

It looks like structuring elements but here, when you add an element, you add its opposite. Points have 3 dimensions.

Definition at line 64 of file neighborhood3d.hh.

7.259.2 Member Typedef Documentation

7.259.2.1 `typedef struct_elt_traits< self_type >::dpoint_type oln::neighborhood3d::dpoint_type`

The associate image's type of dpoint (move point).

Warning:

Prefer the macros `oln_dpoint_type(Pointable)` and `oln_dpoint_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented from [oln::abstract::neighborhoodnd< neighborhood3d >](#).

Definition at line 86 of file neighborhood3d.hh.

Referenced by neighborhood3d().

7.259.2.2 typedef struct_elt_traits< self_type >::iter_type oln::neighborhood3d::iter_type

The associate image's type of iterator.

Warning:

Prefer the macros `oln_iter_type(Iterable)` and `oln_iter_type_(Iterable)` (the same without the 'type-name' keyword)

Reimplemented from [oln::abstract::neighborhood< neighborhood3d >](#).

Definition at line 78 of file neighborhood3d.hh.

7.259.3 Member Function Documentation**7.259.3.1 neighborhood3d& oln::neighborhood3d::add (coord slice, coord row, coord col) [inline]**

Add a point by coordinates to the neighborhood.

- slice The coordinates of the new point.
- row The coordinates of the new point.
- col The coordinates of the new point.

Add a new member by its coordinates to the neighborhood. The coordinates have 3 dimensions.

Definition at line 114 of file neighborhood3d.hh.

References `add()`, and `oln::coord`.

```

115     {
116         return this->add(dpoint3d(slice, row, col));
117     }
```

7.259.3.2 neighborhood3d& oln::neighborhood3d::add (const dpoint_type & dp) [inline]

Add a dpoint (move point) to the neighborhood.

- dp The new point.

Add a new member to the neighborhood. This point must be of 3 dimensions.

Definition at line 98 of file neighborhood3d.hh.

Referenced by `add()`, `oln::mk_neighb_block()`, and `neighborhood3d()`.

```
99      {
100          this->exact().add_(dp);
101          return this->exact().add_(-dp);
102      }
```

7.259.3.3 `coord oln::neighborhood3d::delta_update_ (const dpoint_type & dp)` [inline, protected]

Update delta.

- dp a displacement point.

Returns:

Delta.

If the point is the biggest element of the neighborhood, then this point is assigned to delta.

Definition at line 161 of file neighborhood3d.hh.

References `oln::abstract::window_base< neighborhood< neighborhood3d >, neighborhood3d >::delta_`.

```
162      {
163          delta_(abs(dp.slice()));
164          delta_(abs(dp.row()));
165          delta_(abs(dp.col()));
166          return delta_;
167      }
```

The documentation for this class was generated from the following file:

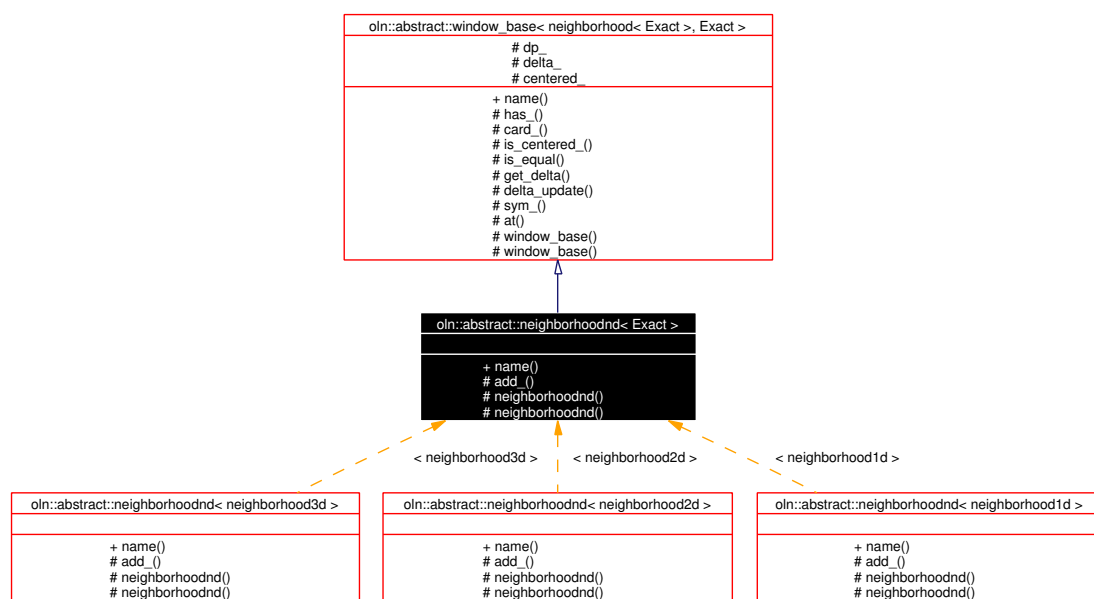
- neighborhood3d.hh

7.260 oln::abstract::neighborhoodnd< Exact > Struct Template Reference

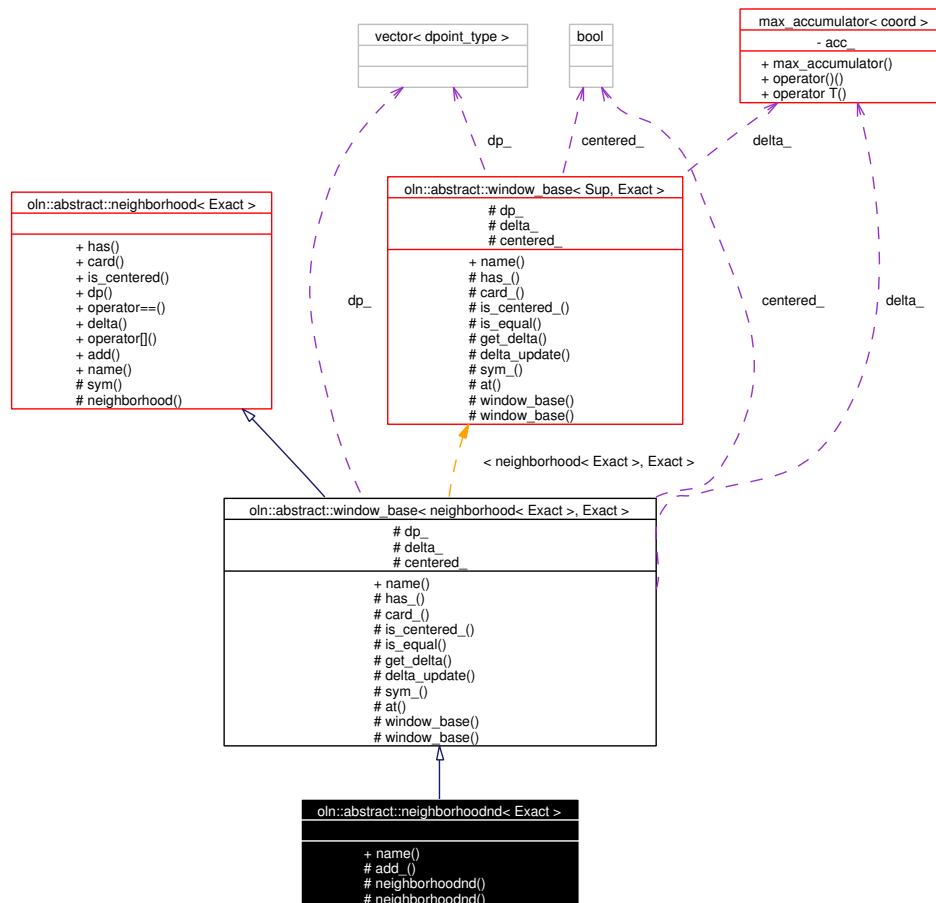
Neighborhood N dimensions.

```
#include <neighborhoodnd.hh>
```

Inheritance diagram for oln::abstract::neighborhoodnd< Exact >:



Collaboration diagram for oln::abstract::neighborhoodnd< Exact >:



Public Types

- typedef `window_base< neighborhood< Exact >, Exact >` `super_type`
Super type.
- typedef `neighborhoodnd< Exact >` `self_type`
Self type.
- typedef `Exact` `exact_type`
Exact type.
- typedef `struct_elt_traits< Exact >::dpoint_type` `dpoint_type`
The associate image's type of dpoint (move point).

Static Public Member Functions

- `std::string name ()`
Return the name of the type.

Protected Member Functions

- [exact_type](#) & [add_](#) (const [dpoint_type](#) &dp)
Add a point to the neighborhood.
 - *dp* The new point.
- [neighborhoodnd](#) ()
Construct a neighborhood N dimensions.
- [neighborhoodnd](#) (unsigned size)
Construct a neighborhood of 'size' elements.
 - *size* The number of elements to reserve for the neighborhood.

Friends

- class [neighborhood](#)< [exact_type](#) >

7.260.1 Detailed Description

template<class Exact> struct oln::abstract::neighborhoodnd< Exact >

Neighborhood N dimensions.

It looks like structuring elements but here, when you add an element, you add its opposite:

$$\forall d \in N, -d \in N$$

Points have N dimensions.

Definition at line 63 of file neighborhoodnd.hh.

7.260.2 Member Typedef Documentation

7.260.2.1 **template<class Exact> typedef struct_elt_traits<Exact>::[dpoint_type](#)**
[oln::abstract::neighborhoodnd](#)< Exact >::[dpoint_type](#)

The associate image's type of dpoint (move point).

Warning:

Prefer the macros `oln_dpoint_type(Pointable)` and `oln_dpoint_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented from [oln::abstract::window_base< neighborhood< Exact >, Exact >](#).

Reimplemented in [oln::neighborhood1d](#), [oln::neighborhood2d](#), and [oln::neighborhood3d](#).

Definition at line 75 of file neighborhoodnd.hh.

7.260.3 Member Function Documentation

7.260.3.1 `template<class Exact> exact_type& oln::abstract::neighborhoodnd< Exact >::add_` `(const dpoint_type & dp)` [`inline`, `protected`]

Add a point to the neighborhood.

- `dp` The new point.

Precondition:

`!dp.is_centered()`.

Add a new member to the neighborhood.

Definition at line 96 of file `neighborhoodnd.hh`.

```
97         {
98             precondition( !dp.is_centered() );
99             this->centered_ = true;
100             if (!(has_(dp)))
101                 this->dp_.push_back(dp);
102             delta_update(dp);
103             return this->exact();
104         }
```

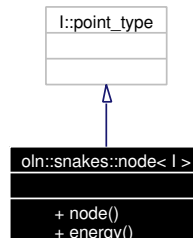
The documentation for this struct was generated from the following file:

- `neighborhoodnd.hh`

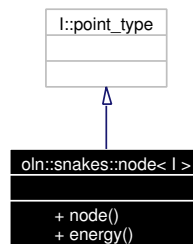
7.261 oln::snakes::node< I > Class Template Reference

```
#include <node.hh>
```

Inheritance diagram for oln::snakes::node< I >:



Collaboration diagram for oln::snakes::node< I >:



Public Types

- typedef I::point_type **point_type**
- typedef I::dpoint_type **dpoint_type**

Public Member Functions

- **node** (point_type point)
- ntg::float_s **energy** (const I &gradient, point_type prev, point_type next) const

Friends

- std::ostream & **operator<<** (std::ostream &, const **node** &)

Print the position of a node n.

7.261.1 Detailed Description

```
template<class I> class oln::snakes::node< I >
```

A node is a point used in ring.

Todo

FIXME: Do not work due to the function energy.

Definition at line 42 of file snakes/node.hh.

7.261.2 Member Function Documentation

7.261.2.1 `template<class I> ntg::float_s oln::snakes::node< I >::energy (const I & gradient,
point_type prev, point_type next) const` `[inline]`

Return the energy

Todo

FIXME: not implemented, do not work

Definition at line 37 of file node.hxx.

```
38     {  
39         return 42; // FIXME: compute the real value.  
40     }
```

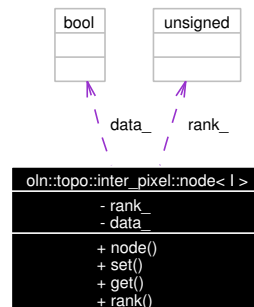
The documentation for this class was generated from the following files:

- snakes/node.hh
- node.hxx

7.262 oln::topo::inter_pixel::node< I > Class Template Reference

```
#include <node.hh>
```

Collaboration diagram for oln::topo::inter_pixel::node< I >:



Public Types

- typedef [oln::topo::inter_pixel::internal::dir_traits< I::dim >::ret](#) **dir_type**
- enum { **dim** = I::dim }

Public Member Functions

- void [set](#) (dir_type i)
Add an adge (a direction).
- bool [get](#) (dir_type i) const
Return true if the direction i joins the node.
- unsigned [rank](#) () const
Degree of the node.

7.262.1 Detailed Description

```
template<class I> class oln::topo::inter_pixel::node< I >
```

Inter pixel node.

A node is a junction of edge; the edge are represented by the directions.

Parameters:

I image.

Definition at line 46 of file topo/inter-pixel/node.hh.

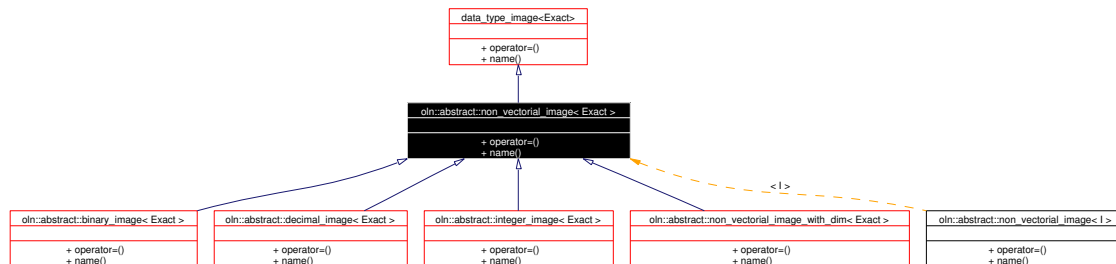
The documentation for this class was generated from the following file:

- topo/inter-pixel/node.hh

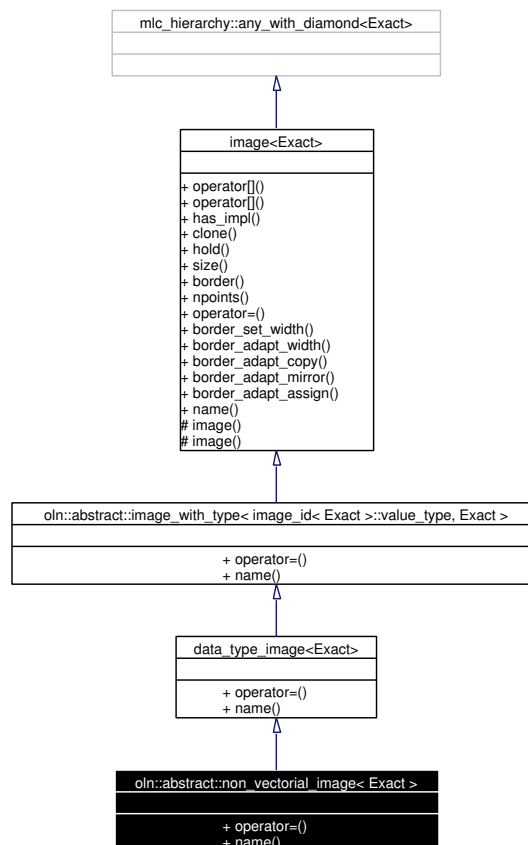
7.263 oln::abstract::non_vectorial_image< Exact > Class Template Reference

```
#include <image_with_type_with_dim.hh>
```

Inheritance diagram for oln::abstract::non_vectorial_image< Exact >:



Collaboration diagram for oln::abstract::non_vectorial_image< Exact >:



Public Types

- typedef `non_vectorial_image< Exact > self_type`

- typedef Exact **exact_type**

Public Member Functions

- exact_type & **operator=** (self_type rhs)

Perform a shallow copy between rhs and the current point, the points will not be duplicated but shared between the two images.

Static Public Member Functions

- std::string **name** ()

7.263.1 Detailed Description

template<class Exact> class oln::abstract::non_vectorial_image< Exact >

This class, when used in a function declaration, forces the function to only accept non vectorial images.

Definition at line 205 of file image_with_type_with_dim.hh.

7.263.2 Member Function Documentation

7.263.2.1 **template<class Exact> exact_type& oln::abstract::non_vectorial_image< Exact >::operator= (self_type rhs)** [inline]

Perform a shallow copy between *rhs* and the current point, the points will not be duplicated but shared between the two images.

See also:

[image::clone\(\)](#)

Reimplemented from [oln::abstract::data_type_image< Exact >](#).

Reimplemented in [oln::abstract::non_vectorial_image_with_dim< Dim, Exact >](#), [oln::abstract::binary_image< Exact >](#), [oln::abstract::binary_image_with_dim< Dim, Exact >](#), [oln::abstract::integer_image< Exact >](#), [oln::abstract::integer_image_with_dim< Dim, Exact >](#), [oln::abstract::decimal_image< Exact >](#), and [oln::abstract::decimal_image_with_dim< Dim, Exact >](#).

Definition at line 205 of file image_with_type_with_dim.hh.

```
273 {
```

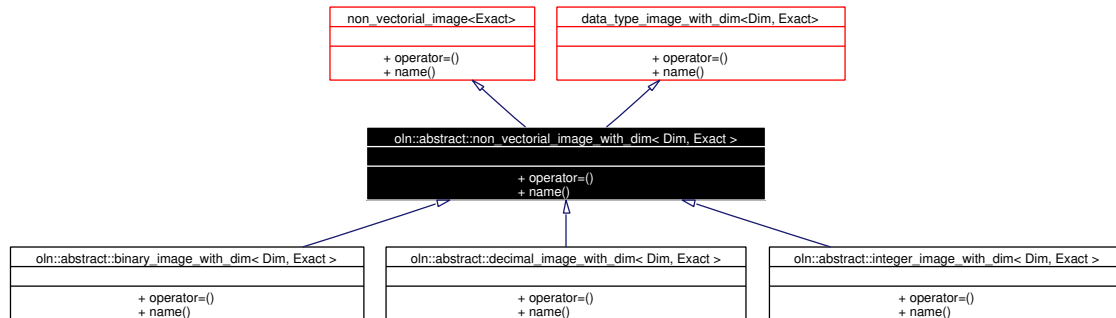
The documentation for this class was generated from the following file:

- image_with_type_with_dim.hh

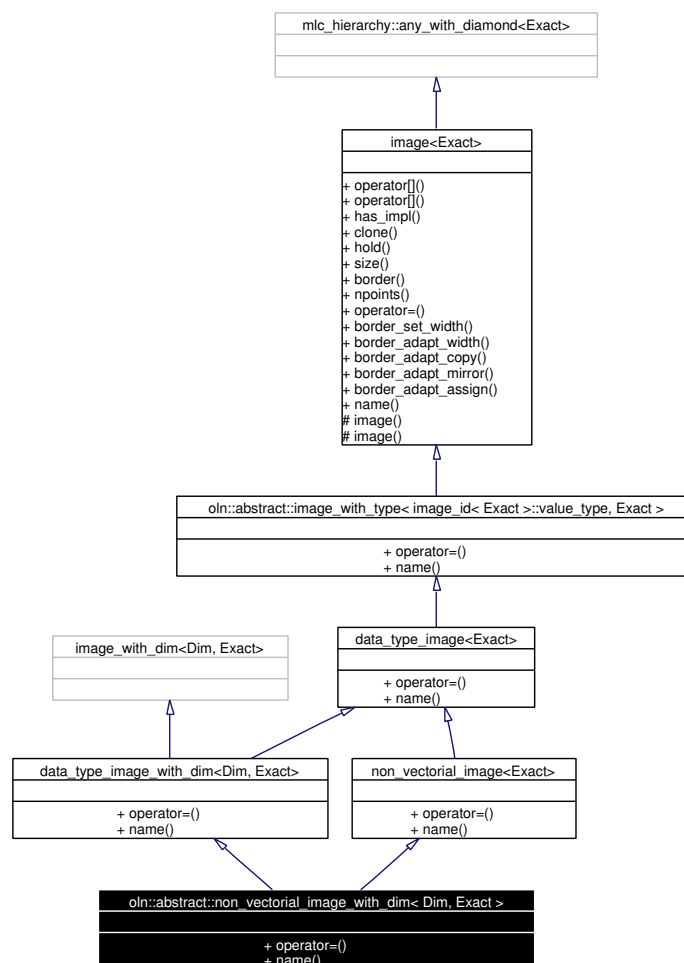
7.264 oln::abstract::non_vectorial_image_with_dim< Dim, Exact > Class Template Reference

```
#include <image_with_type_with_dim.hh>
```

Inheritance diagram for oln::abstract::non_vectorial_image_with_dim< Dim, Exact >:



Collaboration diagram for oln::abstract::non_vectorial_image_with_dim< Dim, Exact >:



Public Types

- typedef [non_vectorial_image_with_dim](#)< Dim, Exact > **self_type**
- typedef Exact **exact_type**

Public Member Functions

- exact_type & [operator=](#) (self_type rhs)
Perform a shallow copy between rhs and the current point, the points will not be duplicated but shared between the two images.

Static Public Member Functions

- std::string **name** ()

7.264.1 Detailed Description

template<unsigned Dim, class Exact> class oln::abstract::non_vectorial_image_with_dim< Dim, Exact >

This class, when used in a function declaration, forces the function to only accept non vectorial images. with a dimension equal to 'Dim'

Definition at line 205 of file image_with_type_with_dim.hh.

7.264.2 Member Function Documentation

7.264.2.1 **template<unsigned Dim, class Exact> exact_type& [non_vectorial_image_with_dim](#)< Dim, Exact >::operator= (self_type rhs)**
 [inline]

Perform a shallow copy between *rhs* and the current point, the points will not be duplicated but shared between the two images.

See also:

[image::clone\(\)](#)

Reimplemented from [oln::abstract::data_type_image_with_dim< Dim, Exact >](#).

Reimplemented in [oln::abstract::binary_image_with_dim< Dim, Exact >](#), [oln::abstract::integer_image_with_dim< Dim, Exact >](#), and [oln::abstract::decimal_image_with_dim< Dim, Exact >](#).

Definition at line 205 of file image_with_type_with_dim.hh.

```
273 {
```

The documentation for this class was generated from the following file:

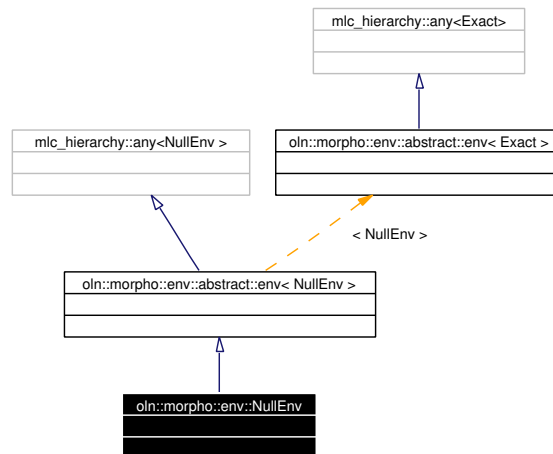
- image_with_type_with_dim.hh

7.265 oln::morpho::env::NullEnv Struct Reference

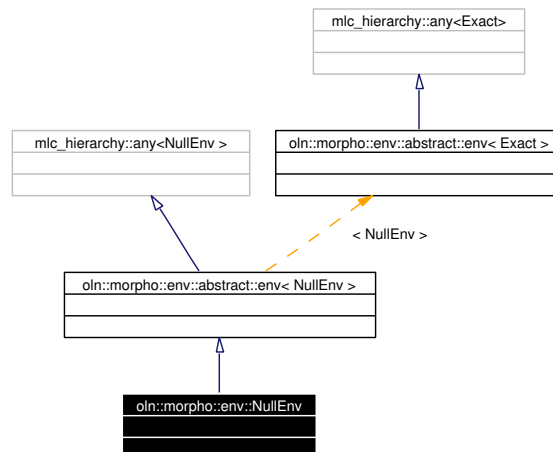
Useless environment.

```
#include <environments.hh>
```

Inheritance diagram for oln::morpho::env::NullEnv:



Collaboration diagram for oln::morpho::env::NullEnv:



7.265.1 Detailed Description

Useless environment.

This environment is an empty one.

Definition at line 59 of file `environments.hh`.

The documentation for this struct was generated from the following file:

- `environments.hh`

7.266 ntg::internal::operator_traits< Op, T, U > Struct Template Reference

Give return type for operators, depending on the input types.

```
#include <global_ops_traits.hh>
```

Inheritance diagram for ntg::internal::operator_traits< Op, T, U >:



Public Types

- typedef undefined_traits **ret**
- typedef undefined_traits **impl**
- enum { **commutative** = false }

7.266.1 Detailed Description

template<class Op, class T, class U> struct ntg::internal::operator_traits< Op, T, U >

Give return type for operators, depending on the input types.

[operator_traits](#) traits should not be used directly. Instead one should use [deduce_from_traits](#), see comments below for more details.

These traits defines 3 properties:

commutative (enum): Tells whether the operator is commutative or not.

ret (typedef): Specifies the return type.

impl (typedef): Specifies the type which implement the operator, that is, the type T such as `optraits<T>::operatorX` is the good implementation.

To specify the concerned operator, one empty class represent each operator. For example, to specify the traits associated to the + operator with T1 and T2 as arguments:

```
template <class T1, class T2> struct operator_traits<operator_plus, T1, T2> { ... }
```

Definition at line 128 of file `global_ops_traits.hh`.

The documentation for this struct was generated from the following file:

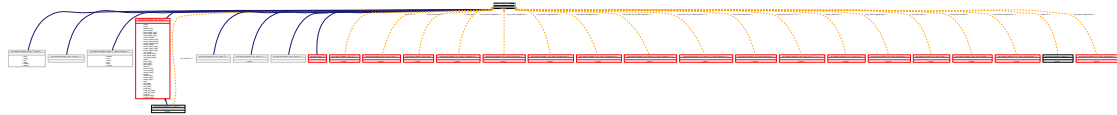
- `global_ops_traits.hh`

7.267 ntg::internal::optraits< T > Struct Template Reference

Associates functions to types.

```
#include <traits.hh>
```

Inheritance diagram for ntg::internal::optraits< T >:



Static Public Member Functions

- `std::string name ()`

7.267.1 Detailed Description

template<class T> struct ntg::internal::optraits< T >

Associates functions to types.

Definition at line 70 of file `traits.hh`.

The documentation for this struct was generated from the following file:

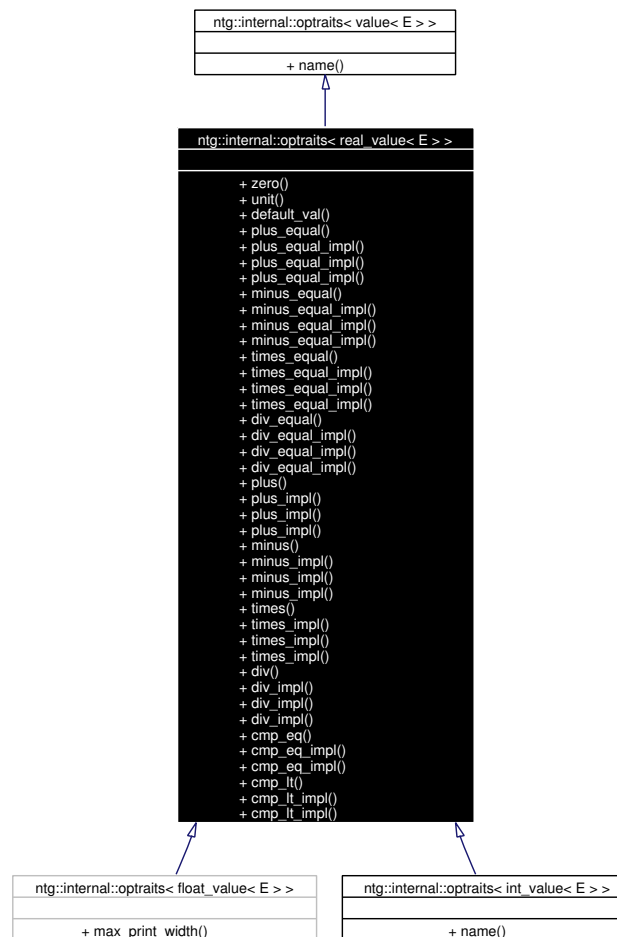
- `traits.hh`

7.268 ntg::internal::optraits< real_value< E > > Class Template Reference

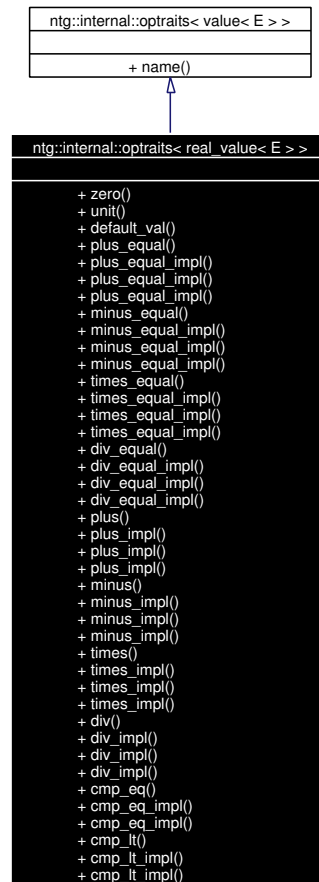
Implement common operators for scalars.

```
#include <optraits_real.hh>
```

Inheritance diagram for ntg::internal::optraits< real_value< E > >:



Collaboration diagram for ntg::internal::optraits< real_value< E > >:



Static Public Member Functions

- storage_type_ **zero** ()
- storage_type_ **unit** ()
- storage_type_ **default_val** ()
- template<class T1, class T2> T1 & **plus_equal** (T1 &lhs, const T2 &rhs)
- template<class T1, class T2> T1 & **plus_equal_impl** (ntg::real_value< T1 > &lhs, const ntg::real_value< T2 > &rhs)
- template<class T1, class T2> T1 & **plus_equal_impl** (ntg::real_value< T1 > &lhs, const ntg::any_const_class< T2 > rhs)
- template<class T1, class T2> T1 & **plus_equal_impl** (ntg::any_class< T1 > lhs, const ntg::real_value< T2 > &rhs)
- template<class T1, class T2> T1 & **minus_equal** (T1 &lhs, const T2 &rhs)
- template<class T1, class T2> T1 & **minus_equal_impl** (ntg::real_value< T1 > &lhs, const ntg::real_value< T2 > &rhs)
- template<class T1, class T2> T1 & **minus_equal_impl** (ntg::real_value< T1 > &lhs, const ntg::any_const_class< T2 > rhs)
- template<class T1, class T2> T1 & **minus_equal_impl** (ntg::any_class< T1 > lhs, const ntg::real_value< T2 > &rhs)
- template<class T1, class T2> T1 & **times_equal** (T1 &lhs, const T2 &rhs)
- template<class T1, class T2> T1 & **times_equal_impl** (ntg::real_value< T1 > &lhs, const ntg::real_value< T2 > &rhs)

- `template<class T1, class T2> T1 & times_equal_impl (ntg::real_value< T1 > &lhs, const ntg::any_const_class< T2 > rhs)`
- `template<class T1, class T2> T1 & times_equal_impl (ntg::any_class< T1 > lhs, const ntg::real_value< T2 > &rhs)`
- `template<class T1, class T2> T1 & div_equal (T1 &lhs, const T2 &rhs)`
- `template<class T1, class T2> T1 & div_equal_impl (ntg::real_value< T1 > &lhs, const ntg::real_value< T2 > &rhs)`
- `template<class T1, class T2> T1 & div_equal_impl (ntg::real_value< T1 > &lhs, const ntg::any_const_class< T2 > rhs)`
- `template<class T1, class T2> T1 & div_equal_impl (ntg::any_class< T1 > lhs, const ntg::real_value< T2 > &rhs)`
- `template<class T1, class T2> ntg::internal::deduce_from_traits< ntg::internal::operator_plus, T1, T2 >::ret plus (const T1 &lhs, const T2 &rhs)`
- `template<class T1, class T2> ntg::internal::deduce_from_traits< ntg::internal::operator_plus, T1, T2 >::ret plus_impl (const ntg::real_value< T1 > &lhs, const ntg::real_value< T2 > &rhs)`
- `template<class T1, class T2> ntg::internal::deduce_from_traits< ntg::internal::operator_plus, T1, T2 >::ret plus_impl (const ntg::real_value< T1 > &lhs, const ntg::any_const_class< T2 > &rhs)`
- `template<class T1, class T2> ntg::internal::deduce_from_traits< ntg::internal::operator_plus, T1, T2 >::ret plus_impl (const ntg::any_const_class< T1 > &lhs, const ntg::real_value< T2 > &rhs)`
- `template<class T1, class T2> ntg::internal::deduce_from_traits< ntg::internal::operator_minus, T1, T2 >::ret minus (const T1 &lhs, const T2 &rhs)`
- `template<class T1, class T2> ntg::internal::deduce_from_traits< ntg::internal::operator_minus, T1, T2 >::ret minus_impl (const ntg::real_value< T1 > &lhs, const ntg::real_value< T2 > &rhs)`
- `template<class T1, class T2> ntg::internal::deduce_from_traits< ntg::internal::operator_minus, T1, T2 >::ret minus_impl (const ntg::real_value< T1 > &lhs, const ntg::any_const_class< T2 > &rhs)`
- `template<class T1, class T2> ntg::internal::deduce_from_traits< ntg::internal::operator_minus, T1, T2 >::ret minus_impl (const ntg::any_const_class< T1 > &lhs, const ntg::real_value< T2 > &rhs)`
- `template<class T1, class T2> ntg::internal::deduce_from_traits< ntg::internal::operator_times, T1, T2 >::ret times (const T1 &lhs, const T2 &rhs)`
- `template<class T1, class T2> ntg::internal::deduce_from_traits< ntg::internal::operator_times, T1, T2 >::ret times_impl (const ntg::real_value< T1 > &lhs, const ntg::real_value< T2 > &rhs)`
- `template<class T1, class T2> ntg::internal::deduce_from_traits< ntg::internal::operator_times, T1, T2 >::ret times_impl (const ntg::real_value< T1 > &lhs, const ntg::any_const_class< T2 > &rhs)`
- `template<class T1, class T2> ntg::internal::deduce_from_traits< ntg::internal::operator_times, T1, T2 >::ret times_impl (const ntg::any_const_class< T1 > &lhs, const ntg::real_value< T2 > &rhs)`
- `template<class T1, class T2> ntg::internal::deduce_from_traits< ntg::internal::operator_div, T1, T2 >::ret div (const T1 &lhs, const T2 &rhs)`
- `template<class T1, class T2> ntg::internal::deduce_from_traits< ntg::internal::operator_div, T1, T2 >::ret div_impl (const ntg::real_value< T1 > &lhs, const ntg::real_value< T2 > &rhs)`
- `template<class T1, class T2> ntg::internal::deduce_from_traits< ntg::internal::operator_div, T1, T2 >::ret div_impl (const ntg::real_value< T1 > &lhs, const ntg::any_const_class< T2 > &rhs)`
- `template<class T1, class T2> ntg::internal::deduce_from_traits< ntg::internal::operator_div, T1, T2 >::ret div_impl (const ntg::any_const_class< T1 > &lhs, const ntg::real_value< T2 > &rhs)`
- `template<class T1, class T2> bool cmp_eq (const T1 &lhs, const T2 &rhs)`
- `template<class T> bool cmp_eq_impl (const ntg::real_value< T > &lhs, const ntg::real_value< T > &rhs)`
- `template<class T> bool cmp_eq_impl (const ntg::any_const_class< T > lhs, const ntg::any_const_class< T > rhs)`
- `template<class T1, class T2> bool cmp_lt (const T1 &lhs, const T2 &rhs)`

- `template<class T> bool cmp_lt_impl (const ntg::real_value< T > &lhs, const ntg::real_value< T > &rhs)`
- `template<class T> bool cmp_lt_impl (const ntg::any_const_class< T > lhs, const ntg::any_const_class< T > rhs)`

7.268.1 Detailed Description

`template<class E> class ntg::internal::optraits< real_value< E > >`

Implement common operators for scalars.

Definition at line 57 of file `optraits_real.hh`.

The documentation for this class was generated from the following file:

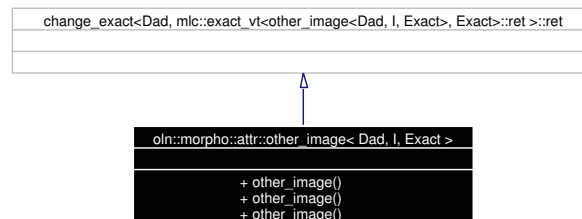
- `optraits_real.hh`

7.269 oln::morpho::attr::other_image< Dad, I, Exact > Class Template Reference

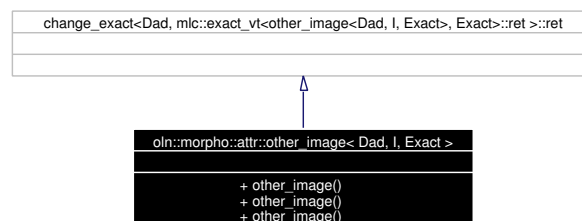
Metaclass used to change attribute behavior.

```
#include <attributes.hh>
```

Inheritance diagram for oln::morpho::attr::other_image< Dad, I, Exact >:



Collaboration diagram for oln::morpho::attr::other_image< Dad, I, Exact >:



Public Types

- typedef `other_image< Dad, I, Exact > self_type`
- typedef `abstract::image< typename mlc::exact< I >::ret > im_type`
- typedef `mlc::exact_vt< self_type, Exact >::ret exact_type`
- typedef `oln::morpho::attr::attr_traits< exact_type >::value_type value_type`
- typedef `oln::morpho::attr::attr_traits< exact_type >::env_type env_type`
- typedef `oln::morpho::attr::attr_traits< exact_type >::lambda_type lambda_type`
- typedef `change_exact< Dad, typename mlc::exact_vt< other_image< Dad, I, Exact >, Exact >::ret>::ret super_type`

Public Member Functions

- `other_image ()`
Constructor.
- `other_image (const lambda_type &lambda)`
lambda_type Constructor.
- `template<typename IM> other_image (const abstract::image< IM > &, const typename mlc::exact< I >::ret::point_type &p, const env_type &e)`

Image Constructor.

7.269.1 Detailed Description

template<class Dad, class I, class Exact = mlc::final> class oln::morpho::attr::other_image< Dad, I, Exact >

Metaclass used to change attribute behavior.

This class do the same job that its Dad parameter, but force it to work on other data.

Definition at line 522 of file attributes.hh.

7.269.2 Constructor & Destructor Documentation

7.269.2.1 template<class Dad, class I, class Exact = mlc::final> oln::morpho::attr::other_image< Dad, I, Exact >::other_image() [inline]

Constructor.

Dispatch to Dad constructor.

Definition at line 538 of file attributes.hh.

```
538                                     : super_type()
539     {
540     };
```

7.269.2.2 template<class Dad, class I, class Exact = mlc::final> oln::morpho::attr::other_image< Dad, I, Exact >::other_image(const lambda_type & lambda) [inline]

lambda_type Constructor.

Dispatch to Dad constructor.

Definition at line 547 of file attributes.hh.

```
547                                     : super_type(lambda)
548     {
549     };
```

7.269.2.3 template<class Dad, class I, class Exact = mlc::final> template<typename IM> oln::morpho::attr::other_image< Dad, I, Exact >::other_image(const abstract::image< IM > &, const typename mlc::exact< I >::ret::point_type & p, const env_type & e) [inline]

Image Constructor.

Dispatch to Dad constructor but substitute image argument with the image contained in the environment.

Definition at line 558 of file attributes.hh.

```
560                                     : super_type(e.getImage(), p, e)
561     {
562     };
```

The documentation for this class was generated from the following file:

- attributes.hh

7.270 oln::morpho::attr::other_point Class Reference

Metaclass used to change attribute behavior.

```
#include <attributes.hh>
```

7.270.1 Detailed Description

Metaclass used to change attribute behavior.

This class do the same job that its Dad parameter, but force it to work on other data.

The documentation for this class was generated from the following file:

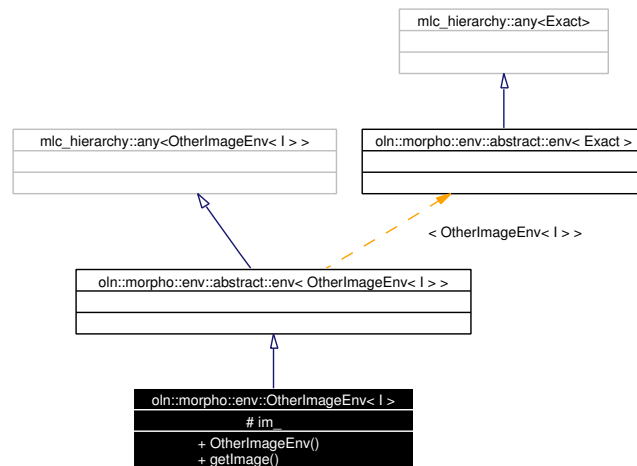
- attributes.hh

7.271 oln::morpho::env::OtherImageEnv< I > Struct Template Reference

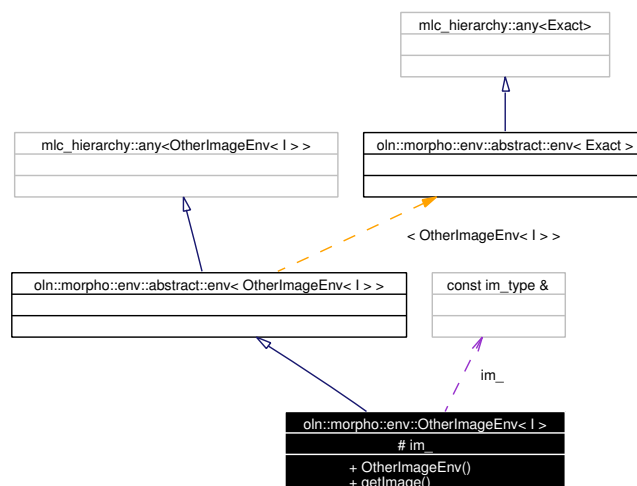
Environment containing image.

```
#include <environments.hh>
```

Inheritance diagram for oln::morpho::env::OtherImageEnv< I >:



Collaboration diagram for oln::morpho::env::OtherImageEnv< I >:



Public Types

- typedef [oln::abstract::image< I >](#) **im_type**

Public Member Functions

- **OtherImageEnv** (const [oln::abstract::image](#)< I > &im)
- const im_type & **getImage** () const

Protected Attributes

- const im_type & **im_**

7.271.1 Detailed Description

template<class I> struct oln::morpho::env::OtherImageEnv< I >

Environment containing image.

Used for image substitution in other_image attribute.

Definition at line 69 of file environments.hh.

The documentation for this struct was generated from the following file:

- environments.hh

7.272 oln::convert::abstract::internal::output< Base, T > Struct Template Reference

Retrieve the result type of a conversion.

```
#include <conversion.hh>
```

7.272.1 Detailed Description

```
template<class Base, class T> struct oln::convert::abstract::internal::output< Base, T >
```

Retrieve the result type of a conversion.

Definition at line 62 of file abstract/conversion.hh.

The documentation for this struct was generated from the following file:

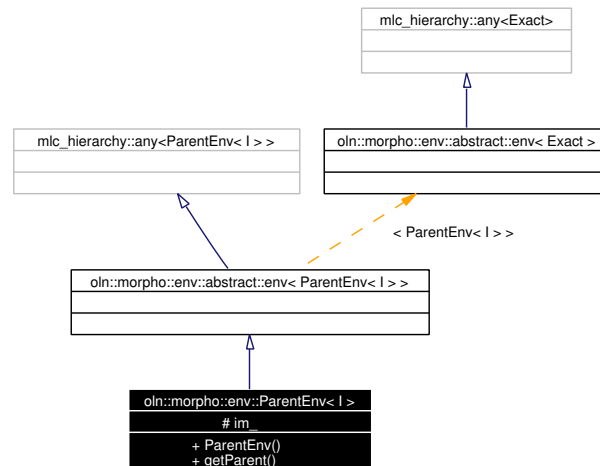
- abstract/conversion.hh

7.273 oln::morpho::env::ParentEnv< I > Struct Template Reference

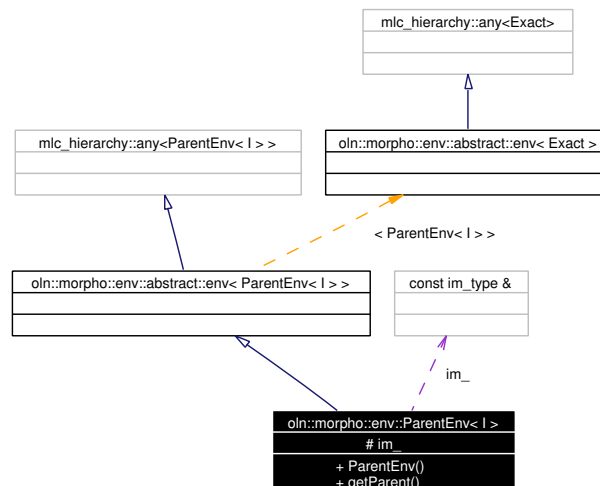
Environment containing point.

```
#include <environments.hh>
```

Inheritance diagram for oln::morpho::env::ParentEnv< I >:



Collaboration diagram for oln::morpho::env::ParentEnv< I >:



Public Types

- typedef [oln::abstract::image< I >](#) **im_type**

Public Member Functions

- **ParentEnv** (const [oln::abstract::image](#)< I > &im)
- const im_type & **getParent** () const

Protected Attributes

- const im_type & **im_**

7.273.1 Detailed Description

template<class I> struct oln::morpho::env::ParentEnv< I >

Environment containing point.

Used for point substitution in other_point attribute.

Definition at line 91 of file environments.hh.

The documentation for this struct was generated from the following file:

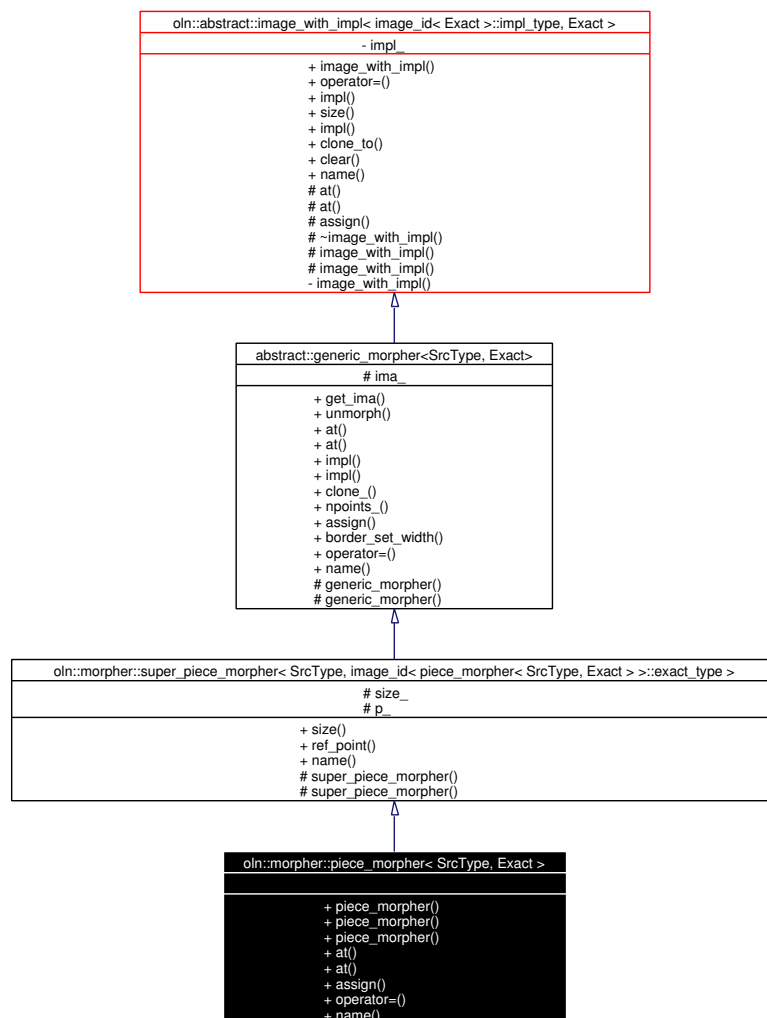
- environments.hh

7.274 oln::morpher::piece_morpher< SrcType, Exact > Struct Template Reference

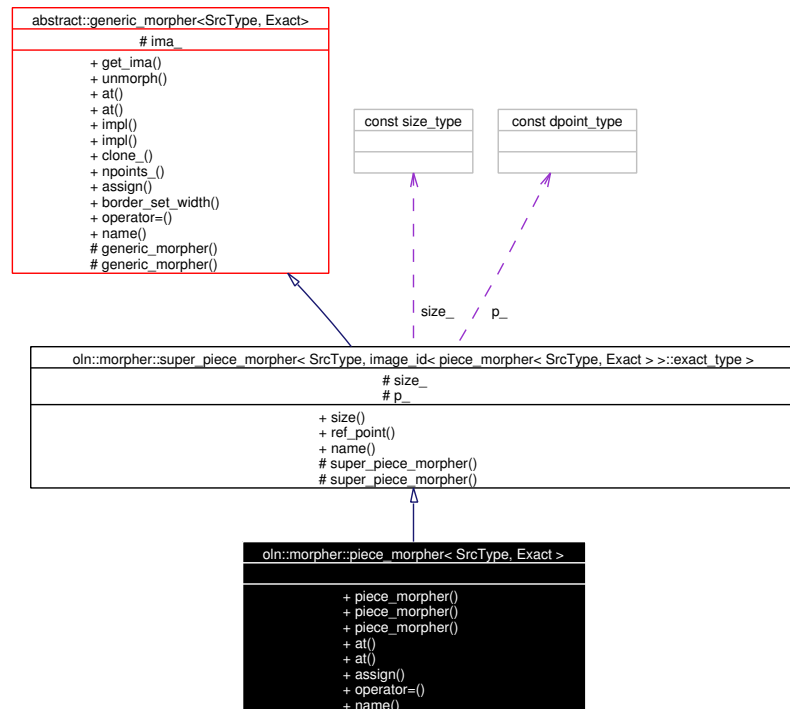
The default piece morpher class.

```
#include <piece_morpher.hh>
```

Inheritance diagram for oln::morpher::piece_morpher< SrcType, Exact >:



Collaboration diagram for oln::morpher::piece_morpher< SrcType, Exact >:



Public Types

- typedef `piece_morpher< SrcType, Exact > self_type`
The self type.
- typedef `image_id< self_type >::exact_type exact_type`
The exact type of the morpher.
- typedef `super_piece_morpher< SrcType, exact_type > super_type`
The upper class.
- typedef `image_id< exact_type >::point_type point_type`
The morpher point type.
- typedef `image_id< exact_type >::dpoint_type dpoint_type`
The morpher dpoint type.
- typedef `image_id< exact_type >::size_type size_type`
The morpher size type.
- typedef `image_id< exact_type >::value_type value_type`
The morpher value type.

Public Member Functions

- [piece_morpher](#) (const SrcType &ima, const [dpoint_type](#) p, const [size_type](#) s)
Construct the piece morpher with an image ima.
- [piece_morpher](#) (const [self_type](#) &r)
Construct the piece morpher with another piece morpher.
- [piece_morpher](#) ()
Empty constructor.
- [value_type](#) & [at](#) (const [point_type](#) &p)
Return the stored value at the point.
– *p The point.*
- const [value_type](#) [at](#) (const [point_type](#) &p) const
Return the stored value at the point.
– *p The point.*
- [self_type](#) & [assign](#) ([self_type](#) &rhs)
- [self_type](#) & [operator=](#) (SrcType &rhs)
This operator= assigns rhs to the current image.

Static Public Member Functions

- std::string [name](#) ()
Useful to debug.

7.274.1 Detailed Description

`template<class SrcType, class Exact> struct oln::morpher::piece_morpher< SrcType, Exact >`

The default piece morpher class.

Using this class, a piece of picture is a picture.

See also:

[oln::morpher::abstract::generic_morpher](#)
[oln::morpher::piece_morph](#)

Definition at line 166 of file `piece_morpher.hh`.

7.274.2 Constructor & Destructor Documentation

7.274.2.1 `template<class SrcType, class Exact> oln::morpher::piece_morpher< SrcType, Exact >::piece_morpher () [inline]`

Empty constructor.

Needed by mlc_hierarchy::any_with_diamond.

Definition at line 197 of file piece_morpher.hh.

```
198      {}
```

7.274.3 Member Function Documentation

7.274.3.1 `template<class SrcType, class Exact> self_type& oln::morpher::piece_morpher< SrcType, Exact >::assign (self_type & rhs) [inline]`

Perform a shallow copy from the decorated image of *rhs* to the current decorated image. The points will be shared by the two images.

Definition at line 228 of file piece_morpher.hh.

References oln::morpher::piece_morpher< SrcType, Exact >::at().

```
229      {
230          oln_iter_type(SrcType)  it(rhs);
231
232          for_all(it)
233              this->at(it) = rhs[it];
234          return this->exact();
235      }
```

7.274.3.2 `template<class SrcType, class Exact> const value_type oln::morpher::piece_morpher< SrcType, Exact >::at (const point_type & p) const [inline]`

Return the stored value at the point.

- p The point.

Returns:

The stored value.

Reimplemented from oln::abstract::image_with_impl< Impl, Exact >.

Definition at line 218 of file piece_morpher.hh.

```
219      {
220          return this->ima_[p + p_];
221      }
```

7.274.3.3 `template<class SrcType, class Exact> value_type& oln::morpher::piece_morpher< SrcType, Exact >::at (const point_type & p) [inline]`

Return the stored value at the point.

- p The point.

Returns:

The stored value.

Reimplemented from [oln::abstract::image_with_impl< Impl, Exact >](#).

Definition at line 206 of file piece_morpher.hh.

Referenced by `oln::morpher::piece_morpher< SrcType, Exact >::assign()`.

```
207     {  
208         return const_cast<value_type &>  
209             ( const_cast<SrcType &>(this->ima_)[p + p_] );  
210     }
```

The documentation for this struct was generated from the following file:

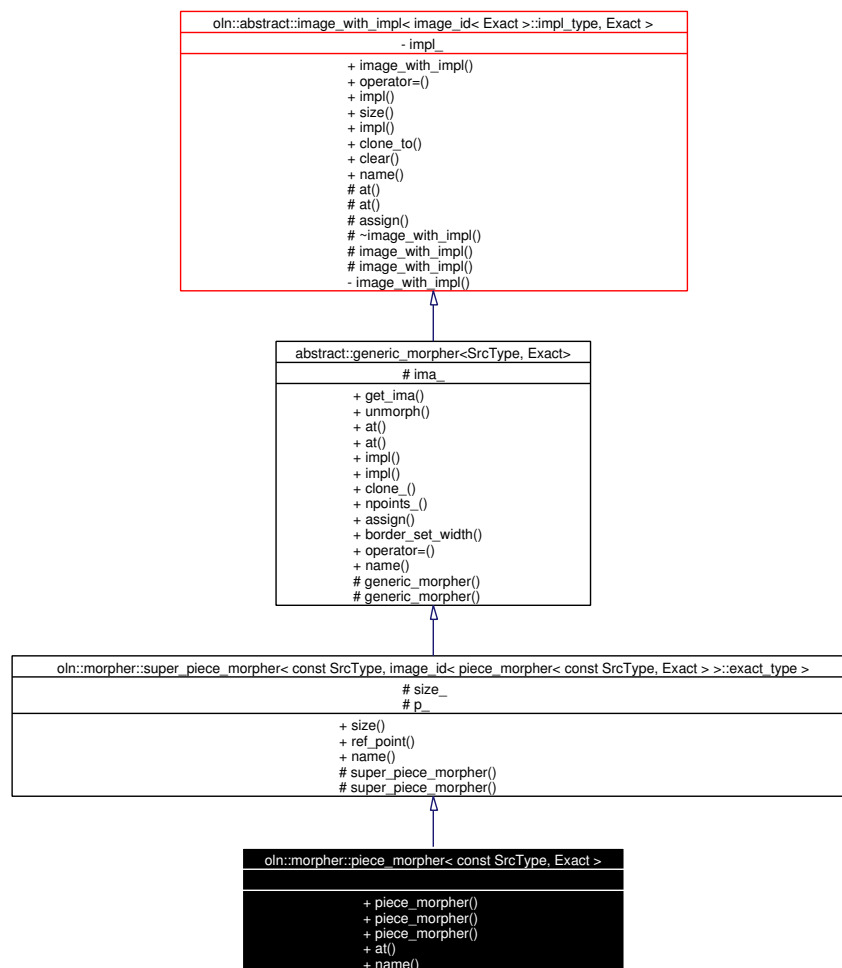
- piece_morpher.hh

7.275 oln::morpher::piece_morpher< const SrcType, Exact > Struct Template Reference

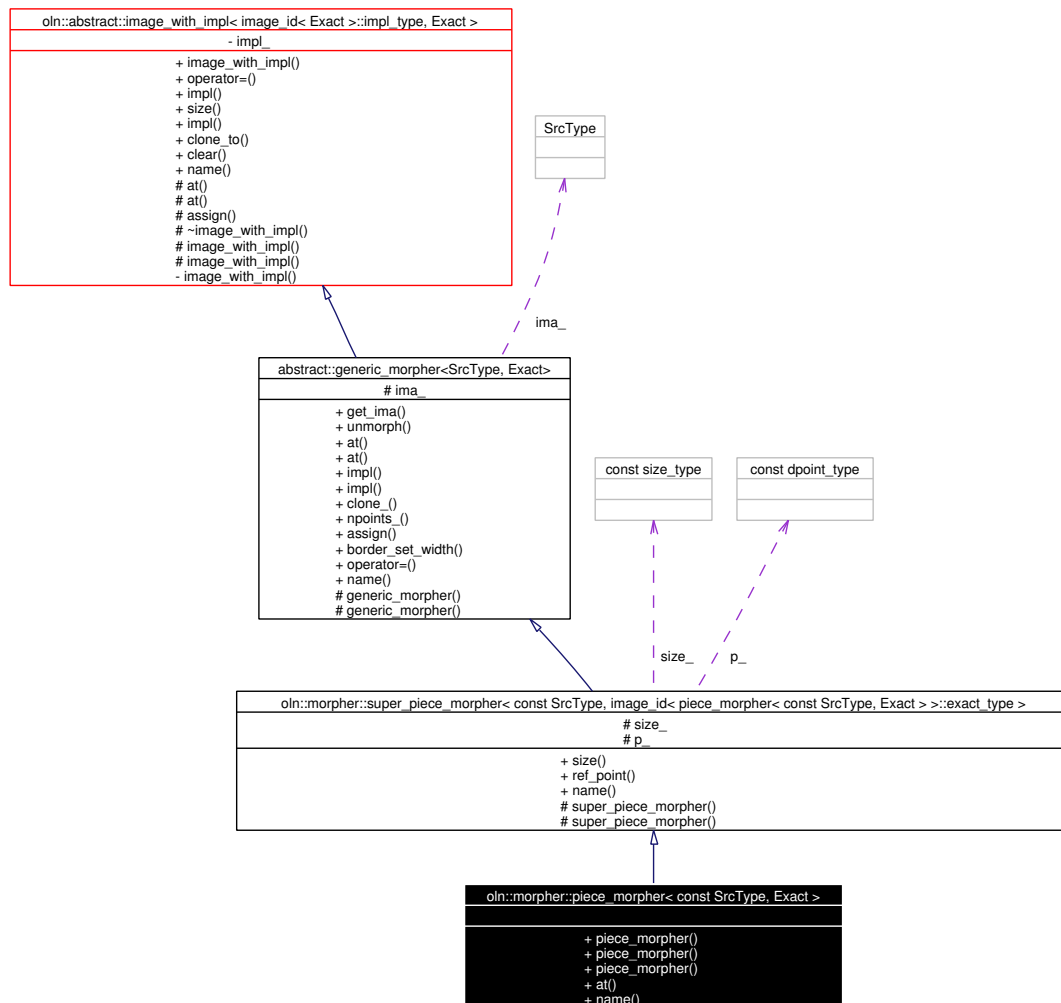
The specialized version for 'const' images.

```
#include <piece_morpher.hh>
```

Inheritance diagram for oln::morpher::piece_morpher< const SrcType, Exact >:



Collaboration diagram for oln::morpher::piece_morpher< const SrcType, Exact >:



Public Types

- typedef `piece_morpher< const SrcType, Exact >` `self_type`
The self type.
- typedef `image_id< self_type >::exact_type` `exact_type`
The exact type of the morpher.
- typedef `super_piece_morpher< const SrcType, exact_type >` `super_type`
The upper class.
- typedef `image_id< exact_type >::point_type` `point_type`
The morpher point type.
- typedef `image_id< exact_type >::dpoint_type` `dpoint_type`
The morpher dpoint type.
- typedef `image_id< exact_type >::size_type` `size_type`

The morpher size type.

- typedef image_id< [exact_type](#) >::value_type value_type

The morpher value type.

Public Member Functions

- [piece_morpher](#) (const SrcType &ima, const [dpoint_type](#) p, const [size_type](#) s)

Construct a piece morpher.

- ima *The image.*
- p *The reference point.*
- s *The size.*

- [piece_morpher](#) (const [self_type](#) &r)

Construct a piece morpher from another one.

- [piece_morpher](#) ()

Empty constructor.

- const [value_type](#) at (const [point_type](#) &p) const

Return the stored value at the point.

- p *The point.*

Static Public Member Functions

- std::string [name](#) ()

Useful to debug.

7.275.1 Detailed Description

template<class SrcType, class Exact> struct oln::morpher::piece_morpher< const SrcType, Exact >

The specialized version for 'const' images.

Definition at line 257 of file piece_morpher.hh.

7.275.2 Member Function Documentation

7.275.2.1 template<class SrcType, class Exact> const [value_type](#) oln::morpher::piece_morpher< const SrcType, Exact >::at (const [point_type](#) &p) const [inline]

Return the stored value at the point.

- p *The point.*

Returns:

The stored value.

Reimplemented from [oln::abstract::image_with_impl< Impl, Exact >](#).

Definition at line 302 of file piece_morpher.hh.

```
303     {  
304         return this->ima_[p + p_];  
305     }
```

7.275.2.2 `template<class SrcType, class Exact> oln::morpher::piece_morpher< const SrcType, Exact >::piece_morpher () [inline]`

Empty constructor.

Needed by `mlc_hierarchy::any_with_diamond`.

Definition at line 294 of file piece_morpher.hh.

```
294 {}
```

The documentation for this struct was generated from the following file:

- piece_morpher.hh

7.276 oln::io::internal::pnm_read_data< V, R > Struct Template Reference

```
#include <pnm_read_data.hh>
```

Static Public Member Functions

- template<class I> bool **read** (std::istream &, I &, const pnm2d_info &)

7.276.1 Detailed Description

template<pnm_type V, reader_id R> struct oln::io::internal::pnm_read_data< V, R >

Default version.

Definition at line 50 of file pnm_read_data.hh.

The documentation for this struct was generated from the following file:

- pnm_read_data.hh

7.277 **oln::io::internal::pnm_read_data< PnmBinary, ReadPnmPlain > Struct Template Reference**

```
#include <pnm_read_data.hh>
```

Static Public Member Functions

- `template<class I> bool read (std::istream &in, I &output, const pnm2d_info &)`

7.277.1 Detailed Description

template<> struct oln::io::internal::pnm_read_data< PnmBinary, ReadPnmPlain >

Specialized version for extracting pnm binary 2d image in plain file.

Todo

FIXME: implement an iterator over data

Definition at line 73 of file `pnm_read_data.hh`.

The documentation for this struct was generated from the following file:

- `pnm_read_data.hh`

7.278 oln::io::internal::pnm_read_data< PnmBinary, ReadPnmRaw > Struct Template Reference

```
#include <pnm_read_data.hh>
```

Static Public Member Functions

- `template<class I> bool read (std::istream &in, I &output, const pnm2d_info &info)`

7.278.1 Detailed Description

`template<> struct oln::io::internal::pnm_read_data< PnmBinary, ReadPnmRaw >`

Specialized version for extracting pnm binary 2d image in raw file.

Definition at line 103 of file `pnm_read_data.hh`.

The documentation for this struct was generated from the following file:

- `pnm_read_data.hh`

7.279 oln::io::internal::pnm_read_data< PnmInteger, ReadPnmPlain > Struct Template Reference

```
#include <pnm_read_data.hh>
```

Static Public Member Functions

- `template<class I> bool read (std::istream &in, I &output, const pnm2d_info &)`

7.279.1 Detailed Description

`template<> struct oln::io::internal::pnm_read_data< PnmInteger, ReadPnmPlain >`

Specialized version for extracting pnm integer 2d image in plain file.

Todo

FIXME: implement an iterator over data

Definition at line 145 of file `pnm_read_data.hh`.

The documentation for this struct was generated from the following file:

- `pnm_read_data.hh`

7.280 oln::io::internal::pnm_read_data< PnmInteger, ReadPnmRaw > Struct Template Reference

```
#include <pnm_read_data.hh>
```

Static Public Member Functions

- `template<class I> bool read (std::istream &in, I &output, const pnm2d_info &info)`

7.280.1 Detailed Description

`template<> struct oln::io::internal::pnm_read_data< PnmInteger, ReadPnmRaw >`

Specialized version for extracting pnm integer 2d image in raw file.

Todo

- FIXME: implement an iterator over data

Definition at line 174 of file `pnm_read_data.hh`.

The documentation for this struct was generated from the following file:

- `pnm_read_data.hh`

7.281 **oln::io::internal::pnm_read_data< PnmVectorial, ReadPnmPlain > Struct Template Reference**

```
#include <pnm_read_data.hh>
```

Static Public Member Functions

- `template<class I> bool read (std::istream &in, I &output, const pnm2d_info &)`

7.281.1 Detailed Description

template<> struct oln::io::internal::pnm_read_data< PnmVectorial, ReadPnmPlain >

Specialized version for extracting pnm vectorial 2d image in plain file.

Todo

FIXME: implement an iterator over data

Definition at line 216 of file `pnm_read_data.hh`.

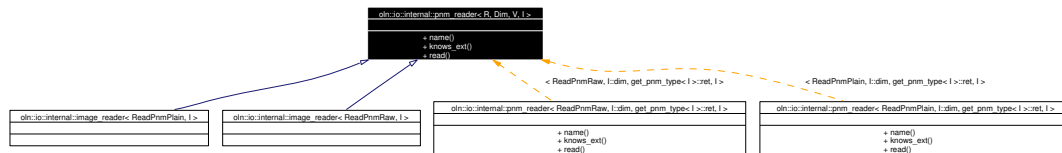
The documentation for this struct was generated from the following file:

- `pnm_read_data.hh`

7.282 oln::io::internal::pnm_reader< R, Dim, V, I > Struct Template Reference

```
#include <pnm_read.hh>
```

Inheritance diagram for oln::io::internal::pnm_reader< R, Dim, V, I >:



Static Public Member Functions

- `std::string name ()`
- `bool knows_ext (const std::string &)`
- `bool read (std::istream &, I &)`

7.282.1 Detailed Description

`template<reader_id R, unsigned Dim, pnm_type V, class I> struct oln::io::internal::pnm_reader< R, Dim, V, I >`

Default version.

Definition at line 55 of file `pnm_read.hh`.

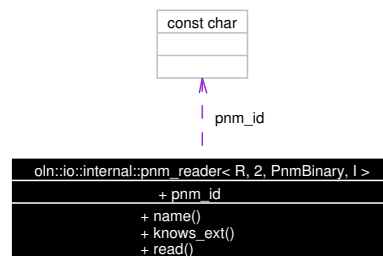
The documentation for this struct was generated from the following file:

- `pnm_read.hh`

7.283 oln::io::internal::pnm_reader< R, 2, PnmBinary, I > Struct Template Reference

```
#include <pnm_read_2d.hh>
```

Collaboration diagram for oln::io::internal::pnm_reader< R, 2, PnmBinary, I >:



Static Public Member Functions

- std::string **name** ()
- bool **knows_ext** (const std::string &ext)
- bool **read** (std::istream &in, I &im)

Initialize im and return true if the headers from in are valid, return false otherwise.

Static Public Attributes

- const char **pnm_id** = (R == ReadPnmPlain) ? '1' : '4'

7.283.1 Detailed Description

template<reader_id R, class I> struct oln::io::internal::pnm_reader< R, 2, PnmBinary, I >

Specialized version for PnmBinary (pbm image format).

Definition at line 96 of file pnm_read_2d.hh.

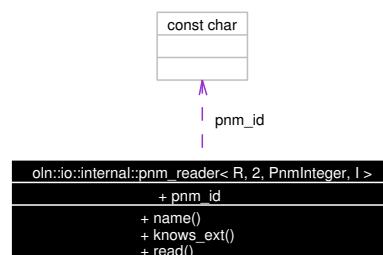
The documentation for this struct was generated from the following file:

- pnm_read_2d.hh

7.284 oln::io::internal::pnm_reader< R, 2, PnmInteger, I > Struct Template Reference

```
#include <pnm_read_2d.hh>
```

Collaboration diagram for oln::io::internal::pnm_reader< R, 2, PnmInteger, I >:



Static Public Member Functions

- std::string **name** ()
- bool **knows_ext** (const std::string &ext)
- bool **read** (std::istream &in, I &im)

Initialize im and return true if in is a valid pgm file stream, return false otherwise.

Static Public Attributes

- const char **pnm_id** = (R == ReadPnmPlain) ? '2' : '5'

7.284.1 Detailed Description

```
template<reader_id R, class I> struct oln::io::internal::pnm_reader< R, 2, PnmInteger, I >
```

Specialized version for PnmInteger (pgm image format).

Definition at line 143 of file pnm_read_2d.hh.

The documentation for this struct was generated from the following file:

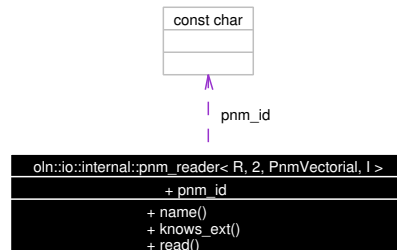
- pnm_read_2d.hh

7.285 oln::io::internal::pnm_reader< R, 2, PnmVectorial, I > Struct Template Reference

```
class pnm_reader<R, 2, PnmVectorial, I>
```

```
#include <pnm_read_2d.hh>
```

Collaboration diagram for oln::io::internal::pnm_reader< R, 2, PnmVectorial, I >:



Static Public Member Functions

- std::string **name** ()
- bool **knows_ext** (const std::string &ext)
- bool **read** (std::istream &in, I &im)

Initialize im and return true if in is a valid ppm file stream, return false otherwise.

Static Public Attributes

- const char **pnm_id** = (R == ReadPnmPlain) ? '3' : '6'

7.285.1 Detailed Description

```
template<reader_id R, class I> struct oln::io::internal::pnm_reader< R, 2, PnmVectorial, I >
```

```
class pnm_reader<R, 2, PnmVectorial, I>
```

Specialized version for pnm_reader<R, 2, PnmVectorial, I> (ppm image format).

Definition at line 196 of file pnm_read_2d.hh.

The documentation for this struct was generated from the following file:

- pnm_read_2d.hh

7.286 `oln::io::internal::pnm_reader< ReadPnmRaw, 3, P, I >` Struct Template Reference

```
#include <pnm_read_3d.hh>
```

Public Types

- typedef `pnm_reader< ReadPnmRaw, 2, P, image2d< typename mlc::exact< I >::ret::value_type >`
 `> reader_2d`
- typedef `image2d< typename mlc::exact< I >::ret::value_type > image2d_type`

Static Public Member Functions

- `std::string name()`
- `bool knows_ext(const std::string &ext)`
- `bool read(std::istream &in, I &im)`

Initialize im and return true if in is a valid pnm file format, return false otherwise.

7.286.1 Detailed Description

`template<pnm_type P, class I> struct oln::io::internal::pnm_reader< ReadPnmRaw, 3, P, I >`

Specialized version for 3d image. Only PnmRaw images can store more than one 2d image. The format is simple: each slice is stored as a bi-dimensional image.

Definition at line 59 of file `pnm_read_3d.hh`.

The documentation for this struct was generated from the following file:

- `pnm_read_3d.hh`

7.287 oln::io::internal::pnm_write Class Reference

```
#include <pnm_write.hh>
```

7.287.1 Detailed Description

Default version.

The documentation for this class was generated from the following file:

- pnm_write.hh

7.288 oln::io::internal::pnm_write_data< V, R > Struct Template Reference

```
#include <pnm_write_data.hh>
```

Static Public Member Functions

- template<class I> bool **write** (std::ostream &, const I &, const pnm2d_info &)

7.288.1 Detailed Description

template<pnm_type V, writer_id R> struct oln::io::internal::pnm_write_data< V, R >

Default version

Definition at line 50 of file pnm_write_data.hh.

The documentation for this struct was generated from the following file:

- pnm_write_data.hh

7.289 oln::io::internal::pnm_write_data< PnmBinary, WritePnmPlain > Struct Template Reference

```
#include <pnm_write_data.hh>
```

Static Public Member Functions

- template<class I> bool **write** (std::ostream &out, const I &input, const pnm2d_info &)

7.289.1 Detailed Description

template<> struct oln::io::internal::pnm_write_data< PnmBinary, WritePnmPlain >

Specialized version for writing pnm binary 3d image in plain file.

Todo

FIXME: implement an iterator over data

Definition at line 74 of file pnm_write_data.hh.

The documentation for this struct was generated from the following file:

- pnm_write_data.hh

7.290 oln::io::internal::pnm_write_data< PnmBinary, WritePnmRaw > Struct Template Reference

```
#include <pnm_write_data.hh>
```

Static Public Member Functions

- template<class I> bool **write** (std::ostream &out, const I &input, const pnm2d_info &info)

7.290.1 Detailed Description

template<> struct oln::io::internal::pnm_write_data< PnmBinary, WritePnmRaw >

Specialized version for writing pnm binary 3d image in raw file.

Todo

FIXME: implement an iterator over data

Definition at line 109 of file pnm_write_data.hh.

The documentation for this struct was generated from the following file:

- pnm_write_data.hh

7.291 oln::io::internal::pnm_write_data< PnmInteger, WritePnmPlain > Struct Template Reference

```
#include <pnm_write_data.hh>
```

Static Public Member Functions

- `template<class I> bool write (std::ostream &out, const I &input, const pnm2d_info &)`

7.291.1 Detailed Description

`template<> struct oln::io::internal::pnm_write_data< PnmInteger, WritePnmPlain >`

Specialized version for writing pnm integer 3d image in plain file.

Todo

FIXME: implement an iterator over data

Definition at line 156 of file `pnm_write_data.hh`.

The documentation for this struct was generated from the following file:

- `pnm_write_data.hh`

7.292 oln::io::internal::pnm_write_data< PnmInteger, WritePnmRaw > Struct Template Reference

```
#include <pnm_write_data.hh>
```

Static Public Member Functions

- template<class I> bool **write** (std::ostream &out, const I &input, const pnm2d_info &info)

7.292.1 Detailed Description

template<> struct oln::io::internal::pnm_write_data< PnmInteger, WritePnmRaw >

Specialized version for writing pnm integer 3d image in raw file.

Todo

FIXME: implement an iterator over data

Definition at line 191 of file pnm_write_data.hh.

The documentation for this struct was generated from the following file:

- pnm_write_data.hh

7.293 **oln::io::internal::pnm_write_data< PnmVectorial, WritePnmPlain > Struct Template Reference**

```
#include <pnm_write_data.hh>
```

Static Public Member Functions

- `template<class I> bool write (std::ostream &out, const I &input, const pnm2d_info &)`

7.293.1 Detailed Description

`template<> struct oln::io::internal::pnm_write_data< PnmVectorial, WritePnmPlain >`

Specialized version for writing pnm vectorial 3d image in plain file.

Todo

FIXME: implement an iterator over data

Definition at line 230 of file `pnm_write_data.hh`.

The documentation for this struct was generated from the following file:

- `pnm_write_data.hh`

7.294 oln::io::internal::pnm_write_data< PnmVectorial, WritePnmRaw > Struct Template Reference

```
#include <pnm_write_data.hh>
```

Static Public Member Functions

- template<class I> bool **write** (std::ostream &out, const I &input, const pnm2d_info &info)

7.294.1 Detailed Description

template<> struct oln::io::internal::pnm_write_data< PnmVectorial, WritePnmRaw >

Specialized version for writing pnm vectorial 3d image in raw file.

Todo

FIXME: implement an iterator over data

Definition at line 268 of file pnm_write_data.hh.

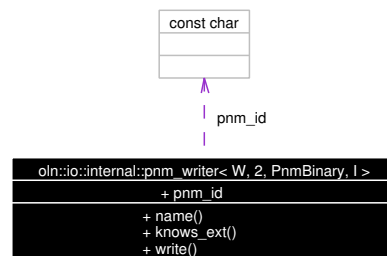
The documentation for this struct was generated from the following file:

- pnm_write_data.hh

7.295 oln::io::internal::pnm_writer< W, 2, PnmBinary, I > Struct Template Reference

```
#include <pnm_write_2d.hh>
```

Collaboration diagram for oln::io::internal::pnm_writer< W, 2, PnmBinary, I >:



Static Public Member Functions

- std::string **name** ()
- bool **knows_ext** (const std::string &ext)
- bool **write** (std::ostream &out, const I &im)
write im on out according to the pbm format, then return true.

Static Public Attributes

- const char **pnm_id** = (W == WritePnmPlain) ? '1' : '4'

7.295.1 Detailed Description

```
template<writer_id W, class I> struct oln::io::internal::pnm_writer< W, 2, PnmBinary, I >
```

Specialized version for pbm images.

Definition at line 76 of file pnm_write_2d.hh.

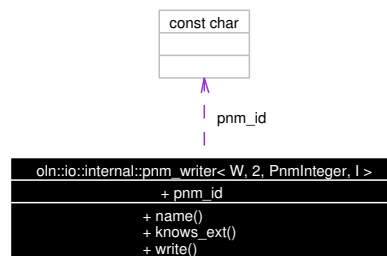
The documentation for this struct was generated from the following file:

- pnm_write_2d.hh

7.296 oln::io::internal::pnm_writer< W, 2, PnmInteger, I > Struct Template Reference

```
#include <pnm_write_2d.hh>
```

Collaboration diagram for oln::io::internal::pnm_writer< W, 2, PnmInteger, I >:



Static Public Member Functions

- std::string **name** ()
- bool **knows_ext** (const std::string &ext)
- bool **write** (std::ostream &out, const I &im)

Write im on out according to the pgm format, then return true.

Static Public Attributes

- const char **pnm_id** = (W == WritePnmPlain) ? '2' : '5'

7.296.1 Detailed Description

```
template<writer_id W, class I> struct oln::io::internal::pnm_writer< W, 2, PnmInteger, I >
```

Specialized version for pgm images.

Definition at line 126 of file pnm_write_2d.hh.

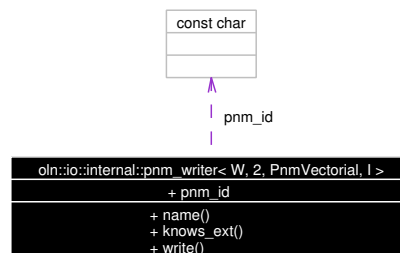
The documentation for this struct was generated from the following file:

- pnm_write_2d.hh

7.297 oln::io::internal::pnm_writer< W, 2, PnmVectorial, I > Struct Template Reference

```
#include <pnm_write_2d.hh>
```

Collaboration diagram for oln::io::internal::pnm_writer< W, 2, PnmVectorial, I >:



Static Public Member Functions

- std::string **name** ()
- bool **knows_ext** (const std::string &ext)
- bool **write** (std::ostream &out, const I &im)

Write im on out according to the ppm format, then return true.

Static Public Attributes

- const char **pnm_id** = (W == WritePnmPlain) ? '3' : '6'

7.297.1 Detailed Description

```
template<writer_id W, class I> struct oln::io::internal::pnm_writer< W, 2, PnmVectorial, I >
```

Specialized version for ppm images.

Definition at line 179 of file pnm_write_2d.hh.

The documentation for this struct was generated from the following file:

- pnm_write_2d.hh

7.298 `oln::io::internal::pnm_writer< WritePnmRaw, 3, P, I >` Struct Template Reference

```
#include <pnm_write_3d.hh>
```

Public Types

- typedef `pnm_writer< WritePnmRaw, 2, P, image2d< typename mlc::exact< I >::ret::value_type > > writer_2d`
- typedef `image2d< typename mlc::exact< I >::ret::value_type > image2d_type`

Static Public Member Functions

- `std::string name()`
- `bool knows_ext(const std::string &ext)`
- `bool write(std::ostream &out, const I &im)`

Write im on out, then return true.

7.298.1 Detailed Description

`template<pnm_type P, class I> struct oln::io::internal::pnm_writer< WritePnmRaw, 3, P, I >`

Specialized version for image 3d. Only PnmRaw images can store more than one 2d image. The format is simple: each slice is stored as a bi-dimensional image.

Definition at line 58 of file `pnm_write_3d.hh`.

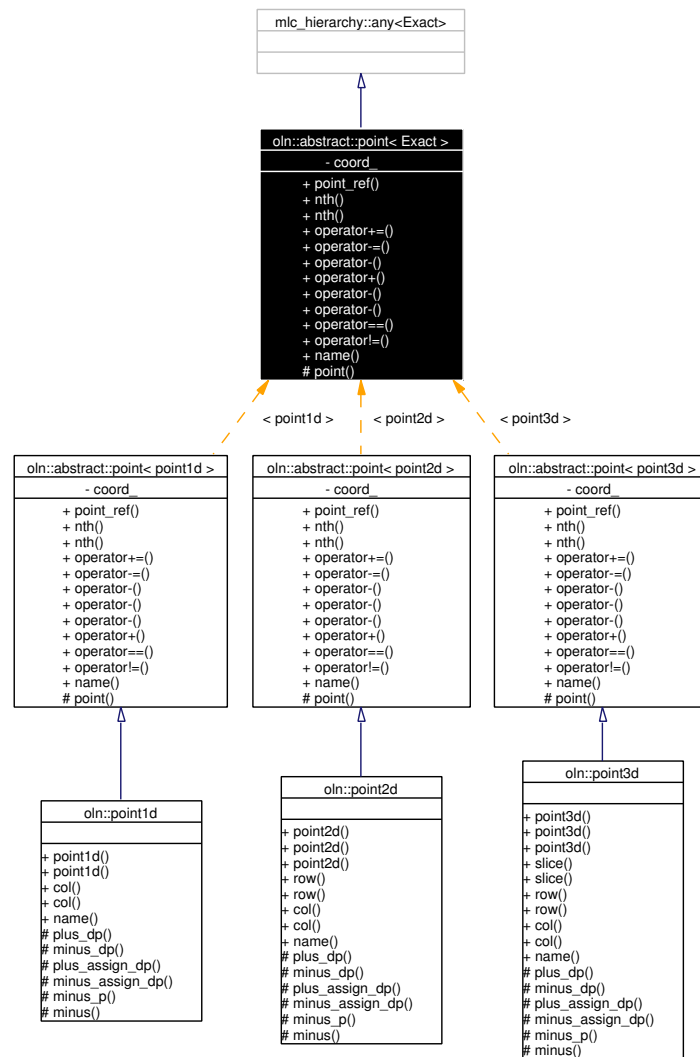
The documentation for this struct was generated from the following file:

- `pnm_write_3d.hh`

7.299 oln::abstract::point< Exact > Struct Template Reference

```
#include <point.hh>
```

Inheritance diagram for oln::abstract::point< Exact >:



Collaboration diagram for oln::abstract::point< Exact >:

Return a point whose coordinates are the opposite of the current point coordinates.

- `bool operator== (const self_type &p) const`
Test if p and the current point have the same coordinates.
- `bool operator!= (const self_type &p) const`
Test if p and the current point do not have the same coordinates.

Static Public Member Functions

- `std::string name ()`

7.299.1 Detailed Description

`template<class Exact> struct oln::abstract::point< Exact >`

The abstract class from whom derives all the others point classes.

Definition at line 71 of file point.hh.

7.299.2 Member Function Documentation

7.299.2.1 `template<class Exact> bool oln::abstract::point< Exact >::operator!= (const self_type &p) const` [inline]

Test if p and the current point do not have the same coordinates.

Returns:

False if p and the current point have the same coordinates, true otherwise.

Definition at line 201 of file point.hh.

```

202     {
203         for (unsigned i = 0; i < dim; ++i)
204             if (p.nth(i) != nth(i))
205                 return true;
206         return false;
207     }
```

7.299.2.2 `template<class Exact> exact_type oln::abstract::point< Exact >::operator+ (const abstract::dpoint< dpoint_type > &dp) const` [inline]

Give the result of the addition of a delta point dp and the current point.

Returns:

The result of the addition of dp and the current point.

Definition at line 148 of file point.hh.

```

149     {
150         return this->exact().plus_dp(dp.exact());
151     }
```

7.299.2.3 `template<class Exact> exact_type& oln::abstract::point< Exact >::operator+=(const abstract::dpoint< dpoint_type > & dp) [inline]`

Add a delta point *dp* to the current point.

Returns:

The current point plus *dp*.

Definition at line 112 of file point.hh.

```
113      {
114          return this->exact().plus_assign_dp(dp.exact());
115      }
```

7.299.2.4 `template<class Exact> exact_type oln::abstract::point< Exact >::operator- (const abstract::dpoint< dpoint_type > & dp) const [inline]`

Give the result of the subtraction of a delta point *dp* and the current point.

Returns:

The result of the subtraction of *dp* and the current point.

Definition at line 161 of file point.hh.

```
162      {
163          return this->exact().minus_dp(dp.exact());
164      }
```

7.299.2.5 `template<class Exact> dpoint_type oln::abstract::point< Exact >::operator- (const self_type & p) const [inline]`

Subtract a point *p* from the current point.

Returns:

A reference to this current point minus *p*.

Definition at line 135 of file point.hh.

```
136      {
137          return this->exact().minus_p(p.exact());
138      }
```

7.299.2.6 `template<class Exact> exact_type& oln::abstract::point< Exact >::operator-= (const abstract::dpoint< dpoint_type > & dp) [inline]`

Subtract a delta point *dp* from the current point.

Returns:

The current point minus *dp*.

Definition at line 124 of file point.hh.

```
125     {  
126         return this->exact().minus_assign_dp(dp.exact());  
127     }
```

7.299.2.7 `template<class Exact> bool oln::abstract::point< Exact >::operator==(const self_type & p) const` [inline]

Test if p and the current point have the same coordinates.

Returns:

True if p and the current point have the same coordinates, false otherwise.

Definition at line 184 of file point.hh.

```
185     {  
186         for (unsigned i = 0; i < dim; ++i)  
187             if (p.nth(i) != nth(i))  
188                 return false;  
189         return true;  
190     }
```

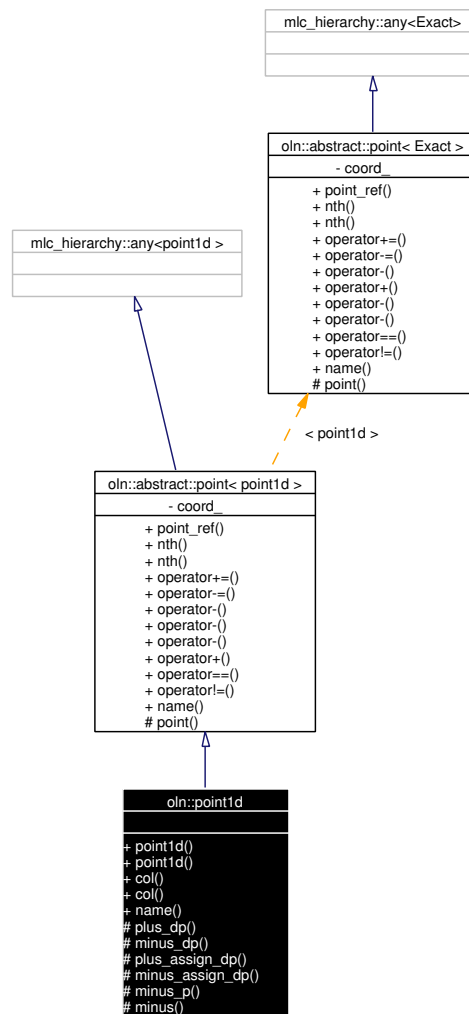
The documentation for this struct was generated from the following file:

- point.hh

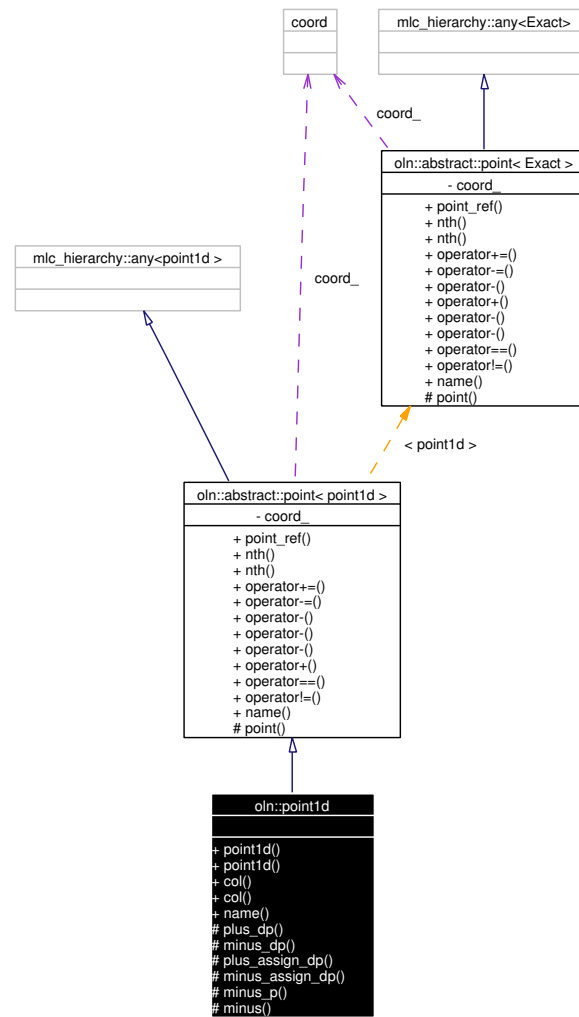
7.300 oln::point1d Class Reference

```
#include <point1d.hh>
```

Inheritance diagram for oln::point1d:



Collaboration diagram for oln::point1d:



Public Types

- typedef `abstract::point< point1d >` **super_type**
- typedef `point_traits< point1d >::dpoint_type` **dpoint_type**

Public Member Functions

- `point1d (coord col)`
The coordinate of the `point1d` is set to `col`.
- `coord col () const`
Return the value of the `point1d` coordinate.
- `coord & col ()`
Return a reference to the `point1d` coordinate.

Static Public Member Functions

- `std::string name ()`

Protected Member Functions

- `point1d plus_dp (const dpoint1d &dp) const`
Return a [point1d](#) whose coordinate is equal to dp coordinate plus the current [point1d](#) coordinate.
- `point1d minus_dp (const dpoint1d &dp) const`
Return a [point1d](#) whose coordinate is equal to the current [point1d](#) coordinate minus dp coordinate.
- `point1d & plus_assign_dp (const dpoint1d &dp)`
Return a reference to the current [point1d](#) plus dp.
- `point1d & minus_assign_dp (const dpoint1d &dp)`
Return a reference to the current [point1d](#) minus dp.
- `dpoint1d minus_p (const point1d &p) const`
Return a [dpoint1d](#) whose coordinate is equal to the current [point1d](#) coordinate minus p coordinate.
- `point1d minus () const`
Return a [point1d](#) whose coordinate is equal to the opposite of the current [point1d](#) coordinate.

Friends

- `class abstract::point< point1d >`

7.300.1 Detailed Description

Subclass of [abstract::point](#), declaration of point for [image1d](#). To instantiate a [point1d](#) on an `oln::image1d<ntg::rgb_8>` for example, use the macro `oln_point_type`.

```
oln_point_type(oln::image1d<ntg::rgb_8>) p();
```

or

```
oln_point_type(oln::image1d<ntg::rgb_8>) p(1);
```

Definition at line 68 of file `point1d.hh`.

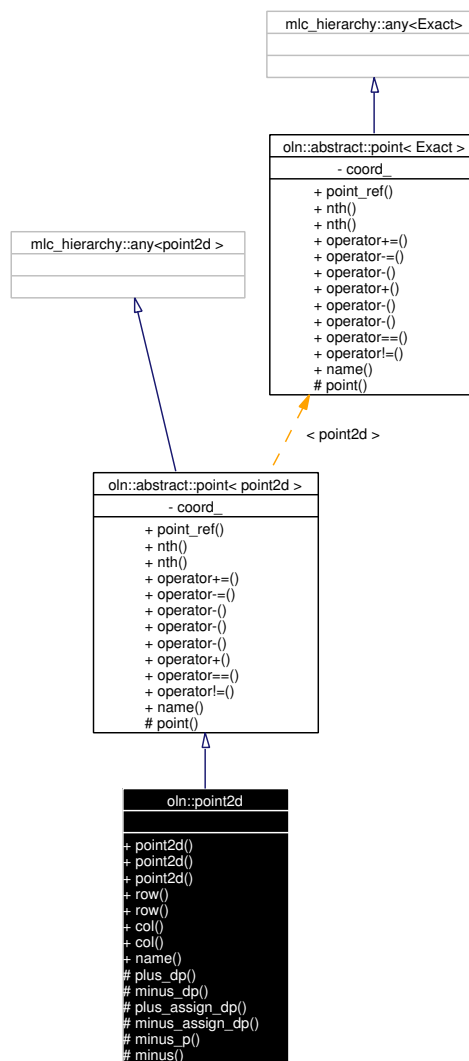
The documentation for this class was generated from the following files:

- `point1d.hh`
- `point1d.hxx`

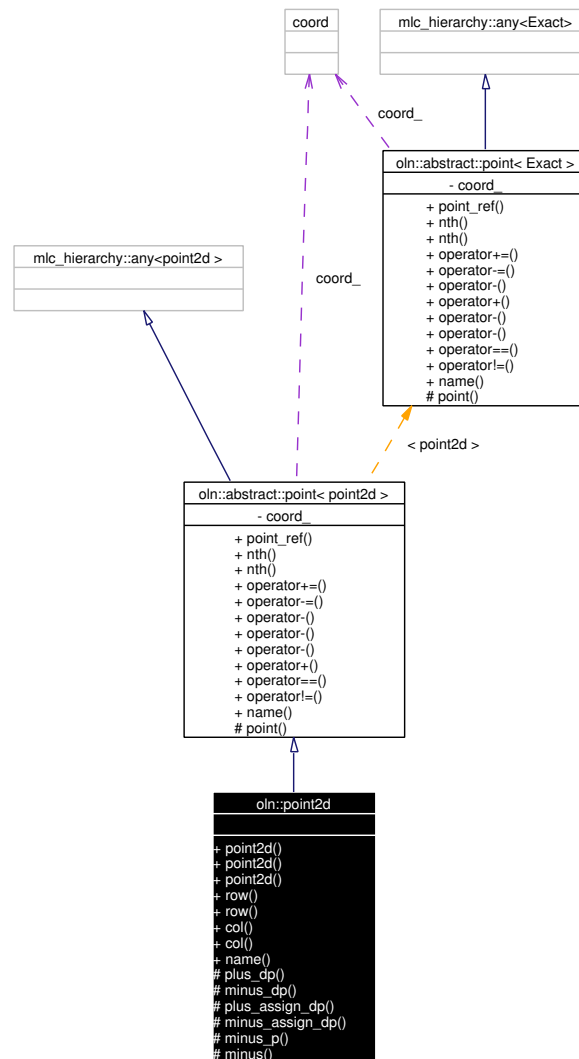
7.301 oln::point2d Class Reference

```
#include <point2d.hh>
```

Inheritance diagram for oln::point2d:



Collaboration diagram for oln::point2d:



Public Types

- typedef `abstract::point< point2d >` **super_type**
- typedef `point_traits< point2d >::dpoint_type` **dpoint_type**

Public Member Functions

- `point2d` (`coord` row, `coord` col)
The coordinates of the `point2d` are set to row and col.
- `point2d` (const `point1d` &p, `coord` col)
The coordinates of the `point2d` are set to p and col.
- `coord` row () const
Return Give the value of the `point2d` row coordinate.

- `coord & row ()`
Return a reference to the [point2d](#) row coordinate.
- `coord col () const`
Return the value of the [point2d](#) col coordinate.
- `coord & col ()`
Return a reference to the [point2d](#) col coordinate.

Static Public Member Functions

- `std::string name ()`

Protected Member Functions

- `point2d plus_dp (const dpoint2d &dp) const`
Return a [point2d](#) whose coordinates are equal to dp coordinates plus the current [point2d](#) coordinates.
- `point2d minus_dp (const dpoint2d &dp) const`
Return a [point2d](#) whose coordinates are equal to the current [point2d](#) coordinates minus dp coordinates.
- `point2d & plus_assign_dp (const dpoint2d &dp)`
Return a reference to the current [point2d](#) plus dp.
- `point2d & minus_assign_dp (const dpoint2d &dp)`
Return a reference to the current [point2d](#) minus dp.
- `dpoint2d minus_p (const point2d &p) const`
Return a [dpoint2d](#) whose coordinates are equal to the current [point2d](#) coordinates minus p coordinates.
- `point2d minus () const`
Return a [point2d](#) whose coordinates are equal to the opposite of the current [point2d](#) coordinates.

Friends

- class `abstract::point< point2d >`

7.301.1 Detailed Description

Subclass of [abstract::point](#), declaration of point for [image2d](#). To instantiate a [dpoint2d](#) on an `oln::image2d<ntg::rgb_8>` for example, use the macro `oln_point_type(I)`.

```
oln_point_type(oln::image2d<ntg::rgb_8>) p();
```

or

```
oln_point_type(oln::image2d<ntg::rgb_8>) p(1, 2);
```

Definition at line 63 of file `point2d.hh`.

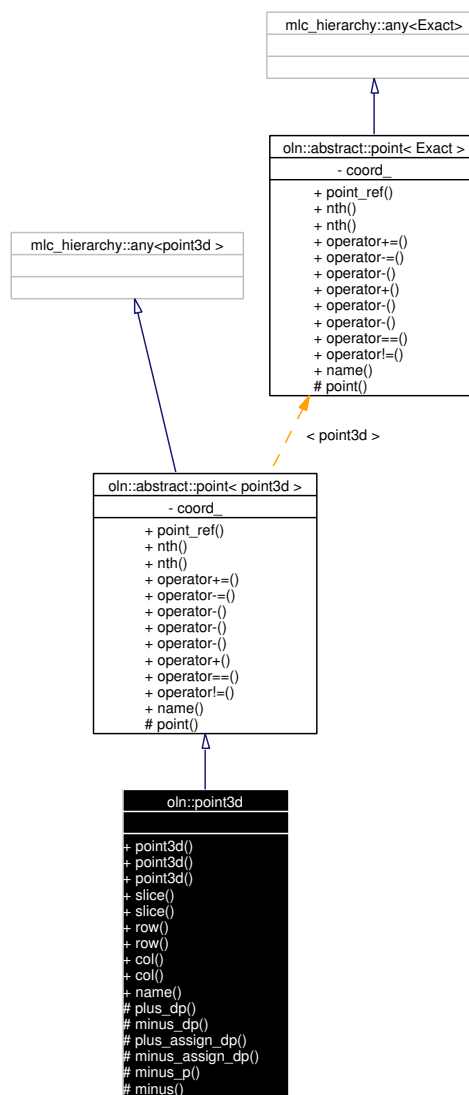
The documentation for this class was generated from the following files:

- point2d.hh
- point2d.hxx

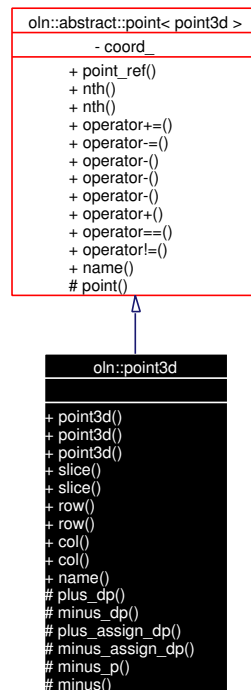
7.302 oln::point3d Class Reference

```
#include <point3d.hh>
```

Inheritance diagram for oln::point3d:



Collaboration diagram for oln::point3d:



Public Types

- typedef `abstract::point< point3d >` **super_type**
- typedef `point_traits< point3d >::dpoint_type` **dpoint_type**

Public Member Functions

- `point3d (coord slice, coord row, coord col)`
The coordinates of the `point3d` are set to slice, row, and col.
- `point3d (const point2d &p, coord slice)`
The coordinates of the `point3d` are set to p, and slice.
- `coord slice () const`
Return the value of the `point3d` slice coordinate.
- `coord & slice ()`
Return a reference to the value of the `point3d` slice coordinate.
- `coord row () const`
Return the value of the `point3d` row coordinate.
- `coord & row ()`
Return a reference to the `point3d` row coordinate.
- `coord col () const`

Return the value of the *point3d* col coordinate.

- *coord* & *col* ()

Return a reference to the *point3d* col coordinate.

Static Public Member Functions

- `std::string name ()`

Protected Member Functions

- *point3d* *plus_dp* (const *dpoint3d* &dp) const
Return a *point3d* whose coordinates are equal to dp coordinates plus the current *point3d* coordinates.
- *point3d* *minus_dp* (const *dpoint3d* &dp) const
Return a *point3d* whose coordinates are equal to the current *point3d* coordinates minus dp coordinates.
- *point3d* & *plus_assign_dp* (const *dpoint3d* &dp)
Return a reference to the current *point3d* plus dp.
- *point3d* & *minus_assign_dp* (const *dpoint3d* &dp)
Return a reference to the current *point3d* minus dp.
- *dpoint3d* *minus_p* (const *point3d* &p) const
Return a *dpoint3d* whose coordinates are equal to the current *point3d* coordinates minus p coordinates.
- *point3d* *minus* () const
Return a *point3d* whose coordinates are equal to the opposite of the current *point3d* coordinates.

Friends

- class `abstract::point< point3d >`

7.302.1 Detailed Description

Subclass of `abstract::point`, declaration of point for *image3d*. To instantiate a *point3d* on an `oln::image3d<ngt::rgb_8>` for example, use the macro `oln_point_type(I)`.

```
oln_point_type(oln::image3d<ngt::rgb_8>) p();
```

or

```
oln_point_type(oln::image3d<ngt::rgb_8>) p(1, 2, 3);
```

Definition at line 67 of file `point3d.hh`.

The documentation for this class was generated from the following files:

- `point3d.hh`
- `point3d.hxx`

7.303 oln::point_traits< abstract::point< Exact > > Struct Template Reference

```
#include <point.hh>
```

7.303.1 Detailed Description

template<class Exact> struct oln::point_traits< abstract::point< Exact > >

The specialized implementation for [abstract::point](#).

Definition at line 57 of file point.hh.

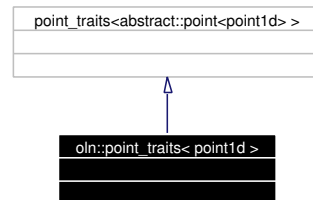
The documentation for this struct was generated from the following file:

- point.hh

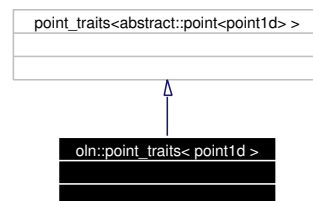
7.304 oln::point_traits< point1d > Struct Template Reference

```
#include <point1d.hh>
```

Inheritance diagram for oln::point_traits< point1d >:



Collaboration diagram for oln::point_traits< point1d >:



Public Types

- typedef [dpoint1d](#) **dpoint_type**
- enum { **dim** = 1 }

7.304.1 Detailed Description

template<> struct oln::point_traits< point1d >

The specialized version for [point1d](#).

Definition at line 51 of file point1d.hh.

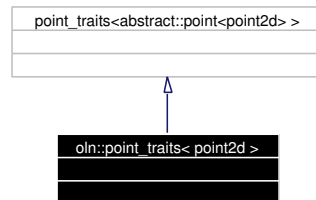
The documentation for this struct was generated from the following file:

- point1d.hh

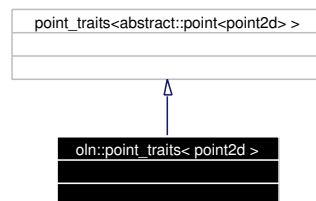
7.305 oln::point_traits< point2d > Struct Template Reference

```
#include <point2d.hh>
```

Inheritance diagram for oln::point_traits< point2d >:



Collaboration diagram for oln::point_traits< point2d >:



Public Types

- typedef `dpoint2d` `dpoint_type`
- enum { `dim` = 2 }

7.305.1 Detailed Description

`template<> struct oln::point_traits< point2d >`

The specialized version for `point2d`.

Definition at line 46 of file `point2d.hh`.

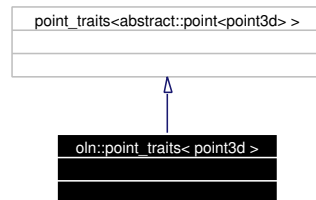
The documentation for this struct was generated from the following file:

- `point2d.hh`

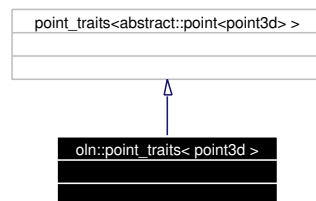
7.306 oln::point_traits< point3d > Struct Template Reference

```
#include <point3d.hh>
```

Inheritance diagram for oln::point_traits< point3d >:



Collaboration diagram for oln::point_traits< point3d >:



Public Types

- typedef [dpoint3d](#) **dpoint_type**
- enum { **dim** = 3 }

7.306.1 Detailed Description

template<> struct oln::point_traits< point3d >

The specialized version for [point3d](#).

Definition at line 50 of file point3d.hh.

The documentation for this struct was generated from the following file:

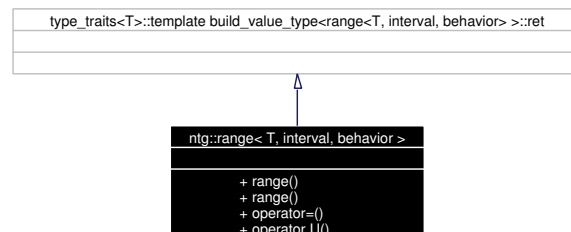
- point3d.hh

7.307 ntg::range< T, interval, behavior > Class Template Reference

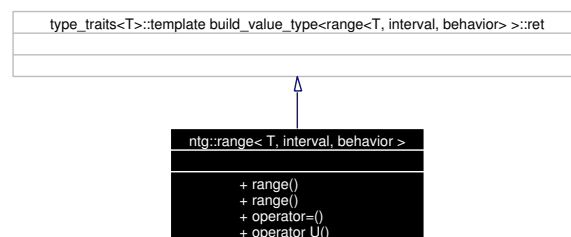
Restrict the interval of a type.

```
#include <range.hh>
```

Inheritance diagram for ntg::range< T, interval, behavior >:



Collaboration diagram for ntg::range< T, interval, behavior >:



Public Types

- typedef `range< T, interval, behavior >` **self**

Public Member Functions

- template<class U> **range** (const U &u)
- template<class U> self & **operator=** (const U &u)
- template<class U> **operator U** () const

7.307.1 Detailed Description

```
template<class T, class interval, class behavior> class ntg::range< T, interval, behavior >
```

Restrict the interval of a type.

Definition at line 84 of file real/range.hh.

The documentation for this class was generated from the following file:

- real/range.hh

7.308 oln::io::internal::readers_trier Struct Reference

Readers trier functor, helper for stream_wrappers.

```
#include <image_read.hh>
```

Static Public Member Functions

- `template<class T> bool doit (T &output, std::istream &in, const std::string ext)`

Readers trier functor, helper for stream_wrappers.

- *output* The new object.
- *in* The stream to read from.
- *ext* The extension.

7.308.1 Detailed Description

Readers trier functor, helper for stream_wrappers.

Definition at line 127 of file image_read.hh.

7.308.2 Member Function Documentation

7.308.2.1 `template<class T> bool oln::io::internal::readers_trier::doit (T & output, std::istream & in, const std::string ext) [inline, static]`

Readers trier functor, helper for stream_wrappers.

- *output* The new object.
- *in* The stream to read from.
- *ext* The extension.

First try to read by extension, then by data.

Definition at line 139 of file image_read.hh.

```
140         {
141             bool result = try_readers<ReadAny,T>::by_extension(output, in, ext);
142             if (!result)
143                 result = try_readers<ReadAny,T>::by_data(output, in);
144             return result;
145         }
```

The documentation for this struct was generated from the following file:

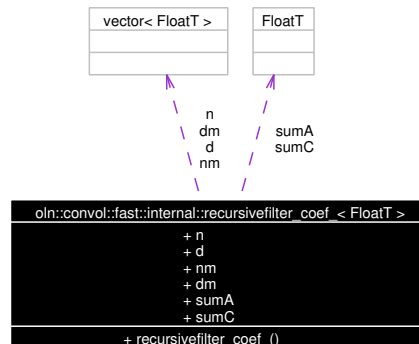
- image_read.hh

7.309 oln::convol::fast::internal::recursivefilter_coef_< FloatT > Struct Template Reference

Data structure for coefficients used for a recursive filter call.

```
#include <fast_gaussian_coefficient.hh>
```

Collaboration diagram for oln::convol::fast::internal::recursivefilter_coef_< FloatT >:



Public Types

- enum **FilterType** { **DericheGaussian**, **DericheGaussianFirstDerivative**, **DericheGaussian-SecondDerivative** }

Public Member Functions

- [recursivefilter_coef_](#) (FloatT a0, FloatT a1, FloatT b0, FloatT b1, FloatT c0, FloatT c1, FloatT w0, FloatT w1, ntg::float_d s, FilterType filter_type)

Constructor.

Public Attributes

- std::vector< FloatT > **n**
- std::vector< FloatT > **d**
- std::vector< FloatT > **nm**
- std::vector< FloatT > **dm**
- FloatT **sumA**
- FloatT **sumC**

7.309.1 Detailed Description

```
template<class FloatT> struct oln::convol::fast::internal::recursivefilter_coef_< FloatT >
```

Data structure for coefficients used for a recursive filter call.

Definition at line 58 of file fast_gaussian_coefficient.hh.

The documentation for this struct was generated from the following file:

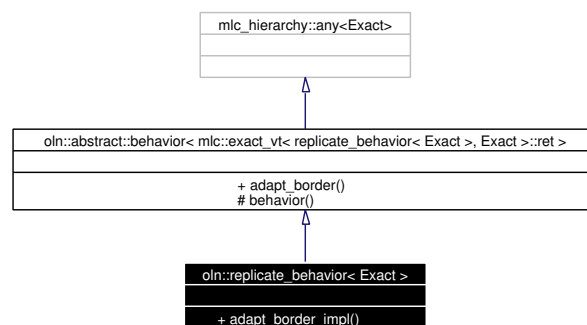
- `fast_gaussian_coefficient.hh`

7.310 oln::replicate_behavior< Exact > Class Template Reference

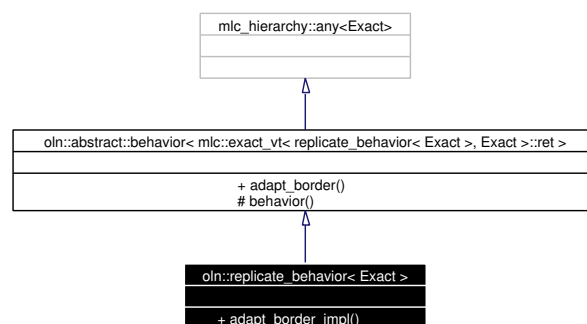
[replicate_behavior](#)

```
#include <behavior.hh>
```

Inheritance diagram for oln::replicate_behavior< Exact >:



Collaboration diagram for oln::replicate_behavior< Exact >:



Public Types

- typedef [replicate_behavior< Exact >](#) [self_type](#)
- typedef `mlc::exact_vt< self_type, Exact >::ret` [exact_type](#)

Public Member Functions

- `template<class I> void adapt_border_impl (abstract::image< I > &im, coord border_size) const`

7.310.1 Detailed Description

```
template<class Exact = mlc::final> class oln::replicate_behavior< Exact >
```

[replicate_behavior](#)

Replicate the border.

See also:

[abstract::image_size::border_](#)

Definition at line 96 of file olena/oln/core/behavior.hh.

7.310.2 Member Typedef Documentation

7.310.2.1 `template<class Exact = mlc::final> typedef mlc::exact_vt< self_type , Exact >::ret
oln::replicate_behavior< Exact >::exact_type`

The exact type.

Reimplemented from [oln::abstract::behavior< Exact >](#).

Definition at line 101 of file olena/oln/core/behavior.hh.

7.310.2.2 `template<class Exact = mlc::final> typedef replicate_behavior<Exact>
oln::replicate_behavior< Exact >::self_type`

The self type.

Reimplemented from [oln::abstract::behavior< Exact >](#).

Definition at line 100 of file olena/oln/core/behavior.hh.

The documentation for this class was generated from the following file:

- olena/oln/core/behavior.hh

7.311 ntg::internal::ret_behavior_if< need_check, Ret > Struct Template Reference

Determine the behavior to use depending on check requirements.

```
#include <behavior.hh>
```

Public Types

- typedef [typetraits](#)< Ret >::abstract_behavior_type **ret**

7.311.1 Detailed Description

template<bool need_check, class Ret> struct ntg::internal::ret_behavior_if< need_check, Ret >

Determine the behavior to use depending on check requirements.

If need_check is true, the returned behavior will be the same as the previously determined return type (generally safe).

In some cases, no check is required, thus type used to perform the calculus does not need to be safe. The force behavior is returned in such cases.

Definition at line 447 of file `integre/ntg/real/behavior.hh`.

The documentation for this struct was generated from the following file:

- `integre/ntg/real/behavior.hh`

7.312 **ntg::saturate** Struct Reference

Bound values to the nearest limit when an overflow occurs.

```
#include <behavior.hh>
```

Static Public Member Functions

- `std::string name ()`

7.312.1 Detailed Description

Bound values to the nearest limit when an overflow occurs.

Definition at line 254 of file `integre/ntg/real/behavior.hh`.

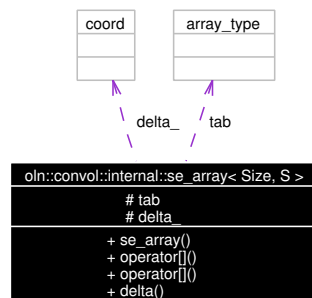
The documentation for this struct was generated from the following file:

- `integre/ntg/real/behavior.hh`

7.313 oln::convol::internal::se_array< Size, S > Struct Template Reference

```
#include <nagao.hh>
```

Collaboration diagram for oln::convol::internal::se_array< Size, S >:



Public Types

- `typedef mlc::exact< oln::abstract::struct_elt< S >>::ret struct_elt_type`
Type of the structuring elements.
- `typedef mlc::array1d< typename mlc::array1d_info< Size >, struct_elt_type > array_type`
Type of the array.
- `enum { size = Size }`
Number of structuring elements.

Public Member Functions

- `struct_elt_type & operator\[\] (int i)`
nth structuring element.
- `struct_elt_type operator\[\] (int i) const`
nth structuring element.
- `coord delta () const`
Maximum of the delta of the structuring elements.

Protected Attributes

- `array_type tab`
Array of structuring elements.
- `coord delta_`
Maximum delta.

7.313.1 Detailed Description

template<unsigned Size, class S> struct oln::convol::internal::se_array< Size, S >

array of structuring elements.

Parameters:

Size Number of structuring elements.

S structuring element type.

Definition at line 50 of file nagao.hh.

7.313.2 Member Data Documentation

7.313.2.1 **template<unsigned Size, class S> coord oln::convol::internal::se_array< Size, S >::delta_** [mutable, protected]

Maximum delta.

Note:

delta_ < 0 means that it must be updated before a read access.

Definition at line 103 of file nagao.hh.

Referenced by oln::convol::internal::se_array< Size, S >::delta(), and oln::convol::internal::se_array< Size, S >::operator[]().

The documentation for this struct was generated from the following file:

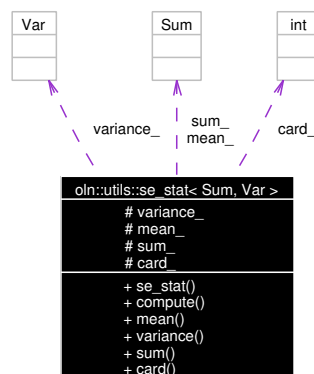
- nagao.hh

7.314 oln::utils::se_stat< Sum, Var > Class Template Reference

Compute the variance and the mean within a window.

```
#include <se_stat.hh>
```

Collaboration diagram for oln::utils::se_stat< Sum, Var >:



Public Types

- typedef Sum [sum_type](#)
type of used to sum values (usually floating point values)
- typedef Var [variance_type](#)
type of the variance (usually floating point values)

Public Member Functions

- template<class I, class S> [se_stat](#) (const [oln::abstract::image](#)< I > &im, const typename mlc::exact< I >::ret::point_type &p, const [oln::abstract::struct_elt](#)< S > &s)
- template<class I, class S> [se_stat](#) & [compute](#) (const [oln::abstract::image](#)< I > &im, const typename mlc::exact< I >::ret::point_type &p, const [oln::abstract::struct_elt](#)< S > &s)
Compute the mean and the variance.
- Sum [mean](#) () const
Mean.
- Var [variance](#) () const
Variance.
- Sum [sum](#) () const
Sum.
- unsigned [card](#) () const
Cardinal of the structuring element.

Protected Attributes

- Var [variance_](#)
Variance.
- Sum [mean_](#)
Mean.
- Sum [sum_](#)
Sum.
- int [card_](#)
Card of the input structuring element.

7.314.1 Detailed Description

`template<typename Sum = ntg::float_s, typename Var = ntg::float_s> class oln::utils::se_stat< Sum, Var >`

Compute the variance and the mean within a window.

Parameters:

Sum type used to compute the sum and the average.

Var type used to compute the variance.

Note:

There are two parameters because for vectorial types the sum is usually a vector, and the variance a non vectorial type.

Definition at line 103 of file `se_stat.hh`.

7.314.2 Constructor & Destructor Documentation

7.314.2.1 `template<typename Sum = ntg::float_s, typename Var = ntg::float_s> template<class I, class S> oln::utils::se_stat< Sum, Var >::se_stat (const oln::abstract::image< I > & im, const typename mlc::exact< I >::ret::point_type & p, const oln::abstract::struct_elt< S > & s) [inline]`

Build a se stat.

- *im* Input image.
- *p* Origin of the structuring element.
- *s* Window.

Definition at line 117 of file `se_stat.hh`.

References `oln::utils::se_stat< Sum, Var >::compute()`.


```
120      {  
121          compute(im, p, s);  
122      }
```

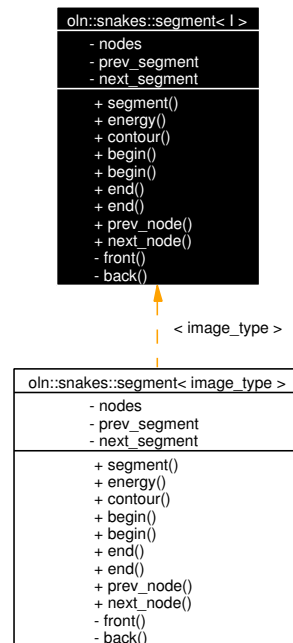
The documentation for this class was generated from the following file:

- se_stat.hh

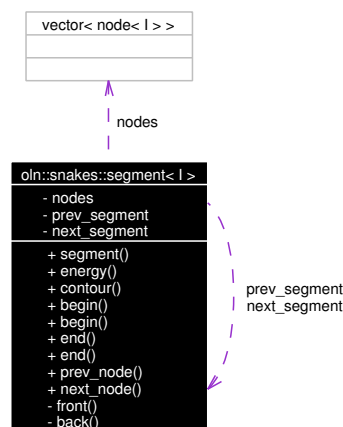
7.315 oln::snakes::segment< I > Class Template Reference

```
#include <segment.hh>
```

Inheritance diagram for oln::snakes::segment< I >:



Collaboration diagram for oln::snakes::segment< I >:



Public Types

- typedef std::vector< [node](#)< I > ::iterator **iter_type**
- typedef std::vector< [node](#)< I > ::const_iterator **const_iter_type**
- typedef I::point_type **point_type**

Public Member Functions

- **segment** (std::list< point_type > &initial_contour)
- ntg::float_s **energy** (const I &gradient) const
- std::list< point_type > **contour** (void) const

Return the points of the segment.

- const_iter_type **begin** (void) const
- iter_type **begin** (void)

Return an iterator on the first node of the segment.

- const_iter_type **end** (void) const
- iter_type **end** (void)

Return an iterator on the last node of the segment.

- const **node**< I > **prev_node** (void) const

Node before the first node of this segment.

- const **node**< I > **next_node** (void) const

Node after the next node of this segment.

Friends

- std::ostream & **operator**<< (std::ostream &, const **segment** &)

7.315.1 Detailed Description

template<class I> class oln::snakes::segment< I >

A segment is a list of node.

Todo

FIXME: Do not work due to the function **node::energy**.

Definition at line 45 of file segment.hh.

7.315.2 Member Function Documentation

7.315.2.1 template<class I> ntg::float_s **oln::snakes::segment**< I >::**energy** (const I & *gradient*)
const [inline]

Just iterate through the vector and sums up point energies.

Attention:

FIXME: Do not work due to the function **node::energy**.

Definition at line 49 of file segment.hxx.

References oln::snakes::segment< I >::next_node(), and oln::snakes::segment< I >::prev_node().

```
50     {
51         ntg::float_s e = 0.f;
52         typename segment<I>::const_iter_type p = nodes.begin();
53         typename segment<I>::const_iter_type c = nodes.begin();
54         typename segment<I>::const_iter_type n = nodes.begin();
55
56         ++n;
57         e += c->energy(gradient, prev_node(), *n);
58         for (++c, ++n; n != nodes.end(); ++p, ++c, ++n)
59             e += c->energy(gradient, *p, *n);
60         e += c->energy(gradient, *p, next_node());
61         return e;
62     }
```

The documentation for this class was generated from the following files:

- segment.hh
- segment.hxx

7.316 oln::utils::select_distrib_sort< reverse > Struct Template Reference

```
#include <histogram.hh>
```

Public Member Functions

- template<class I> void **operator()** (const oln::abstract::image< I > &im, std::vector< typename mlc::exact< I >::ret::point_type > &v)

7.316.1 Detailed Description

template<bool reverse> struct oln::utils::select_distrib_sort< reverse >

Select statically the good distrib_sort.

Parameters:

reverse If the sort should be reverted or not.

Definition at line 749 of file histogram.hh.

The documentation for this struct was generated from the following file:

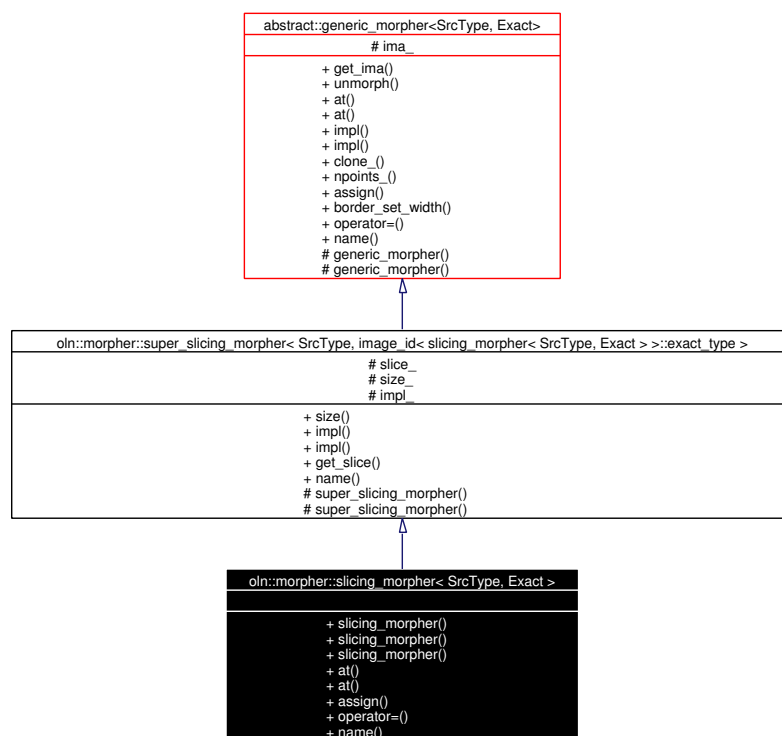
- histogram.hh

7.317 oln::morpher::slicing_morpher< SrcType, Exact > Struct Template Reference

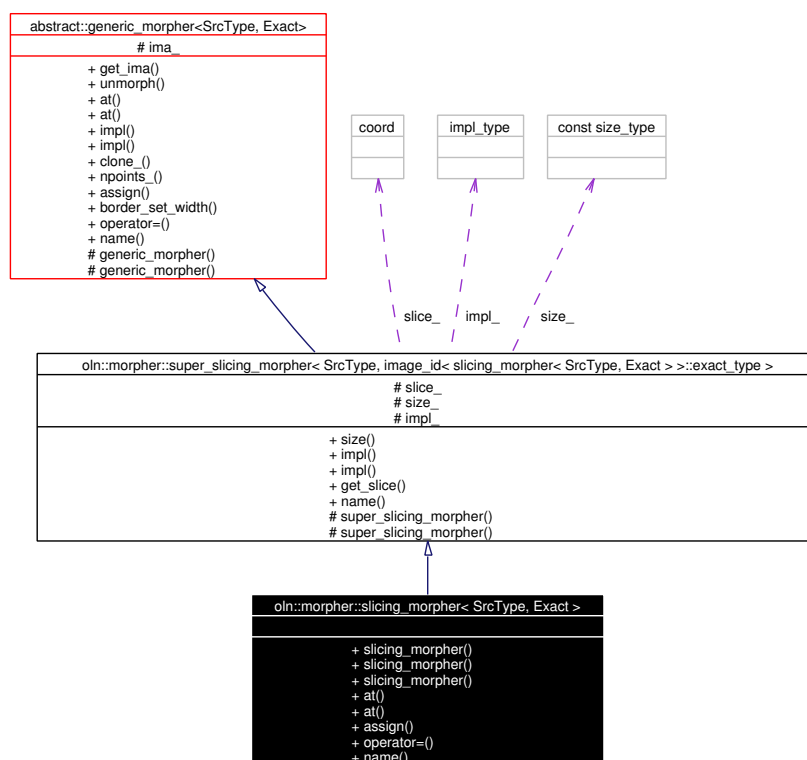
The default slicing morpher class.

```
#include <slicing_morpher.hh>
```

Inheritance diagram for oln::morpher::slicing_morpher< SrcType, Exact >:



Collaboration diagram for oln::morpher::slicing_morpher< SrcType, Exact >:



Public Types

- typedef [slicing_morpher](#)< SrcType, Exact > [self_type](#)
The self type.
- typedef image_id< [self_type](#) >::exact_type exact_type
The exact type of the morpher.
- typedef [super_slicing_morpher](#)< SrcType, exact_type > super_type
The upper class.
- typedef image_id< [exact_type](#) >::point_type point_type
The morpher point type.
- typedef image_id< [exact_type](#) >::img_type **img_type**
- typedef image_id< [exact_type](#) >::value_type value_type
The morpher value type.

Public Member Functions

- [slicing_morpher](#) (const SrcType &ima, [coord](#) slice)
Construct a slicing morpher.
– ima The image.

- *slice* The slice value.
- `slicing_morpher` (const `self_type` &r)
Construct a slicing morpher from another one.
- `slicing_morpher` ()
Empty constructor.
- `value_type` & `at` (const `point_type` &p)
Return the stored value at the point.
 - *p* The point.
- const `value_type` `at` (const `point_type` &p) const
Return a reference to the value stored at p in the current image.
- `self_type` & `assign` (`self_type` &rhs)
Perform a shallow copy from the decorated image of rhs to the current decorated image. The points will be shared by the two images.
- `self_type` & `operator=` (`self_type` &rhs)
This operator= assigns rhs to the current image.

Static Public Member Functions

- `std::string` `name` ()
Useful to debug.

7.317.1 Detailed Description

template<class SrcType, class Exact> struct oln::morpher::slicing_morpher< SrcType, Exact >

The default slicing morpher class.

Using this class, a slicing of picture is a picture.

See also:

[oln::morpher::abstract::generic_morpher](#)
[oln::morpher::slicing_morph](#)

Definition at line 235 of file `slicing_morpher.hh`.

7.317.2 Constructor & Destructor Documentation

7.317.2.1 `template<class SrcType, class Exact> oln::morpher::slicing_morpher< SrcType, Exact >::slicing_morpher () [inline]`

Empty constructor.

Needed by mlc_hierarchy::any_with_diamond.

Definition at line 266 of file slicing_morpher.hh.

```
266 {}
```

7.317.3 Member Function Documentation

7.317.3.1 `template<class SrcType, class Exact> const value_type oln::morpher::slicing_morpher< SrcType, Exact >::at (const point_type & p) const` `[inline]`

Return a reference to the value stored at *p* in the current image.

Warning:

This method must not be overloaded, unlike operator[].

See also:

image::operator[]()

Reimplemented from `oln::abstract::image_with_impl< Impl, Exact >`.

Definition at line 280 of file slicing_morpher.hh.

```
281 {
282     typename SrcType::point_type tmp_p(p, slice_);
283     return this->ima_[tmp_p];
284 }
```

7.317.3.2 `template<class SrcType, class Exact> value_type& oln::morpher::slicing_morpher< SrcType, Exact >::at (const point_type & p) [inline]`

Return the stored value at the point.

- *p* The point.

Returns:

The stored value.

Reimplemented from `oln::abstract::image_with_impl< Impl, Exact >`.

Definition at line 274 of file slicing_morpher.hh.

Referenced by `oln::morpher::slicing_morpher< SrcType, Exact >::assign()`.

```
275 {
276     typename SrcType::point_type tmp_p(p, slice_);
277     return const_cast<value_type &>(this->ima_[tmp_p]);
278 }
```

The documentation for this struct was generated from the following file:

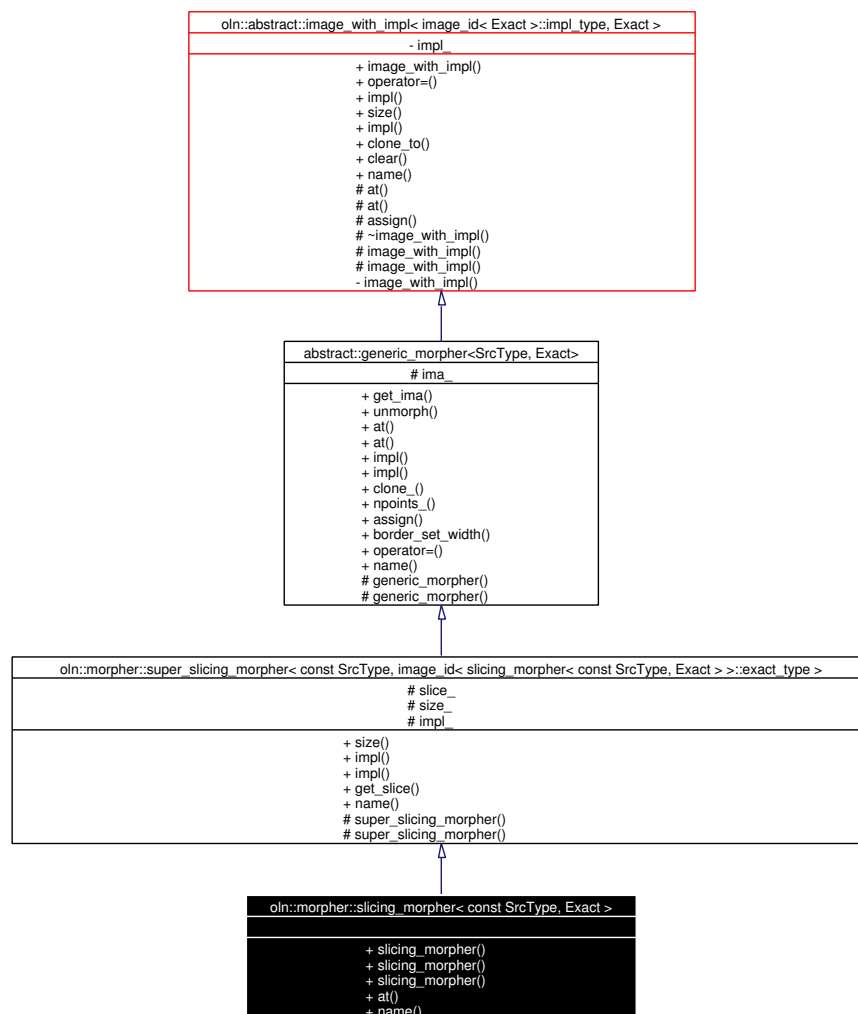
- slicing_morpher.hh

7.318 oln::morpher::slicing_morpher< const SrcType, Exact > Struct Template Reference

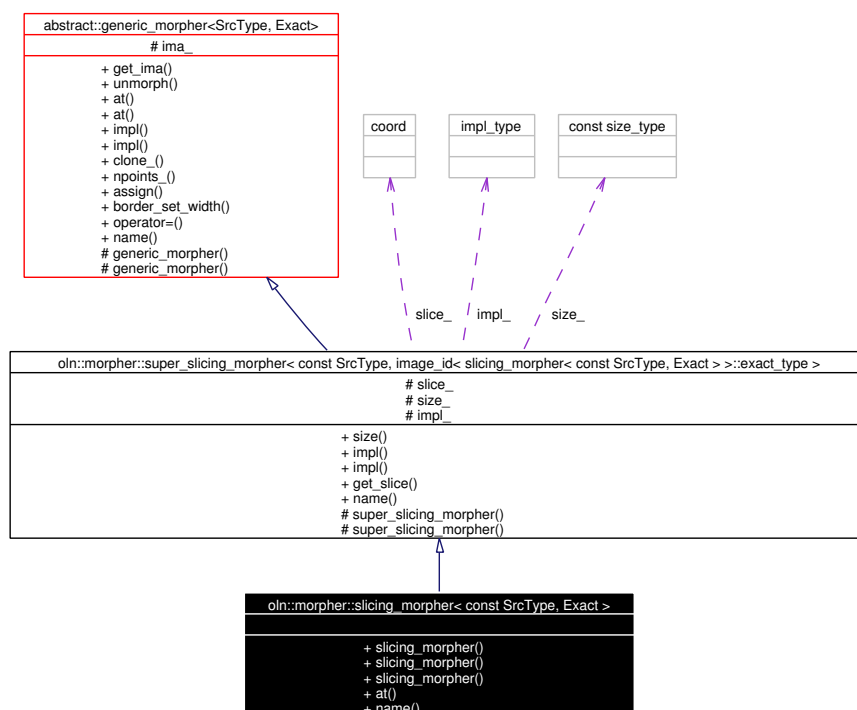
The specialized version for 'const' images.

```
#include <slicing_morpher.hh>
```

Inheritance diagram for oln::morpher::slicing_morpher< const SrcType, Exact >:



Collaboration diagram for oln::morpher::slicing_morpher< const SrcType, Exact >:



Public Types

- typedef [slicing_morpher](#)< const SrcType, Exact > [self_type](#)
The self type.
- typedef image_id< [self_type](#) >::exact_type exact_type
The exact type of the morpher.
- typedef [super_slicing_morpher](#)< const SrcType, exact_type > super_type
The upper class.
- typedef image_id< [exact_type](#) >::point_type point_type
The morpher point type.
- typedef image_id< [exact_type](#) >::value_type value_type
The morpher value type.

Public Member Functions

- [slicing_morpher](#) (const SrcType &ima, [coord](#) slice)
Construct a slicing morpher.
 - ima The image.
 - slice The slice value.
- [slicing_morpher](#) (const [self_type](#) &r)

Construct a slicing morpher from another one.

- [slicing_morpher](#) ()

Empty constructor.

- const [value_type](#) at (const [point_type](#) &p) const

Return the stored value at the point.

- *p* The point.

Static Public Member Functions

- std::string [name](#) ()

Useful to debug.

7.318.1 Detailed Description

```
template<class SrcType, class Exact> struct oln::morpher::slicing_morpher< const SrcType, Exact
>
```

The specialized version for 'const' images.

Definition at line 321 of file slicing_morpher.hh.

7.318.2 Member Function Documentation

7.318.2.1 `template<class SrcType, class Exact> const value_type oln::morpher::slicing_morpher< const SrcType, Exact >::at (const point_type & p) const`
[inline]

Return the stored value at the point.

- *p* The point.

Returns:

The stored value.

Reimplemented from [oln::abstract::image_with_impl< Impl, Exact >](#).

Definition at line 364 of file slicing_morpher.hh.

```
365     {
366         typename SrcType::point_type tmp_p(p, slice_);
367         return this->ima_[tmp_p];
368     }
```

7.318.2.2 `template<class SrcType, class Exact> oln::morpher::slicing_morpher< const SrcType, Exact >::slicing_morpher () [inline]`

Empty constructor.

Needed by `mlc_hierarchy::any_with_diamond`.

Definition at line 355 of file `slicing_morpher.hh`.

```
356      {}
```

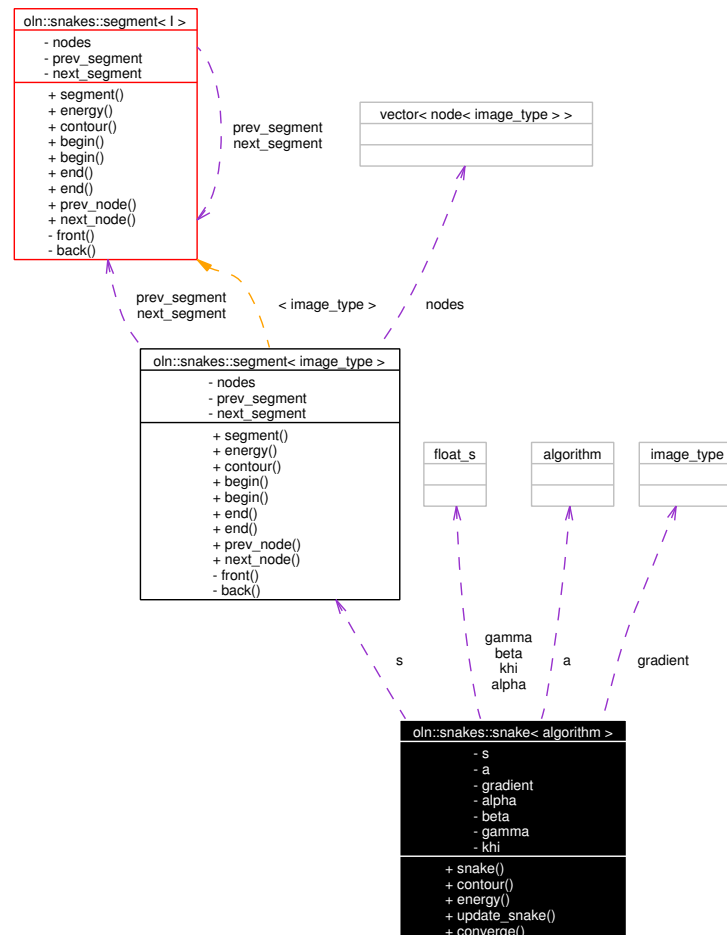
The documentation for this struct was generated from the following file:

- `slicing_morpher.hh`

7.319 oln::snakes::snake< algorithm > Class Template Reference

```
#include <snakes_base.hh>
```

Collaboration diagram for oln::snakes::snake< algorithm >:



Public Types

- typedef `algorithm::image_type` **image_type**
- typedef `image_type::point_type` **point_type**

Public Member Functions

- **snake** (const `image_type` &`image`, std::list< `point_type` > `initial_contour`, ntg::float_s `alpha`, ntg::float_s `beta`, ntg::float_s `gamma`, ntg::float_s `khi`)
- std::list< `point_type` > **contour** (void) const
Return the points of the snake.
- ntg::float_s **energy** (void) const
- int **update_snake** (void)

- void `converge` (void)

Friends

- int `algorithm::update_snake` (const typename `algorithm::image_type` &, `snake` &)
- void `algorithm::converge` (const typename `algorithm::image_type` &, `snake` &)
- `std::ostream & operator<<` (`std::ostream` &, const `snake` &)

7.319.1 Detailed Description

`template<class algorithm> class oln::snakes::snake< algorithm >`

Snake algorithm.

Todo

FIXME: Do not work due to the function `node::energy`.

FIXME: Add doc & test.

Definition at line 44 of file `snakes_base.hh`.

7.319.2 Member Function Documentation

7.319.2.1 `template<class algorithm> void oln::snakes::snake< algorithm >::converge (void)`
[inline]

Calling this method causes the snake to converge. It does so by delegating the method to the algorithm.

Definition at line 75 of file `snakes_base.hxx`.

```

76     {
77         a.converge(gradient, *this);
78     }
```

7.319.2.2 `template<class algorithm> ntg::float_s oln::snakes::snake< algorithm >::energy (void)`
`const`

Return the snake energy. This is not algorithm-dependant.

Todo

FIXME: Do not work due to the function `node::energy`

Definition at line 59 of file `snakes_base.hxx`.

References `oln::snakes::segment< image_type >::energy()`.

```

60     {
61         return s.energy(gradient);
62     }
```

7.319.2.3 `template<class algorithm> int oln::snakes::snake< algorithm >::update_snake (void)`
`[inline]`

Calling this method causes the snake to execute one step. If the method is not iterative, it should fail to compile.

Definition at line 67 of file snakes_base.hxx.

```
68     {  
69         return a.update_snake(gradient, *this);  
70     }
```

The documentation for this class was generated from the following files:

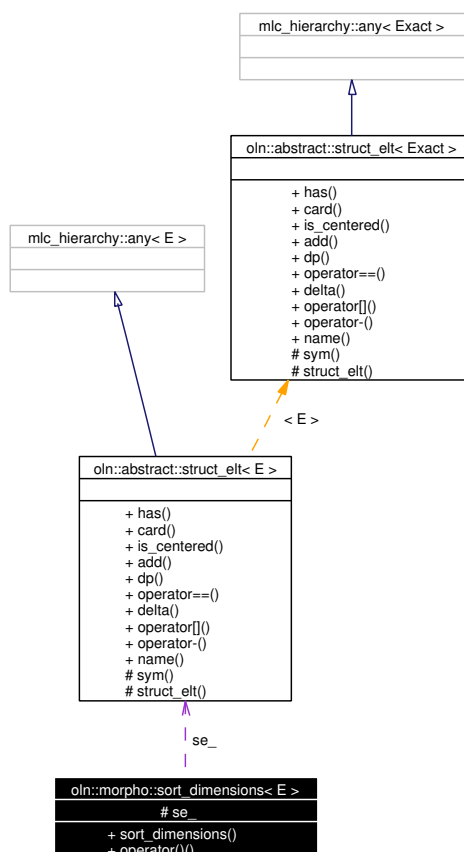
- snakes_base.hh
- snakes_base.hxx

7.320 oln::morpho::sort_dimensions< E > Struct Template Reference

functor to sort dimensions.

```
#include <fast_morpho.hxx>
```

Collaboration diagram for oln::morpho::sort_dimensions< E >:



Public Member Functions

- `sort_dimensions` (`abstract::struct_elt< E > se[mlc::exact< E >::ret::dim]`)

Constructor.

- `bool operator()` (unsigned a, unsigned b)

Parenthesis operator.

Protected Attributes

- `abstract::struct_elt< E > * se_`

Structural element.

7.320.1 Detailed Description

template<class E> struct oln::morpho::sort_dimensions< E >

functor to sort dimensions.

Definition at line 262 of file fast_morpho.hxx.

7.320.2 Member Function Documentation

7.320.2.1 template<class E> bool oln::morpho::sort_dimensions< E >::operator() (unsigned *a*, unsigned *b*) [inline]

Parenthesis operator.

Call it to use the functor.

Definition at line 275 of file fast_morpho.hxx.

References oln::abstract::struct_elt< E >::card(), and oln::morpho::sort_dimensions< E >::se_.

```
276      {  
277          return se_[a].card() > se_[b].card();  
278      }
```

The documentation for this struct was generated from the following file:

- fast_morpho.hxx

7.321 oln::morpho::internal::stat_< I, E, V > Struct Template Reference

Min and Max on a structuring element.

```
#include <stat.hh>
```

Static Public Member Functions

- V [max](#) (const I &input, const typename mlc::exact< I >::ret::point_type &p, const E &se)
Maximum of a structuring element.
- V [min](#) (const I &input, const typename mlc::exact< I >::ret::point_type &p, const E &se)
Minimum of a structuring element.

7.321.1 Detailed Description

```
template<class I, class E, class V = typename mlc::exact< I >::ret::value_type> struct
oln::morpho::internal::stat_< I, E, V >
```

Min and Max on a structuring element.

We need to use this inner definition in order to specialize max and min on binary images.

Parameters:

- I* Image exact type.
- E* Structuring element type.
- V* Associated value type.

Definition at line 51 of file morpho/stat.hh.

7.321.2 Member Function Documentation

```
7.321.2.1 template<class I, class E, class V = typename mlc::exact< I >::ret::value_type>
V oln::morpho::internal::stat_< I, E, V >::max (const I &input, const typename
mlc::exact< I >::ret::point_type &p, const E &se) [inline, static]
```

Maximum of a structuring element.

Look for the maximum in the structuring element se disposed on the image input, at the point p.

- input Input image.
- p Point of the image to move the structuring element on.
- se The structuring element to use.

Definition at line 64 of file morpho/stat.hh.

```

65     {
66         mlc::eq<I::dim, E::dim>::ensure();
67
68         oln_iter_type(E) dp(se);
69         dp = begin;
70         V val = input[p + dp];
71         for_all_remaining (dp)
72             if (val < input[p + dp])
73                 val = input[p + dp];
74         return val;
75     }

```

7.321.2.2 `template<class I, class E, class V = typename mlc::exact< I >::ret::value_type>
V oln::morpho::internal::stat_< I, E, V >::min (const I & input, const typename
mlc::exact< I >::ret::point_type & p, const E & se)` [`inline, static`]

Minimum of a structuring element.

Look for the minimum in the structuring element *se* disposed on the image *input*, at the point *p*.

- *input* Input image.
- *p* Point of the image to move the structuring element on.
- *se* The structuring element to use.

Definition at line 88 of file `morpho/stat.hh`.

```

89     {
90         mlc::eq<I::dim, E::dim>::ensure();
91         oln_iter_type(E) dp(se);
92         dp = begin;
93         V val = input[p + dp];
94         for_all_remaining (dp)
95             if (val > input[p + dp])
96                 val = input[p + dp];
97         return val;
98     }

```

The documentation for this struct was generated from the following file:

- `morpho/stat.hh`

7.322 oln::io::internal::stream_wrapper< W > Struct Template Reference

```
#include <stream_wrapper.hh>
```

Static Public Member Functions

- const std::string & **name** ()
- bool **knows_ext** (const std::string &)
- std::istream * **wrap_in** (std::string &)
- std::ostream * **wrap_out** (std::string &)
- void **find** (std::list< std::string > &, const std::string &)

7.322.1 Detailed Description

template<stream_id W> struct oln::io::internal::stream_wrapper< W >

Default version, if you instantiate one, all the operations on the stream will fail

See also:

[stream_wrapper<StreamFile>](#)
[stream_wrapper<StreamGz>](#)

Definition at line 66 of file stream_wrapper.hh.

The documentation for this struct was generated from the following file:

- stream_wrapper.hh

7.323 `oln::io::internal::stream_wrapper< StreamFile >` Struct Template Reference

```
#include <file.hh>
```

Static Public Member Functions

- `const std::string & name ()`
- `bool knows_ext (const std::string &)`
- `std::istream * wrap_in (std::string &name)`
Open a input stream on the file named name. Return 0 on failure.
- `std::ostream * wrap_out (std::string &name)`
Open a output stream on the file named name. Return 0 on failure.
- `void find (std::list< std::string > &names, const std::string &name)`
Insert in names all the files that have the same suffix as name in the name or the current directory.

7.323.1 Detailed Description

`template<> struct oln::io::internal::stream_wrapper< StreamFile >`

Specialized version for StreamFile.

Definition at line 49 of file file.hh.

The documentation for this struct was generated from the following file:

- file.hh

7.324 `oln::io::internal::stream_wrapper< StreamGz >` Struct Template Reference

```
#include <gz.hh>
```

Static Public Member Functions

- `const std::string & name ()`
Return "gz:".
- `bool knows_ext (const std::string &ext)`
Return true if ext == "gz" or ext == "z".
- `void adjust_name (std::string &name)`
Delete the file extension of name.
- `std::istream * wrap_in (std::string &name)`
Open a input stream on name then return it. On failure wrap_in returns 0.
- `std::ostream * wrap_out (std::string &name)`
Open a output stream on name then return it. On failure wrap_out returns 0.
- `void find (std::list< std::string > &, const std::string &)`

7.324.1 Detailed Description

`template<> struct oln::io::internal::stream_wrapper< StreamGz >`

Specialized version for StreamGz.

Definition at line 47 of file gz.hh.

The documentation for this struct was generated from the following file:

- gz.hh

7.325 oln::io::internal::stream_wrappers_find_files< W > Struct Template Reference

Find files compatible with the given root, extension free filename.

```
#include <stream_wrapper.hh>
```

Static Public Member Functions

- void [doit](#) (std::list< std::string > &names, const std::string &name)
Just a functor.

7.325.1 Detailed Description

template<stream_id W> struct oln::io::internal::stream_wrappers_find_files< W >

Find files compatible with the given root, extension free filename.

Definition at line 103 of file stream_wrapper.hh.

The documentation for this struct was generated from the following file:

- stream_wrapper.hh

7.326 oln::io::internal::stream_wrappers_find_files< StreamNone > Struct Template Reference

Do nothing because of the type of stream.

```
#include <stream_wrapper.hh>
```

Static Public Member Functions

- void **doit** (std::list< std::string > &, const std::string &)

7.326.1 Detailed Description

template<> struct oln::io::internal::stream_wrappers_find_files< StreamNone >

Do nothing because of the type of stream.

Definition at line 121 of file stream_wrapper.hh.

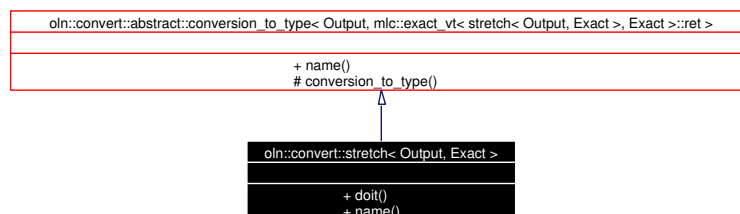
The documentation for this struct was generated from the following file:

- stream_wrapper.hh

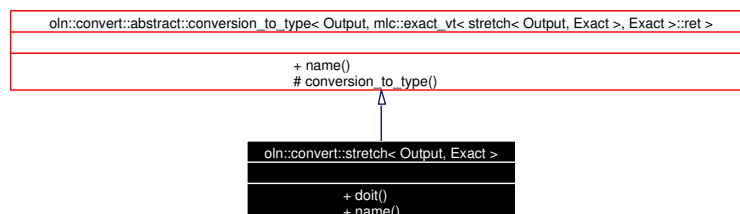
7.327 oln::convert::stretch< Output, Exact > Struct Template Reference

```
#include <stretch.hh>
```

Inheritance diagram for oln::convert::stretch< Output, Exact >:



Collaboration diagram for oln::convert::stretch< Output, Exact >:



Public Member Functions

- `template<class Input> Output doit (const Input &v) const`

Static Public Member Functions

- `std::string name ()`

7.327.1 Detailed Description

`template<class Output, class Exact = mlc::final> struct oln::convert::stretch< Output, Exact >`

Functor to stretch a value from a range Input to a range Output.

See also:

[stretch_balance](#)

```

#include <oln/basics2d.hh>
#include <oln/convert/stretch.hh>
#include <ntg/all.hh>
#include <iostream>

int main()
{

```

```
    oln::image2d<ntg::int_u8> lena = oln::load(IMG_IN "lena256.pgm");  
  
    oln::io::save(apply(oln::convert::stretch<ntg::int_u<3> >(), lena),  
                  IMG_OUT "oln_convert_stretch.pgm");  
}
```



=>



Definition at line 71 of file stretch.hh.

The documentation for this struct was generated from the following file:

- stretch.hh

7.328 ntg::strict Struct Reference

Strict checking, abort in there is a problem.

```
#include <behavior.hh>
```

Static Public Member Functions

- `std::string name ()`

7.328.1 Detailed Description

Strict checking, abort in there is a problem.

Definition at line 179 of file `integre/ntg/real/behavior.hh`.

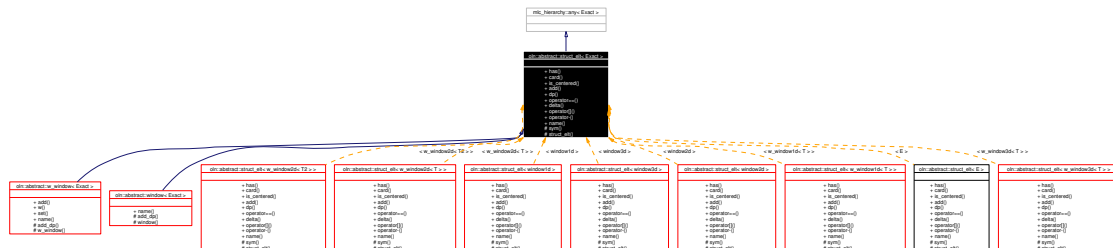
The documentation for this struct was generated from the following file:

- `integre/ntg/real/behavior.hh`

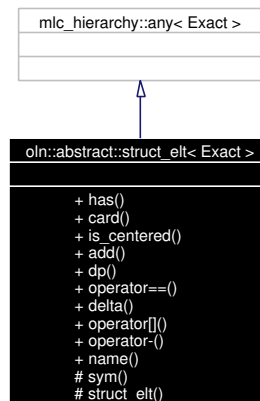
7.329 oln::abstract::struct_elt< Exact > Struct Template Reference

```
#include <struct_elt.hh>
```

Inheritance diagram for oln::abstract::struct_elt< Exact >:



Collaboration diagram for oln::abstract::struct_elt< Exact >:



Public Types

- typedef [struct_elt< Exact >](#) [self_type](#)
Set the exact self type of the class.
- typedef struct_elt_traits< Exact >::abstract_type abstract_type
Set the abstract type of himself.
- typedef struct_elt_traits< Exact >::point_type point_type
Set the point type of the image from which points come.
- typedef struct_elt_traits< Exact >::dpoint_type dpoint_type
Set the dpoint type.
- typedef Exact exact_type
- enum { **dim** = struct_elt_traits<Exact>::dim }

Public Member Functions

- bool **has** (const **abstract::dpoint**< **dpoint_type** > &dp) const
Test if the set of points contains this one.
 - dp a dpoint (deplacement point).
- unsigned **card** () const
Get the number of points.
- bool **is_centered** () const
Test if the structuring elements is centered.
- exact_type & **add** (const **abstract::dpoint**< **dpoint_type** > &dp)
Add a point to the structuring elements.
- **dpoint_type** **dp** (unsigned i) const
Get the nth structuring element.
 - i The nth.
- bool **operator==** (const **self_type** &win) const
Compare two sets of structuring elements.
 - win The structuring elements to compare.
- **coord** **delta** () const
Get the delta of the structuring elements.
- const **dpoint_type** **operator[]** (unsigned i) const
Get the nth structuring element.
 - i The nth.
- exact_type **operator-** () const
Set structuring elements to opposite.

Static Public Member Functions

- std::string **name** ()
Return the name of the type.

Protected Member Functions

- void **sym** ()
Set structuring elements to opposite.
- **struct_elt** ()
Do nothing, used only by sub-classes.

7.329.1 Detailed Description

template<class Exact> struct oln::abstract::struct_elt< Exact >

Structuring elements (set of dpoints).

This abstract class defines several virtual methods for its subclasses. Its goal is to deal with a set of 'move' points.

Definition at line 65 of file struct_elt.hh.

7.329.2 Member Function Documentation

7.329.2.1 template<class Exact> exact_type& oln::abstract::struct_elt< Exact >::add (const abstract::dpoint< dpoint_type > & dp) [inline]

Add a point to the structuring elements.

Add a new member to the structuring elements.

Warning:

Here for convenience (see morpho algorithms). Work with w_windows (associate a default weight set to 1).

Definition at line 130 of file struct_elt.hh.

```
131     {
132         return this->exact().add_dp(dp);
133     }
```

7.329.2.2 template<class Exact> unsigned oln::abstract::struct_elt< Exact >::card () const [inline]

Get the number of points.

Returns:

The number of points.

Definition at line 104 of file struct_elt.hh.

Referenced by oln::convol::slow::convolve(), oln::abstract::w_windownd< w_window3d< T > >::get_weight(), oln::inter(), and oln::uni().

```
105     {
106         return this->exact().card();
107     }
```

7.329.2.3 template<class Exact> coord oln::abstract::struct_elt< Exact >::delta () const [inline]

Get the delta of the structuring elements.

Returns:

Delta.

Delta is the biggest element of the structuring elements.

Definition at line 164 of file struct_elt.hh.

Referenced by `oln::convol::slow::convolve()`, `oln::morpho::dilation()`, `oln::morpho::erosion()`, and `oln::morpho::fast_morpho()`.

```
165     {
166         return this->exact().get_delta();
167     }
```

7.329.2.4 `template<class Exact> dpoint_type oln::abstract::struct_elt< Exact >::dp (unsigned i)` `const [inline]`

Get the nth structuring element.

- `i` The nth.

Returns:

The nth dpoint.

Definition at line 141 of file struct_elt.hh.

Referenced by `oln::convol::slow::convolve()`, `oln::inter()`, and `oln::uni()`.

```
142     {
143         return this->exact().at(i);
144     }
```

7.329.2.5 `template<class Exact> bool oln::abstract::struct_elt< Exact >::has (const abstract::dpoint< dpoint_type > & dp) const [inline]`

Test if the set of points contains this one.

- `dp` a dpoint (displacement point).

Returns:

True if the set of points contains this dpoint.

Definition at line 94 of file struct_elt.hh.

Referenced by `oln::inter()`.

```
95     {
96         return this->exact().has_(dp.exact());
97     }
```


7.329.2.6 `template<class Exact> bool oln::abstract::struct_elt< Exact >::is_centered (void) const`
`[inline]`

Test if the structuring elements is centered.

Returns:

True if it's centered.

Structuring elements are centered when they contains 0.

Definition at line 116 of file struct_elt.hh.

```
117     {
118         return this->exact().is_centered_();
119     }
```

7.329.2.7 `template<class Exact> exact_type oln::abstract::struct_elt< Exact >::operator- ()`
`const [inline]`

Set structuring elements to opposite.

Each point of structuring elements is assigned to its opposite.

Definition at line 186 of file struct_elt.hh.

```
187     {
188         exact_type win(this->exact());
189         win.sym();
190         return win;
191     }
```

7.329.2.8 `template<class Exact> bool oln::abstract::struct_elt< Exact >::operator==(const`
`self_type & win) const [inline]`

Compare two sets of structuring elements.

- win The structuring elements to compare.

Returns:

True if they are the same.

Definition at line 152 of file struct_elt.hh.

```
153     {
154         return this->exact().is_equal(win.exact());
155     }
```

7.329.2.9 `]`

`template<class Exact> const dpoint_type oln::abstract::struct_elt< Exact >::operator[] (unsigned i) const`
`[inline]`

Get the nth structuring element.

- i The nth.

Returns:

The nth dpoint.

Definition at line 175 of file struct_elt.hh.

```
176      {  
177      return this->exact().at(i);  
178      }
```

7.329.2.10 `template<class Exact> void oln::abstract::struct_elt< Exact >::sym ()` [inline, protected]

Set structuring elements to opposite.

Each point of structuring elements is assigned to its opposite.

Definition at line 201 of file struct_elt.hh.

```
202      {  
203      return this->exact().sym_();  
204      }
```

The documentation for this struct was generated from the following file:

- struct_elt.hh

7.330 `oln::struct_elt_traits< abstract::neighborhood< Exact > >` Struct Template Reference

Traits for [abstract::neighborhood](#).

```
#include <neighborhood.hh>
```

Public Types

- typedef [abstract::neighborhood< Exact >](#) `abstract_type`

7.330.1 Detailed Description

```
template<class Exact> struct oln::struct_elt_traits< abstract::neighborhood< Exact > >
```

Traits for [abstract::neighborhood](#).

Definition at line 46 of file `neighborhood.hh`.

The documentation for this struct was generated from the following file:

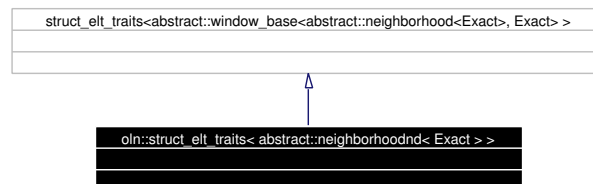
- `neighborhood.hh`

7.331 oln::struct_elt_traits< abstract::neighborhoodnd< Exact > > Struct Template Reference

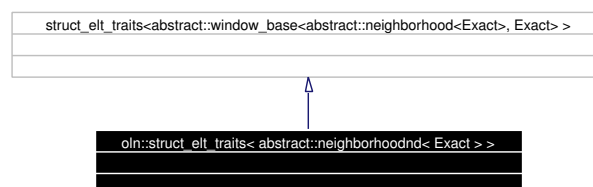
Traits for [abstract::neighborhoodnd](#).

```
#include <neighborhoodnd.hh>
```

Inheritance diagram for oln::struct_elt_traits< abstract::neighborhoodnd< Exact > >:



Collaboration diagram for oln::struct_elt_traits< abstract::neighborhoodnd< Exact > >:



7.331.1 Detailed Description

```
template<class Exact> struct oln::struct_elt_traits< abstract::neighborhoodnd< Exact > >
```

Traits for [abstract::neighborhoodnd](#).

Definition at line 44 of file neighborhoodnd.hh.

The documentation for this struct was generated from the following file:

- neighborhoodnd.hh

7.332 `oln::struct_elt_traits< abstract::struct_elt< Exact > >` Struct Template Reference

Traits for [abstract::struct_elt](#).

```
#include <struct_elt.hh>
```

Public Types

- typedef [abstract::struct_elt< Exact >](#) [abstract_type](#)
Defines the abstract type of the structuring element.

7.332.1 Detailed Description

```
template<class Exact> struct oln::struct_elt_traits< abstract::struct_elt< Exact > >
```

Traits for [abstract::struct_elt](#).

Definition at line 50 of file `struct_elt.hh`.

The documentation for this struct was generated from the following file:

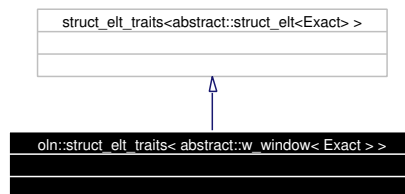
- `struct_elt.hh`

7.333 oln::struct_elt_traits< abstract::w_window< Exact > > Struct Template Reference

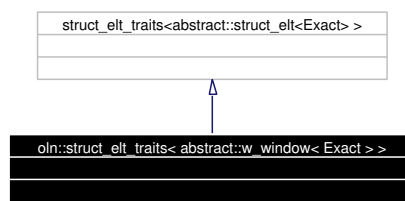
Traits for [abstract::w_window](#).

```
#include <w_window.hh>
```

Inheritance diagram for oln::struct_elt_traits< abstract::w_window< Exact > >:



Collaboration diagram for oln::struct_elt_traits< abstract::w_window< Exact > >:



7.333.1 Detailed Description

```
template<class Exact> struct oln::struct_elt_traits< abstract::w_window< Exact > >
```

Traits for [abstract::w_window](#).

Definition at line 46 of file w_window.hh.

The documentation for this struct was generated from the following file:

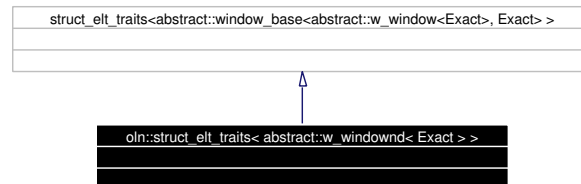
- w_window.hh

7.334 oln::struct_elt_traits< abstract::w_windownd< Exact > > Struct Template Reference

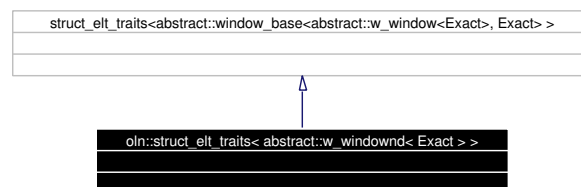
Traits for [abstract::w_windownd](#).

```
#include <w_windownd.hh>
```

Inheritance diagram for oln::struct_elt_traits< abstract::w_windownd< Exact > >:



Collaboration diagram for oln::struct_elt_traits< abstract::w_windownd< Exact > >:



7.334.1 Detailed Description

template<class Exact> struct oln::struct_elt_traits< abstract::w_windownd< Exact > >

Traits for [abstract::w_windownd](#).

Definition at line 44 of file w_windownd.hh.

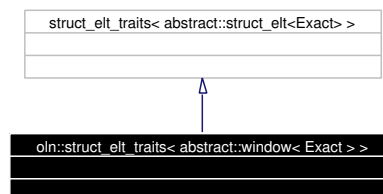
The documentation for this struct was generated from the following file:

- w_windownd.hh

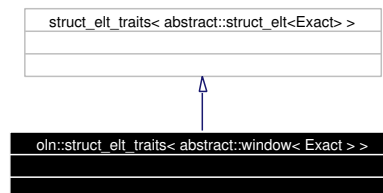
7.335 oln::struct_elt_traits< abstract::window< Exact > > Struct Template Reference

```
#include <window.hh>
```

Inheritance diagram for oln::struct_elt_traits< abstract::window< Exact > >:



Collaboration diagram for oln::struct_elt_traits< abstract::window< Exact > >:



7.335.1 Detailed Description

template<class Exact> struct oln::struct_elt_traits< abstract::window< Exact > >

Traits for [abstract::neighborhood](#)

Definition at line 46 of file window.hh.

The documentation for this struct was generated from the following file:

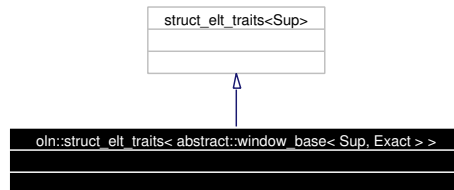
- window.hh

7.336 oln::struct_elt_traits< abstract::window_base< Sup, Exact > > Struct Template Reference

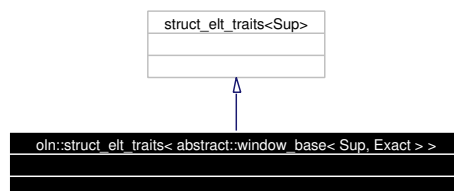
Traits for [abstract::window_base](#).

```
#include <window_base.hh>
```

Inheritance diagram for oln::struct_elt_traits< abstract::window_base< Sup, Exact > >:



Collaboration diagram for oln::struct_elt_traits< abstract::window_base< Sup, Exact > >:



7.336.1 Detailed Description

```
template<class Sup, class Exact> struct oln::struct_elt_traits< abstract::window_base< Sup, Exact
> >
```

Traits for [abstract::window_base](#).

Definition at line 53 of file window_base.hh.

The documentation for this struct was generated from the following file:

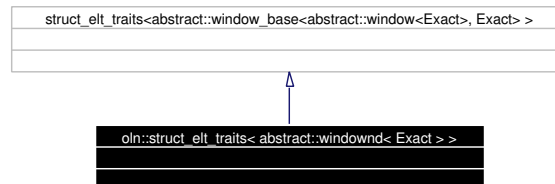
- window_base.hh

7.337 oln::struct_elt_traits< abstract::windownd< Exact > > Struct Template Reference

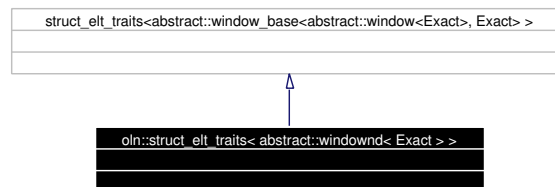
Traits for [abstract::windownd](#).

```
#include <windownd.hh>
```

Inheritance diagram for oln::struct_elt_traits< abstract::windownd< Exact > >:



Collaboration diagram for oln::struct_elt_traits< abstract::windownd< Exact > >:



7.337.1 Detailed Description

template<class Exact> struct oln::struct_elt_traits< abstract::windownd< Exact > >

Traits for [abstract::windownd](#).

Definition at line 44 of file windownd.hh.

The documentation for this struct was generated from the following file:

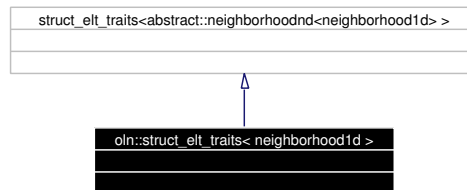
- windownd.hh

7.338 oln::struct_elt_traits< neighborhood1d > Struct Template Reference

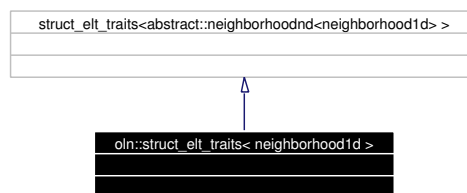
Traits for [neighborhood1d](#).

```
#include <neighborhood1d.hh>
```

Inheritance diagram for oln::struct_elt_traits< neighborhood1d >:



Collaboration diagram for oln::struct_elt_traits< neighborhood1d >:



Public Types

- typedef [point1d](#) [point_type](#)
Type of point.
- typedef [dpoint1d](#) [dpoint_type](#)
Type of dpoint (move).
- typedef winiter< [neighborhood1d](#) > [iter_type](#)
Type of iterator.
- typedef winneighb< [neighborhood1d](#) > [neighb_type](#)
Type of neighbor.
- typedef [window1d](#) [win_type](#)
Type of window.
- enum { **dim** = 1 }

7.338.1 Detailed Description

template<> struct oln::struct_elt_traits< neighborhood1d >

Traits for [neighborhood1d](#).

Definition at line 45 of file neighborhood1d.hh.

The documentation for this struct was generated from the following file:

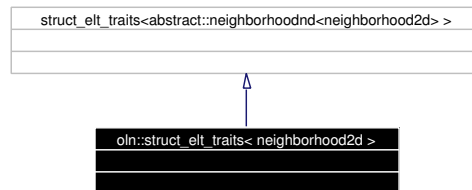
- neighborhood1d.hh

7.339 oln::struct_elt_traits< neighborhood2d > Struct Template Reference

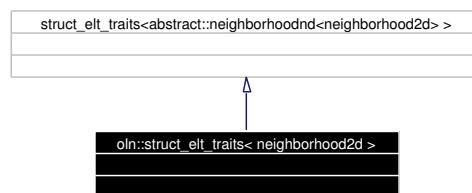
Traits for [neighborhood2d](#).

```
#include <neighborhood2d.hh>
```

Inheritance diagram for oln::struct_elt_traits< neighborhood2d >:



Collaboration diagram for oln::struct_elt_traits< neighborhood2d >:



Public Types

- typedef [point2d](#) [point_type](#)
Type of point.
- typedef [dpoint2d](#) [dpoint_type](#)
Type of dpoint (move).
- typedef winiter< [neighborhood2d](#) > [iter_type](#)
Type of iterator.
- typedef winneighb< [neighborhood2d](#) > [neighb_type](#)
Type of neighbor.
- typedef [window2d](#) [win_type](#)
Type of window.
- enum { **dim** = 2 }

7.339.1 Detailed Description

template<> struct oln::struct_elt_traits< neighborhood2d >

Traits for [neighborhood2d](#).

Definition at line 46 of file neighborhood2d.hh.

The documentation for this struct was generated from the following file:

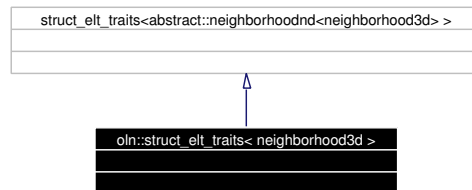
- neighborhood2d.hh

7.340 oln::struct_elt_traits< neighborhood3d > Struct Template Reference

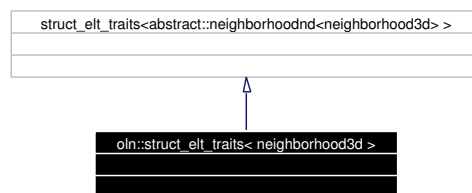
Traits for [neighborhood3d](#).

```
#include <neighborhood3d.hh>
```

Inheritance diagram for oln::struct_elt_traits< neighborhood3d >:



Collaboration diagram for oln::struct_elt_traits< neighborhood3d >:



Public Types

- typedef [point3d](#) [point_type](#)
Type of point.
- typedef [dpoint3d](#) [dpoint_type](#)
Type of dpoint (move).
- typedef winiter< [neighborhood3d](#) > [iter_type](#)
Type of iterator.
- typedef winneighb< [neighborhood3d](#) > [neighb_type](#)
Type of neighbor.
- typedef [window3d](#) [win_type](#)
Type of window.
- enum { **dim** = 3 }

7.340.1 Detailed Description

template<> struct oln::struct_elt_traits< neighborhood3d >

Traits for [neighborhood3d](#).

Definition at line 45 of file neighborhood3d.hh.

The documentation for this struct was generated from the following file:

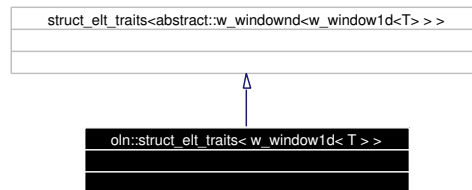
- neighborhood3d.hh

7.341 oln::struct_elt_traits< w_window1d< T > > Struct Template Reference

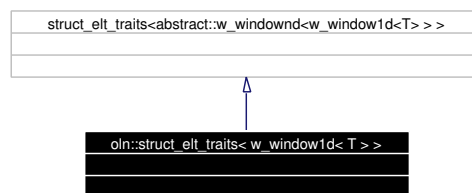
Traits for w_windownd1d.

```
#include <w_window1d.hh>
```

Inheritance diagram for oln::struct_elt_traits< w_window1d< T > >:



Collaboration diagram for oln::struct_elt_traits< w_window1d< T > >:



Public Types

- typedef T [weight_type](#)
Type of weight.
- typedef [point1d](#) [point_type](#)
Type of point.
- typedef [dpoint1d](#) [dpoint_type](#)
Type of dpoint.
- typedef winiter< [w_window1d](#)< T > > [iter_type](#)
Type of iterator.
- typedef winneighb< [w_window1d](#)< T > > [neighb_type](#)
Type of neighbor.
- enum { **dim** = 1 }

7.341.1 Detailed Description

template<class T> struct oln::struct_elt_traits< w_window1d< T > >

Traits for w_windownd1d.

Definition at line 49 of file w_window1d.hh.

The documentation for this struct was generated from the following file:

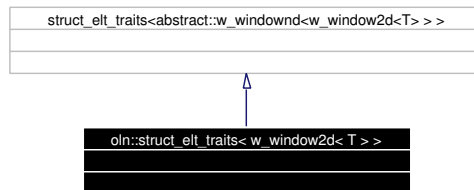
- w_window1d.hh

7.342 oln::struct_elt_traits< w_window2d< T > > Struct Template Reference

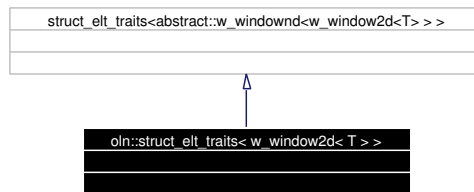
Traits for w_windownd2d.

```
#include <w_window2d.hh>
```

Inheritance diagram for oln::struct_elt_traits< w_window2d< T > >:



Collaboration diagram for oln::struct_elt_traits< w_window2d< T > >:



Public Types

- typedef T [weight_type](#)
Type of weight.
- typedef [point2d](#) [point_type](#)
Type of point.
- typedef [dpoint2d](#) [dpoint_type](#)
Type of dpoint.
- typedef winiter< [w_window2d](#)< T > > [iter_type](#)
Type of iterator.
- typedef winneighb< [w_window2d](#)< T > > [neighb_type](#)
Type of neighbor.
- enum { **dim** = 2 }

7.342.1 Detailed Description

template<class T> struct oln::struct_elt_traits< w_window2d< T > >

Traits for w_windownd2d.

Definition at line 50 of file w_window2d.hh.

The documentation for this struct was generated from the following file:

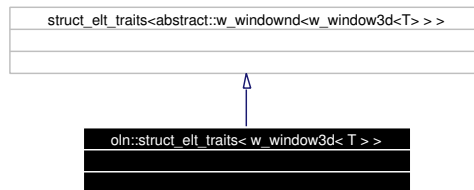
- w_window2d.hh

7.343 oln::struct_elt_traits< w_window3d< T > > Struct Template Reference

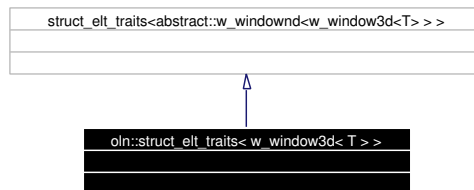
Traits for w_windownd3d.

```
#include <w_window3d.hh>
```

Inheritance diagram for oln::struct_elt_traits< w_window3d< T > >:



Collaboration diagram for oln::struct_elt_traits< w_window3d< T > >:



Public Types

- typedef T [weight_type](#)
Type of weight.
- typedef [point3d](#) [point_type](#)
Type of point.
- typedef [dpoint3d](#) [dpoint_type](#)
Type of dpoint.
- typedef winiter< [w_window3d](#)< T > > [iter_type](#)
Type of iterator.
- typedef winneighb< [w_window3d](#)< T > > [neighb_type](#)
Type of neighbor.
- enum { **dim** = 3 }

7.343.1 Detailed Description

template<class T> struct oln::struct_elt_traits< w_window3d< T > >

Traits for w_windownd3d.

Definition at line 49 of file w_window3d.hh.

The documentation for this struct was generated from the following file:

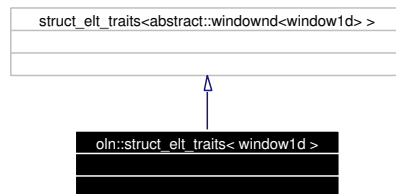
- w_window3d.hh

7.344 oln::struct_elt_traits< window1d > Struct Template Reference

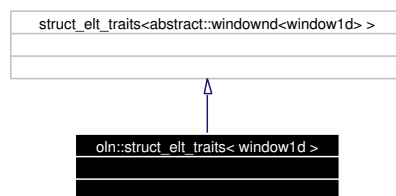
Traits for [window1d](#).

```
#include <window1d.hh>
```

Inheritance diagram for oln::struct_elt_traits< window1d >:



Collaboration diagram for oln::struct_elt_traits< window1d >:



Public Types

- typedef [point1d](#) [point_type](#)
Type of point.
- typedef [dpoint1d](#) [dpoint_type](#)
Type of dpoint.
- typedef winiter< [window1d](#) > [iter_type](#)
Type of iterator.
- typedef winneighb< [window1d](#) > [neighb_type](#)
Type of neighbor.
- enum { **dim** = 1 }

7.344.1 Detailed Description

template<> struct oln::struct_elt_traits< window1d >

Traits for [window1d](#).

Definition at line 46 of file window1d.hh.

The documentation for this struct was generated from the following file:

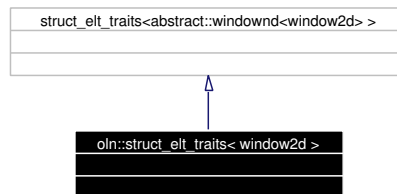
- window1d.hh

7.345 oln::struct_elt_traits< window2d > Struct Template Reference

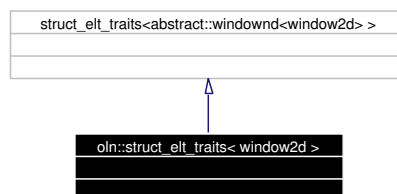
Traits for [window2d](#).

```
#include <window2d.hh>
```

Inheritance diagram for oln::struct_elt_traits< window2d >:



Collaboration diagram for oln::struct_elt_traits< window2d >:



Public Types

- typedef [point2d](#) [point_type](#)
Type of point.
- typedef [dpoint2d](#) [dpoint_type](#)
Type of dpoint.
- typedef winiter< [window2d](#) > [iter_type](#)
Type of iterator.
- typedef winneighb< [window2d](#) > [neighb_type](#)
Type of neighbor.
- enum { **dim** = 2 }

7.345.1 Detailed Description

template<> struct oln::struct_elt_traits< [window2d](#) >

Traits for [window2d](#).

Definition at line 47 of file [window2d.hh](#).

The documentation for this struct was generated from the following file:

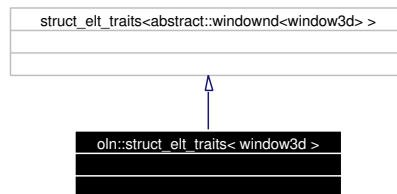
- window2d.hh

7.346 oln::struct_elt_traits< window3d > Struct Template Reference

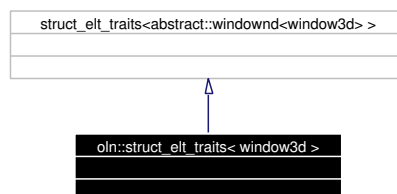
Traits for [window3d](#).

```
#include <window3d.hh>
```

Inheritance diagram for oln::struct_elt_traits< window3d >:



Collaboration diagram for oln::struct_elt_traits< window3d >:



Public Types

- typedef [point3d](#) [point_type](#)
Type of point.
- typedef [dpoint3d](#) [dpoint_type](#)
Type of dpoint.
- typedef winiter< [window3d](#) > [iter_type](#)
Type of iterator.
- typedef winneighb< [window3d](#) > [neighb_type](#)
Type of neighbor.
- enum { **dim** = 3 }

7.346.1 Detailed Description

```
template<> struct oln::struct_elt_traits< window3d >
```

Traits for [window3d](#).

Definition at line 47 of file window3d.hh.

The documentation for this struct was generated from the following file:

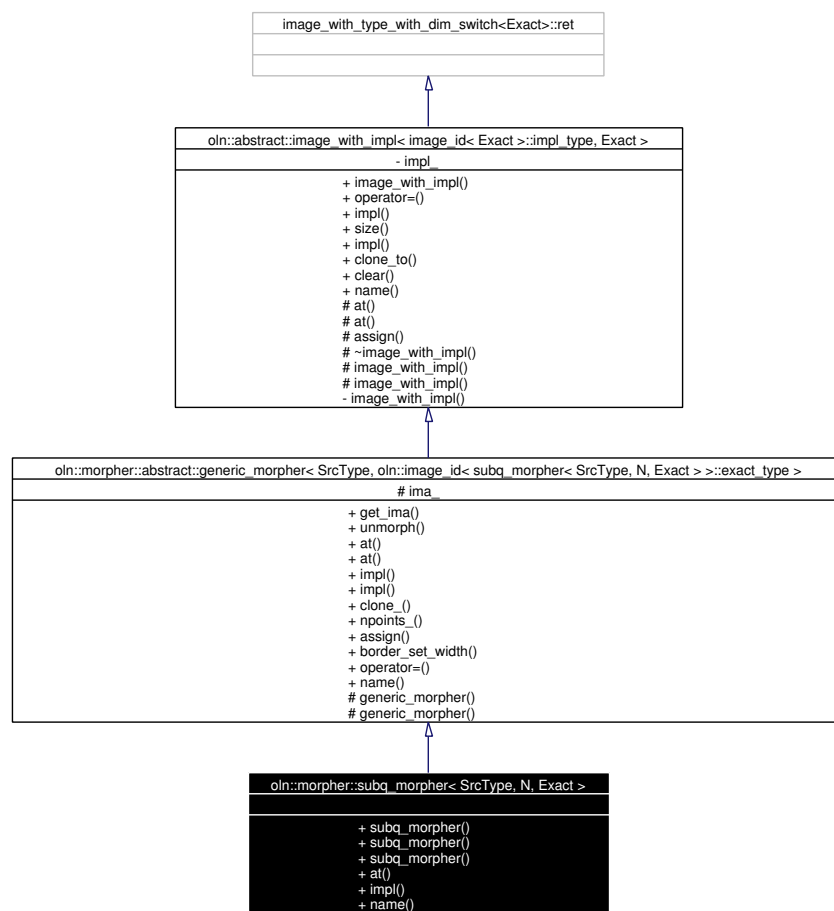
- `window3d.hh`

7.347 oln::morpher::subq_morpher< SrcType, N, Exact > Struct Template Reference

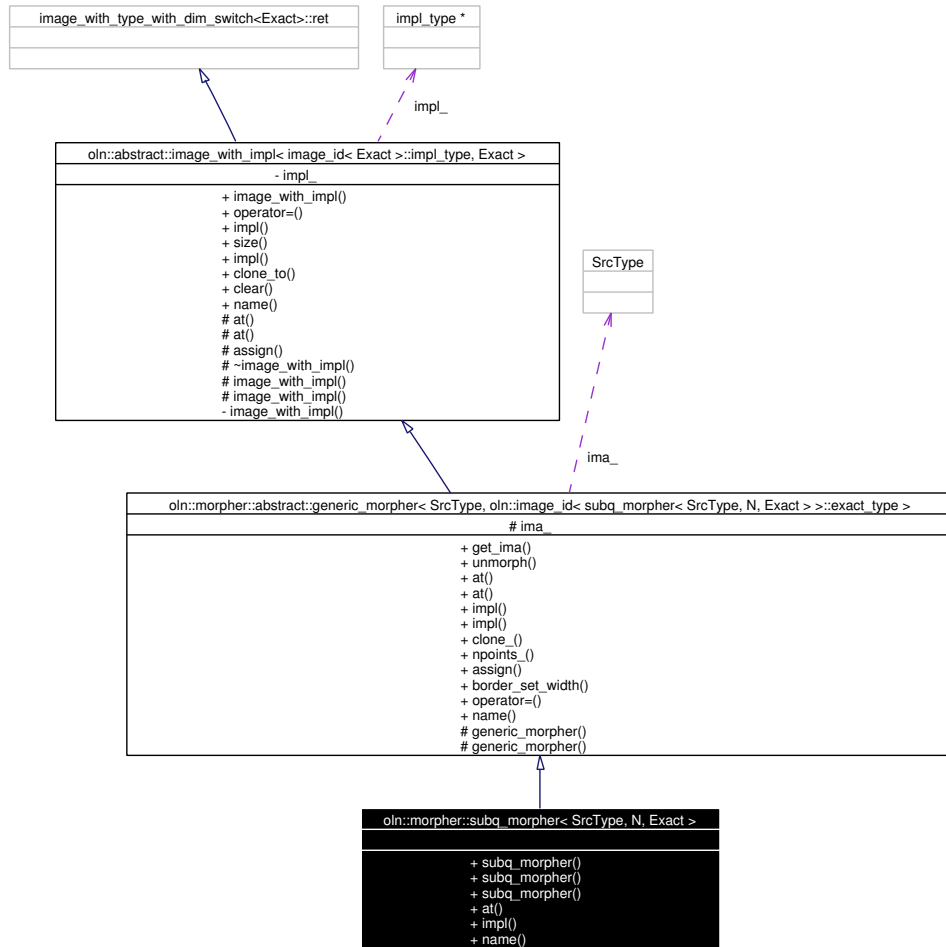
Sub quantify an image.

```
#include <subq_morpher.hh>
```

Inheritance diagram for oln::morpher::subq_morpher< SrcType, N, Exact >:



Collaboration diagram for oln::morpher::subq_morpher< SrcType, N, Exact >:



Public Types

- `typedef oln::image_id< subq_morpher< SrcType, N, Exact > >::exact_type exact_type`
The exact type of this. This class can be derived.
- `typedef abstract::generic_morpher< SrcType, exact_type > super_type`
The upper class.
- `typedef image_id< exact_type >::value_type value_type`
The value point of the resulting image.
- `typedef image_id< exact_type >::point_type point_type`
The morpher point type.
- `typedef image_id< exact_type >::impl_type impl_type`
The morpher underlying implementation.
- `enum { nbcomps = color_mute<typename mlc::exact< SrcType >::ret::value_type, nbcomps = color_mute<typename mlc::exact< SrcType >::ret::value_type }`

Public Member Functions

- [subq_morpher](#) (const SrcType &ima)
Construct the morpher with an image.
- [subq_morpher](#) (const [subq_morpher](#)< SrcType, N > &r)
Construct the morpher with another morpher.
- [subq_morpher](#) ()
- const [value_type](#) at (const [point_type](#) &p) const
Return the value stored at p in the resulting image.
- const [impl_type](#) * [impl](#) () const
Return the implementation.

Static Public Member Functions

- std::string [name](#) ()

7.347.1 Detailed Description

template<class SrcType, unsigned N, class Exact> struct oln::morpher::subq_morpher< SrcType, N, Exact >

Sub quantify an image.

By using this class, an image can be viewed as an image with a lower color depth. [subq_morpher](#)<image2d<rgb_8>, 5> is the same as image2d<rgb_5>.

Parameters:

- SrcType* The input type decorated.
- N* The new numbers of bits by component.
- Exact* The exact type of the morpher.

Definition at line 133 of file subq_morpher.hh.

7.347.2 Constructor & Destructor Documentation

7.347.2.1 **template<class SrcType, unsigned N, class Exact> oln::morpher::subq_morpher< SrcType, N, Exact >::subq_morpher () [inline]**

Empty constructor.

Needed by mlc_hierarchy::any_with_diamond.

Definition at line 166 of file subq_morpher.hh.

```
166 {}
```

The documentation for this struct was generated from the following file:

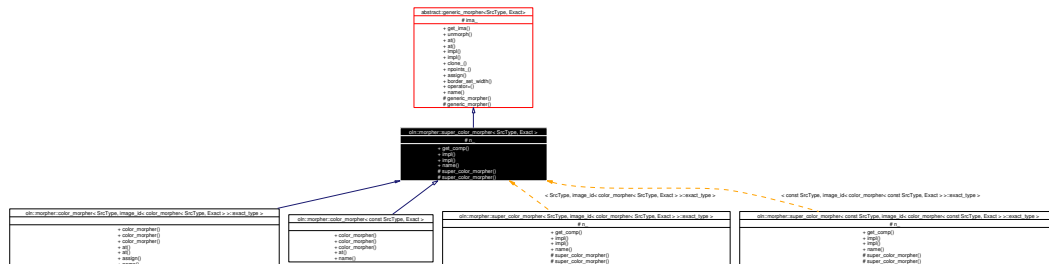
- subq_morpher.hh

7.348 oln::morpher::super_color_morpher< SrcType, Exact > Class Template Reference

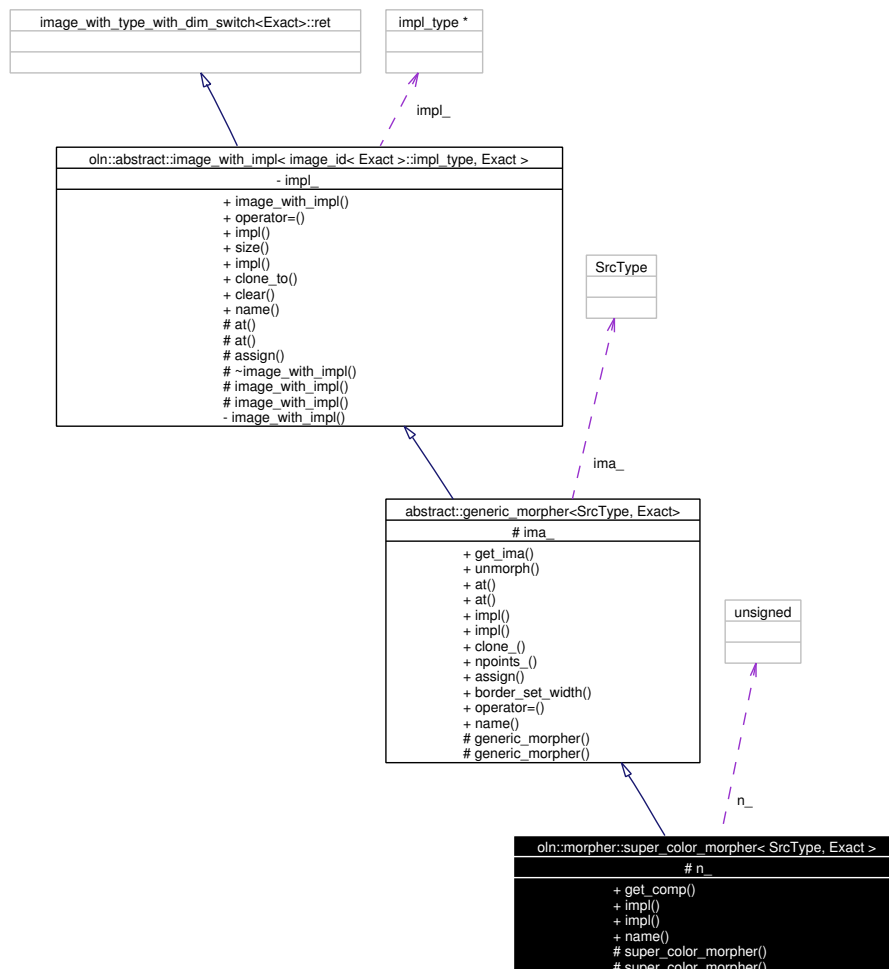
Abstract [color_morpher](#) class used for code factorization.

```
#include <color_morpher.hh>
```

Inheritance diagram for oln::morpher::super_color_morpher< SrcType, Exact >:



Collaboration diagram for oln::morpher::super_color_morpher< SrcType, Exact >:



Public Types

- typedef [abstract::generic_morpher](#)< SrcType, Exact > [super_type](#)
The upper class.
- typedef image_id< Exact >::[impl_type](#) [impl_type](#)
The morpher underlying implementation.

Public Member Functions

- unsigned [get_comp](#) () const
Return the number of the image component to retrieve.
- [impl_type](#) * [impl](#) ()
Return the image implementation.
- const [impl_type](#) * [impl](#) () const
Return the image implementation.

Static Public Member Functions

- std::string [name](#) ()

Protected Member Functions

- [super_color_morpher](#) (const SrcType &ima, unsigned n)
- [super_color_morpher](#) ()

Protected Attributes

- unsigned [n_](#)

7.348.1 Detailed Description

template<class SrcType, class Exact> class oln::morpher::super_color_morpher< SrcType, Exact >

Abstract [color_morpher](#) class used for code factorization.

Parameters:

SrcType Input type decorated.

Exact Exact type

Definition at line 97 of file color_morpher.hh.

7.348.2 Constructor & Destructor Documentation

7.348.2.1 `template<class SrcType, class Exact> oln::morpher::super_color_morpher< SrcType, Exact >::super_color_morpher (const SrcType & ima, unsigned n)` [`inline`, `protected`]

Default constructor.

ima will be the decorated image. One can not use this constructor to instantiate this class since it is protected.

Definition at line 116 of file `color_morpher.hh`.

```

116                                     : super_type(ima)
117     {
118         assert(n == ntg::rgb_R || n == ntg::rgb_G || n == ntg::rgb_B);
119         n_ = n;
120     }
```

7.348.2.2 `template<class SrcType, class Exact> oln::morpher::super_color_morpher< SrcType, Exact >::super_color_morpher ()` [`inline`, `protected`]

Empty constructor.

Needed by `mlc_hierarchy::any_with_diamond`.

Definition at line 126 of file `color_morpher.hh`.

```

127     {}
```

7.348.3 Member Data Documentation

7.348.3.1 `template<class SrcType, class Exact> unsigned oln::morpher::super_color_morpher< SrcType, Exact >::n_` [`protected`]

The component to return.

If *n* is equal to 0, the red component is returned. If *n* is equal to 1, the green component is returned. If *n* is equal to 2, the blue component is returned.

Definition at line 108 of file `color_morpher.hh`.

The documentation for this class was generated from the following file:

- `color_morpher.hh`

Public Types

- typedef [super_piece_morpher](#)< SrcType, Exact > [self_type](#)
The self type.
- typedef image_id< [self_type](#) >::[exact_type](#) [exact_type](#)
The exact type of the morpher.
- typedef [abstract::generic_morpher](#)< SrcType, Exact > [super_type](#)
The upper class.
- typedef image_id< [exact_type](#) >::[dpoint_type](#) [dpoint_type](#)
The morpher dpoint type.
- typedef image_id< [exact_type](#) >::[size_type](#) [size_type](#)
The morpher size type.

Public Member Functions

- const [size_type](#) [size](#) () const
Return the size (different from the original picture).
- const [dpoint_type](#) [ref_point](#) () const
Return the reference point.

Static Public Member Functions

- std::string [name](#) ()
Useful to debug.

Protected Member Functions

- [super_piece_morpher](#) (const SrcType &ima, const [dpoint_type](#) &p, const [size_type](#) &s)
Default constructor.
 - *ima will be the image.*
 - *p The reference point.*
 - *s The size of the piece of image.*
- [super_piece_morpher](#) ()
Empty constructor.

Protected Attributes

- const [size_type](#) [size_](#)
The size of the piece of picture.
- const [dpoint_type](#) [p_](#)
The reference point of the piece of picture.

7.349.1 Detailed Description

template<class SrcType, class Exact> class oln::morpher::super_piece_morpher< SrcType, Exact >

Abstract piece morpher class used for code factorization.

Definition at line 91 of file piece_morpher.hh.

7.349.2 Constructor & Destructor Documentation

7.349.2.1 **template<class SrcType, class Exact> [oln::morpher::super_piece_morpher](#)< SrcType, Exact >::[super_piece_morpher](#) (const SrcType & *ima*, const [dpoint_type](#) & *p*, const [size_type](#) & *s*)** [`inline`, `protected`]

Default constructor.

- *ima* will be the image.
- *p* The reference point.
- *s* The size of the piece of image.

One can not use this constructor to instantiate this class since it is protected.

Definition at line 115 of file piece_morpher.hh.

```
117         : super_type(ima), size_(s), p_(p)
118         {}
```

7.349.2.2 **template<class SrcType, class Exact> [oln::morpher::super_piece_morpher](#)< SrcType, Exact >::[super_piece_morpher](#) ()** [`inline`, `protected`]

Empty constructor.

Needed by `mlc_hierarchy::any_with_diamond`.

Definition at line 129 of file piece_morpher.hh.

```
130     {}
```

The documentation for this class was generated from the following file:

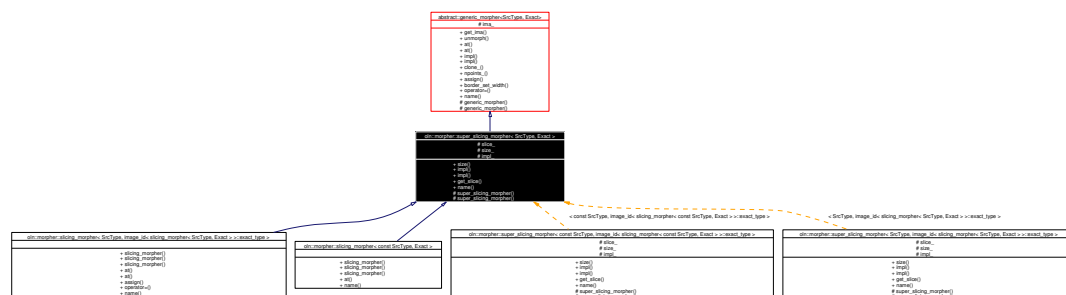
- piece_morpher.hh

7.350 oln::morpher::super_slicing_morpher< SrcType, Exact > Class Template Reference

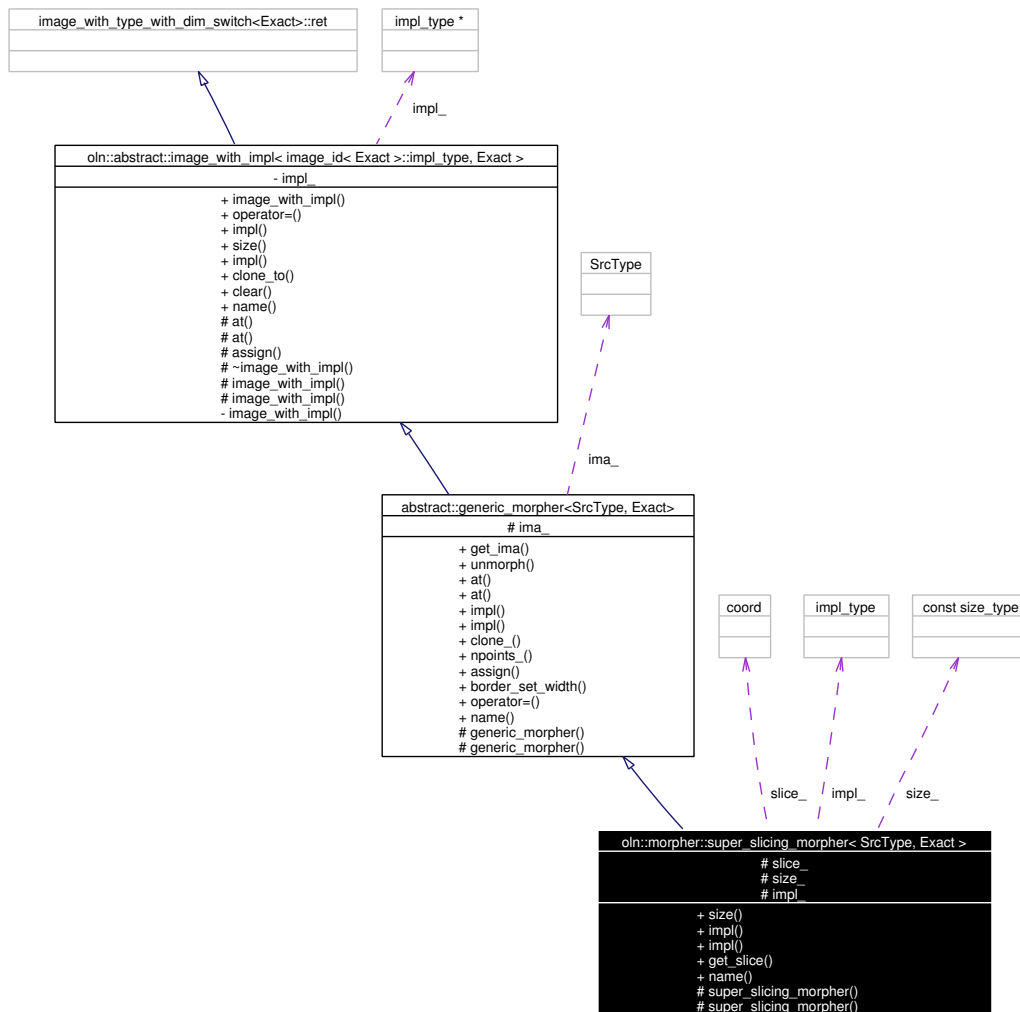
Abstract slicing morpher class used for code factorization.

```
#include <slicing_morpher.hh>
```

Inheritance diagram for oln::morpher::super_slicing_morpher< SrcType, Exact >:



Collaboration diagram for oln::morpher::super_slicing_morpher< SrcType, Exact >:



Public Types

- typedef `super_slicing_morpher< SrcType, Exact >` `self_type`
The self type.
- typedef `image_id< self_type >::exact_type` `exact_type`
The exact type of the morpher.
- typedef `abstract::generic_morpher< SrcType, Exact >` `super_type`
The upper class.
- typedef `image_id< exact_type >::size_type` `size_type`
The morpher size type.
- typedef `image_id< exact_type >::impl_type` `impl_type`
The morpher underlying implementation.

Public Member Functions

- `const size_type & size () const`
Override the size method.
- `const impl_type * impl () const`
Override the impl method.
- `impl_type * impl ()`
Override the impl method.
- `coord get_slice () const`
Return the last coordinate' value.

Static Public Member Functions

- `std::string name ()`
Useful to debug.

Protected Member Functions

- `super_slicing_morpher (const SrcType &ima, const coord slice)`
Default constructor.
 - *ima will be the image.*
 - *slice The last coordinate.*
- `super_slicing_morpher ()`
Empty constructor.

Protected Attributes

- `coord slice_`
The last coordinate.
- `const size_type size_`
The size of the N-1 dimension image.
- `impl_type impl_`
Implementation type of the image. It can be an array, a std::vector, ...

7.350.1 Detailed Description

`template<class SrcType, class Exact> class oln::morpher::super_slicing_morpher< SrcType, Exact >`

Abstract slicing morpher class used for code factorization.

Definition at line 149 of file slicing_morpher.hh.

7.350.2 Constructor & Destructor Documentation

7.350.2.1 `template<class SrcType, class Exact> oln::morpher::super_slicing_morpher< SrcType, Exact >::super_slicing_morpher (const SrcType & ima, const coord slice)` [`inline`, `protected`]

Default constructor.

- *ima* will be the image.
- *slice* The last coordinate.

One can not use this constructor to instantiate this class since it is protected.

Definition at line 207 of file slicing_morpher.hh.

```
208         : super_type(ima), slice_(slice), size_(image_size_dec(ima_.size())), impl_(size_)
209         {}
```

7.350.2.2 `template<class SrcType, class Exact> oln::morpher::super_slicing_morpher< SrcType, Exact >::super_slicing_morpher ()` [`inline`, `protected`]

Empty constructor.

Needed by `mlc_hierarchy::any_with_diamond`.

Todo

create empty constructors for `impl_`, ...

Definition at line 217 of file slicing_morpher.hh.

```
218     {}
```

The documentation for this class was generated from the following file:

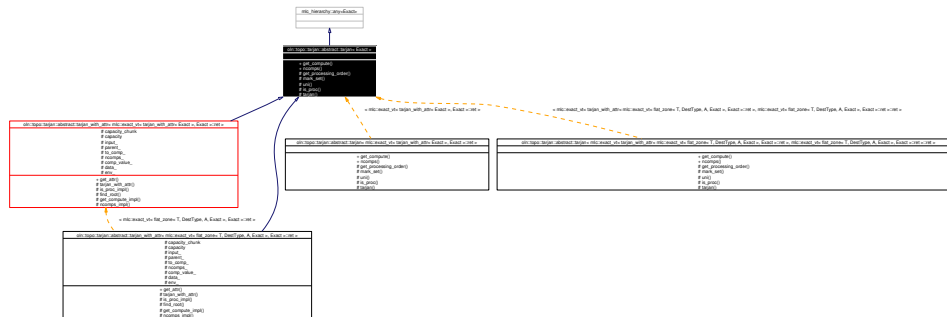
- slicing_morpher.hh

7.351 oln::topo::tarjan::abstract::tarjan< Exact > Struct Template Reference

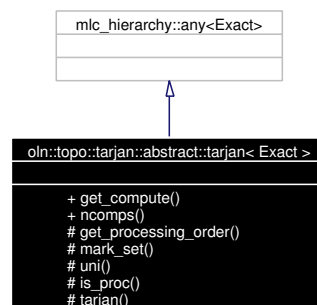
Top of tarjan hierarchy.

```
#include <tarjan.hh>
```

Inheritance diagram for oln::topo::tarjan::abstract::tarjan< Exact >:



Collaboration diagram for oln::topo::tarjan::abstract::tarjan< Exact >:



Public Types

- `typedef oln::topo::tarjan::tarjan_traits< Exact >::input_type input_type`
Type of input image.
- `typedef oln::topo::tarjan::tarjan_traits< Exact >::output_type image_out_type`
Type of output image.
- `typedef mlc::exact< image_out_type >::ret::value_type comp_type`
Type of components.
- `typedef mlc::exact< input_type >::ret::point_type point_type`
Point type of image to process.

Public Member Functions

- `template<class N> image_out_type get_compute (const oln::abstract::neighborhood< N > &Ng)`
Run the algorithm.
- `comp_type ncomps () const`
Give the number of component found.

Protected Member Functions

- `std::vector< point_type > get_processing_order ()`
Abstract method to get the processing order.
- `void mark_set (const point_type &x)`
Mark a point as a new component.
- `void uni (const point_type &n, const point_type &p)`
Perform an union between two components.
- `bool is_proc (const point_type &p) const`
tell if a point has already been processed.
- `tarjan ()`
Make the class abstract.

7.351.1 Detailed Description

`template<typename Exact> struct oln::topo::tarjan::abstract::tarjan< Exact >`

Top of tarjan hierarchy.

Parameters:

- I* Type of image to process.
- D* Type of data of the wanted image.
- Exact* Exact type of the class.

Definition at line 63 of file tarjan.hh.

7.351.2 Member Function Documentation

7.351.2.1 `template<typename Exact> template<class N> image_out_type oln::topo::tarjan::abstract::tarjan< Exact >::get_compute (const oln::abstract::neighborhood< N > &Ng) [inline]`

Run the algorithm.

Warning:

Implement `get_compute_impl` to be able to use this method.

Definition at line 79 of file `tarjan.hh`.

```
80         {
81             mlc_dispatch(get_compute)(Ng);
82         }
```

7.351.2.2 `template<typename Exact> std::vector<point_type>`
`oln::topo::tarjan::abstract::tarjan< Exact >::get_processing_order ()`
`[inline, protected]`

Abstract method to get the processing order.

Warning:

Implement `get_processing_order_impl` to be able to use this method.

Definition at line 103 of file `tarjan.hh`.

Referenced by `oln::topo::tarjan::abstract::tarjan_with_attr< mlc::exact_vt< flat_zone< T, DestType, A, Exact >, Exact >::ret >::get_compute_impl()`.

```
104         {
105             mlc_dispatch(get_processing_order)();
106         }
```

7.351.2.3 `template<typename Exact> bool oln::topo::tarjan::abstract::tarjan< Exact >::is_proc`
`(const point_type & p) const [inline, protected]`

tell if a point has already been processed.

Warning:

Implement `is_proc_impl` to be able to use this method.

Definition at line 139 of file `tarjan.hh`.

Referenced by `oln::topo::tarjan::abstract::tarjan_with_attr< mlc::exact_vt< flat_zone< T, DestType, A, Exact >, Exact >::ret >::get_compute_impl()`.

```
140         {
141             mlc_dispatch(is_proc)(p);
142         };
```

7.351.2.4 `template<typename Exact> void oln::topo::tarjan::abstract::tarjan< Exact`
`>::mark_set (const point_type & x) [inline, protected]`

Mark a point as a new component.

Warning:

Implement `mark_set_impl` to be able to use this method.

Definition at line 115 of file tarjan.hh.

Referenced by oln::topo::tarjan::abstract::tarjan_with_attr< mlc::exact_vt< flat_zone< T, DestType, A, Exact >, Exact >::ret >::get_compute_impl().

```

116         {
117             mlc_dispatch(mark_set)(x);
118         }

```

7.351.2.5 template<typename Exact> comp_type oln::topo::tarjan::abstract::tarjan< Exact >::ncomps() const [inline]

Give the number of component found.

Warning:

Implement ncomps_impl to be able to use this method.

Definition at line 90 of file tarjan.hh.

```

91         {
92             mlc_dispatch(ncomps)();
93         }

```

7.351.2.6 template<typename Exact> void oln::topo::tarjan::abstract::tarjan< Exact >::uni(const point_type & n, const point_type & p) [inline, protected]

Perform an union between two components.

Warning:

Implement uni_impl to be able to use this method.

Definition at line 127 of file tarjan.hh.

Referenced by oln::topo::tarjan::abstract::tarjan_with_attr< mlc::exact_vt< flat_zone< T, DestType, A, Exact >, Exact >::ret >::get_compute_impl().

```

128         {
129             mlc_dispatch(uni)(n, p);
130         }

```

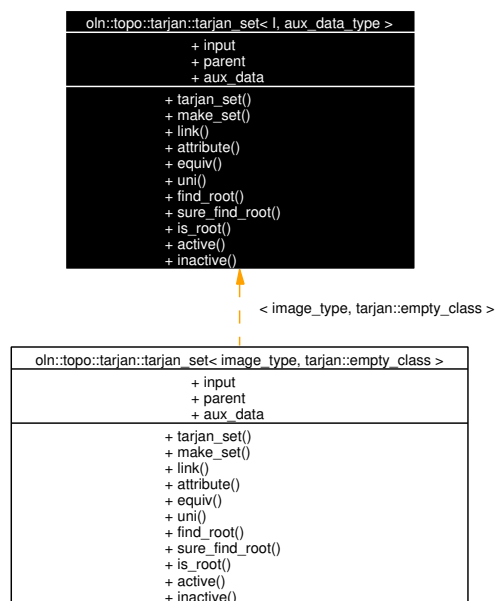
The documentation for this struct was generated from the following file:

- tarjan.hh

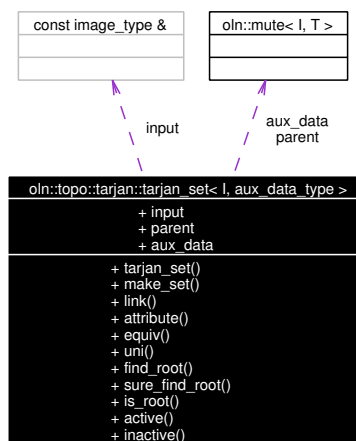
7.352 oln::topo::tarjan::tarjan_set< I, aux_data_type > Struct Template Reference

```
#include <union.hh>
```

Inheritance diagram for oln::topo::tarjan::tarjan_set< I, aux_data_type >:



Collaboration diagram for oln::topo::tarjan::tarjan_set< I, aux_data_type >:



Public Types

- typedef `mlc::exact< I >::ret::point_type` **point_type**
- typedef `mlc::exact< I >::ret::value_type` **data_type**
- typedef `oln::mute< I >::ret` **image_type**

- typedef [mute](#)< I, point_type >::ret **ima_parent_type**
- typedef [mute](#)< I, aux_data_type >::ret **ima_aux_data_type**

Public Member Functions

- **tarjan_set** (const image_type &ima)
- void **make_set** (const point_type &x)
- void **link** (const point_type &x, const point_type &y)
- unsigned int **attribute** (const point_type &x)
- bool **equiv** (const point_type &x, const point_type &y)
- void **uni** (const point_type &n, const point_type &p)
- const point_type & **find_root** (const point_type &x)
- const point_type & **sure_find_root** (const point_type &x)
- const bool **is_root** (const point_type &x)

Static Public Member Functions

- const point_type & **active** ()
- const point_type & **inactive** ()

Public Attributes

- const image_type & **input**
- [ima_parent_type](#) **parent**
- [ima_aux_data_type](#) **aux_data**

7.352.1 Detailed Description

template<class I, class aux_data_type> struct oln::topo::tarjan::tarjan_set< I, aux_data_type >

Tarjan set.

Todo

FIXME: Obsolete since only [obsolete::flat_zone](#) use it.

Definition at line 49 of file union.hh.

The documentation for this struct was generated from the following file:

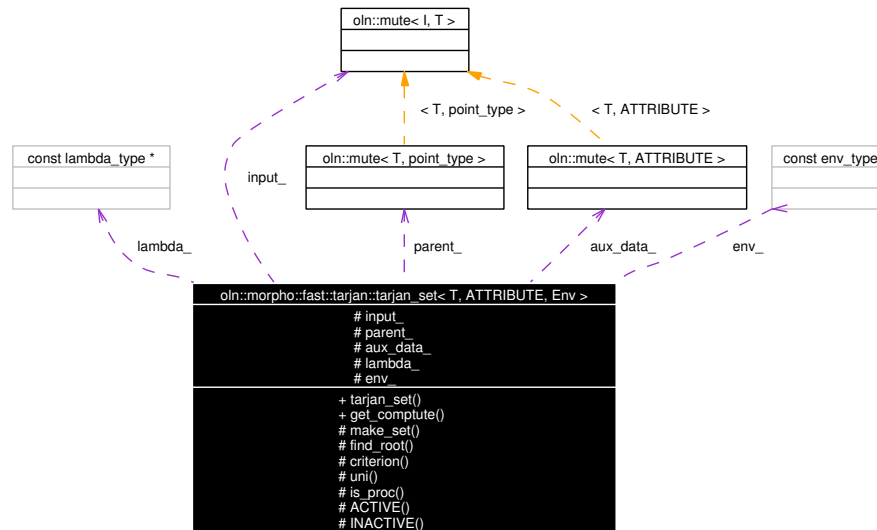
- union.hh

7.353 oln::morpho::fast::tarjan::tarjan_set< T, ATTRIBUTE, Env > Struct Template Reference

Struct that contains everything to compute an attribute opening or closing.

```
#include <attribute_union_find.hh>
```

Collaboration diagram for oln::morpho::fast::tarjan::tarjan_set< T, ATTRIBUTE, Env >:



Public Types

- typedef `mlc::exact< T >::ret::point_type` `point_type`
Associated point type.
- typedef `mlc::exact< T >::ret::value_type` `data_type`
Associated data_type.
- typedef `oln::mute< T >::ret image_type`
Image type to open/close.
- typedef `ATTRIBUTE::lambda_type` `lambda_type`
Threshold type.
- typedef `Env` `env_type`
Environment type.

Public Member Functions

- `tarjan_set` (const `image_type` &ima, const `env_type` &env)
tarjan_set constructor.

- template<bool closing, class N> [image_type](#) [get_comptute](#) (const [lambda_type](#) &lambda, const [abstract::neighborhood](#)< N > &Ng)

Main method to perform an attribute opening/closing.

Protected Member Functions

- void [make_set](#) (const [point_type](#) &x)
Make a new component from a point.
 - x Root of the component.
- [point_type](#) [find_root](#) (const [point_type](#) &x)
find the root of a component.
 - x A point of the component.
- bool [criterion](#) (const [point_type](#) &x, const [point_type](#) &y)
Check if two components should be merged.
 - x A point of the first component.
 - y A point of the second component.
- void [uni](#) (const [point_type](#) &n, const [point_type](#) &p)
Do union of two components.
 - n A point of the first component.
 - p A point of the second component.
- bool [is_proc](#) (const [point_type](#) &p) const
Tells if a point has been proceeded.

Static Protected Member Functions

- const [point_type](#) & [ACTIVE](#) ()
Return the value of an active point.
- const [point_type](#) & [INACTIVE](#) ()
Return the value of an inactive point.

Protected Attributes

- const [image_type](#) & [input_](#)
Input image.
- [mute](#)< T, [point_type](#) >::ret [parent_](#)
Give a parent of a point.
- [mute](#)< T, ATTRIBUTE >::ret [aux_data_](#)
Image to store attributes.

- const [lambda_type](#) * [lambda_](#)
Threshold.
- const [env_type](#) [env_](#)
Environment.

7.353.1 Detailed Description

template<class **T**, class **ATTRIBUTE**, class **Env** = typename **oln::morpho::attr::attr_traits**< **ATTRIBUTE** >::env_type> **struct** **oln::morpho::fast::tarjan::tarjan_set**< **T**, **ATTRIBUTE**, **Env** >

Struct that contains everything to compute an attribute opening or closing.

Parameters:

- T** Exact type of images to process.
ATTRIBUTE Exact type of attribute to use.
Env Type of environment to use.

Todo

FIXME: a similar class is defined in `oln/topo/tarjan/union.hh` ([oln::topo::tarjan::tarjan_set](#)).

Definition at line 56 of file `attribute_union_find.hh`.

7.353.2 Constructor & Destructor Documentation

7.353.2.1 **template**<class **T**, class **ATTRIBUTE**, class **Env** = typename **oln::morpho::attr::attr_traits**< **ATTRIBUTE** >::env_type> [oln::morpho::fast::tarjan::tarjan_set](#)< **T**, **ATTRIBUTE**, **Env** >::[tarjan_set](#) (const [image_type](#) & *ima*, const [env_type](#) & *env*)
[inline]

[tarjan_set](#) constructor.

Parameters:

- ima* Image to open/close.
env Environment to use to compute attributes.

Definition at line 69 of file `attribute_union_find.hh`.

References `oln::morpho::fast::tarjan::tarjan_set`< **T**, **ATTRIBUTE**, **Env** >::aux_data_, `oln::morpho::fast::tarjan::tarjan_set`< **T**, **ATTRIBUTE**, **Env** >::env_, `oln::morpho::fast::tarjan::tarjan_set`< **T**, **ATTRIBUTE**, **Env** >::env_type, `oln::morpho::fast::tarjan::tarjan_set`< **T**, **ATTRIBUTE**, **Env** >::image_type, `oln::morpho::fast::tarjan::tarjan_set`< **T**, **ATTRIBUTE**, **Env** >::INACTIVE(), `oln::morpho::fast::tarjan::tarjan_set`< **T**, **ATTRIBUTE**, **Env** >::input_, and `oln::morpho::fast::tarjan::tarjan_set`< **T**, **ATTRIBUTE**, **Env** >::parent_.

```

69                                     : input_(ima),
70                                     parent_(ima.size()),
71                                     aux_data_(ima.size()),
72                                     env_(env)
73     {
74         level::fill(parent_, INACTIVE());
75     }
```

7.353.3 Member Function Documentation

7.353.3.1 `template<class T, class ATTRIBUTE, class Env = typename oln::morpho::attr::attr_traits< ATTRIBUTE >::env_type> template<bool closing, class N> image_type oln::morpho::fast::tarjan::tarjan_set< T, ATTRIBUTE, Env >::get_comptute (const lambda_type & lambda, const abstract::neighborhood< N > & Ng) [inline]`

Main method to perform an attribute opening/closing.

Parameters:

closing True -> a closing is performed, an opening otherwise.

lambda Threshold to use for attribute growing.

Ng Neighborhood to use in the algorithm.

Returns:

The resulting image.

Definition at line 87 of file attribute_union_find.hh.

References `oln::morpho::fast::tarjan::tarjan_set< T, ATTRIBUTE, Env >::ACTIVE()`, `oln::morpho::fast::tarjan::tarjan_set< T, ATTRIBUTE, Env >::aux_data_`, `oln::abstract::neighborhood< Exact >::delta()`, `oln::morpho::fast::tarjan::tarjan_set< T, ATTRIBUTE, Env >::image_type`, `oln::morpho::fast::tarjan::tarjan_set< T, ATTRIBUTE, Env >::INACTIVE()`, `oln::morpho::fast::tarjan::tarjan_set< T, ATTRIBUTE, Env >::input_`, `oln::morpho::fast::tarjan::tarjan_set< T, ATTRIBUTE, Env >::is_proc()`, `oln::morpho::fast::tarjan::tarjan_set< T, ATTRIBUTE, Env >::lambda_`, `oln::morpho::fast::tarjan::tarjan_set< T, ATTRIBUTE, Env >::make_set()`, `oln::morpho::fast::tarjan::tarjan_set< T, ATTRIBUTE, Env >::parent_`, and `oln::morpho::fast::tarjan::tarjan_set< T, ATTRIBUTE, Env >::uni()`.

```

89         {
90             lambda_ = &lambda;
91
92             std::vector<point_type>    I(input_.npoints());
93
94             oln::utils::select_distrib_sort<closing>()(input_, I);
95
96             level::fill(aux_data_, ntg_sup_val(lambda_type));
97             aux_data_.border_adapt_assign(Ng.delta(), ntg_sup_val(lambda_type));
98
99             // We are ready to perform stuff
100             for (unsigned int p = 0; p < I.size(); ++p)
101             {
102                 point_type p_p = I[p];
103                 make_set(p_p);
104                 oln_neighb_type(N) Q_prime(Ng, p_p);
105                 for_all (Q_prime)
106                     if (is_proc(Q_prime))
107                         uni(Q_prime.cur(), p_p);
108             }
109
110             // Resolving phase
111             image_type output(input_.size());
112             for (int p = I.size() - 1; p >= 0; --p)
113             {
114                 point_type p_p = I[p];
115                 if ((parent_[p_p] == ACTIVE()) || (parent_[p_p] == INACTIVE()))
116                     output[p_p] = input_[p_p];
117                 else
118                     output[p_p] = output[parent_[p_p]];

```

```
119             // this code is equivalent to
120             //      output[I[p].first] = input_[find_root(I[p].first)];
121
122         }
123     return output;
124 }
```

The documentation for this struct was generated from the following file:

- attribute_union_find.hh

7.354 oln::topo::tarjan::tarjan_traits< flat_zone< T, DestType, A, Exact > > Struct Template Reference

Traits specialization for [flat_zone](#).

```
#include <flat-zone.hh>
```

Public Types

- typedef T [input_type](#)
Type of the input image.
- typedef [mute](#)< T, DestType >::ret [output_type](#)
Type of the output image.
- typedef A [attr_type](#)
Type of attribute to use.

7.354.1 Detailed Description

```
template<typename T, typename DestType, typename A, typename Exact> struct  
oln::topo::tarjan::tarjan_traits< flat_zone< T, DestType, A, Exact > >
```

Traits specialization for [flat_zone](#).

Parameters:

- T* Type of the input image.
- DestType* Data type of the output image (label type).
- A* Attribute you want to compute.
- Exact* Exact type of the [flat_zone](#) class.

Definition at line 183 of file flat-zone.hh.

The documentation for this struct was generated from the following file:

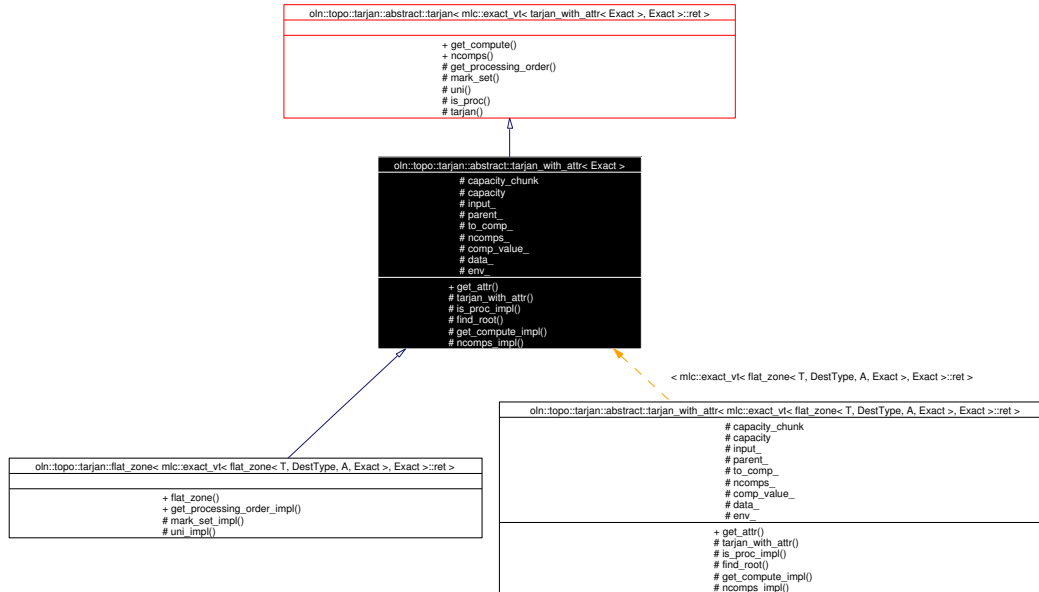
- flat-zone.hh

7.355 oln::topo::tarjan::abstract::tarjan_with_attr< Exact > Struct Template Reference

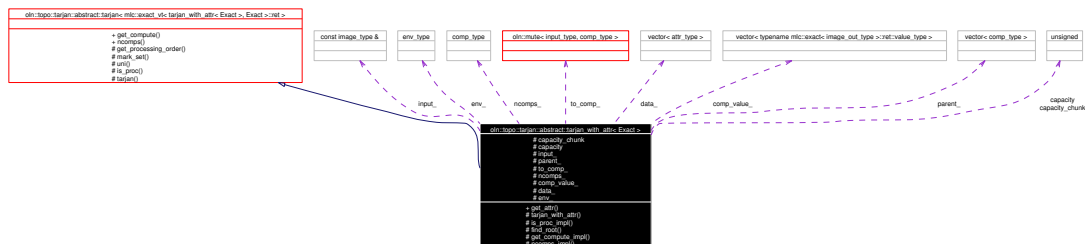
Abstract class to perform a tarjan algorithm on an image with attribute computing .

```
#include <tarjan_with_attr.hh>
```

Inheritance diagram for oln::topo::tarjan::abstract::tarjan_with_attr< Exact >:



Collaboration diagram for oln::topo::tarjan::abstract::tarjan_with_attr< Exact >:



Public Types

- typedef oln::topo::tarjan::tarjan_traits< Exact >::input_type input_type
Type of input image.
- typedef mlc::exact< input_type >::ret::point_type point_type
Point type of image to process.
- typedef mlc::exact< input_type >::ret::value_type data_type
Data type of the input image.

- typedef oln::mute< input_type >::ret image_type
Concrete type of the input image.
- typedef oln::topo::tarjan::tarjan_traits< Exact >::output_type image_out_type
Type of output image.
- typedef oln::topo::tarjan::tarjan_traits< Exact >::attr_type attr_type
Type of attribute to use.
- typedef oln::morpho::attr::attr_traits< attr_type >::env_type env_type
Environment associated to attribute type.
- typedef mlc::exact< image_out_type >::ret::value_type comp_type
Type of components.
- typedef mlc::exact_vt< tarjan_with_attr< Exact >, Exact >::ret exact_type
Exact type of the class.
- typedef tarjan< exact_type > super_type
Type of parent class.

Public Member Functions

- const attr_type & get_attr (const comp_type &i) const
Return the attribute value associated to a component i.

Protected Member Functions

- tarjan_with_attr (const image_type &ima, const env_type &env)
Constructor.
- bool is_proc_impl (const point_type &p) const
Implementation of is_proc().
- comp_type find_root (const comp_type &x) const
Implementation of find_root().
- template<class N> image_out_type get_compute_impl (const oln::abstract::neighborhood< N > &Ng)
Implementation of get_compute().
- comp_type ncomps_impl () const
Implementation of ncomps_impl().

Protected Attributes

- unsigned **capacity_chunk**
- unsigned **capacity**
- const **image_type** & **input_**
- std::vector< **comp_type** > **parent_**
- **oln::mute**< **input_type**, **comp_type** >::**ret to_comp_**
- **comp_type** **ncomps_**
- std::vector< typename mlc::exact< **image_out_type** >::**ret::value_type** > **comp_value_**
- std::vector< **attr_type** > **data_**
- **env_type** **env_**

7.355.1 Detailed Description

template<class Exact> struct oln::topo::tarjan::abstract::tarjan_with_attr< Exact >

Abstract class to perform a tarjan algorithm on an image with attribute computing .

Parameters:

Exact Exact type of the class.

Definition at line 56 of file tarjan_with_attr.hh.

7.355.2 Constructor & Destructor Documentation

7.355.2.1 **template<class Exact> [oln::topo::tarjan::abstract::tarjan_with_attr](#)< Exact >::[tarjan_with_attr](#) (const [image_type](#) & *ima*, const [env_type](#) & *env*)** [inline, protected]

Constructor.

Initialization of data structures.

- *ima* Image to process.
- *env* Environment to use.

Warning:

This constructor is protected, because this class is abstract.

Definition at line 96 of file tarjan_with_attr.hh.

```

97                                     :
98         capacity_chunk((ima.npoints() < 100) ? ima.npoints() : (ima.npoints() / 100)),
99         capacity(capacity_chunk),
100         input_(ima),
101         parent_(),
102         to_comp_(ima.size()),
103         comp_value_(),
104         env_(env)
105     {
106         parent_.reserve(capacity);
107         comp_value_.reserve(capacity);
108         data_.reserve(capacity);
109     }
```


7.355.3 Member Function Documentation

7.355.3.1 `template<class Exact> comp_type oln::topo::tarjan::abstract::tarjan_with_attr< Exact >::find_root (const comp_type &x) const` [inline, protected]

Implementation of `find_root()`.

- x The component you want to find the root.

Warning:

Do not call this method, use `find_root()` instead.

Definition at line 132 of file `tarjan_with_attr.hh`.

Referenced by `oln::topo::tarjan::abstract::tarjan_with_attr< mlc::exact_vt< flat_zone< T, DestType, A, Exact >, Exact >::ret >::find_root()`, `oln::topo::tarjan::abstract::tarjan_with_attr< mlc::exact_vt< flat_zone< T, DestType, A, Exact >, Exact >::ret >::get_attr()`, `oln::topo::tarjan::abstract::tarjan_with_attr< mlc::exact_vt< flat_zone< T, DestType, A, Exact >, Exact >::ret >::get_compute_impl()`, and `oln::topo::tarjan::flat_zone< T, DestType, A, Exact >::uni_impl()`.

```

133         {
134             if (parent_[x] != x)
135                 return parent_[x] = find_root(parent_[x]);
136             return x;
137         }

```

7.355.3.2 `template<class Exact> template<class N> image_out_type oln::topo::tarjan::abstract::tarjan_with_attr< Exact >::get_compute_impl (const oln::abstract::neighborhood< N > &Ng)` [inline, protected]

Implementation of `get_compute()`.

- Ng Neighborhood to use.

Warning:

Do not call this method, use `get_compute()` instead.

Definition at line 149 of file `tarjan_with_attr.hh`.

```

150         {
151             std::vector<point_type> I(get_processing_order());
152
153             level::fill(to_comp_, ntg_sup_val(comp_type));
154             to_comp_.border_adapt_assign(Ng.delta(), ntg_sup_val(comp_type));
155             ncomps_ = 0;
156             parent_.push_back(0);
157             comp_value_.push_back(0);
158             data_.push_back(attr_type());
159
160             // We are ready to perform stuff
161             for (unsigned int p = 0; p < I.size(); ++p)
162             {
163                 point_type p_p = I[p];
164                 mark_set(p_p);
165             }

```

```

166         oln_neighb_type(N) Q_prime(Ng, p_p);
167         for_all (Q_prime)
168             if (is_proc(Q_prime))
169                 uni(Q_prime.cur(), p_p);
170         if (to_comp[p_p] == (ncomps_ + 1)) // new component
171             ++ncomps_;
172         else
173             {
174                 comp_value_.pop_back();
175                 parent_.pop_back();
176                 data_.pop_back();
177             }
178     }
179
180     // Resolving phase
181     std::map<comp_type, comp_type>          cmps;
182     ncomps_ = 0;
183     for (int p = I.size() - 1; p >= 0; --p)
184     {
185         point_type p_p = I[p];
186         if (cmps.find(find_root(to_comp[p_p])) == cmps.end())
187             {
188                 cmps[comp_value_[find_root(to_comp[p_p])] = ncomps_;
189                 comp_value_[find_root(to_comp[p_p])] = ncomps_++;
190             }
191     }
192
193     image_out_type output(input_.size());
194     for (int p = I.size() - 1; p >= 0; --p)
195     {
196         point_type p_p = I[p];
197         output[p_p] = comp_value_[find_root(to_comp[p_p])];
198     }
199     return output;
200 }

```

7.355.3.3 `template<class Exact> bool oln::topo::tarjan::abstract::tarjan_with_attr< Exact >::is_proc_impl (const point_type &p) const` [inline, protected]

Implementation of `is_proc()`.

- p Point you want to check.

Warning:

Do not call this method, use `is_proc()` instead.

Definition at line 118 of file `tarjan_with_attr.hh`.

```

119     {
120         return to_comp[p] != ntg_sup_val(comp_type);
121     };

```

7.355.3.4 `template<class Exact> comp_type oln::topo::tarjan::abstract::tarjan_with_attr< Exact >::ncomps_impl () const` [inline, protected]

Implementation of `ncomps_impl()`.

Warning:

Do not call this method, use `ncomps()` instead.

Definition at line 207 of file tarjan_with_attr.hh.

```
208         {  
209             return ncomps_;  
210         }
```

The documentation for this struct was generated from the following file:

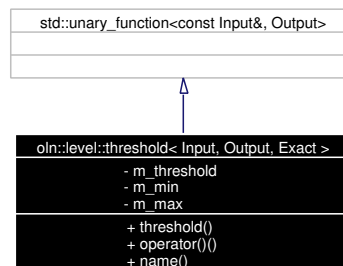
- tarjan_with_attr.hh

7.356 oln::level::threshold< Input, Output, Exact > Class Template Reference

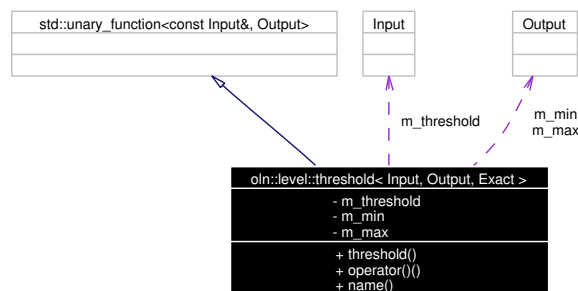
Threshold the value of the image.

```
#include <threshold.hh>
```

Inheritance diagram for oln::level::threshold< Input, Output, Exact >:



Collaboration diagram for oln::level::threshold< Input, Output, Exact >:



Public Member Functions

- [threshold](#) (const Input &[threshold](#), const Output &min=[ntg::type_traits](#)< Output >::min(), const Output &max=[ntg::type_traits](#)< Output >::max())
- Output [operator\(\)](#) (const Input &v) const

Static Public Member Functions

- std::string [name](#) ()

7.356.1 Detailed Description

```
template<class Input, class Output, class Exact = mlc::final> class oln::level::threshold< Input, Output, Exact >
```

Threshold the value of the image.

```
#include <oln/basics2d.hh>
#include <oln/level/threshold.hh>
#include <ntg/all.hh>
using namespace ntg;
int main()
{
    oln::image2d<int_u8> in = oln::load(IMG_IN "lena256.pgm");
    int_u8 th      = 127;
    rgb_8 low      = rgb_8(100, 0, 0);
    rgb_8 height = rgb_8(0, 200, 255);

    oln::image2d<rgb_8> out
        = apply(oln::level::threshold<int_u8, rgb_8 >(th, low, height),
               in);
    save(out, IMG_OUT "oln_level_threshold.ppm");
}
```



=>



Definition at line 65 of file threshold.hh.

7.356.2 Constructor & Destructor Documentation

7.356.2.1 `template<class Input, class Output, class Exact = mlc::final> oln::level::threshold< Input, Output, Exact >::threshold (const Input & threshold, const Output & min = ntg::type_traits< Output >::min(), const Output & max = ntg::type_traits< Output >::max()) [inline]`

- Threshold any value heigher or equal to this value will return *min*.
- min Value returned if the input is smaller than threshold.
- max Value returned if the input is greater or equal to the threshold.

Definition at line 74 of file threshold.hh.

```
76                                     :  
77         m_threshold(threshold), m_min(min), m_max(max)  
78     {}
```

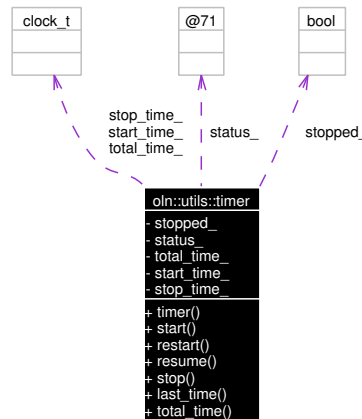
The documentation for this class was generated from the following file:

- threshold.hh

7.357 oln::utils::timer Class Reference

```
#include <timer.hh>
```

Collaboration diagram for oln::utils::timer:



Public Member Functions

- [timer](#) ()
- void [start](#) ()
- float [restart](#) ()
- void [resume](#) ()
- float [stop](#) ()
- float [last_time](#) () const
Time since the last [resume\(\)](#) or [start\(\)](#).
- float [total_time](#) () const
Total time elapsed.

7.357.1 Detailed Description

Timer class.

Definition at line 39 of file timer.hh.

7.357.2 Constructor & Destructor Documentation

7.357.2.1 oln::utils::timer::timer () [inline]

Constructor.

Note:

The timer is not stopped, it is in a so-called 'unknown state'. Therefore, Restart can not be called.

Definition at line 47 of file timer.hh.

```
48     {
49         status_ = e_unknown;
50         total_time_ = 0;
51         start_time_ = clock();
52     }
```

7.357.3 Member Function Documentation

7.357.3.1 float oln::utils::timer::restart () [inline]

Restart the timer.

Precondition:

The timer should have been start at least one time.

Definition at line 72 of file timer.hh.

References start(), and total_time().

```
73     {
74         assertion(status_ != e_unknown);
75         float val = total_time();
76         start();
77         return val;
78     }
```

7.357.3.2 void oln::utils::timer::resume () [inline]

Resume the timer.

Precondition:

The timer should be running.

Definition at line 85 of file timer.hh.

```
86     {
87         assertion(status_ == e_stopped);
88         status_ = e_running;
89         start_time_ = clock();
90     }
```

7.357.3.3 void oln::utils::timer::start () [inline]

Start the timer.

Precondition:

The timer should not be running.

Definition at line 59 of file timer.hh.

Referenced by restart().


```
60     {
61         assertion(status_ != e_running);
62         total_time_ = 0;
63         stop_time_ = 0;
64         status_ = e_running;
65         start_time_ = clock();
66     }
```

7.357.3.4 float oln::utils::timer::stop () [inline]

Stop the timer.

Precondition:

The timer should be running.

Definition at line 97 of file timer.hh.

References `total_time()`.

```
98     {
99         assertion(status_ == e_running);
100         stop_time_ = clock();
101         total_time_ += (stop_time_ - start_time_);
102         status_ = e_stopped;
103         return total_time();
104     }
```

The documentation for this class was generated from the following file:

- timer.hh

7.358 oln::io::internal::try_readers< R, T > Struct Template Reference

Image's reader.

```
#include <image_read.hh>
```

Static Public Member Functions

- bool [by_extension](#) (T &output, std::istream &in, const std::string &ext)
Read an object from a stream. Try to deduce the image format from the extension.
 - *output* The new object to write.
 - *in* The stream.
 - *ext* The extension's filename.
- bool [by_data](#) (T &output, std::istream &in)
Read an object from a stream. Try to match the file format referring to the data only.
 - *output* The new object to write.
 - *in* The stream.

7.358.1 Detailed Description

```
template<reader_id R, typename T> struct oln::io::internal::try_readers< R, T >
```

Image's reader.

Parameters:

R Type of reader.

Definition at line 58 of file image_read.hh.

The documentation for this struct was generated from the following file:

- image_read.hh

7.359 oln::io::internal::try_readers< ReadNone, T > Struct Template Reference

Reader for image of type None.

```
#include <image_read.hh>
```

Static Public Member Functions

- bool [by_extension](#) (T &, std::istream &, const std::string &)
Read an object from a stream. Try to deduce the image format from the extension.
- bool [by_data](#) (T &, std::istream &)
Read an object from a stream. Try to match the file format referring to the data only.

7.359.1 Detailed Description

template<typename T> struct oln::io::internal::try_readers< ReadNone, T >

Reader for image of type None.

In fact, do nothing because of the type of reader. Used when an errors occurs on others types of reader.

Definition at line 101 of file image_read.hh.

The documentation for this struct was generated from the following file:

- image_read.hh

7.360 oln::io::internal::try_stream_wrappers_in< W, T, Reader > Struct Template Reference

Read a stream.

```
#include <stream_wrapper.hh>
```

Static Public Member Functions

- bool [by_extension](#) (T &output, const std::string &name, const std::string &ext)
Open file as a stream, take extension and call the "real" reader. If the extension is not known, reiterate with a stream_id - 1.
- bool [by_data](#) (T &output, const std::string &name)
Open file as a stream, take extension and call the "real" reader. If the extension is not known, reiterate with a stream_id - 1.
 - *output* The new object.
 - *name* The filename.

7.360.1 Detailed Description

template<stream_id W, typename T, class Reader> struct oln::io::internal::try_stream_wrappers_in< W, T, Reader >

Read a stream.

Parameters:

- W** The type of stream
- Reader** The real reader of this stream.

Definition at line 135 of file stream_wrapper.hh.

7.360.2 Member Function Documentation

7.360.2.1 **template<stream_id W, typename T, class Reader> bool [oln::io::internal::try_stream_wrappers_in](#)< W, T, Reader >::by_data** (T & *output*, const std::string & *name*)
[inline, static]

Open file as a stream, take extension and call the "real" reader. If the extension is not known, reiterate with a stream_id - 1.

- *output* The new object.
- *name* The filename.

Todo

FIXME: it sounds strange to read wrapped file without matching its extension.

Definition at line 171 of file stream_wrapper.hh.

```
172     {
173         std::string wrapped_name = name;
174         if (std::istream* in = stream_wrapper<W>::wrap_in(wrapped_name))
175         {
176             std::string wrapped_ext = utils::extension(wrapped_name);
177             bool result = Reader::doit(output, *in, wrapped_ext);
178             delete in;
179             if (result)
180                 return true;
181         }
182         return try_stream_wrappers_in<stream_id(unsigned(W)-1), T, Reader>
183             ::by_data(output, name);
184     }
```

The documentation for this struct was generated from the following file:

- stream_wrapper.hh

7.361 oln::io::internal::try_stream_wrappers_in< StreamNone, T, Reader > Struct Template Reference

Read a stream of type None.

```
#include <stream_wrapper.hh>
```

Static Public Member Functions

- bool [by_extension](#) (T &, const std::string &, const std::string &)
Read a stream of type None by data.
- bool [by_data](#) (T &, const std::string &)

7.361.1 Detailed Description

template<typename T, class Reader> struct oln::io::internal::try_stream_wrappers_in< StreamNone, T, Reader >

Read a stream of type None.

In fact, do nothing because of the type of stream. Used when an errors occurs on others types of stream.

Definition at line 194 of file stream_wrapper.hh.

The documentation for this struct was generated from the following file:

- stream_wrapper.hh

7.362 oln::io::internal::try_stream_wrappers_out< W, T, Writer > Struct Template Reference

Write a stream.

```
#include <stream_wrapper.hh>
```

Static Public Member Functions

- bool [by_extension](#) (const T &input, const std::string &name, const std::string &ext)
Open file as a stream, take extension and call the "real" writer. If the extension is not known, reiterate with a stream_id - 1.

7.362.1 Detailed Description

`template<stream_id W, typename T, class Writer> struct oln::io::internal::try_stream_wrappers_out< W, T, Writer >`

Write a stream.

Parameters:

- W** The type of stream
- Writer** The "real" writer of this stream.

Definition at line 218 of file stream_wrapper.hh.

The documentation for this struct was generated from the following file:

- stream_wrapper.hh

7.363 oln::io::internal::try_stream_wrappers_out< StreamNone, T, Reader > Struct Template Reference

Write a stream of type None.

```
#include <stream_wrapper.hh>
```

Static Public Member Functions

- bool [by_extension](#) (const T &, const std::string &, const std::string &)
< Do nothing because of the type of stream.

7.363.1 Detailed Description

template<typename T, class Reader> struct oln::io::internal::try_stream_wrappers_out< StreamNone, T, Reader >

Write a stream of type None.

In fact, do nothing because of the type of stream. Used when an errors occurs on others types of stream.

Definition at line 255 of file stream_wrapper.hh.

The documentation for this struct was generated from the following file:

- stream_wrapper.hh

7.364 oln::io::internal::try_writers< W, T > Struct Template Reference

Image's writer.

```
#include <image_write.hh>
```

Static Public Member Functions

- bool [by_extension](#) (const T &input, std::ostream &out, const std::string &ext)

Write object on the stream. Try to deduce the image format from the extension.

- *input* The object to read.
- *out* The stream.
- *ext* The extension's filename.

- bool [by_data](#) (const T &input, std::ostream &out, const std::string &ext)

Try to match the file format referring to the data only.

- *input* The object to read.
- *out* The stream.

7.364.1 Detailed Description

```
template<writer_id W, typename T> struct oln::io::internal::try_writers< W, T >
```

Image's writer.

Parameters:

W Type of writer.

Definition at line 58 of file image_write.hh.

The documentation for this struct was generated from the following file:

- image_write.hh

7.365 oln::io::internal::try_writers< WriteNone, T > Struct Template Reference

Reader for image of type None.

```
#include <image_write.hh>
```

Static Public Member Functions

- bool [by_extension](#) (const T &, std::ostream &, const std::string &)
Write object on the stream. Try to deduce the image format from the extension.
- bool [by_data](#) (const T &, std::ostream &, const std::string &)
Write object on the stream. Try to match the file format referring to the data only.

7.365.1 Detailed Description

template<typename T> struct oln::io::internal::try_writers< WriteNone, T >

Reader for image of type None.

In fact, do nothing because of the type of writer. Used when an errors occurs on others types of writer.

Definition at line 105 of file image_write.hh.

The documentation for this struct was generated from the following file:

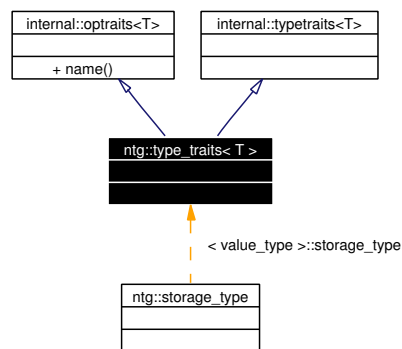
- image_write.hh

7.366 ntg::type_traits< T > Class Template Reference

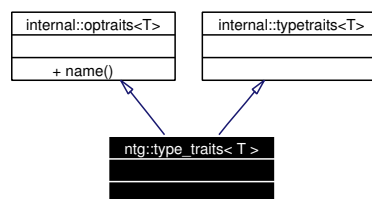
Associates properties and methods to types.

```
#include <type_traits.hh>
```

Inheritance diagram for ntg::type_traits< T >:



Collaboration diagram for ntg::type_traits< T >:



7.366.1 Detailed Description

```
template<class T> class ntg::type_traits< T >
```

Associates properties and methods to types.

`type_traits` gives a variable set of properties, depending on the type considered. For example, `type_traits<int_u8>` can give `larger_type`, `max()`, ...

It refers to the exact type, this means `type_traits<any<int_u8> >` will give the same results than `type_traits<int_u8>`.

Traits can be defined for builtin values, so that `type_traits<int>` works, this is the main advantage on `T::xxx` approach to define properties.

Definition at line 61 of file `type_traits.hh`.

The documentation for this class was generated from the following file:

- `type_traits.hh`

7.367 ntg::internal::typetraits< T > Struct Template Reference

Associates types to types.

```
#include <traits.hh>
```

Inheritance diagram for ntg::internal::typetraits< T >:



Public Types

- typedef [data_type](#) **abstract_type**

7.367.1 Detailed Description

```
template<class T> struct ntg::internal::typetraits< T >
```

Associates types to types.

Specialize it for every types.

Please note that typetraits only associates types, and not methods. This is necessary to handle mutual instantiation problems, as optraits usually needs typetraits to be instantiated.

Definition at line 56 of file traits.hh.

The documentation for this struct was generated from the following file:

- traits.hh

7.368 ntg::unsafe Struct Reference

No check performed.

```
#include <behavior.hh>
```

Static Public Member Functions

- `std::string name ()`

7.368.1 Detailed Description

No check performed.

Same behavior as the underlying builtin type.

Definition at line 80 of file `integre/ntg/real/behavior.hh`.

The documentation for this struct was generated from the following file:

- `integre/ntg/real/behavior.hh`

7.369 oln::io::internal::utils Struct Reference

Utils for io (get extension of a file).

```
#include <utils.hh>
```

Static Public Member Functions

- `std::string extension` (const `std::string &name`)

Return the extension of a filename.

– *name* The filename.

7.369.1 Detailed Description

Utils for io (get extension of a file).

Definition at line 44 of file `utils.hh`.

7.369.2 Member Function Documentation

7.369.2.1 `std::string oln::io::internal::utils::extension` (const `std::string &name`) [`inline`, `static`]

Return the extension of a filename.

- *name* The filename.

Returns:

The extension (lower case).

Definition at line 53 of file `utils.hh`.

```
54     {
55         std::string ext;
56         int pos = name.rfind('.');
57         if (pos > 0)
58         {
59             ext.assign(name, pos + 1, name.size() - pos);
60             for (std::string::iterator i = ext.begin(); i != ext.end(); ++i)
61                 *i = tolower(*i);
62         }
63         return ext;
64     }
```

The documentation for this struct was generated from the following file:

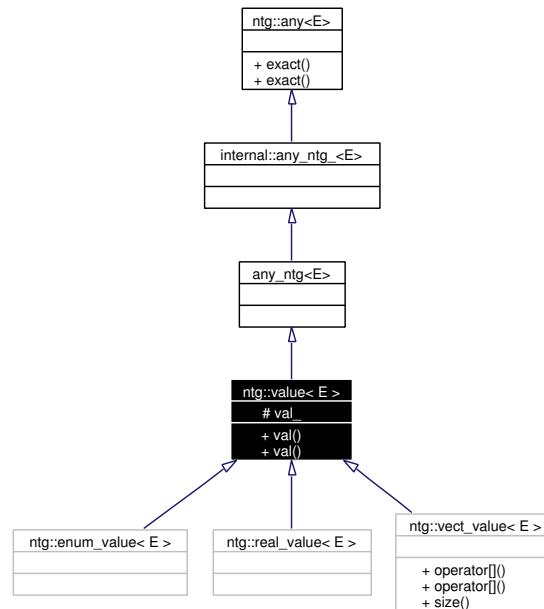
- `utils.hh`

7.370 ntg::value< E > Class Template Reference

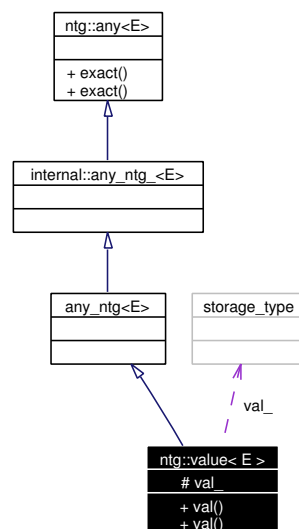
Concrete value storage class.

```
#include <value.hh>
```

Inheritance diagram for ntg::value< E >:



Collaboration diagram for ntg::value< E >:



Public Member Functions

- [storage_type](#) & `val ()`

- const [storage_type](#) & val () const

Protected Attributes

- [storage_type](#) val_

7.370.1 Detailed Description

template<class E> class ntg::value< E >

Concrete value storage class.

This class actually store the real value (usually a builtin). The value type is determined from the exact type. For example, `int_u8::val()` returns an "unsigned char".

Definition at line 91 of file `value.hh`.

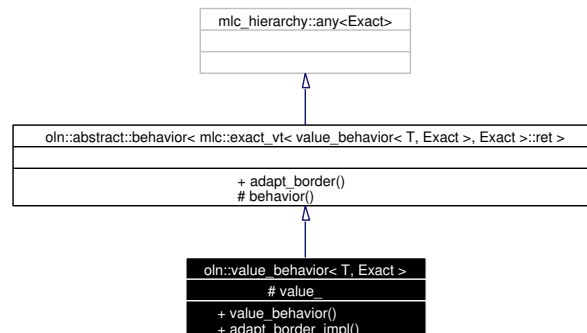
The documentation for this class was generated from the following file:

- `value.hh`

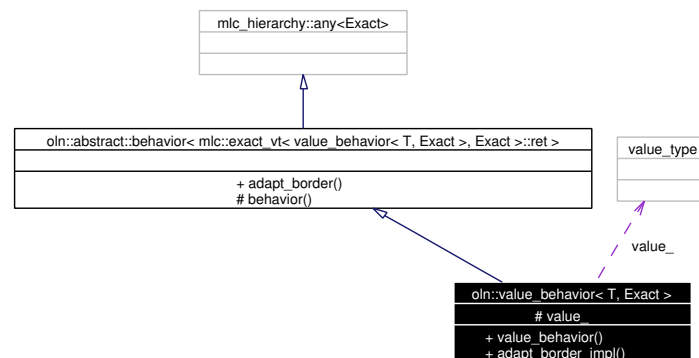
7.371 oln::value_behavior< T, Exact > Class Template Reference

```
#include <behavior.hh>
```

Inheritance diagram for oln::value_behavior< T, Exact >:



Collaboration diagram for oln::value_behavior< T, Exact >:



Public Types

- typedef `value_behavior< T, Exact >` `self_type`
- typedef `mlc::exact_vt< self_type, Exact >::ret exact_type`
- typedef `T value_type`

Public Member Functions

- `value_behavior` (`value_type value`)
- template<class I> void `adapt_border_impl` (`abstract::image< I > &im`, `coord border_size`) const

Protected Attributes

- `value_type value_`

7.371.1 Detailed Description

template<class T, class Exact = mlc::final> class oln::value_behavior< T, Exact >

Set the border to a specific value.

See also:

[abstract::image_size::border_](#)

Definition at line 65 of file olena/oln/core/behavior.hh.

7.371.2 Member Typedef Documentation

7.371.2.1 template<class T, class Exact = mlc::final> typedef mlc::exact_vt< [self_type](#) , Exact >::[ret oln::value_behavior](#)< T, Exact >::[exact_type](#)

The exact type.

Reimplemented from [oln::abstract::behavior< Exact >](#).

Definition at line 70 of file olena/oln/core/behavior.hh.

7.371.2.2 template<class T, class Exact = mlc::final> typedef [value_behavior](#)<T, Exact> [oln::value_behavior](#)< T, Exact >::[self_type](#)

The self type.

Reimplemented from [oln::abstract::behavior< Exact >](#).

Definition at line 69 of file olena/oln/core/behavior.hh.

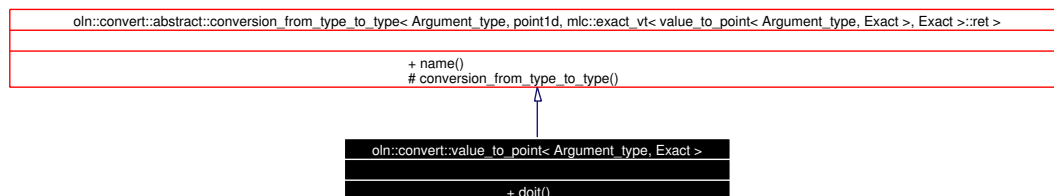
The documentation for this class was generated from the following file:

- olena/oln/core/behavior.hh

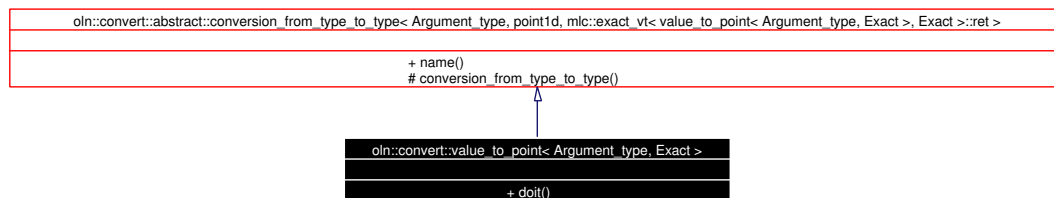
7.372 oln::convert::value_to_point< Argument_type, Exact > Struct Template Reference

```
#include <value_to_point.hh>
```

Inheritance diagram for oln::convert::value_to_point< Argument_type, Exact >:



Collaboration diagram for oln::convert::value_to_point< Argument_type, Exact >:



Public Types

- typedef [point1d](#) [result_type](#)
By default return a [point1d](#).
- typedef Argument_type [argument_type](#)

Public Member Functions

- [result_type](#) [doit](#) (const argument_type &input) const

7.372.1 Detailed Description

```
template<typename Argument_type, class Exact = mlc::final> struct oln::convert::value_to_point<
Argument_type, Exact >
```

Convert a value of pixel to a point.

For example, transform an RGB color to a 3D point (ntg::rgb_8 => [oln::point3d](#)). This function is useful to build the histogram.

Example:

```
** f(oln::convert::value_to_point<ntg::rgb_8>()(ntg::rgb_8(1,6,64)));
** // is equivalent to:
```

```
** f(oln::point3d(1, 6, 64));  
**
```

Definition at line 52 of file value_to_point.hh.

The documentation for this struct was generated from the following file:

- value_to_point.hh

7.373 oln::convert::value_to_point< Argument_type, Exact >::doit_binary< O, I > Struct Template Reference

Convert a binary to a point.

```
#include <value_to_point.hh>
```

Static Public Member Functions

- O doit (const I &input)

7.373.1 Detailed Description

```
template<typename Argument_type, class Exact = mlc::final>template<typename O, typename I>  
struct oln::convert::value_to_point< Argument_type, Exact >::doit_binary< O, I >
```

Convert a binary to a point.

Definition at line 71 of file value_to_point.hh.

The documentation for this struct was generated from the following file:

- value_to_point.hh

7.374 `oln::convert::value_to_point< Argument_type, Exact >::doit_not_binary< O, I >` Struct Template Reference

Convert a non vectorial to a point.

```
#include <value_to_point.hh>
```

Static Public Member Functions

- `O doit` (const I &input)

7.374.1 Detailed Description

```
template<typename Argument_type, class Exact = mlc::final>template<typename O, typename I>  
struct oln::convert::value_to_point< Argument_type, Exact >::doit_not_binary< O, I >
```

Convert a non vectorial to a point.

Definition at line 82 of file `value_to_point.hh`.

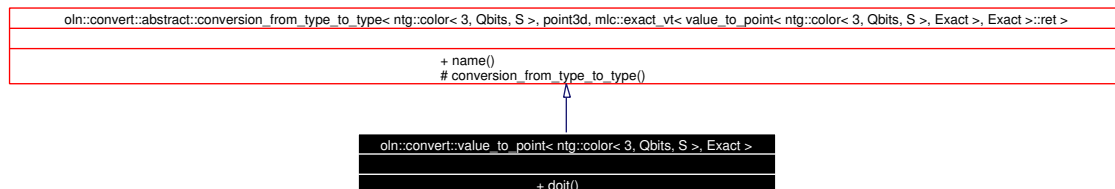
The documentation for this struct was generated from the following file:

- `value_to_point.hh`

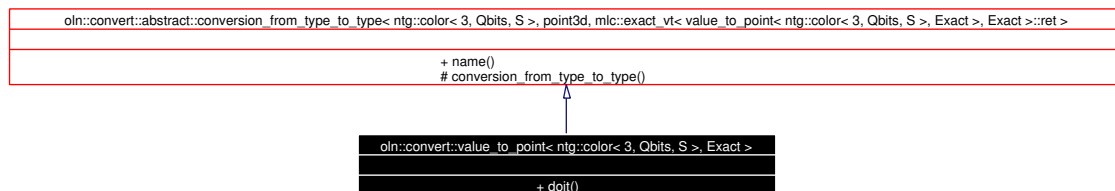
7.375 oln::convert::value_to_point< ntg::color< 3, Qbits, S >, Exact > Struct Template Reference

```
#include <value_to_point.hh>
```

Inheritance diagram for oln::convert::value_to_point< ntg::color< 3, Qbits, S >, Exact >:



Collaboration diagram for oln::convert::value_to_point< ntg::color< 3, Qbits, S >, Exact >:



Public Types

- typedef [point3d](#) **result_type**
- typedef [ntg::color< 3, Qbits, S >](#) **argument_type**

Public Member Functions

- result_type **doit** (const argument_type &input) const

7.375.1 Detailed Description

template<unsigned Qbits, template< unsigned > class S, class Exact> struct oln::convert::value_to_point< ntg::color< 3, Qbits, S >, Exact >

Specialization for color of three dimension.

Todo

Could be generalized to n dimensions if there were a trait that give a pointkd for a given dimension k.

Definition at line 108 of file value_to_point.hh.

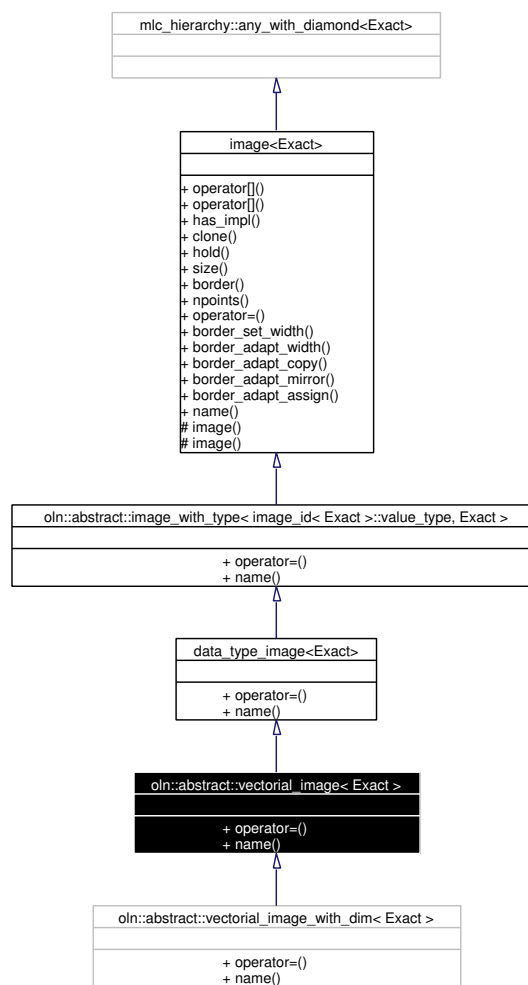
The documentation for this struct was generated from the following file:

- value_to_point.hh

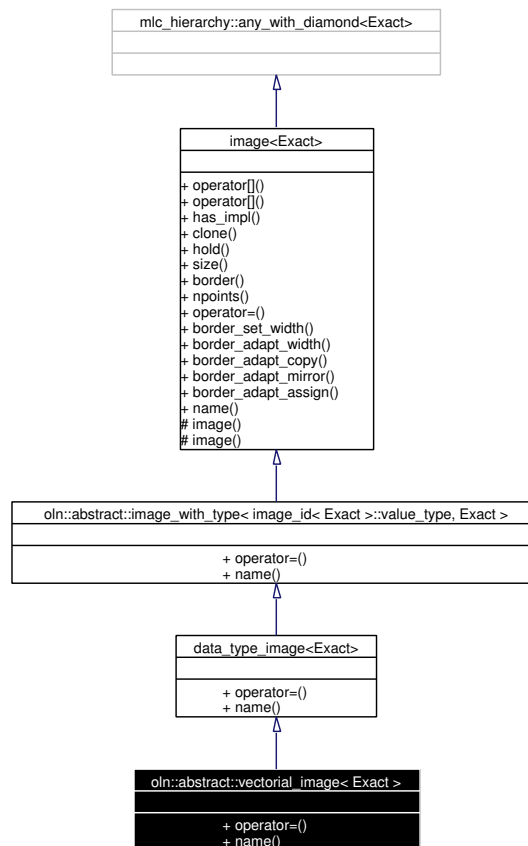
7.376 oln::abstract::vectorial_image< Exact > Class Template Reference

```
#include <image_with_type_with_dim.hh>
```

Inheritance diagram for oln::abstract::vectorial_image< Exact >:



Collaboration diagram for oln::abstract::vectorial_image< Exact >:



Public Types

- typedef `vectorial_image< Exact >` **self_type**
- typedef `Exact` **exact_type**

Public Member Functions

- `exact_type & operator= (self_type rhs)`
Perform a shallow copy between rhs and the current point, the points will not be duplicated but shared between the two images.

Static Public Member Functions

- `std::string name ()`

7.376.1 Detailed Description

`template<class Exact> class oln::abstract::vectorial_image< Exact >`

This class, when used in a function declaration, forces the function to only accept vectorial images.

Definition at line 191 of file image_with_type_with_dim.hh.

7.376.2 Member Function Documentation

7.376.2.1 `template<class Exact> exact_type& oln::abstract::vectorial_image< Exact
>::operator= (self_type rhs) [inline]`

Perform a shallow copy between *rhs* and the current point, the points will not be duplicated but shared between the two images.

See also:

[image::clone\(\)](#)

Reimplemented from [oln::abstract::data_type_image](#)< [Exact](#) >.

Definition at line 191 of file image_with_type_with_dim.hh.

273 {

The documentation for this class was generated from the following file:

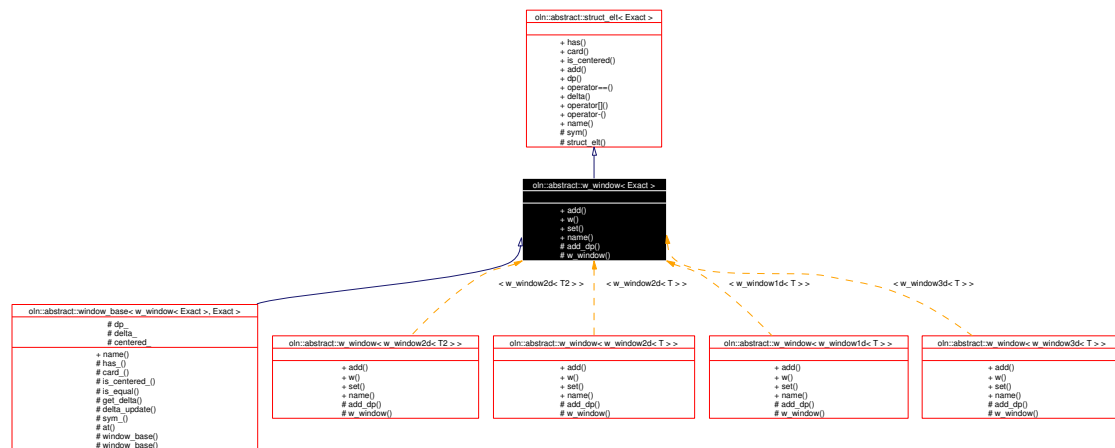
- image_with_type_with_dim.hh

7.377 oln::abstract::w_window< Exact > Struct Template Reference

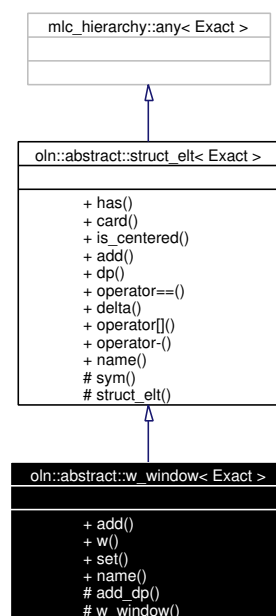
Weight Window.

```
#include <w_window.hh>
```

Inheritance diagram for oln::abstract::w_window< Exact >:



Collaboration diagram for oln::abstract::w_window< Exact >:



Public Types

- typedef Exact [exact_type](#)

Set the exact type.

- typedef [struct_elt](#)< Exact > [super_type](#)

Set the super type.

- typedef [struct_elt_traits](#)< Exact >::[dpoint_type](#) [dpoint_type](#)

The associate image's type of dpoint (move point).

- typedef [struct_elt_traits](#)< Exact >::[weight_type](#) [weight_type](#)

Set the type of weight.

Public Member Functions

- [exact_type](#) & [add](#) (const [abstract::dpoint](#)< [dpoint_type](#) > &dp, const [weight_type](#) &w=1)

Add a point (with weight) to the [w_window](#).

- [weight_type](#) w (unsigned i) const

Get the weight of a point.

– i The nth point.

- const [weight_type](#) & [set](#) (const [abstract::dpoint](#)< [dpoint_type](#) > &dp, const [weight_type](#) &weight)

Set the weight of a point if it exists. Otherwise create a new entry.

– dp The point.

– weight The weight of the point.

Static Public Member Functions

- std::string [name](#) ()

Return the name of the type.

Protected Member Functions

- [exact_type](#) & [add_dp](#) (const [abstract::dpoint](#)< [dpoint_type](#) > &dp)

Add a new point with a weight of 1.

– dp The point.

- [w_window](#) ()

Do nothing, used only by sub-classes.

Friends

- class [struct_elt](#)< [exact_type](#) >

7.377.1 Detailed Description

template<class Exact> struct oln::abstract::w_window< Exact >

Weight Window.

A [w_window](#) is a set of points associated with a weight. This class defines how to deal with.

Definition at line 62 of file [w_window.hh](#).

7.377.2 Member Typedef Documentation

7.377.2.1 **template<class Exact> typedef struct_elt_traits<Exact>::dpoint_type**
oln::abstract::w_window< Exact >::dpoint_type

The associate image's type of dpoint (move point).

Warning:

Prefer the macros `oln_dpoint_type(Pointable)` and `oln_dpoint_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented from [oln::abstract::struct_elt< Exact >](#).

Reimplemented in [oln::abstract::w_windownd< Exact >](#), [oln::w_window1d< T >](#), [oln::w_window2d< T >](#), [oln::w_window3d< T >](#), [oln::abstract::w_windownd< w_window2d< T2 > >](#), [oln::abstract::w_windownd< w_window2d< T > >](#), [oln::abstract::w_windownd< w_window1d< T > >](#), [oln::abstract::w_windownd< w_window3d< T > >](#), [oln::abstract::window_base< w_window< w_window1d< T > >](#), [w_window1d< T > >](#), [oln::abstract::window_base< w_window< Exact >](#), [Exact >](#), [oln::abstract::window_base< w_window< w_window2d< T2 > >](#), [w_window2d< T2 > >](#), [oln::abstract::window_base< w_window< w_window2d< T > >](#), [w_window2d< T > >](#), [oln::abstract::window_base< w_window< w_window3d< T > >](#), [w_window3d< T > >](#), and [oln::w_window2d< T2 >](#).

Definition at line 72 of file [w_window.hh](#).

7.377.3 Member Function Documentation

7.377.3.1 **template<class Exact> exact_type& oln::abstract::w_window< Exact >::add (const**
abstract::dpoint< dpoint_type > & dp, const weight_type & w = 1) [inline]

Add a point (with weight) to the [w_window](#).

Add a new member to the [w_window](#).

Todo

FIXME: Add dpoint with default weight (multiplication neutral element).

Definition at line 94 of file [w_window.hh](#).

Referenced by [oln::abstract::w_window< w_window3d< T > >::add_dp\(\)](#), and [oln::abstract::w_windownd< w_window3d< T > >::set_\(\)](#).

```

95         {
96             return this->exact().add_(dp.exact(), w);
97         }

```

7.377.3.2 `template<class Exact> exact_type& oln::abstract::w_window< Exact >::add_dp (const abstract::dpoint< dpoint_type > & dp)` [inline, protected]

Add a new point with a weight of 1.

- dp The point.

Todo

FIXME: Add dpoint with default weight (multiplication neutral element).

Definition at line 134 of file w_window.hh.

```
135     {
136         return this->add(dp.exact(), 1);
137     }
```

7.377.3.3 `template<class Exact> const weight_type& oln::abstract::w_window< Exact >::set (const abstract::dpoint< dpoint_type > & dp, const weight_type & weight)` [inline]

Set the weight of a point if it exists. Otherwise create a new entry.

- dp The point.
- weight The weight of the point.

Returns:

The weight of the point.

Definition at line 118 of file w_window.hh.

```
120     {
121         return this->exact().set_(dp.exact(), weight);
122     }
```

7.377.3.4 `template<class Exact> weight_type oln::abstract::w_window< Exact >::w (unsigned i)` const [inline]

Get the weight of a point.

- i The nth point.

Returns:

The weight of the point.

Definition at line 105 of file w_window.hh.

Referenced by oln::convol::slow::convolve().

```
106     {
107         return this->exact().get_weight(i);
108     }
```

The documentation for this struct was generated from the following file:

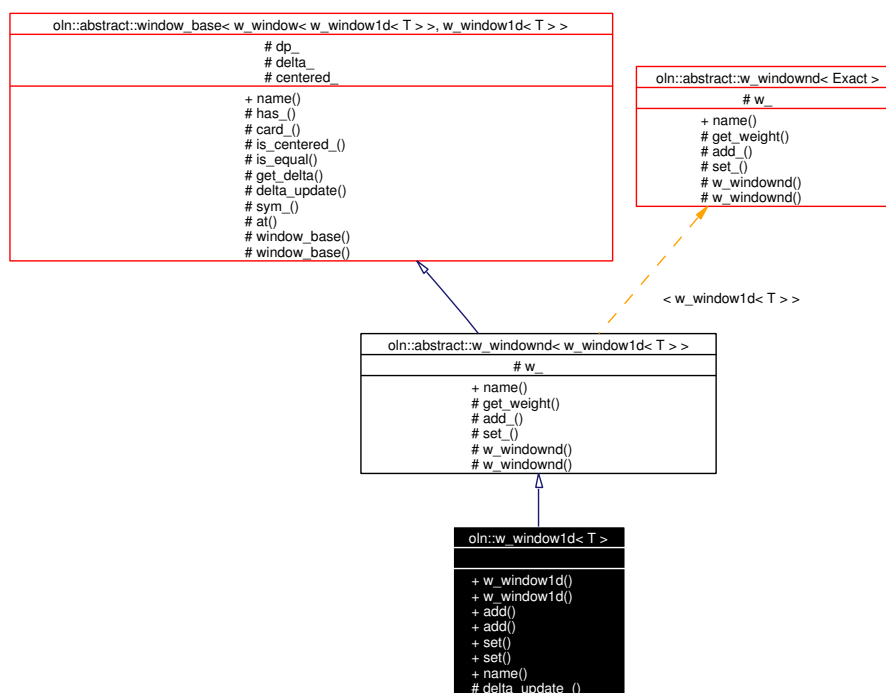
- w_window.hh

7.378 oln::w_window1d< T > Class Template Reference

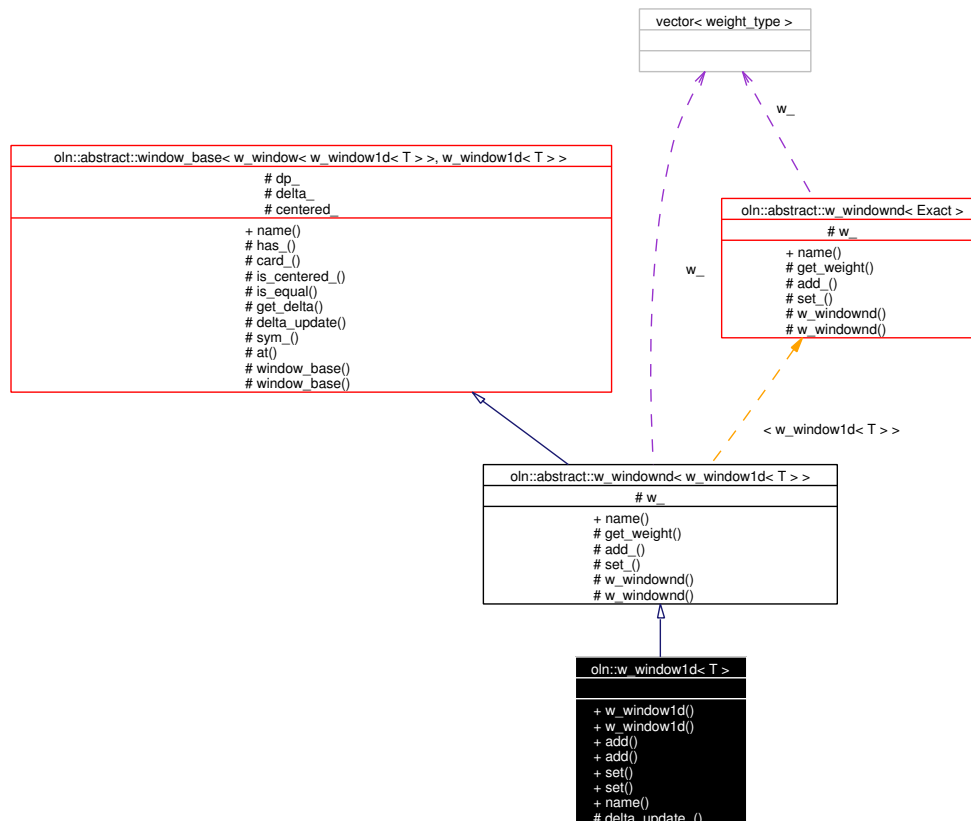
Window 1 dimension with weight.

```
#include <w_window1d.hh>
```

Inheritance diagram for oln::w_window1d< T >:



Collaboration diagram for oln::w_window1d< T >:



Public Types

- typedef `w_window1d< T > self_type`
The self type.
- typedef struct_elt_traits< `self_type` >::iter_type iter_type
The associate image's type of iterator.
- typedef struct_elt_traits< `self_type` >::neighb_type neighb_type
Type of neighbor.
- typedef struct_elt_traits< `self_type` >::dpoint_type dpoint_type
The associate image's type of dpoint (move point).
- typedef struct_elt_traits< `self_type` >::weight_type weight_type
Type of weight.

Public Member Functions

- `w_window1d()`
Construct a w_window of 1 dimension.

- `w_window1d` (unsigned size)
Construct a `w_window` of 1 dimension.
– `size` The number of element.
- `w_window1d< T > & add` (const `dpoint_type` &dp, const `weight_type` &w)
Add a `dpoint` (move point) to the `w_window`.
– `dp` The new point.
– `w` The weight of the new point.
- `w_window1d< T > & add` (`coord` col, const `weight_type` &weight)
Add a point by coordinates to the `w_window`.
– `col` The coordinate of the new point (1 dimension).
– `weight` The weight of the new point.
- `const weight_type & set` (const `dpoint_type` &dp, const `weight_type` &weight)
Set the weight of a point.
– `dp` The point to set the weight.
– `weight` The weight of the point.
- `const weight_type & set` (`coord` col, const `weight_type` &weight)
Set the weight of a point by coordinates.
– `col` The coordinates of the point.
– `weight` The weight of the point.

Static Public Member Functions

- `std::string name` ()
Return the name of the type.

Protected Member Functions

- `coord delta_update_` (const `dpoint_type` &dp)
Update delta.
– `dp` a move point.

Friends

- `class abstract::window_base< abstract::w_window< w_window1d >, w_window1d >`

7.378.1 Detailed Description

template<class T> class oln::w_window1d< T >

Window 1 dimension with weight.

A window is a set of points. This class defines how to deal with. These points have 1 dimension.

Definition at line 67 of file w_window1d.hh.

7.378.2 Member Typedef Documentation

7.378.2.1 template<class T> typedef struct_elt_traits< self_type >::dpoint_type oln::w_window1d< T >::dpoint_type

The associate image's type of dpoint (move point).

Warning:

Prefer the macros oln_dpoint_type(Pointable) and oln_dpoint_type_(Pointable) (the same without the 'typename' keyword)

Reimplemented from [oln::abstract::w_windownd< w_window1d< T > >](#).

Definition at line 92 of file w_window1d.hh.

Referenced by oln::w_window1d< T >::add(), and oln::w_window1d< T >::set().

7.378.2.2 template<class T> typedef struct_elt_traits< self_type >::iter_type oln::w_window1d< T >::iter_type

The associate image's type of iterator.

Warning:

Prefer the macros oln_iter_type(Iterable) and oln_iter_type_(Iterable) (the same without the 'type-name' keyword)

Definition at line 82 of file w_window1d.hh.

7.378.3 Member Function Documentation

7.378.3.1 template<class T> w_window1d<T>& oln::w_window1d< T >::add (coord col, const weight_type & weight) [inline]

Add a point by coordinates to the w_window.

- col The coordinate of the new point (1 dimension).
- weight The weight of the new point.

Add a new member by its coordinates to the w_window. The coordinates are only the column number because the w_window is of 1 dimension.

Definition at line 136 of file w_window1d.hh.

References oln::w_window1d< T >::add(), oln::coord, and oln::w_window1d< T >::dpoint_type.

```

137     {
138         return add(dpoint_type(col), weight);
139     }

```

7.378.3.2 `template<class T> w_window1d<T>& oln::w_window1d< T >::add (const dpoint_type & dp, const weight_type & w) [inline]`

Add a dpoint (move point) to the w_window.

- dp The new point.
- w The weight of the new point.

Add a new member to the w_window. This point must be of 1 dimension.

Definition at line 121 of file w_window1d.hh.

Referenced by oln::w_window1d< T >::add(), and oln::mk_w_win_from_win().

```

122     {
123         return this->exact().add_(dp, w);
124     }

```

7.378.3.3 `template<class T> coord oln::w_window1d< T >::delta_update_ (const dpoint_type & dp) [inline, protected]`

Update delta.

- dp a move point.

Returns:

Delta.

If the point is the biggest element of the w_window, then this point is assigned to delta.

Definition at line 184 of file w_window1d.hh.

References oln::abstract::window_base< w_window< w_window1d< T > >, w_window1d< T >::delta_.

```

185     {
186         delta_(abs(dp.col()));
187         return this->delta_;
188     }

```

7.378.3.4 `template<class T> const weight_type& oln::w_window1d< T >::set (coord col, const weight_type & weight) [inline]`

Set the weight of a point by coordinates.

- col The coordinates of the point.
- weight The weight of the point.

The coordinates are only the column number because the w_window is of 1 dimension.

Definition at line 161 of file w_window1d.hh.

References `oln::coord`, `oln::w_window1d< T >::dpoint_type`, and `oln::w_window1d< T >::set()`.

```
162     {  
163         return set(dpoint_type(col), weight);  
164     }
```

The documentation for this class was generated from the following file:

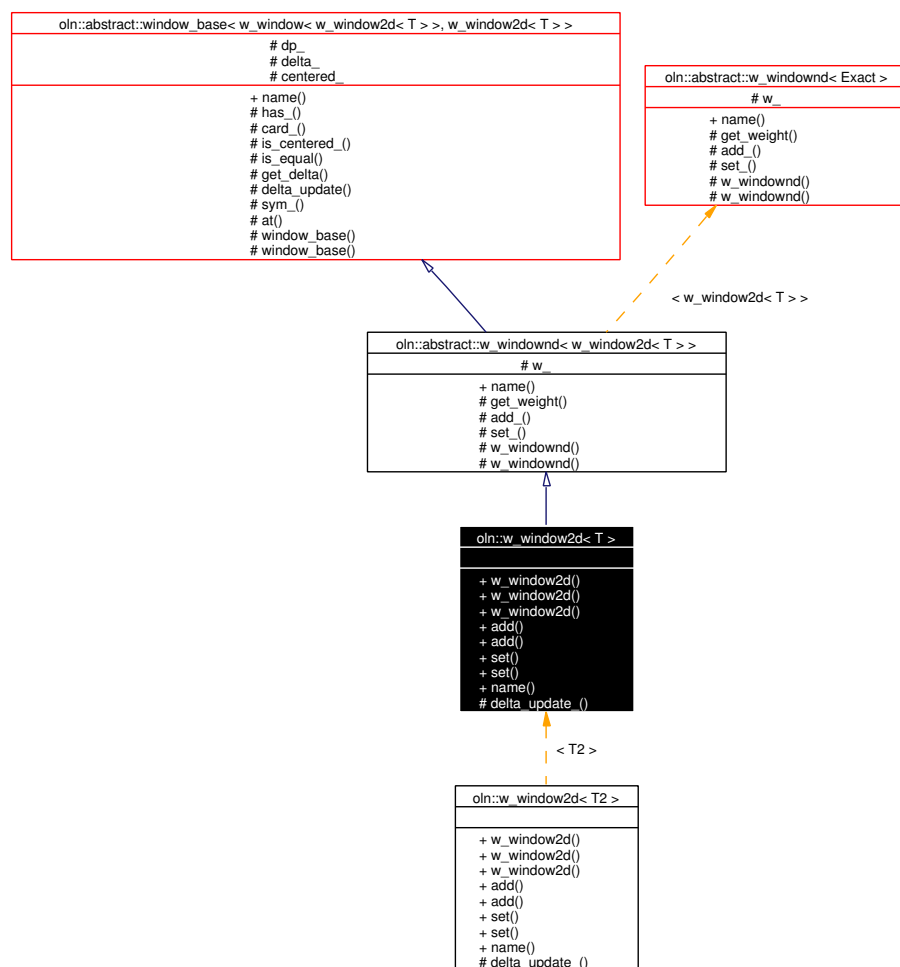
- w_window1d.hh

7.379 oln::w_window2d< T > Class Template Reference

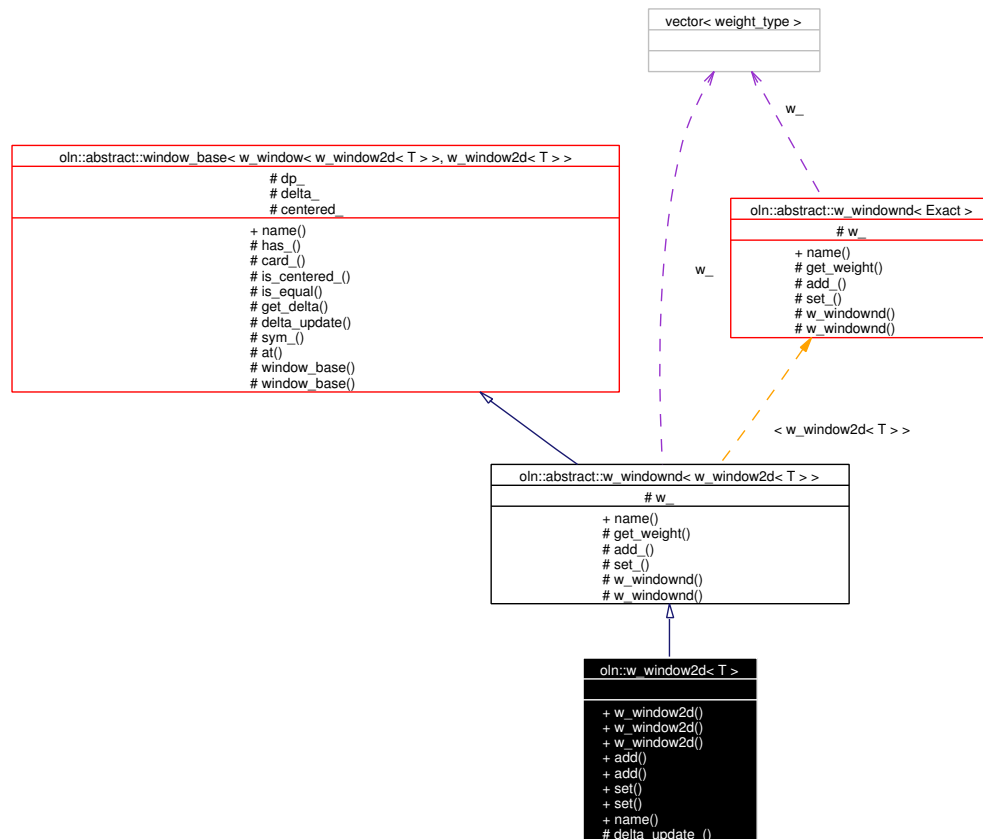
Window 2 dimensions with weight.

```
#include <w_window2d.hh>
```

Inheritance diagram for oln::w_window2d< T >:



Collaboration diagram for oln::w_window2d< T >:



Public Types

- typedef `w_window2d< T > self_type`
The self type.
- typedef struct_elt_traits< `self_type` >::iter_type iter_type
The associate image's type of iterator.
- typedef struct_elt_traits< `self_type` >::neighb_type neighb_type
Type of neighbor.
- typedef struct_elt_traits< `self_type` >::dpoint_type dpoint_type
The associate image's type of dpoint (move point).
- typedef struct_elt_traits< `self_type` >::weight_type weight_type
Type of weight.

Public Member Functions

- `w_window2d()`
Construct a w_window of 2 dimensions.

- `w_window2d` (unsigned size)
Construct a `w_window` of 2 dimensions.
– `size` The number of element.
- `template<class I, class T2> w_window2d` (const `mlc::array2d< I, T2 > &arr`)
Construct a `w_window` of 2 dimensions from an array of elements.
– `arr` The array of elements.
- `w_window2d< T > & add` (`coord` row, `coord` col, const `weight_type` &weight)
Add a point by coordinates to the `w_window`.
– `row` The coordinate (row) of the new point (2 dimensions).
– `col` The coordinate (col) of the new point (2 dimensions).
– `weight` The weight of the new point.
- `w_window2d< T > & add` (const `dpoint_type` &dp, const `weight_type` &w)
Add a `dpoint` (move point) to the `w_window`.
– `dp` The new point.
– `w` The weight of the new point.
- const `weight_type` & `set` (const `dpoint_type` &dp, const `weight_type` &weight)
Set the weight of a point.
– `dp` The point to set the weight.
– `weight` The weight of the point.
- const `weight_type` & `set` (`coord` row, `coord` col, const `weight_type` &weight)
Set the weight of a point by coordinates.
– `row` The coordinates (row) of the point.
– `col` The coordinates (col) of the point.
– `weight` The weight of the point.

Static Public Member Functions

- `std::string name` ()
Return the name of the type.

Protected Member Functions

- `coord delta_update_` (const `dpoint_type` &dp)
Update delta.
– `dp` a move point.

Friends

- class `abstract::window_base< abstract::w_window< w_window2d >, w_window2d >`

7.379.1 Detailed Description

template<class T> class oln::w_window2d< T >

Window 2 dimensions with weight.

A window is a set of points. This class defines how to deal with. These points have 2 dimensions.

Definition at line 68 of file w_window2d.hh.

7.379.2 Member Typedef Documentation

7.379.2.1 **template<class T> typedef struct_elt_traits< [self_type](#) >::dpoint_type**
[oln::w_window2d< T >::dpoint_type](#)

The associate image's type of dpoint (move point).

Warning:

Prefer the macros `oln_dpoint_type(Pointable)` and `oln_dpoint_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented from [oln::abstract::w_windownd< w_window2d< T > >](#).

Definition at line 93 of file w_window2d.hh.

Referenced by `oln::w_window2d< T2 >::add()`, and `oln::w_window2d< T2 >::set()`.

7.379.2.2 **template<class T> typedef struct_elt_traits< [self_type](#) >::iter_type [oln::w_window2d< T >::iter_type](#)**

The associate image's type of iterator.

Warning:

Prefer the macros `oln_iter_type(Iterable)` and `oln_iter_type_(Iterable)` (the same without the 'type-name' keyword)

Definition at line 83 of file w_window2d.hh.

7.379.3 Constructor & Destructor Documentation

7.379.3.1 **template<class T> template<class I, class T2> [oln::w_window2d< T >::w_window2d](#)**
(const `mlc::array2d< I, T2 > &arr`) [inline]

Construct a w_window of 2 dimensions from an array of elements.

- `arr` The array of elements.

Todo

FIXME: this constructor is not in [w_window1d.hh](#) nor [w_window3d.hh](#). Is it really useful ? This constructor is used to build a chamfer distance.

Definition at line 121 of file w_window2d.hh.


```

121                                     :
122     super_type(I::card)
123     {
124         unsigned i = 0;
125         for (coord row = -I::center_row; row <= I::nrows - I::center_row - 1; ++row)
126             for (coord col = -I::center_col; col <= I::ncols - I::center_col - 1; ++col)
127                 add(row, col, arr[i++]);
128     }

```

7.379.4 Member Function Documentation

7.379.4.1 `template<class T> w_window2d<T>& oln::w_window2d< T >::add (const dpoint_type & dp, const weight_type & w) [inline]`

Add a dpoint (move point) to the w_window.

- dp The new point.
- w The weight of the new point.

Add a new member to the w_window. This point must be of 2 dimensions.

Definition at line 155 of file w_window2d.hh.

```

156     {
157         return this->exact().add_(dp, w);
158     }

```

7.379.4.2 `template<class T> w_window2d<T>& oln::w_window2d< T >::add (coord row, coord col, const weight_type & weight) [inline]`

Add a point by coordinates to the w_window.

- row The coordinate (row) of the new point (2 dimensions).
- col The coordinate (col) of the new point (2 dimensions).
- weight The weight of the new point.

Add a new member by its coordinates to the w_window. The coordinates are only the column number because the w_window is of 2 dimensions.

Definition at line 141 of file w_window2d.hh.

Referenced by `oln::w_window2d< T2 >::add()`, `oln::mk_w_win_from_win()`, and `oln::w_window2d< T2 >::w_window2d()`.

```

142     {
143         return add(dpoint_type(row, col), weight);
144     }

```

7.379.4.3 `template<class T> coord oln::w_window2d< T >::delta_update_ (const dpoint_type & dp)` [inline, protected]

Update delta.

- dp a move point.

Returns:

Delta.

If the point is the biggest element of the w_window, then this point is assigned to delta.

Definition at line 204 of file w_window2d.hh.

```
205     {  
206         delta_(abs(dp.row()));  
207         delta_(abs(dp.col()));  
208         return this->delta_;  
209     }
```

7.379.4.4 `template<class T> const weight_type& oln::w_window2d< T >::set (coord row, coord col, const weight_type & weight)` [inline]

Set the weight of a point by coordinates.

- row The coordinates (row) of the point.
- col The coordinates (col) of the point.
- weight The weight of the point.

The coordinates are only the column number because the w_window is of 2 dimensions.

Definition at line 181 of file w_window2d.hh.

```
182     {  
183         return set(dpoint_type(row, col), weight);  
184     }
```

The documentation for this class was generated from the following file:

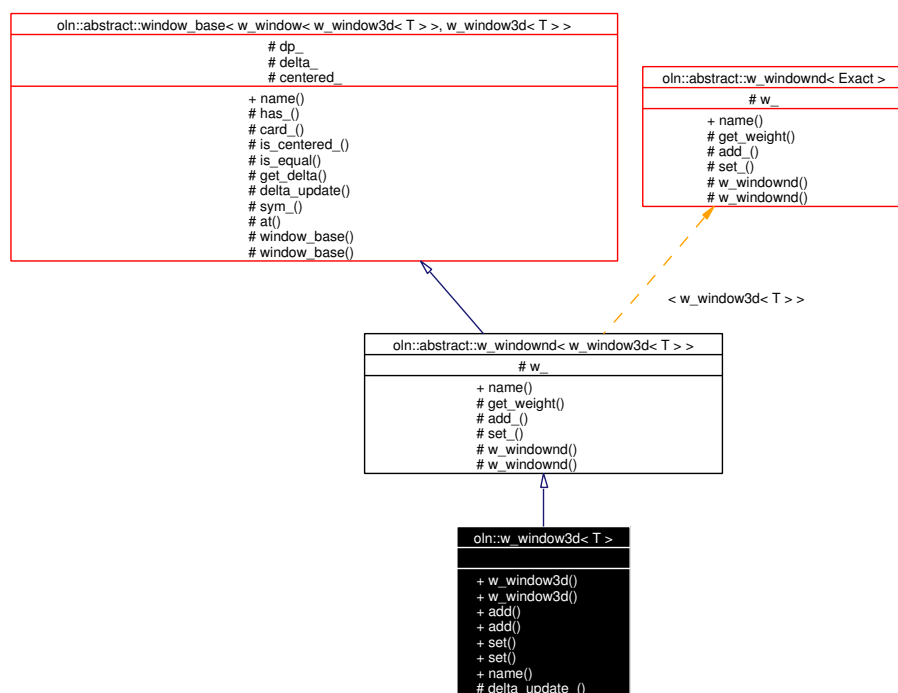
- w_window2d.hh

7.380 oln::w_window3d< T > Class Template Reference

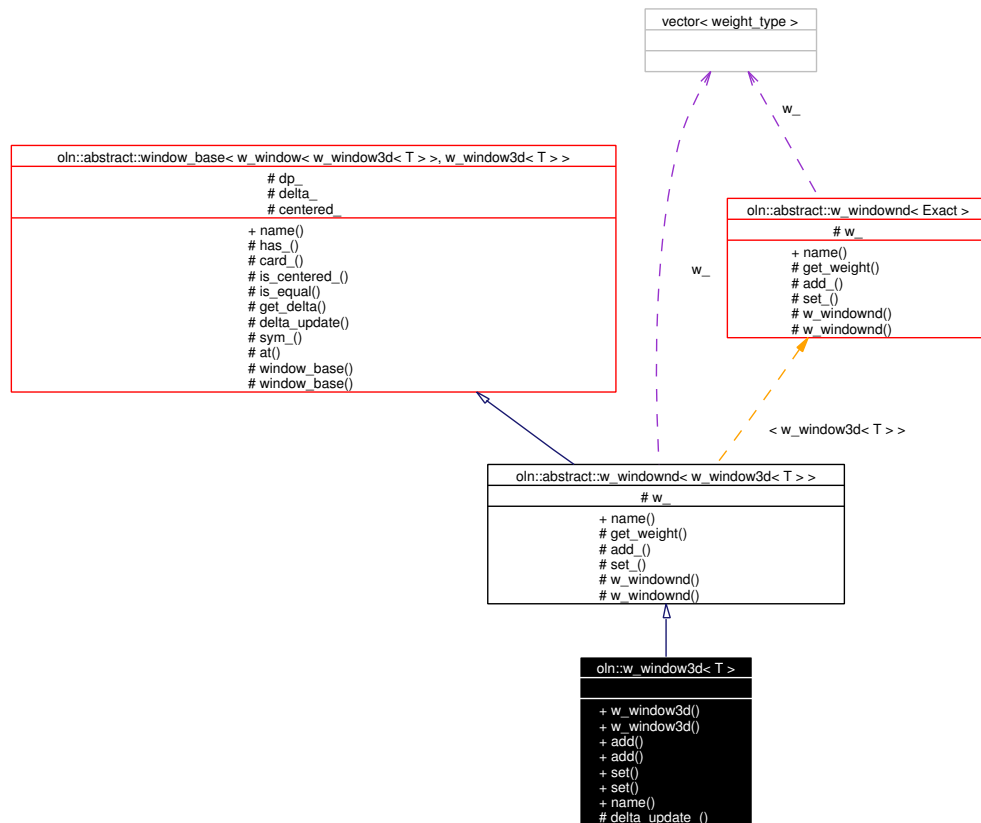
Window 3 dimensions with weight.

```
#include <w_window3d.hh>
```

Inheritance diagram for oln::w_window3d< T >:



Collaboration diagram for oln::w_window3d< T >:



Public Types

- typedef `w_window3d< T > self_type`
The *self* type.
- typedef struct_elt_traits< `self_type` >::iter_type iter_type
The associate image's type of iterator.
- typedef struct_elt_traits< `self_type` >::neighb_type neighb_type
Type of neighbor.
- typedef struct_elt_traits< `self_type` >::dpoint_type dpoint_type
The associate image's type of dpoint (move point).
- typedef struct_elt_traits< `self_type` >::weight_type weight_type
Type of weight.

Public Member Functions

- `w_window3d()`
Construct a *w_window* of 3 dimensions.

- `w_window3d` (unsigned size)
Construct a `w_window` of 3 dimensions.
– `size` The number of element.
- `w_window3d< T > & add` (const `dpoint_type` &dp, const `weight_type` &w)
Add a `dpoint` (move point) to the `w_window`.
– `dp` The new point.
– `w` The weight of the new point.
- `w_window3d< T > & add` (`coord` slice, `coord` row, `coord` col, const `weight_type` &weight)
Add a point by coordinates to the `w_window`.
– `slice` The coordinate (slice) of the new point (3 dimensions).
– `row` The coordinate (row) of the new point (3 dimensions).
– `col` The coordinate (col) of the new point (3 dimensions).
– `weight` The weight of the new point.
- `const weight_type & set` (const `dpoint_type` &dp, const `weight_type` &weight)
Set the weight of a point.
– `dp` The point to set the weight.
– `weight` The weight of the point.
- `const weight_type & set` (`coord` slice, `coord` row, `coord` col, const `weight_type` &weight)
Set the weight of a point by coordinates.
– `slice` The coordinates (slice) of the point.
– `row` The coordinates (row) of the point.
– `col` The coordinates (col) of the point.
– `weight` The weight of the point.

Static Public Member Functions

- `std::string name` ()
Return the name of the type.

Protected Member Functions

- `coord delta_update_` (const `dpoint_type` &dp)
Update delta.
– `dp` a move point.

Friends

- `class abstract::window_base< abstract::w_window< w_window3d >, w_window3d >`

7.380.1 Detailed Description

template<class T> class oln::w_window3d< T >

Window 3 dimensions with weight.

A window is a set of points. This class defines how to deal with. These points have 3 dimensions.

Definition at line 68 of file w_window3d.hh.

7.380.2 Member Typedef Documentation

**7.380.2.1 template<class T> typedef struct_elt_traits< [self_type](#) >::dpoint_type
 [oln::w_window3d< T >::dpoint_type](#)**

The associate image's type of dpoint (move point).

Warning:

Prefer the macros oln_dpoint_type(Pointable) and oln_dpoint_type_(Pointable) (the same without the 'typename' keyword)

Reimplemented from [oln::abstract::w_windownd< w_window3d< T > >](#).

Definition at line 94 of file w_window3d.hh.

Referenced by oln::w_window3d< T >::add(), and oln::w_window3d< T >::set().

7.380.2.2 template<class T> typedef struct_elt_traits< [self_type](#) >::iter_type [oln::w_window3d< T >::iter_type](#)

The associate image's type of iterator.

Warning:

Prefer the macros oln_iter_type(Iterable) and oln_iter_type_(Iterable) (the same without the 'type-name' keyword)

Definition at line 83 of file w_window3d.hh.

7.380.3 Member Function Documentation

**7.380.3.1 template<class T> [w_window3d<T>& oln::w_window3d< T >::add](#) ([coord slice](#),
 [coord row](#), [coord col](#), const [weight_type](#) & *weight*) [inline]**

Add a point by coordinates to the w_window.

- slice The coordinate (slice) of the new point (3 dimensions).
- row The coordinate (row) of the new point (3 dimensions).
- col The coordinate (col) of the new point (3 dimensions).
- weight The weight of the new point.

Add a new member by its coordinates to the w_window. The coordinates are only the column number because the w_window is of 3 dimensions.

Definition at line 140 of file w_window3d.hh.

References oln::w_window3d< T >::add(), oln::coord, and oln::w_window3d< T >::dpoint_type.

```

141     {
142         return add(dpoint_type(slice, row, col), weight);
143     }
```

7.380.3.2 template<class T> w_window3d<T>& oln::w_window3d< T >::add (const dpoint_type & dp, const weight_type & w) [inline]

Add a dpoint (move point) to the w_window.

- dp The new point.
- w The weight of the new point.

Add a new member to the w_window. This point must be of 3 dimensions.

Definition at line 123 of file w_window3d.hh.

Referenced by oln::w_window3d< T >::add(), and oln::mk_w_win_from_win().

```

124     {
125         return this->exact().add_(dp, w);
126     }
```

7.380.3.3 template<class T> coord oln::w_window3d< T >::delta_update_ (const dpoint_type & dp) [inline, protected]

Update delta.

- dp a move point.

Returns:

Delta.

If the point is the biggest element of the w_window, then this point is assigned to delta.

Definition at line 190 of file w_window3d.hh.

References oln::abstract::window_base< w_window< w_window3d< T > >, w_window3d< T >::delta_.

```

191     {
192         delta_(abs(dp.slice()));
193         delta_(abs(dp.row()));
194         delta_(abs(dp.col()));
195         return this->delta_;
196     }
```

7.380.3.4 `template<class T> const weight_type& oln::w_window3d< T >::set (coord slice, coord row, coord col, const weight_type & weight) [inline]`

Set the weight of a point by coordinates.

- slice The coordinates (slice) of the point.
- row The coordinates (row) of the point.
- col The coordinates (col) of the point.
- weight The weight of the point.

The coordinates are only the column number because the w_window is of 3 dimensions.

Definition at line 167 of file w_window3d.hh.

References `oln::coord`, `oln::w_window3d< T >::dpoint_type`, and `oln::w_window3d< T >::set()`.

```
168     {  
169         return set(dpoint_type(slice, row, col), weight);  
170     }
```

The documentation for this class was generated from the following file:

- w_window3d.hh

7.381 oln::abstract::w_windownd< Exact > Struct Template Reference

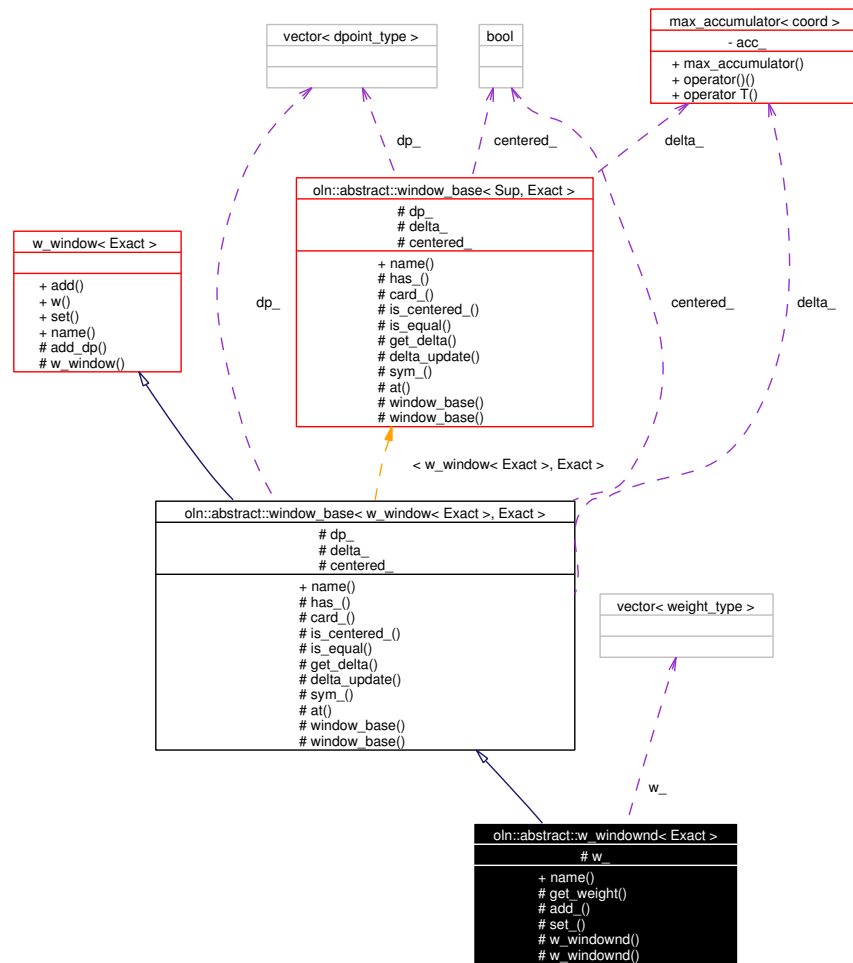
Weight Window N dimensions.

```
#include <w_windownd.hh>
```

Inheritance diagram for oln::abstract::w_windownd< Exact >:



Collaboration diagram for oln::abstract::w_windownd< Exact >:



Public Types

- typedef [window_base](#)< [abstract::w_window](#)< Exact >, Exact > [super_type](#)
Set the super type.
- typedef [w_windownd](#)< Exact > [self_type](#)
Set the self type.
- typedef Exact [exact_type](#)
Set the exact type.
- typedef struct_elt_traits< Exact >::weight_type [weight_type](#)
Set the type of weight.
- typedef struct_elt_traits< Exact >::dpoint_type [dpoint_type](#)
The associate image's type of dpoint (move point).

Static Public Member Functions

- `std::string name ()`
Return the name of the type.

Protected Member Functions

- `weight_type get_weight (unsigned i) const`
Get the weight of the nth point of the window.
– *i* The nth point.
- `exact_type & add_ (const dpoint_type &dp, const weight_type &w)`
Add a point (with weight) to the window.
– *dp* The new point.
– *w* The weight of this new point.
- `const weight_type & set_ (const dpoint_type &dp, const weight_type &w)`
Set the weight of a point if it exists. Otherwise create a new entry.
– *dp* The point.
– *weight* The weight of the point.
- `w_windownd ()`
Construct a w_window.
- `w_windownd (unsigned size)`
Construct a w_window of 'size' elements.
– *size* The number of elements to reserve for the window.

Protected Attributes

- `std::vector< weight_type > w_`
List of point's weight.

Friends

- `class w_window< exact_type >`

7.381.1 Detailed Description

`template<class Exact> struct oln::abstract::w_windownd< Exact >`

Weight Window N dimensions.

A `w_window` is a set of points associated with a weight. This class defines how to deal with. These points have N dimensions.

Definition at line 59 of file `w_windownd.hh`.

7.381.2 Member Typedef Documentation

7.381.2.1 `template<class Exact> typedef struct_elt_traits<Exact>::dpoint_type oln::abstract::w_windownd< Exact >::dpoint_type`

The associate image's type of dpoint (move point).

Warning:

Prefer the macros `oln_dpoint_type(Pointable)` and `oln_dpoint_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented from `oln::abstract::window_base< w_window< Exact >, Exact >.`

Reimplemented in `oln::w_window1d< T >`, `oln::w_window2d< T >`, `oln::w_window3d< T >`, and `oln::w_window2d< T2 >.`

Definition at line 73 of file `w_windownd.hh`.

7.381.3 Member Function Documentation

7.381.3.1 `template<class Exact> exact_type& oln::abstract::w_windownd< Exact >::add_ (const dpoint_type & dp, const weight_type & w) [inline, protected]`

Add a point (with weight) to the window.

- dp The new point.
- w The weight of this new point.

Precondition:

`!has(dp).`

Add a new member to the window (ignored if ist weight is 0).

Definition at line 108 of file `w_windownd.hh`.

```

109     {
110         precondition(! has_(dp));
111         if (w == 0)           // Don't add 0 weighted entries
112             return this->exact();
113         if (dp.is_centered())
114             this->centered_ = true;
115         this->dp_.push_back(dp);
116         delta_update(dp);
117         w_.push_back(w);
118         return this->exact();
119     }
```

7.381.3.2 `template<class Exact> weight_type oln::abstract::w_windownd< Exact >::get_weight (unsigned i) const [inline, protected]`

Get the weight of the nth point of the window.

- i The nth point.

Returns:

The weight of this point.

Precondition:

$i < \text{card}()$.

Definition at line 93 of file w_windownd.hh.

```
94     {
95         precondition(i < this->card());
96         return w_[i];
97     }
```

7.381.3.3 `template<class Exact> const weight_type& oln::abstract::w_windownd< Exact >::set_`
(const dpoint_type & dp, const weight_type & w) [inline, protected]

Set the weight of a point if it exists. Otherwise create a new entry.

- dp The point.
- weight The weight of the point.

Returns:

The weight of the point.

Definition at line 129 of file w_windownd.hh.

```
130     {
131         // if the dp exists, return a ref to the existing entry
132         for (unsigned i = 0; i < this->card_(); ++i)
133             if (this->dp_[i] == dp)
134                 {
135                     w_[i] = w;
136                     return w_[i];
137                 }
138
139         // otherwise, create new entry
140         add(dp, w);
141         return w_.back();
142     }
```

The documentation for this struct was generated from the following file:

- w_windownd.hh

7.382 oln::internal::wavelet_coeffs_< T, N, Self > Struct Template Reference

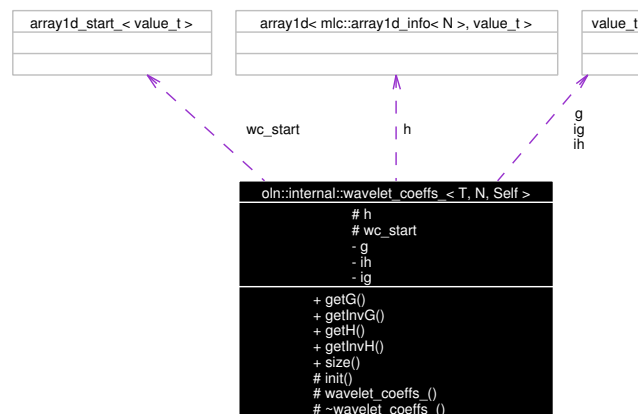
Wavelet coefficient data structure.

```
#include <dwt.hh>
```

Inheritance diagram for oln::internal::wavelet_coeffs_< T, N, Self >:



Collaboration diagram for oln::internal::wavelet_coeffs_< T, N, Self >:



Public Types

- typedef T [value_t](#)
Type of data used.
- typedef Self [self_t](#)
Self type.

Public Member Functions

- const [value_t](#) [getG](#) (unsigned i) const
Accessor to ith element of g.
- const [value_t](#) [getInvG](#) (unsigned i) const
Accessor to ith element of ig.
- const [value_t](#) [getH](#) (unsigned i) const
Accessor to ith element of h.

- const [value_t](#) [getInvH](#) (unsigned i) const
Accessor to ith element of ih.
- const unsigned [size](#) () const
Give the size of the arrays.

Protected Member Functions

- void [init](#) ()
Initialization method.
- [wavelet_coeffs_](#) ()
Constructor.
- [~wavelet_coeffs_](#) ()
Destructor.

Protected Attributes

- [mlc::array1d](#)< [mlc::array1d_info](#)< N >, [value_t](#) > [h](#)
array of value_t.
- [mlc::internal::array1d_start_](#)< [value_t](#) > [wc_start](#)
First coefficient.

7.382.1 Detailed Description

template<class T, unsigned N, class Self> struct oln::internal::wavelet_coeffs_< T, N, Self >

Wavelet coefficient data structure.

Parameters:

- T* Coefficients data type.
- N* Number of coefficients.
- Self* Self type.

Definition at line 87 of file dwt.hh.

7.382.2 Constructor & Destructor Documentation

7.382.2.1 **template<class T, unsigned N, class Self> [oln::internal::wavelet_coeffs_](#)< T, N, Self >::[~wavelet_coeffs_](#) ()** [inline, protected]

Destructor.

Its aim is to check N is pair.

Definition at line 134 of file dwt.hh.

```
135     {
136         mlc::is_false<N % 2>::ensure();
137     }
```

7.382.3 Member Function Documentation

7.382.3.1 `template<class T, unsigned N, class Self> void oln::internal::wavelet_coeffs_< T, N, Self >::init ()` [inline, protected]

Initialization method.

Fill coefficients.

Definition at line 112 of file dwt.hh.

```
113     {
114         for (unsigned i = 0; i < size_; i += 2) {
115             g[i] = -h[size_ - 1 - i];
116             g[i + 1] = h[size_ - 2 - i];
117             ig[size_ - 1 - i] = g[i + 1];
118             ig[size_ - 2 - i] = h[i + 1];
119             ih[size_ - 1 - i] = g[i];
120             ih[size_ - 2 - i] = h[i];
121         }
122     }
```

The documentation for this struct was generated from the following file:

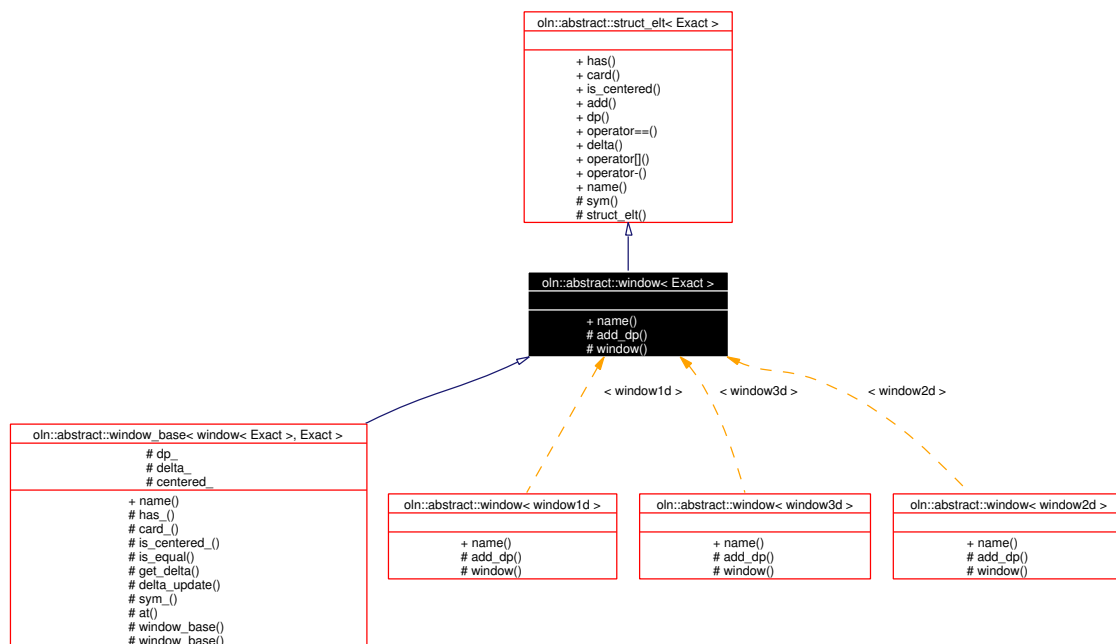
- dwt.hh

7.383 oln::abstract::window< Exact > Struct Template Reference

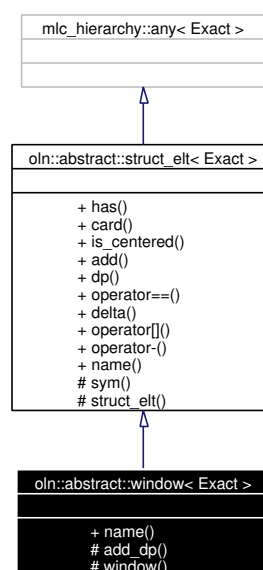
Window.

```
#include <window.hh>
```

Inheritance diagram for oln::abstract::window< Exact >:



Collaboration diagram for oln::abstract::window< Exact >:



Public Types

- typedef [struct_elt](#)< Exact > [super_type](#)
Set super type.
- typedef Exact [exact_type](#)
Set exact type.
- typedef struct_elt_traits< Exact >::[dpoint_type](#) [dpoint_type](#)
The associate image's type of dpoint (move point).

Static Public Member Functions

- std::string [name](#) ()
Return the name of the type.

Protected Member Functions

- [exact_type](#) & [add_dp](#) (const [abstract::dpoint](#)< [dpoint_type](#) > &dp)
Add a point to the window.
- [window](#) ()
Do nothing, used only by sub-classes.

Friends

- class [struct_elt](#)< [exact_type](#) >

7.383.1 Detailed Description

`template<class Exact> struct oln::abstract::window< Exact >`

Window.

A window is a set of points and this class defines how to deal with.

Definition at line 62 of file `window.hh`.

7.383.2 Member Typedef Documentation

7.383.2.1 `template<class Exact> typedef struct_elt_traits<Exact>::dpoint_type oln::abstract::window< Exact >::dpoint_type`

The associate image's type of dpoint (move point).

Warning:

Prefer the macros `oln_dpoint_type(Pointable)` and `oln_dpoint_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented from [oln::abstract::struct_elt< Exact >](#).

Reimplemented in [oln::abstract::windownd< Exact >](#), [oln::window1d](#), [oln::window2d](#), [oln::window3d](#), [oln::abstract::window_base< window< Exact >, Exact >](#), [oln::abstract::window_base< window< window3d >, window3d >](#), [oln::abstract::window_base< window< window2d >, window2d >](#), [oln::abstract::window_base< window< window1d >, window1d >](#), [oln::abstract::windownd< window1d >](#), [oln::abstract::windownd< window3d >](#), and [oln::abstract::windownd< window2d >](#).

Definition at line 72 of file window.hh.

7.383.3 Member Function Documentation**7.383.3.1** `template<class Exact> exact_type& oln::abstract::window< Exact >::add_dp (const abstract::dpoint< dpoint_type > & dp) [inline, protected]`

Add a point to the window.

Add a new member to the window.

Definition at line 91 of file window.hh.

```

92     {
93         return this->exact().add_(dp.exact());
94     }
```

The documentation for this struct was generated from the following file:

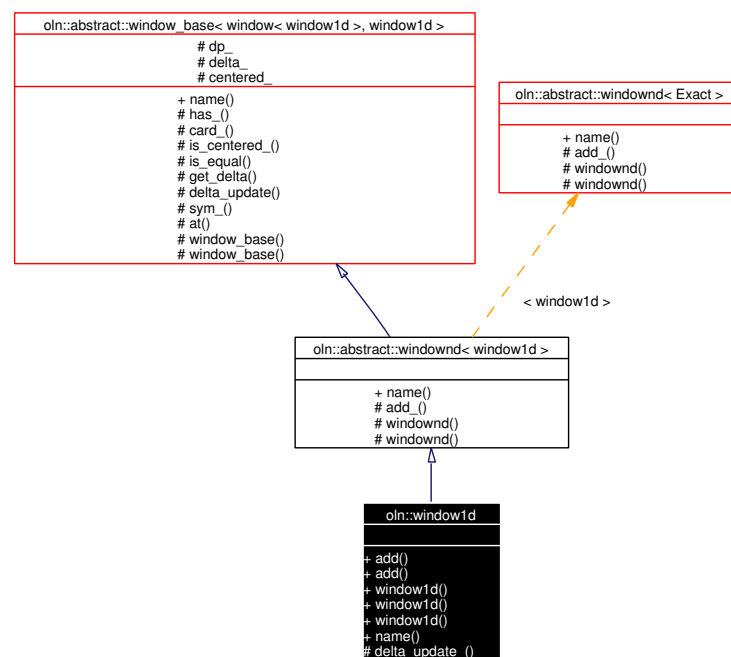
- window.hh

7.384 oln::window1d Class Reference

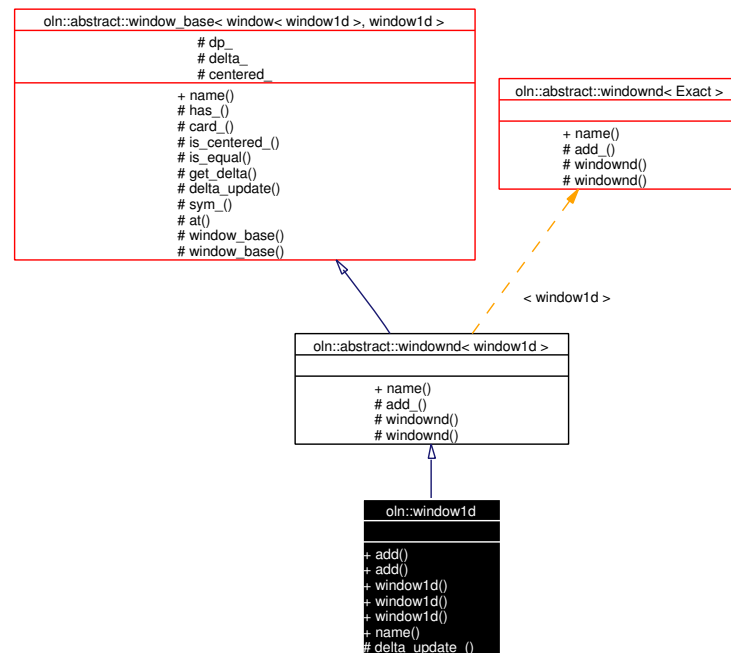
Window 1 dimension.

```
#include <window1d.hh>
```

Inheritance diagram for oln::window1d:



Collaboration diagram for oln::window1d:



Public Types

- typedef `abstract::windownd< window1d > super_type`
The *super type*.
- typedef `window1d self_type`
The *self type*.
- typedef struct_elt_traits< `self_type` >::iter_type iter_type
The *associate image's type of iterator*.
- typedef struct_elt_traits< `self_type` >::neighb_type neighb_type
Type of *neighbor*.
- typedef struct_elt_traits< `self_type` >::dpoint_type dpoint_type
The *associate image's type of dpoint (move point)*.

Public Member Functions

- `window1d & add (const dpoint_type &dp)`
Add a *dpoint (move point)* to the window.
– *dp* The new point.
- `window1d & add (coord col)`
Add a point by *coordinates* to the window.
– *col* The coordinate of the new point (1 dimension).

- `window1d ()`
Construct a window of 1 dimension.
- `window1d (unsigned size)`
Construct a window of 1 dimension.
 - *size The number of element.*
- `window1d (unsigned n, const coord crd[])`
Construct a window of 1 dimension from several points.
 - *n The number of element.*
 - *crd The coordinates of the elements.*

Static Public Member Functions

- `std::string name ()`
Return the name of the type.

Protected Member Functions

- `coord delta_update_ (const dpoint_type &dp)`
Update delta.
 - *dp a move point.*

Friends

- `class abstract::window_base< abstract::window< window1d >, window1d >`

7.384.1 Detailed Description

Window 1 dimension.

A window is a set of points. This class defines how to deal with. These points have 1 dimension.

Definition at line 63 of file `window1d.hh`.

7.384.2 Member Typedef Documentation

7.384.2.1 `typedef struct_elt_traits< self_type >::dpoint_type oln::window1d::dpoint_type`

The associate image's type of dpoint (move point).

Warning:

Prefer the macros `oln_dpoint_type(Pointable)` and `oln_dpoint_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented from [oln::abstract::windownd< window1d >](#).

Definition at line 87 of file window1d.hh.

Referenced by [add\(\)](#), and [window1d\(\)](#).

7.384.2.2 `typedef struct_elt_traits< self_type >::iter_type oln::window1d::iter_type`

The associate image's type of iterator.

Warning:

Prefer the macros `oln_iter_type(Iterable)` and `oln_iter_type_(Iterable)` (the same without the 'type-name' keyword)

Definition at line 77 of file window1d.hh.

7.384.3 Member Function Documentation

7.384.3.1 `window1d& oln::window1d::add(coord col) [inline]`

Add a point by coordinates to the window.

- `col` The coordinate of the new point (1 dimension).

Add a new member by its coordinates to the window. The coordinates are only the column number because the window is of 1 dimension.

Definition at line 113 of file window1d.hh.

References [add\(\)](#), [oln::coord](#), and [dpoint_type](#).

```

114     {
115         return this->add(dpoint_type(col));
116     }
```

7.384.3.2 `window1d& oln::window1d::add(const dpoint_type & dp) [inline]`

Add a dpoint (move point) to the window.

- `dp` The new point.

Add a new member to the neighborhood. This point must be of 1 dimension.

Definition at line 99 of file window1d.hh.

Referenced by [add\(\)](#), [oln::mk_win_from_neighb\(\)](#), [oln::mk_win_segment\(\)](#), and [window1d\(\)](#).

```

100     {
101         return this->exact().add_(dp);
102     }
```

7.384.3.3 `coord` `oln::window1d::delta_update_ (const dpoint_type & dp)` [`inline`,
`protected`]

Update delta.

- `dp` a move point.

Returns:

Delta.

If the point is the biggest element of the window. then this point is assigned to delta.

Definition at line 160 of file `window1d.hh`.

References `oln::abstract::window_base< window< window1d >, window1d >::delta_`.

```
161     {  
162         delta_(abs(dp.col()));  
163         return delta_;  
164     }
```

The documentation for this class was generated from the following file:

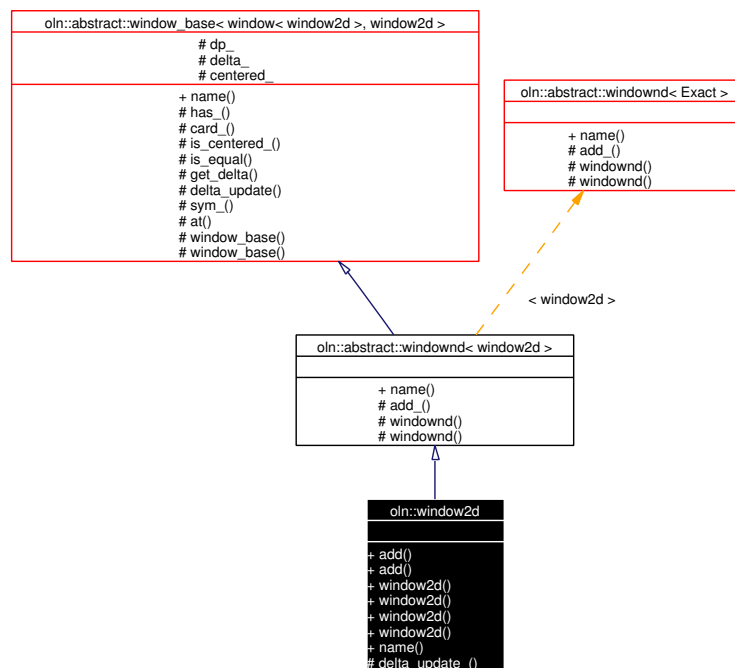
- `window1d.hh`

7.385 oln::window2d Class Reference

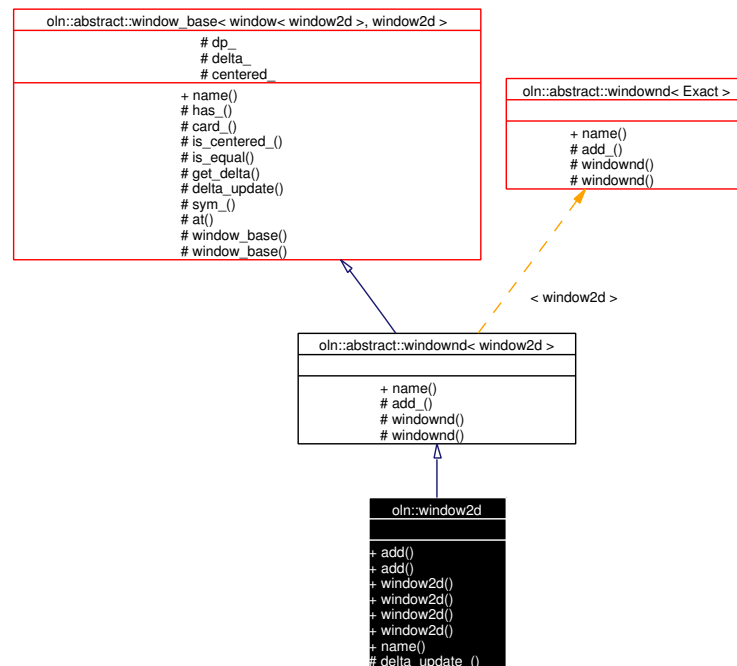
Window 2 dimensions.

```
#include <window2d.hh>
```

Inheritance diagram for oln::window2d:



Collaboration diagram for oln::window2d:



Public Types

- typedef `abstract::windownd< window2d >` `super_type`
The super type.
- typedef `window2d` `self_type`
The self type.
- typedef `struct_elt_traits< self_type >::iter_type` `iter_type`
The associate image's type of iterator.
- typedef `struct_elt_traits< self_type >::neighb_type` `neighb_type`
- typedef `struct_elt_traits< self_type >::neighb_type` `neighb_type`
Type of neighbor.
- typedef `struct_elt_traits< self_type >::dpoint_type` `dpoint_type`
The associate image's type of dpoint (move point).

Public Member Functions

- `window2d` & `add` (const `dpoint_type` &dp)
Add a dpoint (move point) to the window.
– dp The new point.
- `window2d` & `add` (coord row, coord col)
Add a point by coordinates to the window.

- *row* The coordinate (row) of the new point.
- *col* The coordinate (col) of the new point.

- [window2d](#) ()

Construct a window of 2 dimensions.

- [window2d](#) (unsigned size)

Construct a window of 2 dimensions.

- *size* The number of element.

- [window2d](#) (unsigned n, const [coord](#) crd[])

Construct a window of 2 dimensions from several points.

- *n* The number of element.
- *crd* The coordinates of the elements.

- [window2d](#) (const [io::internal::anything](#) &r)

Static Public Member Functions

- `std::string` [name](#) ()

Return the name of the type.

Protected Member Functions

- [coord](#) [delta_update_](#) (const [dpoint_type](#) &dp)

Update delta.

- *dp* a move point.

Friends

- class [abstract::window_base](#)< [abstract::window](#)< [window2d](#) >, [window2d](#) >

7.385.1 Detailed Description

Window 2 dimensions.

A window is a set of points. This class defines how to deal with. These points have 2 dimension.

Definition at line 63 of file [window2d.hh](#).

7.385.2 Member Typedef Documentation

7.385.2.1 `typedef struct_elt_traits< self_type >::dpoint_type oln::window2d::dpoint_type`

The associate image's type of dpoint (move point).

Warning:

Prefer the macros `oln_dpoint_type(Pointable)` and `oln_dpoint_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented from [oln::abstract::windownd< window2d >](#).

Definition at line 88 of file `window2d.hh`.

Referenced by `window2d()`.

7.385.2.2 `typedef struct_elt_traits< self_type >::iter_type oln::window2d::iter_type`

The associate image's type of iterator.

Warning:

Prefer the macros `oln_iter_type(Iterable)` and `oln_iter_type_(Iterable)` (the same without the 'typename' keyword)

Definition at line 77 of file `window2d.hh`.

7.385.3 Member Function Documentation

7.385.3.1 `window2d& oln::window2d::add(coord row, coord col) [inline]`

Add a point by coordinates to the window.

- row The coordinate (row) of the new point.
- col The coordinate (col) of the new point.

Add a new member by its coordinates to the window. The coordinates are the row number and the column number because the window has 2 dimensions.

Definition at line 115 of file `window2d.hh`.

References `add()`, `oln::coord`, and `oln::abstract::struct_elt< window2d >::dp()`.

```

116     {
117         dpoint_type dp(row, col);
118         return add(dp);
119     }
```

7.385.3.2 `window2d& oln::window2d::add(const dpoint_type & dp) [inline]`

Add a dpoint (move point) to the window.

- dp The new point.

Add a new member to the window. This point must be of 2 dimensions.

Definition at line 100 of file `window2d.hh`.

Referenced by `add()`, `oln::topo::tarjan::obsolete::flat_zone< I >::doit()`, `oln::mk_win_ellipse()`, `oln::mk_win_from_neighb()`, `oln::mk_win_rectangle()`, and `window2d()`.

```
101     {  
102         return this->exact().add_(dp);  
103     }
```

7.385.3.3 coord oln::window2d::delta_update_ (const dpoint_type & dp) [inline, protected]

Update delta.

- dp a move point.

Returns:

Delta.

If the point is the biggest element of the window. then this point is assigned to delta.

Definition at line 169 of file window2d.hh.

References oln::abstract::window_base< window< window2d >, window2d >::delta_.

```
170     {  
171         delta_(abs(dp.row()));  
172         delta_(abs(dp.col()));  
173         return delta_;  
174     }
```

The documentation for this class was generated from the following file:

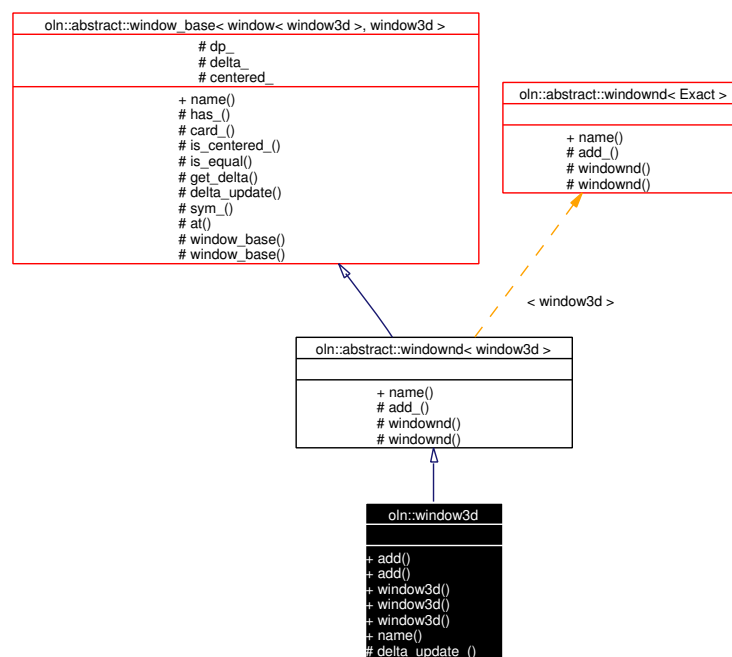
- window2d.hh

7.386 oln::window3d Class Reference

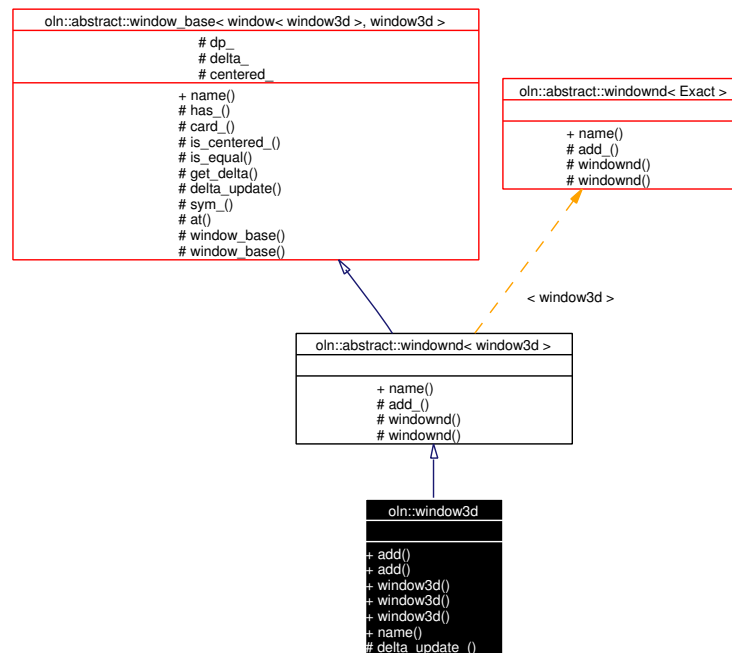
Window 3 dimensions.

```
#include <window3d.hh>
```

Inheritance diagram for oln::window3d:



Collaboration diagram for oln::window3d:



Public Types

- typedef `abstract::windownd< window3d > super_type`
The *super type*.
- typedef `window3d self_type`
The *self type*.
- typedef struct_elt_traits< `self_type` >::iter_type iter_type
The *associate image's type of iterator*.
- typedef struct_elt_traits< `self_type` >::neighb_type neighb_type
Type of *neighbor*.
- typedef struct_elt_traits< `self_type` >::dpoint_type dpoint_type
The *associate image's type of dpoint (move point)*.

Public Member Functions

- `window3d & add (const dpoint_type &dp)`
Add a *dpoint (move point)* to the window.
– *dp* The new point.
- `window3d & add (coord slice, coord row, coord col)`
Add a point by coordinates to the window.
– *slice* The coordinate (*slice*) of the new point.

- *row* The coordinate (row) of the new point.
- *col* The coordinate (col) of the new point.
- [window3d](#) ()
Construct a window of 3 dimensions.
- [window3d](#) (unsigned size)
Construct a window of 3 dimensions.
 - *size* The number of element.
- [window3d](#) (unsigned n, const [coord](#) crd[])
Construct a window of 3 dimensions from several points.
 - *n* The number of element.
 - *crd* The coordinates of the elements.

Static Public Member Functions

- `std::string` [name](#) ()
Return the name of the type.

Protected Member Functions

- [coord](#) [delta_update_](#) (const [dpoint_type](#) &dp)
Update delta.
 - *dp* a move point.

Friends

- class `abstract::window_base`< `abstract::window`< `window3d` >, `window3d` >

7.386.1 Detailed Description

Window 3 dimensions.

A window is a set of points. This class defines how to deal with. These points have 3 dimensions.

Definition at line 63 of file `window3d.hh`.

7.386.2 Member Typedef Documentation

7.386.2.1 `typedef struct_elt_traits`< [self_type](#) >::[dpoint_type](#) `oln::window3d::dpoint_type`

The associate image's type of dpoint (move point).

Warning:

Prefer the macros `oln_dpoint_type(Pointable)` and `oln_dpoint_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented from [oln::abstract::windownd< window3d >](#).

Definition at line 87 of file window3d.hh.

Referenced by [add\(\)](#), and [window3d\(\)](#).

7.386.2.2 typedef struct_elt_traits< self_type >::iter_type oln::window3d::iter_type

The associate image's type of iterator.

Warning:

Prefer the macros `oln_iter_type(Iterable)` and `oln_iter_type_(Iterable)` (the same without the 'type-name' keyword)

Definition at line 77 of file window3d.hh.

7.386.3 Member Function Documentation

7.386.3.1 [window3d](#)& oln::window3d::add (coord slice, coord row, coord col) [inline]

Add a point by coordinates to the window.

- slice The coordinate (slice) of the new point.
- row The coordinate (row) of the new point.
- col The coordinate (col) of the new point.

Add a new member by its coordinates to the window. The coordinates are only the column number, row number and column number because the window has 3 dimensions.

Definition at line 113 of file window3d.hh.

References [add\(\)](#), [oln::coord](#), and [dpoint_type](#).

```

114     {
115         return this->add(dpoint_type(slice, row, col));
116     }
```

7.386.3.2 [window3d](#)& oln::window3d::add (const dpoint_type & dp) [inline]

Add a dpoint (move point) to the window.

- dp The new point.

Add a new member to the window. This point must be of 3 dimensions.

Definition at line 98 of file window3d.hh.

Referenced by [add\(\)](#), [oln::mk_win_block\(\)](#), [oln::mk_win_ellipsoid\(\)](#), [oln::mk_win_from_neighb\(\)](#), and [window3d\(\)](#).

```

99     {
100         return this->exact().add_(dp);
101     }
```

7.386.3.3 `coord oln::window3d::delta_update_ (const dpoint_type & dp)` [`inline`,
`protected`]

Update delta.

- *dp* a move point.

Returns:

Delta.

If the point is the biggest element of the window. then this point is assigned to delta.

Definition at line 160 of file window3d.hh.

References `oln::abstract::window_base< window< window3d >, window3d >::delta_`.

```
161     {  
162         delta_(abs(dp.slice()));  
163         delta_(abs(dp.row()));  
164         delta_(abs(dp.col()));  
165         return delta_;  
166     }
```

The documentation for this class was generated from the following file:

- window3d.hh

7.387 oln::abstract::window_base< Sup, Exact > Struct Template Reference

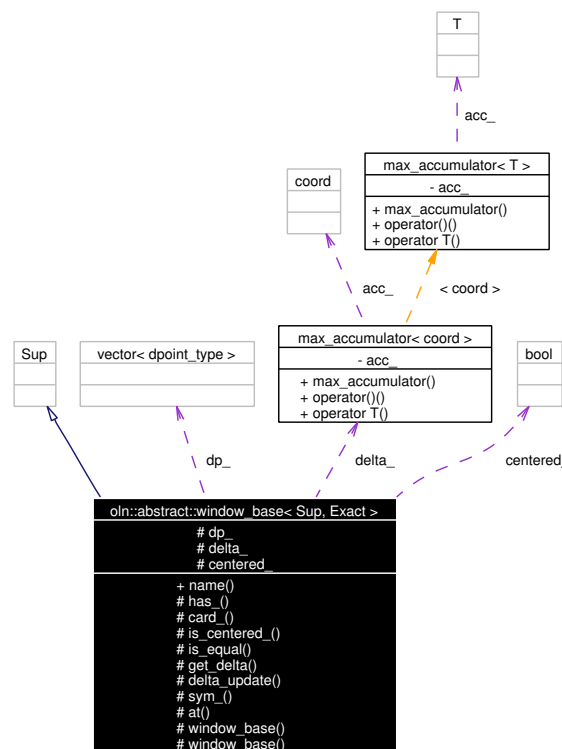
Window Base.

```
#include <window_base.hh>
```

Inheritance diagram for oln::abstract::window_base< Sup, Exact >:



Collaboration diagram for oln::abstract::window_base< Sup, Exact >:



Public Types

- typedef `window_base< Sup, Exact > self_type`
Set self type.
- typedef struct_elt_traits< Exact >::point_type point_type
The associate image's type of point.
- typedef struct_elt_traits< Exact >::dpoint_type dpoint_type

The associate image's type of dpoint (move point).

- typedef Exact [exact_type](#)
Set exact type.
- typedef Sup [super_type](#)
Set type of class inherited.
- enum { **dim** = struct_elt_traits<Exact >::dim }

Static Public Member Functions

- std::string [name](#) ()
Return the name of the type.

Protected Member Functions

- bool [has_](#) (const [dpoint_type](#) &dp) const
Test if the set of points contains this one.
– dp a dpoint (deplacement point).
- unsigned [card_](#) () const
Get the number of points.
- bool [is_centered_](#) () const
Test if the set of points is centered.
- bool [is_equal](#) (const [exact_type](#) &win) const
Compare two sets of points.
– win The set of point to compare.
- [coord get_delta](#) () const
Get the delta of the set of points.
- [coord delta_update](#) (const [dpoint_type](#) &dp)
Update delta.
– dp a deplacement point.
- void [sym_](#) ()
Set a set of point to opposite.
- const [dpoint_type](#) [at](#) (unsigned i) const
Get the nth structuring element.
– i The nth.
- [window_base](#) ()
CTor.

- [window_base](#) (unsigned size)
Used only by sub-classes
 - *size* The number of point.

Protected Attributes

- `std::vector< dpoint_type > dp_`
The list of point.
- `max_accumulator< coord > delta_`
Delta : the maximale point of the list.
- `bool centered_`
Is the set of point centered ?

Friends

- class `struct_elt< Exact >`
- class `neighborhood< Exact >`

7.387.1 Detailed Description

`template<class Sup, class Exact> struct oln::abstract::window_base< Sup, Exact >`

Window Base.

A window is a set of points and this class defines how to deal with. This class regroups common things for window, [w_window](#) (weight window) and neighborhood. Here, a set of point is a window or a weigh window or a neighborhood.

Definition at line 103 of file `window_base.hh`.

7.387.2 Member Typedef Documentation

7.387.2.1 `template<class Sup, class Exact> typedef struct_elt_traits<Exact>::dpoint_type oln::abstract::window_base< Sup, Exact >::dpoint_type`

The associate image's type of dpoint (move point).

Warning:

Prefer the macros `oln_dpoint_type(Pointable)` and `oln_dpoint_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented in `oln::abstract::neighborhoodnd< Exact >, oln::abstract::w_windownd< Exact >, oln::abstract::windownd< Exact >, oln::neighborhood1d, oln::neighborhood2d, oln::neighborhood3d, oln::w_window1d< T >, oln::w_window2d< T >, oln::w_window3d< T >, oln::window1d, oln::window2d, oln::window3d, oln::abstract::neighborhoodnd< neighborhood3d >, oln::abstract::neighborhoodnd< neighborhood2d >, oln::abstract::neighborhoodnd< neighborhood1d`

>, [oln::abstract::w_windownd< w_window2d< T2 > >](#), [oln::abstract::w_windownd< w_window2d< T > >](#), [oln::abstract::w_windownd< w_window1d< T > >](#), [oln::abstract::w_windownd< w_window3d< T > >](#), [oln::abstract::windownd< window1d >](#), [oln::abstract::windownd< window3d >](#), [oln::abstract::windownd< window2d >](#), and [oln::w_window2d< T2 >](#).

Definition at line 122 of file `window_base.hh`.

7.387.2.2 `template<class Sup, class Exact> typedef struct_elt_traits<Exact>::point_type` [oln::abstract::window_base< Sup, Exact >::point_type](#)

The associate image's type of point.

Warning:

Prefer the macros `oln_point_type(Pointable)` and `oln_point_type_(Pointable)` (the same without the 'typename' keyword)

Definition at line 115 of file `window_base.hh`.

7.387.3 Constructor & Destructor Documentation

7.387.3.1 `template<class Sup, class Exact> oln::abstract::window_base< Sup, Exact >::window_base\(\) [inline, protected]`

CTor.

Used only by sub-classes

Definition at line 251 of file `window_base.hh`.

```

251                                     : super_type(), dp_(), delta_(0)
252     {
253         centered_ = false;
254     }
```

7.387.3.2 `template<class Sup, class Exact> oln::abstract::window_base< Sup, Exact >::window_base\(unsigned size\) [inline, protected]`

Used only by sub-classes

- size The number of point.

Set the number of points this object will get. Used only by sub-classes

Definition at line 263 of file `window_base.hh`.

```

263                                     : super_type(), dp_(), delta_(0)
264     {
265         dp_.reserve(size);
266         centered_ = false;
267     }
```

7.387.4 Member Function Documentation

7.387.4.1 `template<class Sup, class Exact> const dpoint_type oln::abstract::window_base< Sup, Exact >::at (unsigned i) const` [inline, protected]

Get the nth structuring element.

- *i* The nth.

Returns:

The nth dpoint.

Precondition:

i < card().

Definition at line 240 of file window_base.hh.

```
241      {
242          precondition(i < this->card());
243          return dp_[i];
244      }
```

7.387.4.2 `template<class Sup, class Exact> unsigned oln::abstract::window_base< Sup, Exact >::card_() const` [inline, protected]

Get the number of points.

Returns:

The number of points.

Definition at line 162 of file window_base.hh.

```
163      {
164          return dp_.size();
165      }
```

7.387.4.3 `template<class Sup, class Exact> coord oln::abstract::window_base< Sup, Exact >::delta_update (const dpoint_type & dp)` [inline, protected]

Update delta.

- *dp* a displacement point.

Returns:

Delta.

If the point is the biggest element of the set of points, then this point is assigned to delta.

Definition at line 216 of file window_base.hh.

```
217      {
218          return this->exact().delta_update_(dp);
219      }
```

7.387.4.4 `template<class Sup, class Exact> coord oln::abstract::window_base< Sup, Exact
 >::get_delta () const [inline, protected]`

Get the delta of the set of points.

Returns:

Delta.

Delta is the bigger element of the set of points.

Definition at line 202 of file window_base.hh.

```
203     {
204         return delta_;
205     }
```

7.387.4.5 `template<class Sup, class Exact> bool oln::abstract::window_base< Sup, Exact
 >::has_ (const dpoint_type & dp) const [inline, protected]`

Test if the set of points contains this one.

- dp a dpoint (deplacement point).

Returns:

True if the set of points contains this dpoint.

Definition at line 152 of file window_base.hh.

```
153     {
154         return std::find(dp_.begin(), dp_.end(), dp) != dp_.end();
155     }
```

7.387.4.6 `template<class Sup, class Exact> bool oln::abstract::window_base< Sup, Exact
 >::is_centered_ () const [inline, protected]`

Test if the set of points is centered.

Returns:

True if it's centered.

Centered means :

- at least one element for neighborhood;
- list of point contains 0 for window.

Definition at line 176 of file window_base.hh.

```
177     {
178         return centered_;
179     }
```


7.387.4.7 `template<class Sup, class Exact> bool oln::abstract::window_base< Sup, Exact >::is_equal (const exact_type & win) const` `[inline, protected]`

Compare two sets of points.

- win The set of point to compare.

Returns:

True if they are the same.

Definition at line 187 of file window_base.hh.

```

188      {
189          for (typename std::vector<dpoint_type>::const_iterator it = dp_.begin(); it != dp_.end(); ++it)
190              if (std::find(win.dp_.begin(), win.dp_.end(), *it) == win.dp_.end())
191                  return false;
192          return true;
193      }
```

7.387.4.8 `template<class Sup, class Exact> void oln::abstract::window_base< Sup, Exact >::sym_()` `[inline, protected]`

Set a set of point to opposite.

Each point of the set of point is assigned to its opposite.

Definition at line 227 of file window_base.hh.

```

228      {
229          for (unsigned i = 0; i < this->card(); ++i)
230              dp_[i] = - dp_[i];
231      }
```

7.387.5 Friends And Related Function Documentation

7.387.5.1 `template<class Sup, class Exact> friend class struct_elt< Exact >` `[friend]`

Todo

FIXME: this has been commented out to satisfy icc and comeau. I don't know who is right between them and gcc.

Definition at line 133 of file window_base.hh.

The documentation for this struct was generated from the following file:

- window_base.hh

7.388 oln::window_base_friend_traits< abstract::neighborhood< Exact > > Struct Template Reference

```
#include <window_base.hh>
```

Public Types

- typedef [abstract::neighborhood](#)< Exact > **ret**

7.388.1 Detailed Description

```
template<class Exact> struct oln::window_base_friend_traits< abstract::neighborhood< Exact > >
```

If window_base inherits from neighborhood, then mother is neighborhood.

Definition at line 68 of file window_base.hh.

The documentation for this struct was generated from the following file:

- window_base.hh

7.389 oln::window_base_friend_traits< abstract::w_window< Exact > > Struct Template Reference

```
#include <window_base.hh>
```

Public Types

- typedef [abstract::struct_elt](#)< Exact > **ret**

7.389.1 Detailed Description

template<class Exact> struct oln::window_base_friend_traits< abstract::w_window< Exact > >

If window_base inherits from w_window, then mother is struct_elt.

Definition at line 86 of file window_base.hh.

The documentation for this struct was generated from the following file:

- window_base.hh

7.390 oln::window_base_friend_traits< abstract::window< Exact > > Struct Template Reference

```
#include <window_base.hh>
```

Public Types

- typedef [abstract::struct_elt](#)< Exact > **ret**

7.390.1 Detailed Description

template<class Exact> struct oln::window_base_friend_traits< abstract::window< Exact > >

If window_base inherits from window, then mother is struct_elt.

Definition at line 77 of file window_base.hh.

The documentation for this struct was generated from the following file:

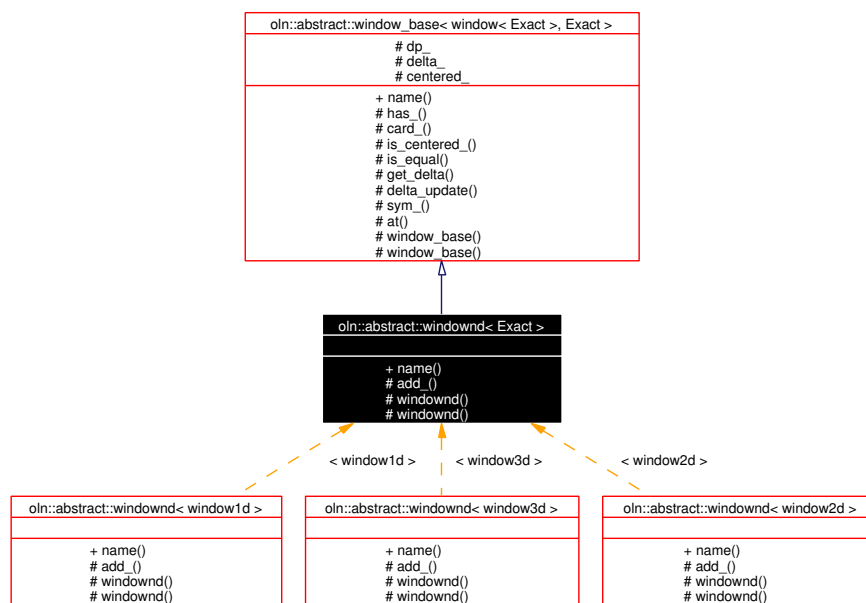
- window_base.hh

7.391 oln::abstract::windownd< Exact > Struct Template Reference

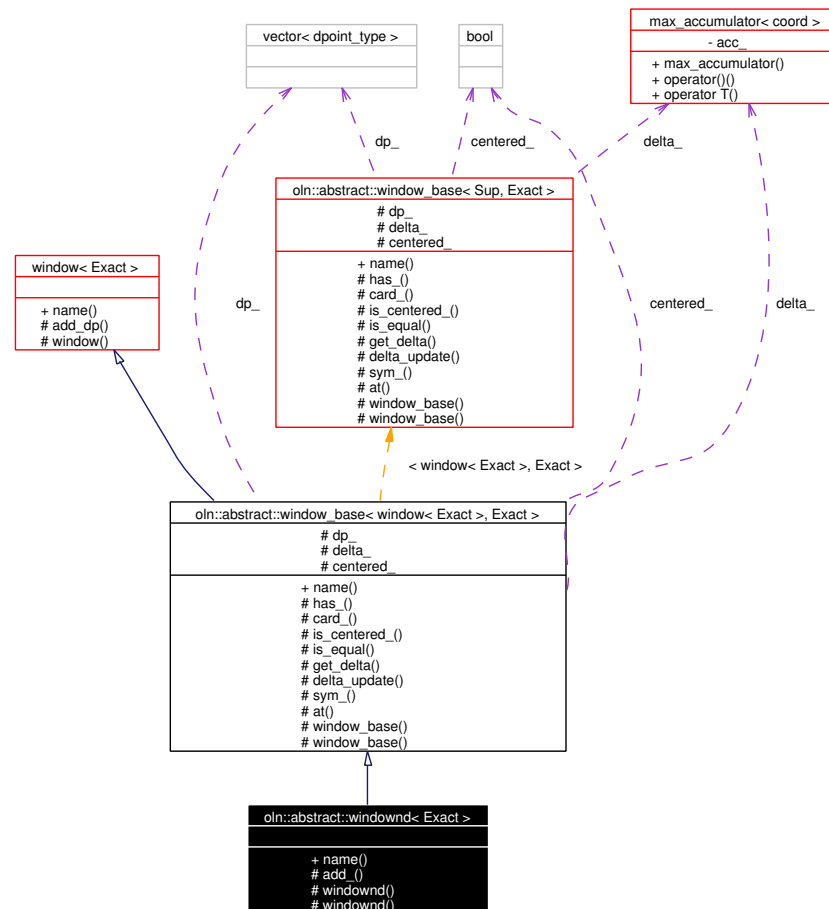
Window N dimensions.

```
#include <windownd.hh>
```

Inheritance diagram for oln::abstract::windownd< Exact >:



Collaboration diagram for oln::abstract::windownd< Exact >:



Public Types

- typedef `window_base< window< Exact >, Exact >` `super_type`
Set the super type.
- typedef `windownd< Exact >` `self_type`
Set the self type.
- typedef `Exact` `exact_type`
Set the exact type.
- typedef `struct_elt_traits< Exact >::dpoint_type` `dpoint_type`
The associate image's type of dpoint (move point).

Static Public Member Functions

- `std::string name ()`
Return the name of the type.

Protected Member Functions

- [exact_type](#) & [add_](#) (const [dpoint_type](#) &dp)
Add a point to the window.
 - *dp* The new point.
- [windownd](#) ()
Construct a window.
- [windownd](#) (unsigned size)
Construct a [w_window](#) of 'size' elements.
 - *size* The number of elements to reserve for the window.

Friends

- class [window](#)< [exact_type](#) >

7.391.1 Detailed Description

`template<class Exact> struct oln::abstract::windownd< Exact >`

Window N dimensions.

A window is a set of points. This class defines how to deal with. These points have N dimensions.

Definition at line 59 of file windownd.hh.

7.391.2 Member Typedef Documentation

7.391.2.1 `template<class Exact> typedef struct_elt_traits<Exact>::dpoint_type
oln::abstract::windownd< Exact >::dpoint_type`

The associate image's type of dpoint (move point).

Warning:

Prefer the macros `oln_dpoint_type(Pointable)` and `oln_dpoint_type_(Pointable)` (the same without the 'typename' keyword)

Reimplemented from [oln::abstract::window_base](#)< [window](#)< Exact >, Exact >.

Reimplemented in [oln::window1d](#), [oln::window2d](#), and [oln::window3d](#).

Definition at line 71 of file windownd.hh.

7.391.3 Member Function Documentation

7.391.3.1 `template<class Exact> exact_type& oln::abstract::windownd< Exact >::add_ (const
dpoint_type & dp) [inline, protected]`

Add a point to the window.

- dp The new point.

Add a new member to the window.

Definition at line 91 of file windownd.hh.

```
92     {
93         if (dp.is_centered())
94             this->centered_ = true;
95         if (!(has_(dp)))
96             this->dp_.push_back(dp);
97         this->delta_update(dp);
98         return this->exact();
99     }
```

The documentation for this struct was generated from the following file:

- windownd.hh

7.392 `ntg::internal::with_arith< T >` Struct Template Reference

Return a type which supports inc and dec.

```
#include <pred_succ.hh>
```

Public Types

- `typedef mlc::internal::wrap< typename mlc::internal::is_a< sizeof(mlc::form::get< non_vectorial >)) >::check< typename ntg::type_traits< T >::abstract_type, non_vectorial > >::ensure_type non_v)`
- `typedef int_u< 1 > bool_with_arith`
- `typedef T non_vectorial_with_arith`
- `typedef mlc::if_< mlc::internal::wrap< typename mlc::internal::is_a< sizeof(mlc::form::get< ntg::binary >)) >::check< typename ntg::type_traits< T >::abstract_type, ntg::binary > >::ret, id_< bool_with_arith >, id_< non_vectorial_with_arith > >::ret::re ret)`

7.392.1 Detailed Description

`template<typename T> struct ntg::internal::with_arith< T >`

Return a type which supports inc and dec.

Definition at line 41 of file `pred_succ.hh`.

The documentation for this struct was generated from the following file:

- `pred_succ.hh`

7.393 oln::io::internal::writers_trier Struct Reference

Writers trier functor, helper for stream_wrappers.

```
#include <image_write.hh>
```

Static Public Member Functions

- `template<class T> bool doit (const T &input, std::ostream &out, const std::string ext)`

Writers trier functor, helper for stream_wrappers.

- *output* The new object.
- *in* The stream to read from.
- *ext* The extension.

7.393.1 Detailed Description

Writers trier functor, helper for stream_wrappers.

Definition at line 131 of file image_write.hh.

7.393.2 Member Function Documentation

7.393.2.1 `template<class T> bool oln::io::internal::writers_trier::doit (const T &input, std::ostream &out, const std::string ext) [inline, static]`

Writers trier functor, helper for stream_wrappers.

- *output* The new object.
- *in* The stream to read from.
- *ext* The extension.

First try to write by extension, then by data.

Definition at line 143 of file image_write.hh.

```
144     {
145         bool result = try_writers<WriteAny,T>::by_extension(input, out, ext);
146         if (!result)
147             result = try_writers<WriteAny,T>::by_data(input, out, ext);
148         return result;
149     }
```

The documentation for this struct was generated from the following file:

- image_write.hh

7.394 oln::io::gz::zfilebuf Class Reference

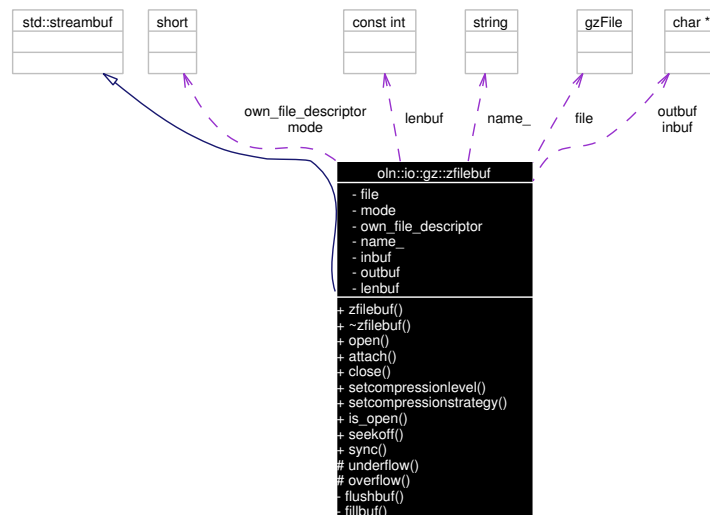
Performs operation on compressed files.

```
#include <gz_stream.hh>
```

Inheritance diagram for oln::io::gz::zfilebuf:



Collaboration diagram for oln::io::gz::zfilebuf:



Public Member Functions

- `zfilebuf * open` (const char *name, int io_mode)

Return a stream on the file name regarding the opening mode: io_mode.

- `zfilebuf * attach` (int file_descriptor, int io_mode)
Attach a stream on file_descriptor regarding the opening mode: io_mode.
- `zfilebuf * close` ()
Close the stream.
- int `setcompressionlevel` (short comp_level)
- int `setcompressionstrategy` (short comp_strategy)
- int `is_open` () const
Return true if the stream is open, false otherwise.
- virtual std::streampos `seekoff` (std::streamoff off, std::ios::seekdir dir, int)
- virtual int `sync` ()
Flush the buffer associated to the stream.

Protected Member Functions

- virtual int `underflow` ()
Return the next character in the stream. On failure, EOF is returned.
- virtual int `overflow` (int c=EOF)
Flush the output buffer associated to the stream then write c. On failure, EOF is returned.

7.394.1 Detailed Description

Performs operation on compressed files.

Definition at line 47 of file gz_stream.hh.

The documentation for this class was generated from the following file:

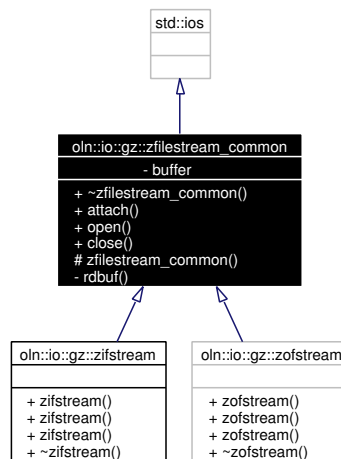
- gz_stream.hh

7.395 oln::io::gz::zfilestream_common Class Reference

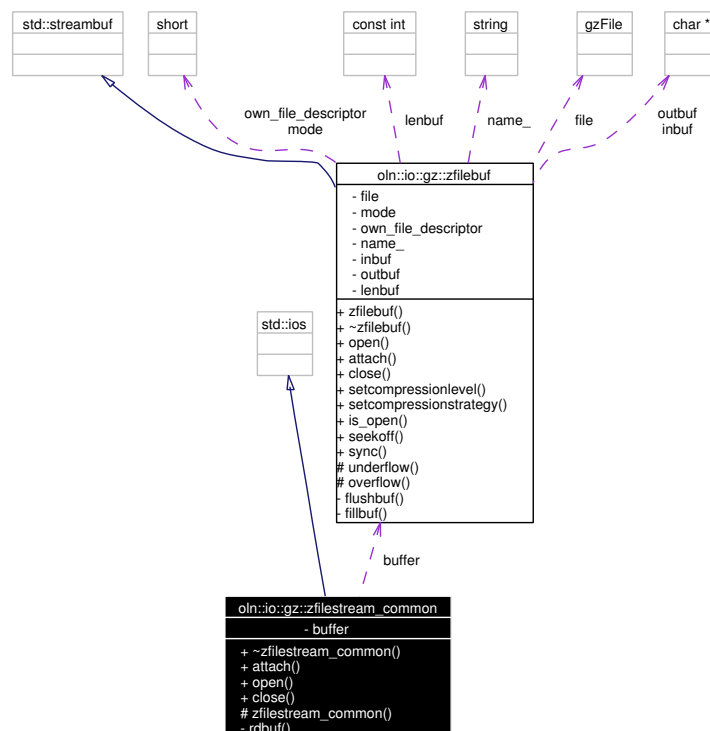
Define an interface for compressed file stream manipulation.

```
#include <gz_stream.hh>
```

Inheritance diagram for oln::io::gz::zfilestream_common:



Collaboration diagram for oln::io::gz::zfilestream_common:



Public Member Functions

- void [attach](#) (int fd, int io_mode)
Open the stream on the file descriptor: fd regarding the opening mode.
- void [open](#) (const char *name, int io_mode)
Open the stream on the file named name regarding the opening mode.
- void [close](#) ()
Close the current stream.

Protected Member Functions

- [zfstream_common](#) ()
Prevent instantiation.

Friends

- zofstream & [setcompressionlevel](#) (zofstream &, int)
Set the compression level of s to l.
- zofstream & [setcompressionstrategy](#) (zofstream &, int)
Set the compression strategy of s to l.

7.395.1 Detailed Description

Define an interface for compressed file stream manipulation.

Definition at line 300 of file gz_stream.hh.

The documentation for this class was generated from the following file:

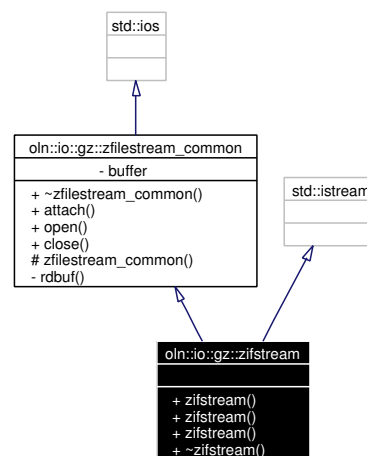
- gz_stream.hh

7.396 oln::io::gz::zifstream Class Reference

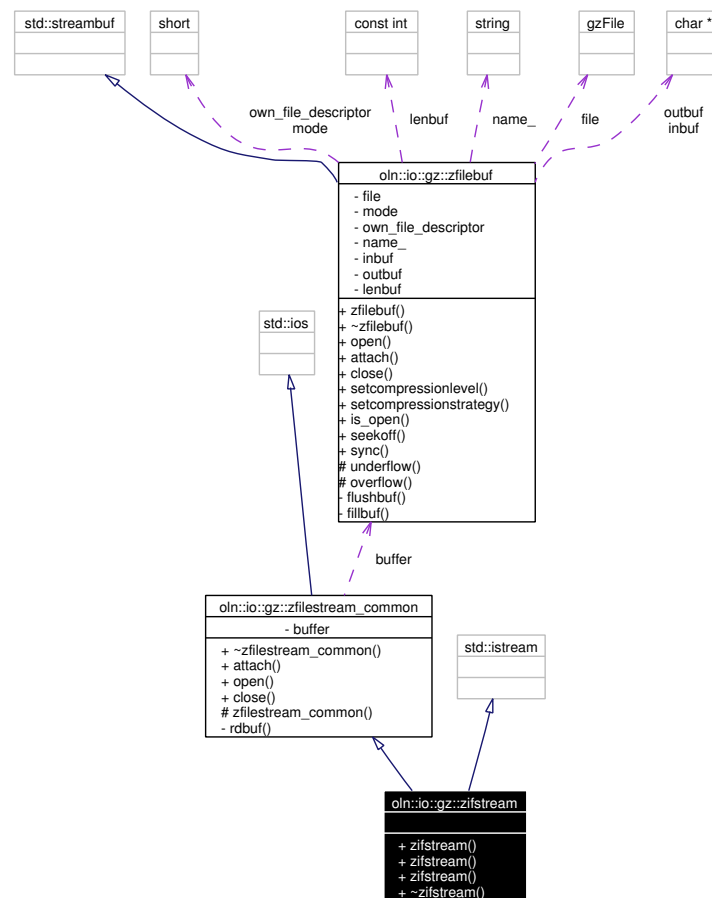
Read only zstream.

```
#include <gz_stream.hh>
```

Inheritance diagram for oln::io::gz::zifstream:



Collaboration diagram for oln::io::gz::zifstream:



Public Member Functions

- [zifstream](#) (const char *name, int io_mode=std::ios::in)
Open a read only stream on the file named name.
- [zifstream](#) (int fd, int io_mode=std::ios::in)
Open a read only stream on the file descriptor fd.

7.396.1 Detailed Description

Read only zstream.

Definition at line 360 of file gz_stream.hh.

The documentation for this class was generated from the following file:

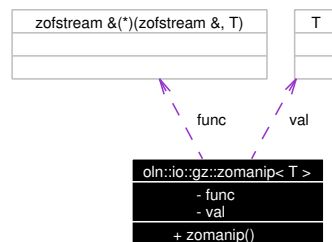
- gz_stream.hh

7.397 oln::io::gz::zomanip< T > Class Template Reference

Define a pair func / val to perform manipulation on zofstream.

```
#include <gz_stream.hh>
```

Collaboration diagram for oln::io::gz::zomanip< T >:



Public Member Functions

- **zomanip** (zofstream &(*) (zofstream &, T), T v)

Friends

- zofstream & **operator** (zofstream &, const [zomanip](#)< T > &)

7.397.1 Detailed Description

```
template<class T> class oln::io::gz::zomanip< T >
```

Define a pair func / val to perform manipulation on zofstream.

Definition at line 426 of file gz_stream.hh.

The documentation for this class was generated from the following file:

- gz_stream.hh

Chapter 8

Olena File Documentation

8.1 cmap.hh File Reference

```
#include <oln/topo/tarjan/flat-zone.hh>
#include <oln/topo/inter-pixel/inter-pixel.hh>
#include <oln/topo/combinatorial-map/internal/zeta.hh>
#include <oln/topo/combinatorial-map/internal/allfunc.hh>
#include <oln/topo/combinatorial-map/internal/level.hh>
#include <algorithm>
#include <iterator>
```

Include dependency graph for cmap.hh:



Namespaces

- namespace `oln`
- namespace `oln::topo`
- namespace `oln::topo::combinatorial_map`

Functions

- `template<class I> std::ostream & operator<< (std::ostream &ostr, const oln::topo::combinatorial_map::cmap< I > &cm)`

8.1.1 Detailed Description

Todo

FIXME: There is some problems in the directory `topo/combinatorial-map/`, such as non static functions, or the redefinition of the class any.

FIXME: The documentation is not good enough.

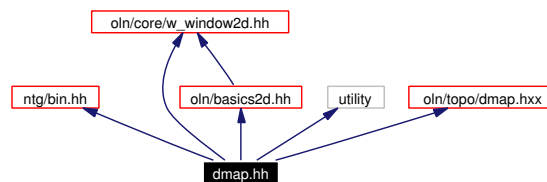
FIXME: Real test are missing. FIXME: Some part of this file are inside "#if 0" comments. FIXME:
This file force to keep an obsolete version of flat-zone.

Definition in file [cmap.hh](#).

8.2 dmap.hh File Reference

```
#include <ntg/bin.hh>
#include <oln/basics2d.hh>
#include <oln/core/w_window2d.hh>
#include <utility>
#include <oln/topo/dmap.hxx>
```

Include dependency graph for dmap.hh:



Namespaces

- namespace `oln`
- namespace `oln::topo`

8.2.1 Detailed Description

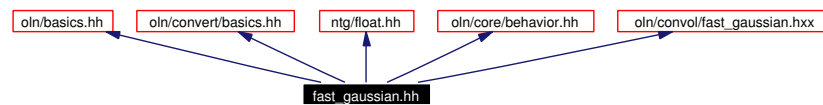
REF: B.J.H. Verwer, Local distances for distance transformations in two and three dimensions, Pattern Recognition Letters 12 (1991) 671-682

Definition in file [dmap.hh](#).

8.3 fast_gaussian.hh File Reference

```
#include <oln/basics.hh>
#include <oln/convert/basics.hh>
#include <ntg/float.hh>
#include <oln/core/behavior.hh>
#include <oln/convol/fast_gaussian.hxx>
```

Include dependency graph for fast_gaussian.hh:



Namespaces

- namespace `oln`
- namespace `oln::convol`
- namespace `oln::convol::fast`

8.3.1 Detailed Description

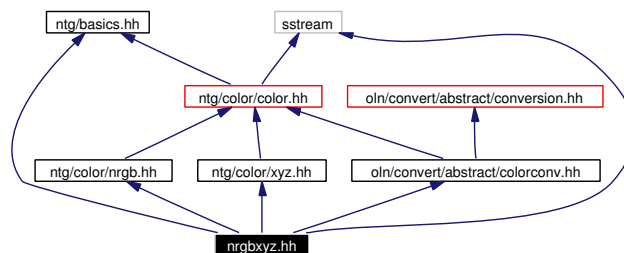
Gaussian filter implementation from "Recursively implementing the gaussian and its derivatives" Deriche 93 INRIA REPORT.

Definition in file [fast_gaussian.hh](#).

8.4 nrgbxyz.hh File Reference

```
#include <oln/convert/abstract/colorconv.hh>
#include <ntg/color/nrgb.hh>
#include <ntg/color/xyz.hh>
#include <ntg/basics.hh>
#include <sstream>
```

Include dependency graph for nrgbxyz.hh:



Namespaces

- namespace `oln`
- namespace `oln::convert`

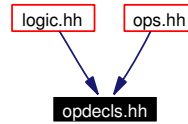
8.4.1 Detailed Description

REF: The formulas used here come from “Digital Image Processing Algorithms and Applications”, I. Pitas; Wiley-Interscience.

Definition in file [nrgbxyz.hh](#).

8.5 opdecls.hh File Reference

This graph shows which files directly or indirectly include this file:



Defines

- #define [oln_arith_declare_binrecval_functor_](#)(OPNAME, OPCODE)
- #define [oln_arith_declare_binrecvalcst_functor_](#)(OPNAME, OPCODE_CST)
- #define [oln_arith_declare_binrecval_functors_](#)(OPNAME, OPCODE, OPCODE_CST)
- #define [oln_arith_declare_binfixedtype_functor_](#)(OPNAME, OPCODE, TYPE)
- #define [oln_arith_declare_binfixedtypecst_functor_](#)(OPNAME, OPCODE_CST, TYPE)
- #define [oln_arith_declare_binfixedtype_functors_](#)(NAME, TYPE, CODE, CODE_CST)
- #define [default_functor_return_type_](#)(OPNAME, I1, I2)
Shortcut.
- #define [default_functor_type_cst_](#)(OPNAME, I1, T2)
Shortcut.
- #define [default_functor_return_type_cst_](#)(OPNAME, I1, T2) typename default_functor_type_cst_(OPNAME, I1, T2)::result_type
Shortcut.
- #define [oln_arith_declare_binop_procs_](#)(OPNAME)
Declare front-end functions.
- #define [oln_arith_declare_binopcst_procs_](#)(OPNAME)
Apply OPNAME with a constant as second operand.
- #define [oln_arith_declare_all_binop_procs_](#)(OPNAME)
- #define [oln_arith_declare_nongenericbinop_procs_](#)(OPNAME)
Same as oln_arith_declare_nongenericbinop_procs_ but for non template functors.
- #define [oln_arith_declare_nongenericbinopcst_procs_](#)(OPNAME)
Apply OPNAME with a constant as second operand.
- #define [oln_arith_declare_all_nongenericbinop_procs_](#)(OPNAME)
- #define [oln_arith_declare_unfixedtype_functor_](#)(OPNAME, TYPE, OPCODE)
- #define [oln_arith_declare_nongenericunop_procs_](#)(OPNAME)
- #define [oln_arith_declare_unop_procs_](#)(OPNAME)

8.5.1 Detailed Description

Operations are defined between two images and between one image and one constant value (with the cst suffix). The two main components are:

1. Define functors for each operations, taking two values and returning the result.
2. Define front-end functions applying a functor on the whole image.
3. Versions are defined, leaving the possibility to specify the return type automatically, manually or using a conversion (from convert).

Todo

FIXME: These macros should be rewritten / split into real code to make things clearer.

Definition in file [opdecls.hh](#).

8.5.2 Define Documentation

8.5.2.1 #define default_functor_return_type_(OPNAME, I1, I2)

Value:

```
typename f_##OPNAME<oln_value_type(I1), \
                    oln_value_type(I2), \
                    ntg_return_type(OPNAME, \
                                    oln_value_type(I1), \
                                    oln_value_type(I2))>::result_type
```

Shortcut.

Definition at line 163 of file opdecls.hh.

8.5.2.2 #define default_functor_type_cst_(OPNAME, I1, T2)

Value:

```
f_##OPNAME##_cst<oln_value_type(I1), \
                 T2, \
                 ntg_return_type(OPNAME, \
                                 oln_value_type(I1), \
                                 T2)>
```

Shortcut.

Definition at line 171 of file opdecls.hh.

8.5.2.3 #define oln_arith_declare_all_binop_procs_(OPNAME)

Value:

```
oln_arith_declare_binop_procs_(OPNAME) \
    oln_arith_declare_binopcst_procs_(OPNAME)
```

Definition at line 268 of file opdecls.hh.

8.5.2.4 #define oln_arith_declare_all_nongenericbinop_procs_(OPNAME)**Value:**

```

oln_arith_declare_nongenericbinop_procs_(OPNAME) \
    oln_arith_declare_nongenericbinopcst_procs_(OPNAME)

```

Definition at line 319 of file opdecls.hh.

8.5.2.5 #define oln_arith_declare_binfixedtype_functor_(OPNAME, OPCODE, TYPE)**Value:**

```

struct f_##OPNAME : std::binary_function< const TYPE&, const TYPE&, TYPE> \
{ \
    const result_type \
    operator()(first_argument_type val1, \
               second_argument_type val2) const \
    { \
        return OPCODE; \
    } \
} /* no ; */

```

For binary functions that work on a single known datatype.

- OPNAME Will produce the name of the function.
- OPCODE Operation that may use *val1* and *val2*.
- TYPE Type that can be used.

Definition at line 123 of file opdecls.hh.

8.5.2.6 #define oln_arith_declare_binfixedtype_functors_(NAME, TYPE, CODE, CODE_CST)**Value:**

```

oln_arith_declare_binfixedtype_functor_(NAME, CODE, TYPE); \
    oln_arith_declare_binfixedtypecst_functor_(NAME, CODE_CST, TYPE)

```

`oln_arith_declare_binfixedtype_functor_ & oln_arith_declare_binfixedtypecst_functor_.`

Definition at line 158 of file opdecls.hh.

8.5.2.7 #define oln_arith_declare_binfixedtypecst_functor_(OPNAME, OPCODE_CST, TYPE)**Value:**

```

struct f_##OPNAME##_cst: std::unary_function<const TYPE, TYPE > \
{ \
    f_##OPNAME##_cst(TYPE cst) : cst_(cst) {} \
    \
    const result_type \
    operator()(argument_type val) const \
    { \
        \
    } \
}

```

```

        return OPCODE_CST;
    }
private:
    TYPE cst_;
} /* no ; */

```

For Binary functions that work on a single known datatype.

- OPNAME Will produce the name of the function.
- OPCODE_CST Operation that may use *val1* and *cst_*
- TYPE Type that can be used.

Definition at line 141 of file opdecls.hh.

8.5.2.8 #define oln_arith_declare_binrecval_functor_(OPNAME, OPCODE)

Value:

```

template<class T1, class T2, class Ret>
    struct f_##OPNAME : std::binary_function<const T1&,
        const T2&,
        Ret>
    {
        typedef f_##OPNAME self_type;
        typename self_type::result_type
        operator()(typename self_type::first_argument_type val1,
            typename self_type::second_argument_type val2) const
        {
            return OPCODE;
        }
    };

template <class T1, class T2 = T1>
struct default_f_##OPNAME
    : public f_##OPNAME< T1, T2, ntg_return_type(OPNAME, T1, T2)>
{} /* no ; */

```

Binary functors.

Produce a functor named *f_##OPNAME* using the code OPCODE.

- OPNAME Will produce the name of the function
- OPCODE Operation that may use *val1* and *val2*

Definition at line 55 of file opdecls.hh.

8.5.2.9 #define oln_arith_declare_binrecval_functors_(OPNAME, OPCODE, OPCODE_CST)

Value:

```

oln_arith_declare_binrecval_functor_(OPNAME, OPCODE);
oln_arith_declare_binrecvalcst_functor_(OPNAME, OPCODE_CST)

```

Produces an unary function and a binary and a Unary Functor using a constant.

- OPNAME Will produce the name of the function.
- OPCODE Operation that may use *val1* and *val2*.
- OPCODE_CST Operation that may use *val1* and *cst_*.

See also:

[oln_arith_declare_binrecval_funcutor_](#)
[oln_arith_declare_binrecvalcst_funcutor_](#)

Definition at line 113 of file opdecls.hh.

8.5.2.10 #define oln_arith_declare_binrecvalcst_funcutor_(OPNAME, OPCODE_CST)

Value:

```
template<class T1, class T2, class Ret>
    struct f_##OPNAME##_cst : std::unary_function<const T1&,
        Ret>
    {
        typedef f_##OPNAME##_cst self_type;
        f_##OPNAME##_cst(T2 cst) : cst_(cst) {}

        typename self_type::result_type
        operator()(typename self_type::argument_type val) const
        {
            return OPCODE_CST;
        }
    private:
        T2 cst_;
    } /* no ; */
```

Unary functor, using a constant.

Produce a functor named `f_##OPNAME` using the code `OPCODE`. The object is constructed using a constant.

- OPNAME Will produce the name of the function
- OPCODE_CST Operation that may use *val1* and *cst_*

Definition at line 86 of file opdecls.hh.

8.5.2.11 #define oln_arith_declare_nongenericbinop_procs_(OPNAME)

Value:

```
/* Standard application of OPNAME */
template<class I1, class I2> inline
typename mute<I1, typename f_##OPNAME::result_type>::ret
OPNAME(const abstract::image<I1>& input1, const abstract::image<I2>& input2)
{
    return apply2<f_##OPNAME >(input1, input2);
}

/* Same as above, plus conversion. */
template<class C, class B, class I1, class I2> inline
typename mute<I1,
```

```

    typename convoutput<C, B, typename f_##OPNAME::result_type>::ret>::ret      \
    OPNAME(const convert::abstract::conversion<C, B>& conv,                  \
            const abstract::image<I1>& input1, const abstract::image<I2>& input2) \
    {                                                                         \
        return apply2(convert::compconv2(conv, f_##OPNAME()),               \
                       input1, input2);                                       \
    }                                                                           \

```

Same as oln_arith_declare_nongenericbinop_procs_ but for non template functors.

Definition at line 273 of file opdecls.hh.

8.5.2.12 #define oln_arith_declare_nongenericbinopcst_procs_(OPNAME)

Value:

```

template<class I, class T> inline                                           \
    typename mute<I, typename f_##OPNAME##_cst::result_type>::ret         \
    OPNAME##_cst(const abstract::image<I>& input, T val)                  \
    {                                                                       \
        return apply(f_##OPNAME##_cst(val), input);                       \
    }                                                                       \
                                                                            \
/* Same as above, plus conversion. */                                     \
template<class C, class B, class I, class T> inline                       \
    typename mute<I,                                                       \
        typename convoutput<C, B,                                         \
            typename f_##OPNAME##_cst::result_type>::ret>::ret           \
    OPNAME##_cst(const convert::abstract::conversion<C, B>& conv,         \
        const abstract::image<I>& input, T val)                           \
    {                                                                       \
        return apply(convert::compconv1(conv, f_##OPNAME##_cst(val)),     \
                       input);                                             \
    }                                                                       \

```

Apply OPNAME with a constant as second operand.

Definition at line 297 of file opdecls.hh.

8.5.2.13 #define oln_arith_declare_nongenericunop_procs_(OPNAME)

Value:

```

/* Standard application of OPNAME */                                     \
template<class I> inline                                                  \
    typename mute<I, typename f_##OPNAME::result_type>::ret               \
    OPNAME(const abstract::image<I>& input1)                               \
    {                                                                       \
        return apply<f_##OPNAME >(input1);                                \
    }                                                                       \
                                                                            \
/* Same as above, plus conversion. */                                     \
template<class C, class B, class I> inline                               \
    typename mute<I,                                                       \
        typename convoutput<C, B,                                         \
            typename f_##OPNAME::result_type>::ret>::ret                 \
    OPNAME(const convert::abstract::conversion<C, B>& conv, const abstract::image<I>& input1) \
    {                                                                       \
        return apply(convert::compconv2(conv, f_##OPNAME()), input1);     \
    }                                                                       \

```

Definition at line 338 of file opdecls.hh.

8.5.2.14 #define oln_arith_declare_unfixedtype_functor_(OPNAME, TYPE, OPCODE)**Value:**

```

struct f_##OPNAME : std::unary_function< const TYPE&, TYPE>
{
    const result_type operator()(argument_type val) const
    {
        return OPCODE;
    }
} /* no ; */

```

Unary function for binary functions that work on a single known datatype.

Definition at line 328 of file opdecls.hh.

8.5.2.15 #define oln_arith_declare_unop_procs_(OPNAME)**Value:**

```

/* Standard application of OPNAME */
template<class I> inline
typename mute<I, typename f_##OPNAME<oln_value_type(I)>::result_type>::ret
OPNAME(const abstract::image<I>& input1)
{
    return apply(f_##OPNAME<oln_value_type(I)>(), input1);
}

/* Same as above, plus conversion. */
template<class C, class B, class I> inline
typename mute<I,
    typename convoutput<C, B,
        typename f_##OPNAME<oln_value_type(I)>::result_type>::ret>::ret
OPNAME(const convert::abstract::conversion<C>& conv, const abstract::image<I>& input1)
{
    return apply(convert::compconv2(conv, f_##OPNAME<oln_value_type(I)>()), input1);
}

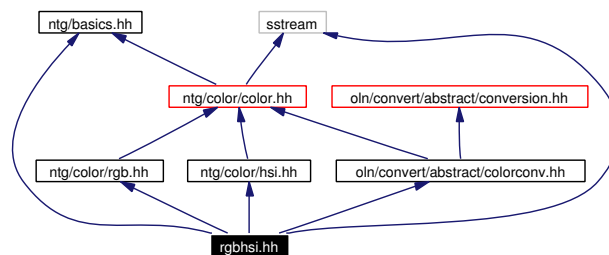
```

Definition at line 356 of file opdecls.hh.

8.6 rgbhsi.hh File Reference

```
#include <oln/convert/abstract/colorconv.hh>
#include <ntg/basics.hh>
#include <ntg/color/rgb.hh>
#include <ntg/color/hsi.hh>
#include <sstream>
```

Include dependency graph for rgbhsi.hh:



Namespaces

- namespace `oln`
- namespace `oln::convert`

8.6.1 Detailed Description

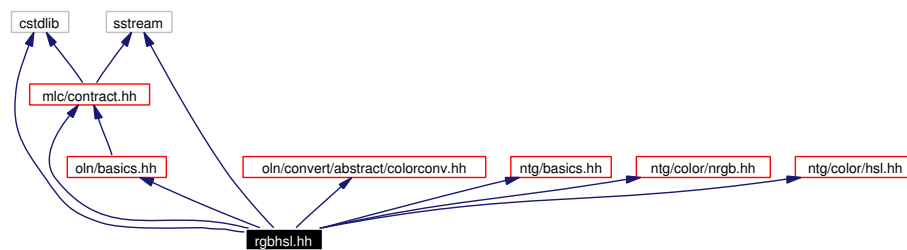
REF: The formulas used here come from “Digital Image Processing Algorithms and Applications”, I. Pitas; Wiley-Interscience.

Definition in file `rgbhsi.hh`.

8.7 rgbhsl.hh File Reference

```
#include <oln/basics.hh>
#include <oln/convert/abstract/colorconv.hh>
#include <ntg/basics.hh>
#include <ntg/color/nrgb.hh>
#include <ntg/color/hsl.hh>
#include <mlc/contract.hh>
#include <cstdlib>
#include <sstream>
```

Include dependency graph for rgbhsl.hh:



Namespaces

- namespace `oln`
- namespace `oln::convert`
- namespace `oln::convert::internal`

8.7.1 Detailed Description

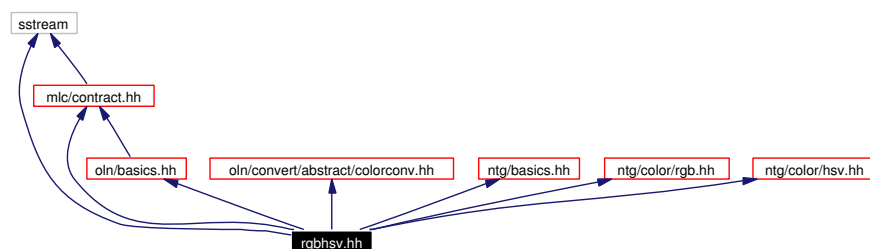
REF: The formulas used here come from “Color space conversion”; Paul Bourke.

Definition in file [rgbhsl.hh](#).

8.8 rgbhsv.hh File Reference

```
#include <oln/basics.hh>
#include <oln/convert/abstract/colorconv.hh>
#include <ntg/basics.hh>
#include <ntg/color/rgb.hh>
#include <ntg/color/hsv.hh>
#include <mlc/contract.hh>
#include <sstream>
```

Include dependency graph for rgbhsv.hh:



Namespaces

- namespace `oln`
- namespace `oln::convert`

8.8.1 Detailed Description

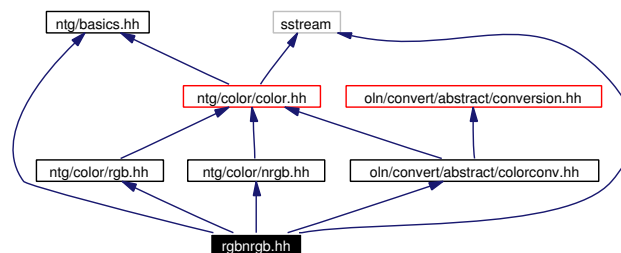
REF: The formulas used here come from “Color Conversion Algorithms”

Definition in file `rgbhsv.hh`.

8.9 rgbnrgb.hh File Reference

```
#include <oln/convert/abstract/colorconv.hh>
#include <ntg/color/rgb.hh>
#include <ntg/color/nrgb.hh>
#include <ntg/basics.hh>
#include <sstream>
```

Include dependency graph for rgbnrgb.hh:



Namespaces

- namespace `oln`
- namespace `oln::convert`

8.9.1 Detailed Description

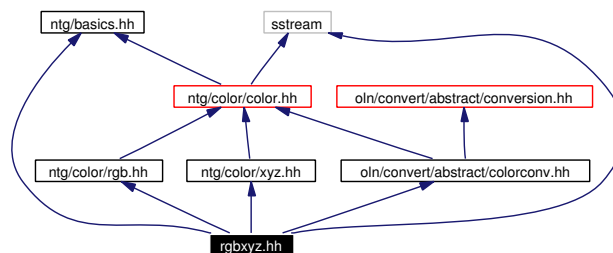
REF: The formulas used here come from “Digital Image Processing Algorithms and Applications”, I. Pitas; Wiley-Interscience.

Definition in file `rgbnrgb.hh`.

8.10 rgbxyz.hh File Reference

```
#include <oln/convert/abstract/colorconv.hh>
#include <ntg/color/rgb.hh>
#include <ntg/color/xyz.hh>
#include <ntg/basics.hh>
#include <sstream>
```

Include dependency graph for rgbxyz.hh:



Namespaces

- namespace `oln`
- namespace `oln::convert`

8.10.1 Detailed Description

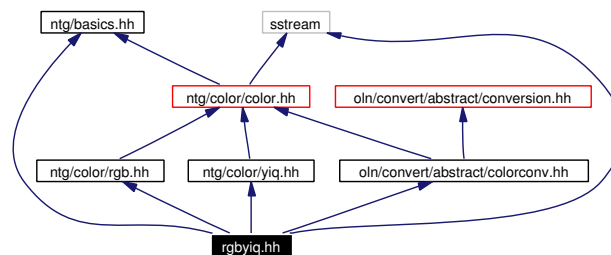
REF: The formulas used here come from “Digital Image Processing Algorithms and Applications”, I. Pitas; Wiley-Interscience.

Definition in file [rgbxyz.hh](#).

8.11 `rgbyiq.hh` File Reference

```
#include <oln/convert/abstract/colorconv.hh>
#include <ntg/color/rgb.hh>
#include <ntg/color/yiq.hh>
#include <ntg/basics.hh>
#include <sstream>
```

Include dependency graph for `rgbyiq.hh`:



Namespaces

- namespace `oln`
- namespace `oln::convert`

8.11.1 Detailed Description

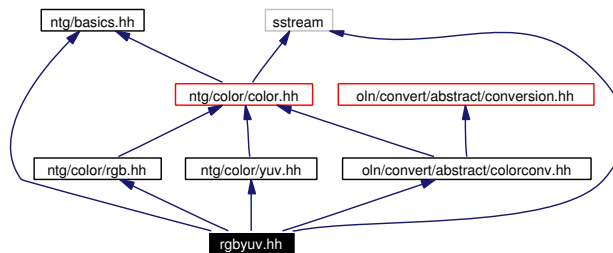
REF: The formulas used here come from “Digital Image Processing Algorithms and Applications”, I. Pitas; Wiley-Interscience.

Definition in file `rgbyiq.hh`.

8.12 rgbyuv.hh File Reference

```
#include <oln/convert/abstract/colorconv.hh>
#include <ntg/color/rgb.hh>
#include <ntg/color/yuv.hh>
#include <ntg/basics.hh>
#include <sstream>
```

Include dependency graph for rgbyuv.hh:



Namespaces

- namespace `oln`
- namespace `oln::convert`

8.12.1 Detailed Description

REF: The formulas used here come from “Colour Space Conversions”, IAdrian Ford and Alan Roberts; August 11,1998.

Definition in file `rgbyuv.hh`.

Chapter 9

Olena Page Documentation

9.1 Todo List

Class `oln::topo::combinatorial_map::internal::any< Inf >` FIXME: totally obsolete.

Class `oln::topo::combinatorial_map::internal::anyfunc< U, V, Inf >` FIXME: It has nothing to do there.

Member `oln::io::internal::anything::anything()` FIXME: these constructors are required by swig

Member `oln::io::internal::anything::assign(T &output) const` FIXME: call output.clear()?

Member `oln::topo::inter_pixel::bkd_dir_iter::operator=(U u)` FIXME: I am not sure that this respect the new paradigm.

Class `oln::convert::bound< Output, Exact >` FIXME: is this really useful with new types ?

Member `oln::topo::inter_pixel::internal::dir_traits< 2 >::opposite(ret i)` FIXME: no modulus.

Class `oln::level::f_invert< T >` FIXME: the specialisation is done within the class.

Class `oln::utils::internal::f_to_float_< DestT, SrcT >` FIXME: There should be a way to use the standard conversion, but I failed to find a good one. The problem is that a `color<...>` derived of `vec<..>` but the conversion between color and vec should be done through `c.to_float()`.

Example:

Class `oln::topo::tarjan::obsolete::flat_zone< I >` FIXME: many assertions are missing.

Member `oln::topo::inter_pixel::fwd_dir_iter::operator=(U u)` FIXME: I am not sure that this respect the new paradigm.

Class `oln::io::internal::get_pnm_type< I >` FIXME: this could be done by using labels images eg:
`read(binary_image_with_dim<2>& ima) { // ... }`

Class `oln::utils::histogram< T, CPT, V2P, Exact >` FIXME: An image is inside the histogram. This is incorrect because it is not exactly an image (no border needed).

Class `oln::topo::inter_pixel::interpixel< I >` FIXME: Test the output values in the tests.

Member `oln::topo::inter_pixel::interpixel::folw(const head_type &in) const` FIXME: add doc.

Member `oln::topo::inter_pixel::interpixel::operator[] (const point_type &p) const` FIXME: add doc.

Class `oln::snakes::node< I >` FIXME: Do not work due to the function energy.

Member `oln::snakes::node::energy(const I &gradient, point_type prev, point_type next) const`
 FIXME: not implemented, do not work

Class `oln::io::internal::pnm_read_data< PnmBinary, ReadPnmPlain >` FIXME: implement an iterator over data

Class `oln::io::internal::pnm_read_data< PnmInteger, ReadPnmPlain >` FIXME: implement an iterator over data

Class `oln::io::internal::pnm_read_data< PnmInteger, ReadPnmRaw >` FIXME: implement an iterator over data

Class `oln::io::internal::pnm_read_data< PnmInteger, ReadPnmRaw >` FIXME: implement an iterator over data

Class `oln::io::internal::pnm_read_data< PnmVectorial, ReadPnmPlain >` FIXME: implement an iterator over data

Class `oln::io::internal::pnm_write_data< PnmBinary, WritePnmPlain >` FIXME: implement an iterator over data

Class `oln::io::internal::pnm_write_data< PnmBinary, WritePnmRaw >` FIXME: implement an iterator over data

Class `oln::io::internal::pnm_write_data< PnmInteger, WritePnmPlain >` FIXME: implement an iterator over data

Class `oln::io::internal::pnm_write_data< PnmInteger, WritePnmRaw >` FIXME: implement an iterator over data

Class `oln::io::internal::pnm_write_data< PnmVectorial, WritePnmPlain >` FIXME: implement an iterator over data

Class `oln::io::internal::pnm_write_data< PnmVectorial, WritePnmRaw >` FIXME: implement an iterator over data

Class `oln::snakes::segment< I >` FIXME: Do not work due to the function `node::energy`.

Class `oln::snakes::snake< algorithm >` FIXME: Do not work due to the function `node::energy`.
 FIXME: Add doc & test.

Member `oln::snakes::snake::energy(void) const` FIXME: Do not work due to the function `node::energy`

Member `oln::morpher::super_slicing_morpher::super_slicing_morpher()` create empty constructors for `impl_`, ...

Class `oln::topo::tarjan::tarjan_set< I, aux_data_type >` FIXME: Obsolete since only `obsolete::flat_zone` use it.

Class `oln::morpho::fast::tarjan::tarjan_set< T, ATTRIBUTE, Env >` FIXME: a similar class is defined in `oln/topo/tarjan/union.hh` (`oln::topo::tarjan::tarjan_set`).

Member `oln::io::internal::try_stream_wrappers_in::by_data(T &output, const std::string &name)`
 FIXME: it sounds strange to read wrapped file without matching its extension.

Class `oln::convert::value_to_point< ntg::color< 3, Qbits, S >, Exact >` Could be generalized to n dimensions if there were a trait that give a `pointkd` for a given dimension `k`.

Member `oln::abstract::w_window::add(const abstract::dpoint< dpoint_type > &dp, const weight_type &w=1)`
 FIXME: Add `dpoint` with default weight (multiplication neutral element).

Member `oln::abstract::w_window::add_dp(const abstract::dpoint< dpoint_type > &dp)` FIXME:
 Add `dpoint` with default weight (multiplication neutral element).

Member [oln::w_window2d::w_window2d](#)(const mlc::array2d< I, T2 > &arr) **FIXME:** this constructor is not in [w_window1d.hh](#) nor [w_window3d.hh](#). Is it really useful ? This constructor is used to build a chamfer distance.

Member [oln::abstract::window_base::struct_elt](#)< [Exact](#) > **FIXME:** this has been commented out to satisfy icc and comeau. I don't know who is right between them and gcc.

File [cmap.hh](#) **FIXME:** There is some problems in the directory topo/combinatorial-map/, such as non static functions, or the redefinition of the class any.

FIXME: The documentation is not good enough.

FIXME: Real test are missing. **FIXME:** Some part of this file are inside "#if 0" comments. **FIXME:** This file force to keep an obsolete version of flat-zone.

File [opdecls.hh](#) **FIXME:** These macros should be rewritten / split into real code to make things clearer.

Member [oln::apply2](#)(const abstract::image< I > &input1, const abstract::image< I > &input2)
FIXME: Don't we want to name these functions 'apply()' too?

Member [oln::apply2](#)(const abstract::image< I1 > &input1, const abstract::image< I2 > &input2)
FIXME: Don't we want to name these functions 'apply()' too?

Member [oln::apply2](#)(const abstract::image< I1 > &input1, const abstract::image< I2 > &input2)
FIXME: Don't we want to name these functions 'apply()' too?

Member [oln::apply2](#)(AdaptableBinaryFun f, const abstract::image< I1 > &input1, const abstract::image< I2 > &input2)
FIXME: Don't we want to name these functions 'apply()' too?
FIXME: Don't we want to name these functions 'apply()' too?

Member [oln::fold](#)(AdaptableBinaryFun f, const abstract::image< I > &input) **FIXME:** Ensure that first_argument_type == result_type.

Member [oln::fold](#)(AdaptableBinaryFun f, typename mlc::typeadj< typename AdaptableBinaryFun::result_type >::mu)
FIXME: Ensure that first_argument_type == result_type.

Member [oln::convol::fast::internal::recursivefilter](#)(I &image, const recursivefilter_coef_< FloatType > &c, const oln::point &point, const oln::point &point)
FIXME: Until something clever is designed, the line is defined by two points (START and FINISH) and a displacement dpoint (D).

Member [oln::convol::slow::convolve](#)(const abstract::image< I > &input, const mlc::array2d< Info, Win > &arr)
FIXME: don't use array1d, ..., arraynd.

Member `oln::convol::slow::convolve`(const abstract::image< I > &input, const abstract::w_window< Win > &win)
 FIXME: we must always specify DestValue.

Member `oln::level::frontp_connected_component`(const abstract::binary_image< I > &input, const abstract::neighborhood< N > &n)
 FIXME: Should probably be turned into a class.

Namespace `oln::math` FIXME: I'm not proud of the code below think it could be better..
 FIXME: this code sounds really odd. Why does the operator() take value<Self> instead of Self directly ? FIXME: Self should be renamed into Exact.

Member `oln::morpho::fast_morpho`(const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)
 FIXME: This algorithm should be moved to the namespace `oln::morpho::fast`.
 FIXME: add tests.

Member `oln::morpho::fast_morpho`(const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)
 FIXME: REMOVE ME.

Member `oln::morpho::laplacian`(const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)
 FIXME: Not instantiated in swilena (see tools/swilena/generate_morpho_instantiations.py)

Member `oln::morpho::watershed_seg_or`(const abstract::non_vectorial_image< I1 > &I1, const abstract::non_vectorial_image< I2 > &I2, const abstract::struct_elt< E > &se)
 FIXME: Not instantiated in swilena (see tools/swilena/generate_morpho_instantiations.py)

Member `oln::morpho::fast::laplacian`(const abstract::non_vectorial_image< I > &input, const abstract::struct_elt< E > &se)
 FIXME: Not instantiated in swilena (see tools/swilena/generate_morpho_instantiations.py)

Member `oln::morpho::internal::find_struct_elts`(const abstract::struct_elt< E1 > &se, E2 se_add[mlc::exact< E1 >::r])
 FIXME: add(dp) on w_windows associates a default weight set to 1

Member `oln::topo::mk_chamfer_3x3`(float d10, float d11) FIXME: This highly not thread safe !

Member `oln::topo::mk_chamfer_3x3`(float coef=1.f) FIXME: This highly not thread safe !

9.2 Deprecated List

Class `oln::convert::f_nrgb_to_xyz< inbits, outbits >` A composition should be performed with `nrgb->rgb` and `rgb->xyz`. It has not been replaced within the function because a double conversion 'reduces' the color space. See the following example:

```
// Obsolete:
//
// #include <oln/convert/nrgbxyz.hh>
// #include <ntg/all.hh>
// int main(int argc, char **argv)
// {
//     ntg::nrgb_8 in(100, 60, 64);
//     ntg::xyz_8 out = oln::convert::f_nrgb_to_xyz<8, 8>()(in);
// }
//
// Should be replaced by:
//
#include <oln/convert/rgbxyz.hh>
#include <oln/convert/rgbnrgb.hh>
#include <ntg/all.hh>
int main()
{
    ntg::nrgb_8 in(100, 60, 64);
    ntg::xyz_8 out = oln::convert::f_rgb_to_xyz<8, 8>()
        (oln::convert::f_nrgb_to_rgb<8, 8>()(in));
}
```

Class `oln::convert::f_xyz_to_nrgb< inbits, outbits >` A composition should be performed with `xyz->rgb` and `rgb->nrgb`.

Member `oln::convert::nrgb_to_xyz(const color< 3, inbits, nrgb_traits > &v)` A composition should be performed with `nrgb->rgb` and `rgb->xyz`.

Member `oln::convert::xyz_to_nrgb(const color< 3, inbits, xyz_traits > &v)` a composition should be performed with `xyz->rgb` and `rgb->nrgb`.