The MILENA Image Processing Library

Thierry Géraud, Roland Levillain, Guillaume Lazzara

EPITA Research and Development Laboratory (LRDE)

27/06/2012

The Olena Platform

What? Olena: a platform for Image Processing (IP), including:

- Milena: an IP library in C++
- Some Python bindings over Milena
- Dedicated (applied) modules

  e.g. for Document Image Analysis (DIA)

Who?

- Authors:
  EPITA Research and Development Laboratory (LRDE)
- Users/public: IP practitioners

How?

- Requirements: a C++98/03 (or more recent) compiler
- Distribution:
  - tarballs (`.tar.gz`, `.tar.bz2`)
  - Debian packages (`.deb`)
  - MacPorts package,
  - next winter: RPM package(s) (`.rpm`)

Olena: Model, Organization & Infrastructure

Model Free and Open Source Software

- License: GNU General Public Licence, version 2
- Virtually all material is public (code, documentation, access to tools, etc.)

Infrastructure Based on Free Software

- Public source code repository (Git)
- Automatic build and continuous integration (BuildBot)
- Ticket management/bug tracker (Trac)
- Mailing lists (Mailman)
- Online (HTML & PDF) reference documentation (Doxygen)
- Wiki (TWiki)

Organization

- 2 maintainers (*permanent* staff)
- More than 50 contributors for almost 15 years

A Common Use Case

A couple of algorithms on 2D images...

input image

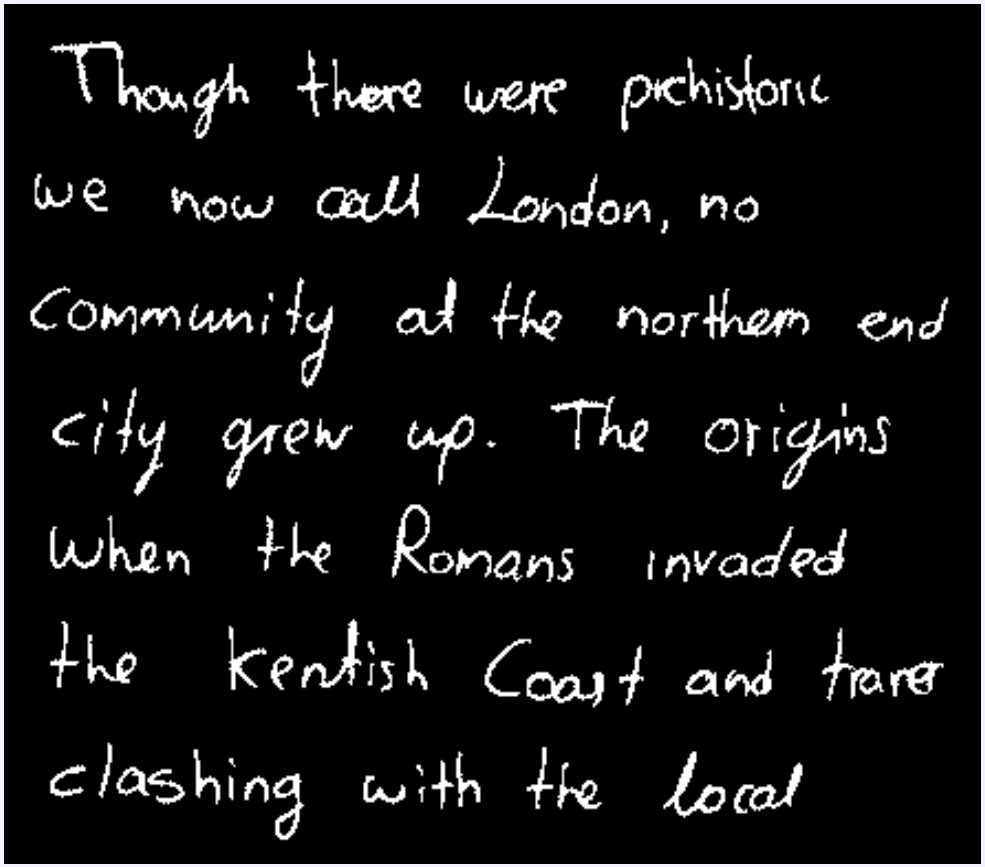
$\longrightarrow$ component labeling

$\longrightarrow$ influence zone transform

...so classical that they are featured by every IP library.

A Common Use Case

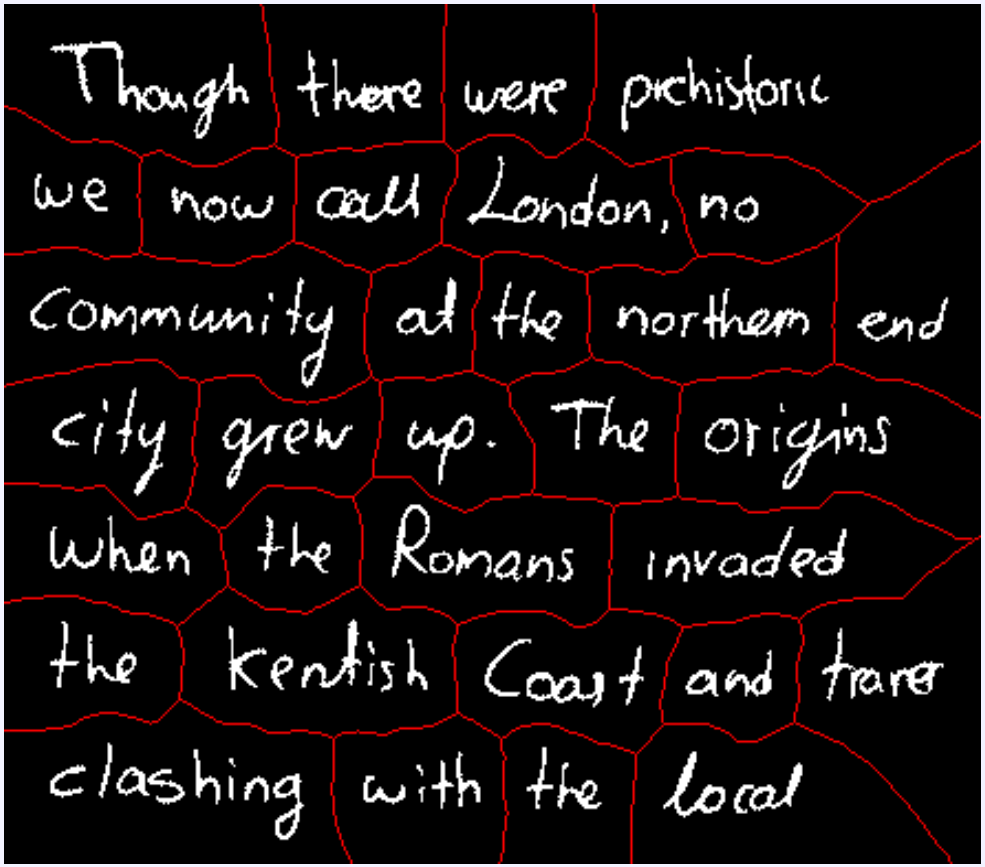
A *very basic* problem to solve: text line identification



input

A Common Use Case

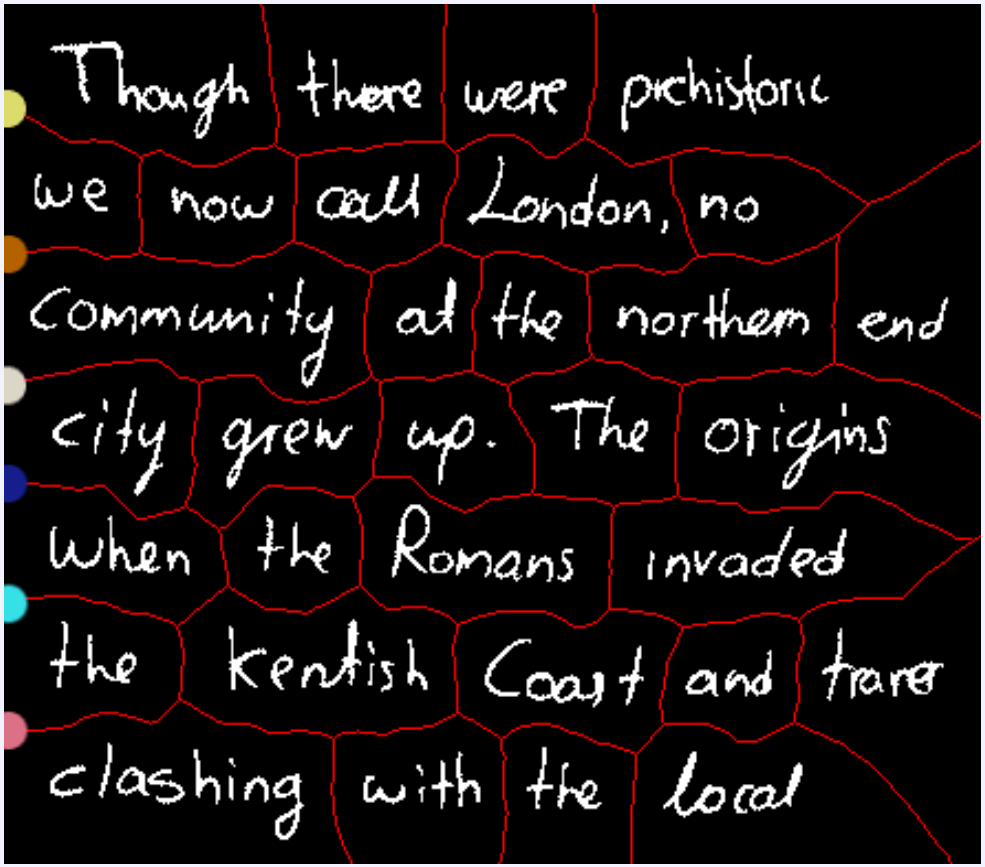
We have this intermediate result:



word segmentation

A Common Use Case

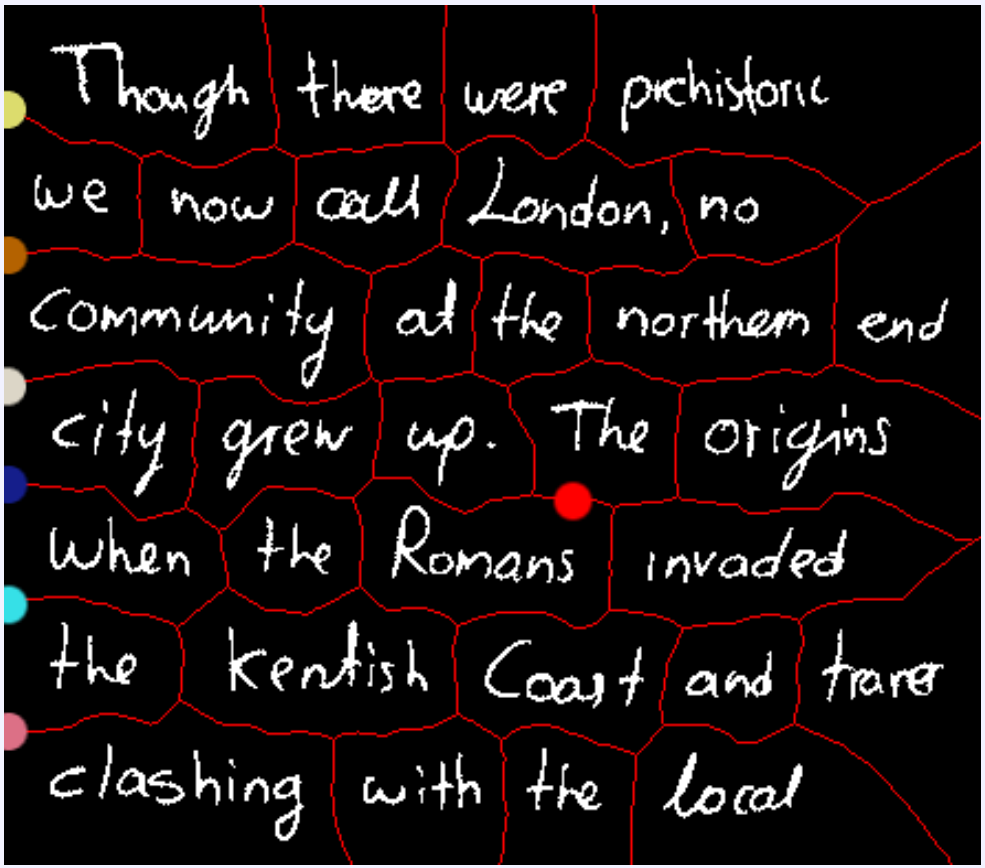
an idea:



identify each line space = label the 1st column points

A Common Use Case

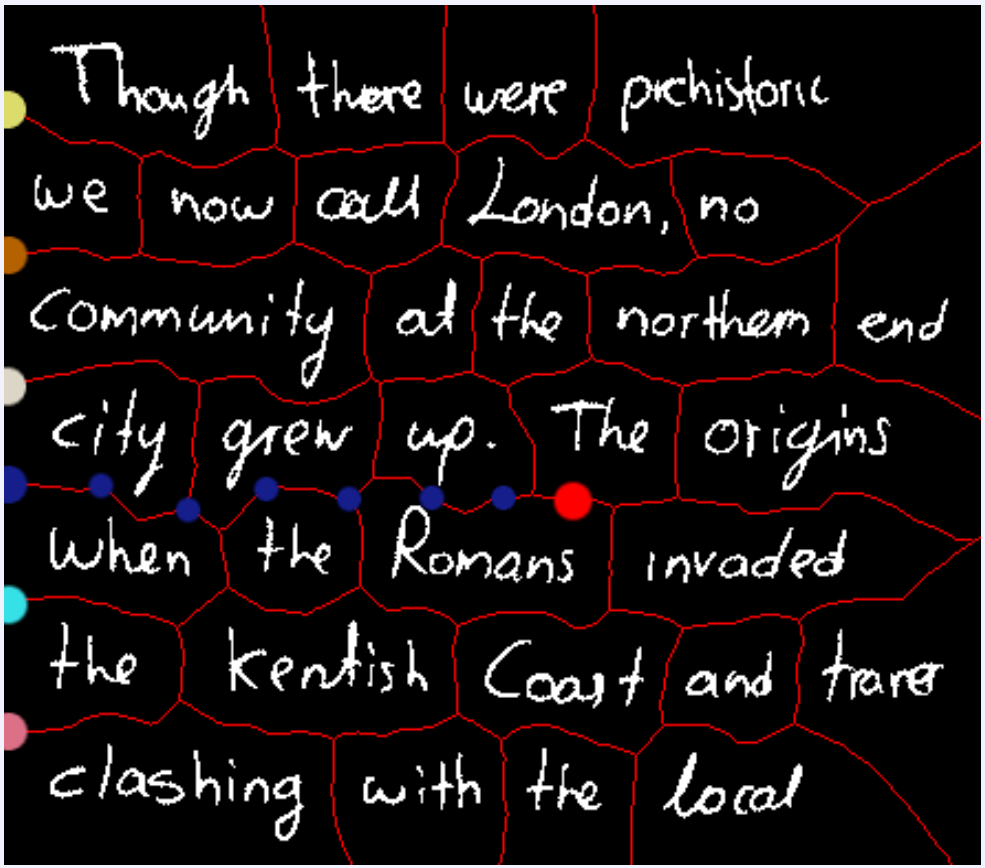
an idea (cont'd):



a point belongs to a given line space...

A Common Use Case

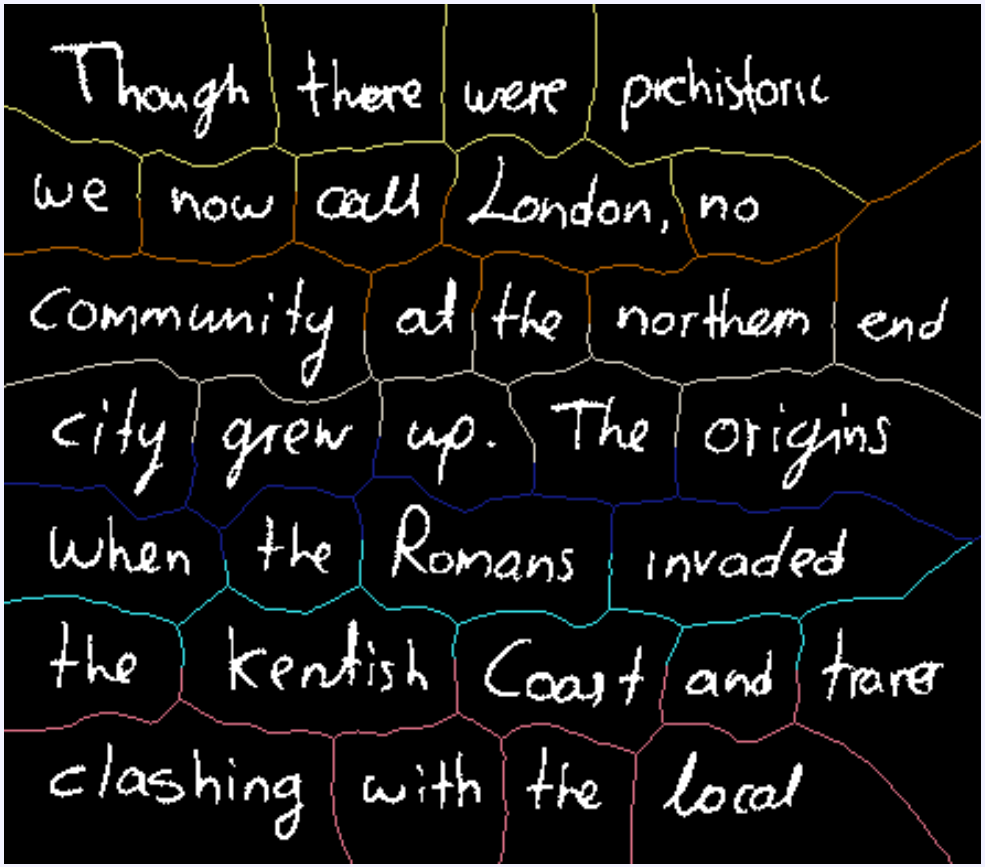
an idea (cont'd):



...if it is in the influence zone of the corresponding label

A Common Use Case

How difficult is it to get this result...



...with commonly used IP software / environments?

A Common Use Case

Possible answers:

- I just cannot do it. :-(

Just pick one.

A Common Use Case

Possible answers:

- I just cannot do it. :-(
- Maybe I can do it but I'm not too sure about that... :-(

Just pick one.

A Common Use Case

Possible answers:

- I just cannot do it. :-(
- Maybe I can do it but I'm not too sure about that... :-(
- I can do it but... I need to copy-paste-modify some code. :-(

Just pick one.

A Common Use Case

Possible answers:

- I just cannot do it. :-(
- Maybe I can do it but I'm not too sure about that... :-(
- I can do it but... I need to copy-paste-modify some code. :-(
- I can do it; yet it will take many steps (and hence some time)... :-(

Just pick one.

A Common Use Case

Possible answers:

- I just cannot do it. :-(
- Maybe I can do it but I'm not too sure about that... :-(
- I can do it but... I need to copy-paste-modify some code. :-(
- I can do it; yet it will take many steps (and hence some time)... :-(
- I can do it in a very few lines lines of code. :-)

Just pick one.

A Common Untruth

"If I need to process some particular images

- e.g., parts of images, 3D images, videos, or huge 2D images,
- 2D+t data, n-D data, graphs, meshes, etc.

it's normal:

- that my tool can't do it,
- that I should get a specific tool."

A Common Untruth

"If I need to process some particular images

- e.g., parts of images, 3D images, videos, or huge 2D images,
- 2D+t data, n-D data, graphs, meshes, etc.

~~it's normal:~~

- ~~that my tool can't do it,~~
- ~~that I should get a specific tool.~~" No, that's NOT normal at all!

A Common Untruth

"If I need to process some particular images

- e.g., parts of images, 3D images, videos, or huge 2D images,
- 2D+t data, n-D data, graphs, meshes, etc.

~~it's normal:~~

- ~~that my tool can't do it,~~
- ~~that I should get a specific tool.~~" No, that's NOT normal at all!

Some clues:

- a tool has a perimeter *but* this perimeter should *not* to be so limited...
- do *not* accept to be so limited, just change your tool!

Main Objective

The MILENA Library

Favor the potential for having features (idea of *capability / non-limitation*) versus the number of provided features (idea of *immediate availability*).

Main Objective

The MILENA Library

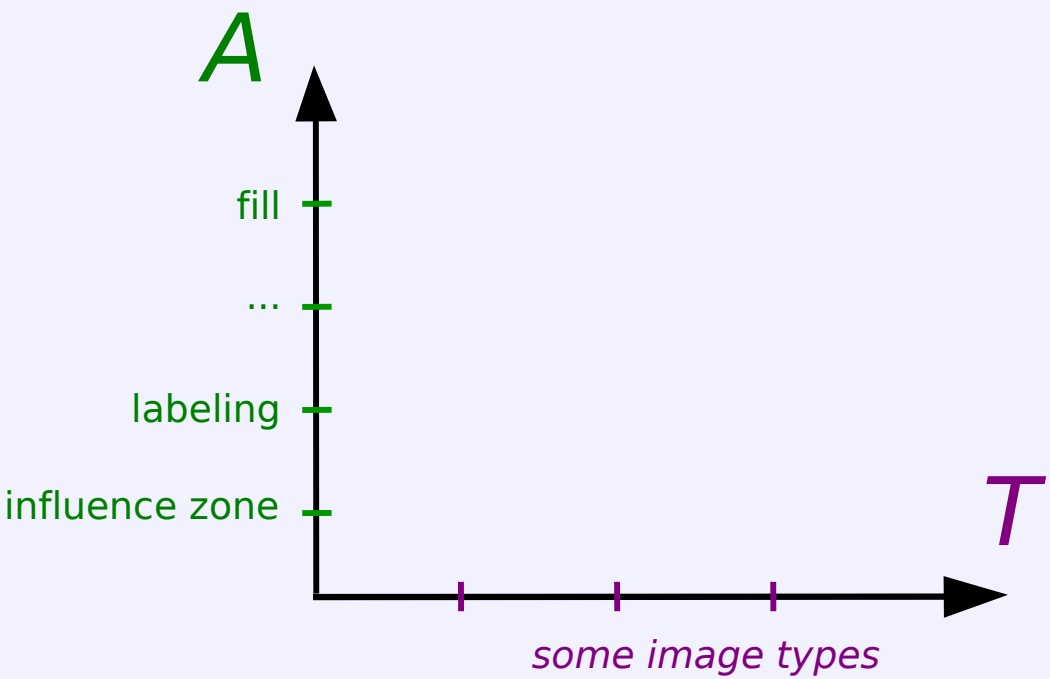
Favor the potential for having features (idea of *capability / non-limitation*) versus the number of provided features (idea of *immediate availability*).

Thus we propose:

- a library that is *not* dedicated
  - to an IP applicative domain,
  - to a class of IP operators.
- a library where any theoretically possible processing...

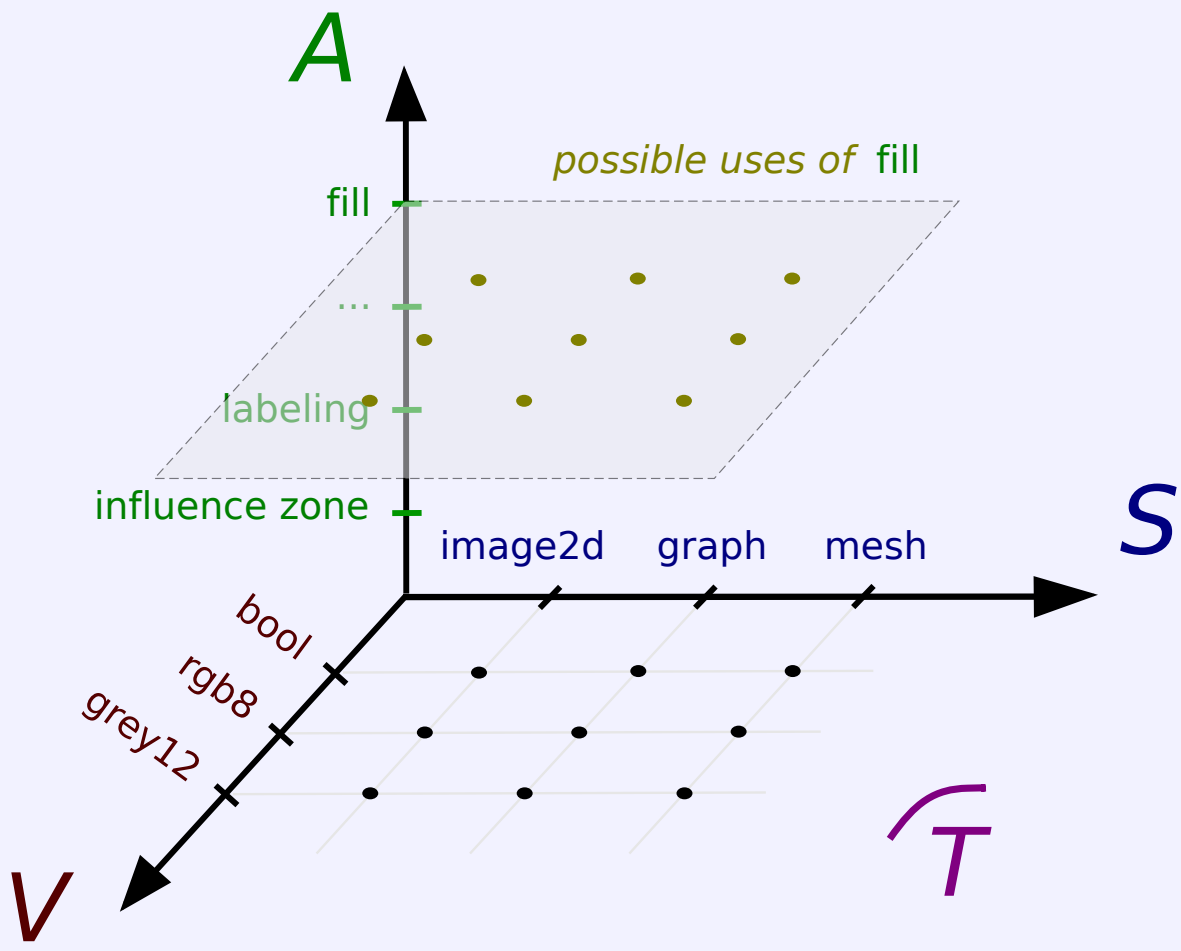
...is (or will be) possible in practice!

Genericity



classical decomposition: algorithms and image types

Genericity



not to be limited

= covering *as much as possible* the space of possibilities

Genericity

Myths:

- "with a generic library, one is forced to code in a generic way" (no)
- "if it's generic, it's much pain for the user" (no)

Genericity

Myths:

- "with a generic library, one is forced to code in a generic way" (no)
- "if it's generic, it's much pain for the user" (no)
- "a library is either generic or not" (no)
- "to obtain a generic lib, adding the `template` keyword suffices" (no)

Genericity

Myths:

- "with a generic library, one is forced to code in a generic way" (no)
- "if it's generic, it's much pain for the user" (no)
- "a library is either generic or not" (no)
- "to obtain a generic lib, adding the `template` keyword suffices" (no)

MILENA features

- make it and keep it simple for the user,
- efficient so usable in industrial context,
- an effective high level of genericity.

(and that was *very hard* to get those three features at the same time...)

About User-Friendliness

- Provide many value types + image types + other IP classical entities,

About User-Friendliness

- Provide many value types + image types + other IP classical entities,

  $\longrightarrow$ the user just has to code IP solutions or algorithms .

About User-Friendliness

- Provide many value types + image types + other IP classical entities,

  $\longrightarrow$ the user just has to code IP solutions or algorithms .

- Provide many tools so that

About User-Friendliness

- Provide many value types + image types + other IP classical entities,

  $\longrightarrow$ the user just has to code IP solutions or algorithms .

- Provide many tools so that
  - an IP chain means "passing objects to procedures",

About User-Friendliness

- Provide many value types + image types + other IP classical entities,

  $\longrightarrow$ the user just has to code IP solutions or algorithms .

- Provide many tools so that
  - an IP chain means "passing objects to procedures",

    $\longrightarrow$ just like in C

About User-Friendliness

- Provide many value types + image types + other IP classical entities,

  $\longrightarrow$ the user just has to code IP solutions or algorithms .

- Provide many tools so that
  - an IP chain means "passing objects to procedures",

    $\longrightarrow$ just like in C
  - the writing of algorithms is very close to the practitioner's language.

About User-Friendliness

- Provide many value types + image types + other IP classical entities,

  $\longrightarrow$ the user just has to code IP solutions or algorithms .

- Provide many tools so that
  - an IP chain means "passing objects to procedures",

    $\longrightarrow$ just like in C
  - the writing of algorithms is very close to the practitioner's language.
- Keep it simple!

About User-Friendliness

- Provide many value types + image types + other IP classical entities,

  $\longrightarrow$ the user just has to code IP solutions or algorithms .

- Provide many tools so that
  - an IP chain means "passing objects to procedures",

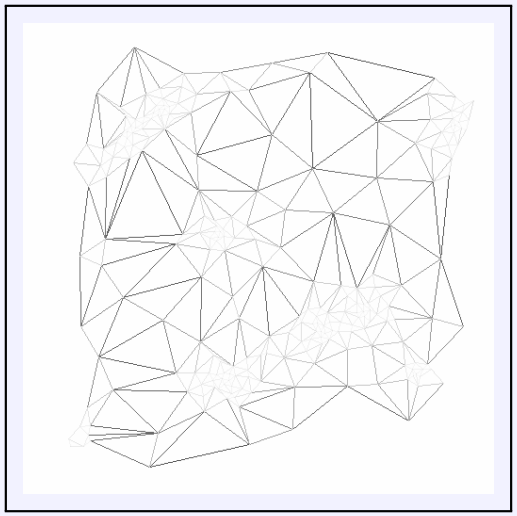
    $\longrightarrow$ just like in C
  - the writing of algorithms is very close to the practitioner's language.

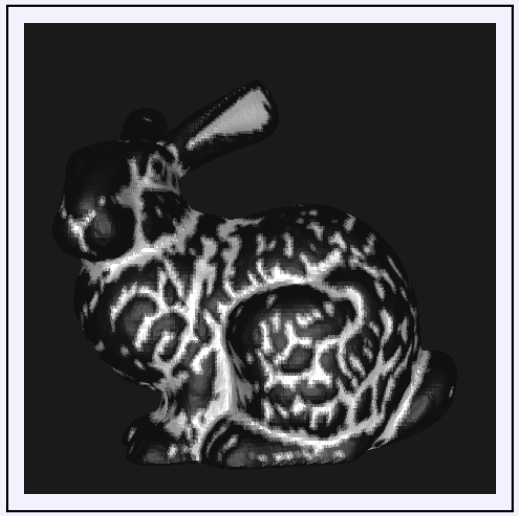
- Keep it simple!

  $\longrightarrow$ so that the user can *really* focus on IP (not on programming).

Sample Runs

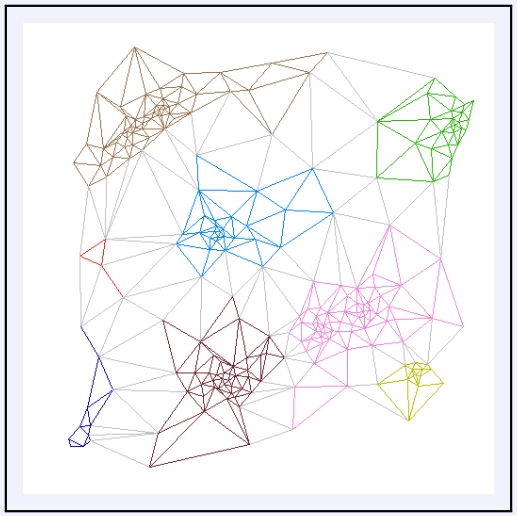
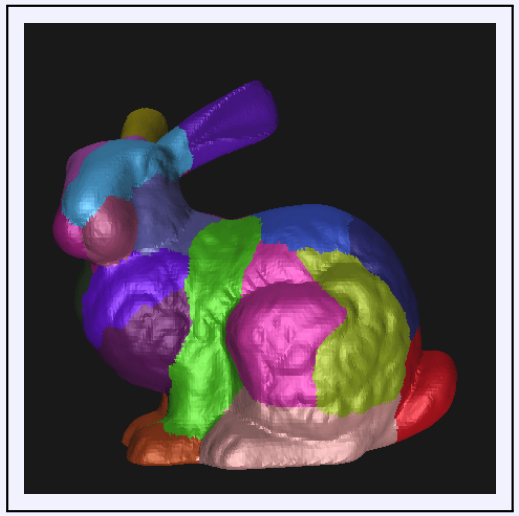
Some kinds of images we may *need* to process:





Sample Runs

Some results that we *can* get:

In MILENA, the very same algorithm runs on those images.

Hum...

YADAI!

Hum...

Yet Another *Dilation* Algorithm Implementation!

sorry...

Ease of Use

Translation of an algorithm

$\mathcal{D}$: domain of *ima*

$\mathcal{N}$: neighborhood

V : type of values of *ima*

$\forall\, p \in \mathcal{D}$

$out(p) \leftarrow \inf(V)$

$\forall\, n \in \mathcal{N}(p)$

$out(p) \leftarrow \sup(out(p),\ ima(n))$

Ease of Use

Translation of an algorithm into C++ with MILENA:

▼ ▼

$\mathcal{D}$: domain of *ima*

$\mathcal{N}$: neighborhood

V : type of values of *ima*

$\forall\, p \in \mathcal{D}$

  $out(p) \leftarrow \inf(V)$

  $\forall\, n \in \mathcal{N}(p)$

    $out(p) \leftarrow \sup(out(p),\ ima(n))$

```
mln_piter(I)    p(D);        // p ∈ D
mln_niter(N)    n(N, p);     // n ∈ N(p)
typedef  mln_value(I)  V;
```

```
for_all(p)
  out(p)  =  mln_inf(V);
  for_all(n)  if (ima.has(n))
    out(p)  =  sup(out(p), ima(n));
```

Ease of Use

Translation of an algorithm into C++ with MILENA:

▼ ▼

$\mathcal{D}$: domain of *ima*

$\mathcal{N}$: neighborhood

V : type of values of *ima*

$\forall\, p \in \mathcal{D}$

  $out(p) \leftarrow \inf(V)$

  $\forall\, n \in \mathcal{N}(p)$

    $out(p) \leftarrow \sup(out(p),\ ima(n))$

```
mln_piter(I)    p(D);       // p ∈ D
mln_niter(N)    n(N, p);    // n ∈ N(p)
typedef  mln_value(I)  V;
```

```
for_all(p)
  out(p)  =  mln_inf(V);
  for_all(n)  if (ima.has(n))
    out(p)  =  sup(out(p), ima(n));
```

Highly generic / Easy to read / Really that hard to write?

Tackling Practical Problems

- Different domains:
  - medical (anatomic pathology, 2D MRI, 2D+t MRI, ultrasound),
  - astronomy / satellite images, $\longrightarrow$ what huge images!
  - document images / natural images,
  - 3D objects / meshes,
  - videos.
- Different tasks:
  - filtering, registration, segmentation,
  - pattern recognition.
- Industrial products integrating IP solutions based on MILENA:

Tackling Practical Problems

- Different domains:
  - medical (anatomic pathology, 2D MRI, 2D+t MRI, ultrasound),
  - astronomy / satellite images, $\longrightarrow$ what huge images!
  - document images / natural images,
  - 3D objects / meshes,
  - videos.
- Different tasks:
  - filtering, registration, segmentation,
  - pattern recognition.
- Industrial products integrating IP solutions based on MILENA:

Mandriva nuxeo WALLIX Infrastructure and Security Solutions XWIKI™

We've said we **favor capabilities**,
yet we also have some (in fact, many) features **immediately available**.

Back to the Initial Example

What we need:

'label' is a 2D image of labels having the same domain as the 'input' image
fill 'label' with zeros

('connected component labeling' step)
label the points of watershed line *restricted to* the first column
and paste the result into 'label'

('influence zone transform' step)
compute the influence zones of 'label' *only on* the watershed frontiers
and paste the result into 'label'

('visualization' step)

'output' is a colorization with RGB of the 'label' image

fill in white the 'output' image *only for* object points

that is, just some IP classic work!

Back to the Initial Example

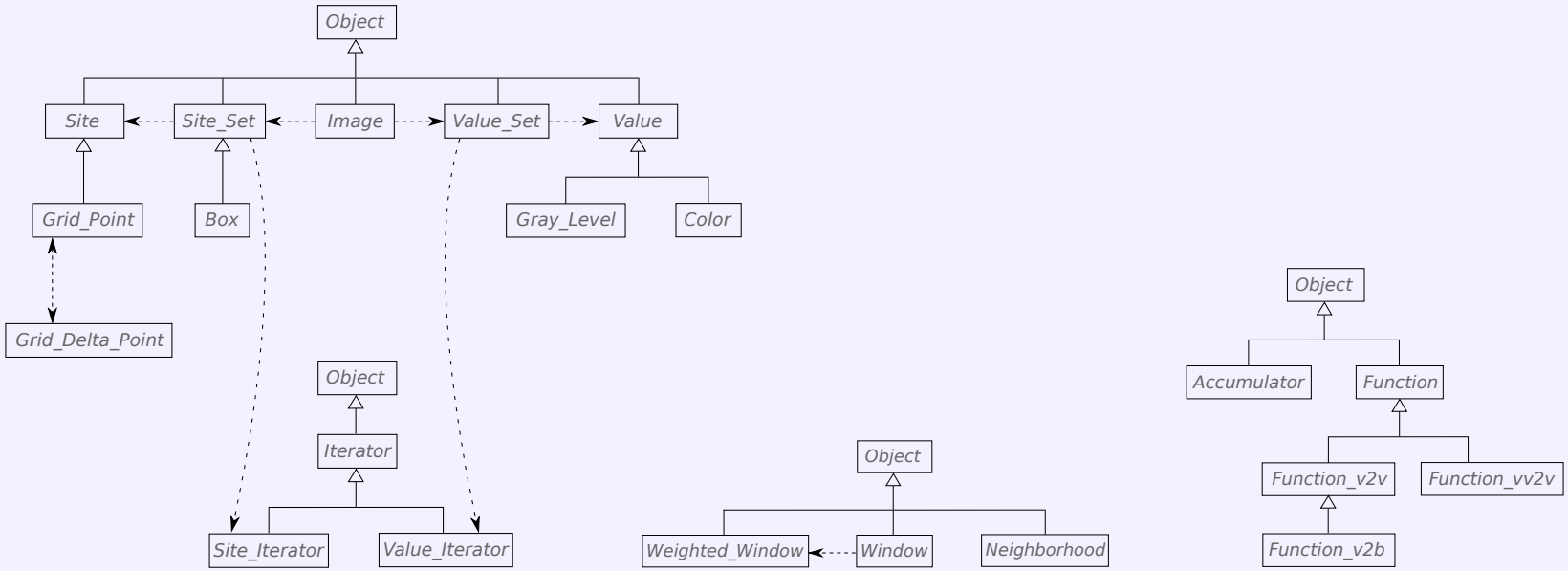
The actual code with milena:

```
image2d<unsigned> label(input.domain());
data::fill(label, 0);

// labeling
unsigned nlabels;
data::paste(labeling::value(ws | make::box2d(0,0,input.nrows()-1,0),
                            0, c4(), nlabels),
            label);

// influence zone computation
data::paste(transform::influence_zone(label | (pw::value(ws) == 0),
                                      c8(), w_win2d_5_7_11),
            label);

// visualization
output = labeling::colorize(value::rgb8(), label);
data::fill((output | pw::value(input)).rw(), literal::white);
```

that is, the IP classic work applied!

Insight of the Core Abstractions

The few basic concepts (image, neighborhood, etc.) used in IP are mapped into MILENA:



That's not many to "learn"...

State of the Project

- Mature project: usable as-is, right now
  - Stable core
  - Late work emphasized on algorithms and applications
  - Some cleanup tasks planned though
- Future Work
  - Make users' life easier
    - ★ Better documentation (tutorials, examples, etc.)
    - ★ More non-C++ APIs (Python, Ruby, etc.)
    - ★ More non-programming tools (GUIs, etc.)
  - Formal work on abstractions, properties and algorithms
    - ★ < see publications on our web site >

The End

Not the end my friend! The project is living and keeps growing...

Any questions?

olena@lrde.epita.fr

http://olena.lrde.epita.fr