Morphology on color images
Theory and application in Milena

Alexandre "Goulagman" Abraham
abraham@lrde.epita.fr

LRDE – EPITA Research & Development Laboratory

January 14, 2009
1. Introduction

2. Mathematical morphology
   - Theory
   - Implementation in Milena

3. Color spaces

4. Morphology on color images
   - Theory
   - Implementation in Milena
   - Algorithms and applications
1 Introduction

2 Mathematical morphology

3 Color spaces

4 Morphology on color images
What are Olena and Milena?
What is mathematical morphology? What is a color?
What is mathematical morphology on color images?
How can we implement it in Milena?
All processed images of this presentation have been treated thanks to Milena.

In all sample images, white pixels represent the object and black pixels the background.
Outline

1 Introduction

2 Mathematical morphology

3 Color spaces

4 Morphology on color images
1. Introduction

2. Mathematical morphology
   - Theory
   - Implementation in Milena

3. Color spaces

4. Morphology on color images
Quick definition
Principle and purpose

- Focuses on shapes
- Notion of structuring element
- Generalized to complete lattices

Basic operators:
- Dilation
- Erosion
Basic operators

Dilation

Sample image

Structuring element

Dilated image
Basic operators

Dilation

Sample image

Structuring element

Dilated image
Basic operators

Dilation

Sample image

Dilated image

Structuring element
Basic operators

Dilation

Sample image

Structuring element

Dilated image
Basic operators

Dilation

Sample image

Structuring element

Dilated image
Basic operators

Dilation

Sample image

Dilated image

Structuring element
Basic operators

Dilation

Sample image

Structuring element

Dilated image
Basic operators

Dilation

Sample image

Structuring element

Dilated image
Basic operators

Dilation

Sample image

Structuring element

Dilated image

Introduction Mathematical morphology Color spaces Morphology on color images
Basic operators

Dilation

Sample image

Structuring element

Dilated image
Basic operators

Dilation

Sample image

Structuring element

Dilated image
Basic operators

Dilation

Sample image

Structuring element

Dilated image
Basic operators

Erosion

Sample image

Structuring element

Eroded image
Basic operators

Erosion

Sample image

Structuring element

Eroded image

Introduction Mathematical morphology Color spaces Morphology on color images
Basic operators

Erosion

Sample image

Structuring element

Eroded image

Introduction Mathematical morphology Color spaces Morphology on color images
Basic operators

Erosion

Sample image

Structuring element

Eroded image
Basic operators

Erosion

Sample image

Structuring element

Eroded image
Basic operators

Erosion

Sample image

Structuring element

Eroded image
Basic operators

Erosion

Sample image

Structuring element

Eroded image
Basic operators

Erosion

Sample image

Structuring element

Eroded image

Introduction Mathematical morphology Color spaces Morphology on color images
Basic operators

Erosion

Sample image

Structuring element

Eroded image
Basic operators

Erosion

Sample image

Structuring element

Eroded image
Basic operators

Erosion

Sample image

Structuring element

Eroded image
Basic operators

Erosion

Sample image

Structuring element

Eroded image
Basic operator composition:

- Opening: erosion followed by dilation
- Closing: dilation followed by erosion
- Hit or miss: erosion by a pair of disjoint structuring elements

Morphological filters:

- Segmentation: watershed...
- Levelings
- Skeletonization
Examples

Closing example

Squamous epithelium cells

Processed image: cell nuclei are removed

Structuring element
### Examples

hit-or-miss example

**Blood cells**

![Blood cells](image)

**Processed image: red blood cells are spotted**

![Processed image](image)

**Structuring elements**
Examples
hit-or-miss example

Blood cells

Processed image: red blood cells are spotted

Structuring elements
1. Introduction

2. Mathematical morphology
   - Theory
   - Implementation in Milena

3. Color spaces

4. Morphology on color images
I morphological_algorithm (Image<I> input, Window<W> output, Operator<O> op) {
    // Output image
    I output;

    // Accumulator that take value of neighbours and
    // output the result of an operation
    A accu = op.accu(ima);

    // Fill the border with a neutral value not to
    // have to test each time if we are on a border
    border::fill(input, op.neutral());

    mln_piter(I) p(input.domain); // Psite iterator
    mln_qiter(W) q(win, p);       // Psite neighbor iterator

    for_all(p) {
        accu.init();
        for_all(q) if (input.has(q))
            accu.take(input(q));
        output(p) = accu.to_result();
    }

    return output;
}
```cpp
struct erosion_op
{
    template <typename I>
    mln_morpho_select_accu(I, land_basic, min)
    accu(const Image<I>&) const
    {
        mln_morpho_select_accu(I, land_basic, min) tmp;
        return tmp;
    }

    template <typename I>
    mln_morpho_select_accu(I, land, min_h)
    accu_incr(const Image<I>&) const
    {
        mln_morpho_select_accu(I, land, min_h) tmp;
        return tmp;
    }

    template <typename I>
    mln_value(I) neutral(const Image<I>&) const
    {
        return internal::neutral<I>::supremum();
    }
};
```

Basic operator needed for erosion.
1. Introduction
2. Mathematical morphology
3. Color spaces
4. Morphology on color images
Characteristics

- A value for each color (Red, Green, Blue)
- Most used color space
- Far from human perception

RGB color space represented by a cube
Characteristics

- Cyan, Magenta, Yellow, Key
- Subtractive model
- Used by print houses
Characteristics

- Hue, Saturation, Lightness/Value
- Closest to human perception
- Well suited for image processing

HSL and HSV color spaces
Characteristics

- One luminance (Y) and two chrominances (UV)
- Used for data transmission
1. Introduction
2. Mathematical morphology
3. Color spaces
4. Morphology on color images
What is needed
... to perform mathematical morphology on any kind of data

- Considering each component like a plain image (e.g. : red erosion)
- A (partial) order on the set of data and, if possible, a minimum and a maximum (or an infimum and a supremum)

Warning

Some minimum operators may introduce non previously existing values in the image (e.g. \( \min((2, 12, 7), (15, 1, 5)) = (2, 1, 5) \)). The choice of this behavior is done according to the purpose. Keeping values previously existing in the image is generally better if it is a preprocessing filter. Creating new colors looks better to the human eye.
Introduction

Mathematical morphology

Color spaces

Morphology on color images
  - Theory
  - Implementation in Milena
  - Algorithms and applications
Multivariate data ordering

Lexicographical order

Principle
- For a $n$-dimensional data, compare the first value, if not equal, compare the second and so on.
- This is a total order (if the values are totally ordered).

Restriction
A well suited color space is required such as HSI.
Multivariate data ordering
Scalar valued functions

Principle and characteristics
- Based on the comparison of a linear combination of the components
- Treat each value according to its importance
- This is a partial order.

Example for RGB values
Human eyes are more sensitive to the modification of the green component than red or blue one. There are commonly used values to order RGB values.

\[
a \leq b \iff \begin{cases} 
  v_a = 0.3 \times a_{red} + 0.6 \times a_{green} + 0.1 \times a_{blue} \\
  v_b = 0.3 \times b_{red} + 0.6 \times b_{green} + 0.1 \times b_{blue} \\
  v_a \leq v_b 
\end{cases}
\]
Distance Computation
...can we get rid of data ordering?

Principle
Instead of using an order, one can simply try to minimize the distance between the center of the structuring element and its neighbors.

Example
Later we describe an algorithm using this method and computing an Euclidean distance between RGB values.
Function types

- **v2v function**: one way function (modulus...)
- **v2w2v function**: or bijective function, a function that has an inverse (cos/arccos, ...)
- **v2w_w2v function**: or two-way function, a function that can be inversed knowing the previous state of the value (norm, ...).

Problem

- The way to access the red component of a value may be different form one type to another
- Value may be read only

⇒ Creation of meta-function concept
Example of custom red function

```cpp
struct custom_rgb
{
    typedef int value;
    value red;
    value green;
    value blue;
};

struct function< meta::red< custom_rgb > > : public Function_v2w_w2v<function< meta::red< custom_rgb > > >
{
    typedef typename custom_rgb::value result;
    result read(const custom_rgb& c)
    {
        return c.red;
    }

    typedef typename custom_rgb::value& lresult;
    lresult write(custom_rgb& c)
    {
        return c.red;
    }
};
```
We want it to be:

- simple
- type safe
- generic and efficient (as usual)

```cpp
struct not_comparable {
  int value;
};

struct make_comparable {
  bool less(not_comparable &lhs, not_comparable &rhs) {
    return lhs.value < rhs.value;
  }
};
```
template <typename T, typename O>
struct mixin: T
{
};

template <typename T, typename O>
bool operator <(mixin<T, O> &lhs, mixin<T, O> &rhs)
{
    static O op;
    return op.less(lhs, rhs);
}
template <typename T, typename O>
struct mixin: T
{
};

template <typename T, typename O>
bool operator< (mixin<T, O> &lhs, mixin<T, O> &rhs)
{
    static O op;
    return op.less(lhs, rhs);
}

We need a way to cast not_comparable into mixin.

template <typename T, typename A>
T & violent_cast(A & a)
{
    return *(T*) (void*)(&a);
}
int main ()
{
    not_comparable a, b;
    a.value = 3;
    b.value = 5;

    typedef mixin<not_comparable, make_comparable> comp;
    std::cout << (violent_cast<comp>(a) < violent_cast<comp>(b));
}

Caution

The violent cast works because the size of the two objects are the same. One can obviously not add an attribute to an object, it would segfault (this assertion is made at compilation in Milena).
struct red_only
{
    value::rgb8 max () const
    {
        return value::rgb8(255, 0, 0);
    }

template <unsigned n>
bool less(const value::rgb<n>& a, const value::rgb<n>& b)
{
    return a.red() < b.red();
}
};

Operator needed for red erosion
1 Introduction

2 Mathematical morphology

3 Color spaces

4 Morphology on color images
   - Theory
   - Implementation in Milena
   - Algorithms and applications
Vector Median Filter
an efficient denoising filter

Principle
Minimize the distance between the center of the structuring element and all its neighbors.

\[ \sum_{i \in N} || \vec{v}_{vm} - \vec{v}_i || \leq \sum_{i,j \in N} || \vec{v}_j - \vec{v}_i || \]

Advantages
- Keep image edge
- Do not create new colors
Vector Median Filter applied on Lena

(a) Original image  (b) Vector Median Filter  (c) Gaussian Blur

Figure: Comparative of denoising filters. Size of window is $9 \times 9$. 

Introduction Mathematical morphology Color spaces Morphology on color images
A new canvas
more and more complexity!

**General canvas**

For each point \( p \), all the neighbors \( q \) are treated. Complexity is \( O(n \times s) \) (with \( n \) size of the image and \( s \) size of the structuring element).

**New canvas**

For each point \( p \), for each neighbor \( q \), each neighbor \( q' \neq q \) is treated. Complexity is \( O(n \times s^2) \).
A new canvas
simplified code

```c
morphological_algorithm (Image I input, Window W output, Operator O op)
{
    I output;

    // Accumulators
    A accu = op.accu(ima);
    A naccu = op.accu_neighb(ima);

    // Psite iterator
    mln_piter(I) p(input.domain);

    // Psite neighbor iterators
    mln_qiter(W) q(win, p), r(win, p);

    for_all(p) {
        accu.init();
        for_all(q) if (input.domain().has(q)) {
            naccu.init(q);
            for_all(r) if (input.domain().has(r))
                naccu.take(input(r));
            accu.take(naccu.to_result());
        }
        output(p) = accu.to_result();
    }

    return output;
}
```
Principle

The leveling is a morphological filter which purpose is to “cut crests” and “fill gaps”. Basically, it is just reducing all values of an image that is greater than a maximum value to this maximum value, creating “flat levels”.

Figure: An example of leveling, the crests are cut.
Leveling according to a function

**Principle**

$g$ is a marker function. The rule is simple: if $g \leq f$ (resp. $g \geq f$), the leveling must create a flat zone that is as low (resp. high) as possible without passing under (resp. above) the $g$ function.

**Figure:** Result of the leveling according to the dotted marker function is dashed.
Autarkical Leveling
a little more complicated

Preamble

- Algorithm from [Gomila and Meyer, 1999]
- Does not create any color (that is why it is called autarkical)
- Colors are viewed as 3D points

Notations and variables

$I$ is the input image
$M$ is the Vector Median Filter of $I$

$I_p$ is the data of $I$ at point $p$
$N_p$ is the neighborhood of point $p$
Same side

Three points $a, b, c$ are on the same side if the angle $\widehat{abc}$ is acute. That is to say if $a$ does not belong to the disk of diameter $[bc]$. It will be written $\sim abc$.

Closest vector

The closest vector to $I_p$ is the vector $M_q$ that minimises its Euclidean distance from $I_p$ such as $M_q = \min_{q \in N_p} ||\vec{I}_p - \vec{M}_q||$. It will be written $\hat{I}_p$. 

**Autarkical Leveling**

the algorithm (at last)

---

**Formal definition**

\[
\forall p \in I, I_p = \begin{cases} 
\hat{I}_p & \text{if } \forall (q, q') \in N_p \mid q \neq q', I_p \overbrace{M_q M_{q'}}^\sim \\
I_p & \text{otherwise}
\end{cases}
\]

This operation is repeated until stability.

---

**Implementation**

Even if the algorithm seems to follow the canvas described above, it cannot be implemented as is easily. A future work may be the integration of this algorithm in the canvas.
Autarkical Leveling

Example

(a) Original image  (b) Vector Median Filter  (c) Autarkical Leveling

Figure: Après le bain, femme s’essuyant la nuque, Degas, 1898.
Autarkical Leveling

Another example

(a) Original image  (b) Vector Median Filter  (c) Autarkical Leveling

Figure: The toilette, Toulouse-Lautrec, 1896.
**Conclusion**

toward the end of the... presentation

<table>
<thead>
<tr>
<th>Work done</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Fun images have been improved</td>
</tr>
<tr>
<td>- It is now possible to use mathematical morphology on color images</td>
</tr>
<tr>
<td>- Some algorithms have been implemented like red erosion or autarkical leveling</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Future work</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Consider the viability of the new canvas</td>
</tr>
<tr>
<td>- Implement more tests</td>
</tr>
<tr>
<td>- Furnish default operators to apply morphological filters on color images</td>
</tr>
</tbody>
</table>
Any question?
It’s now or never!

CAPTAIN ANSWER


Introduction aux représentations d’images couleur et à leur traitement.

Color types in milena.

Fast median and bilateral filtering. Special Interest Group in GRAPHics.

Wiktionary (2003). 
morphology. Website. 