Morph-M: Image Processing Library Specialized in Mathematical Morphology

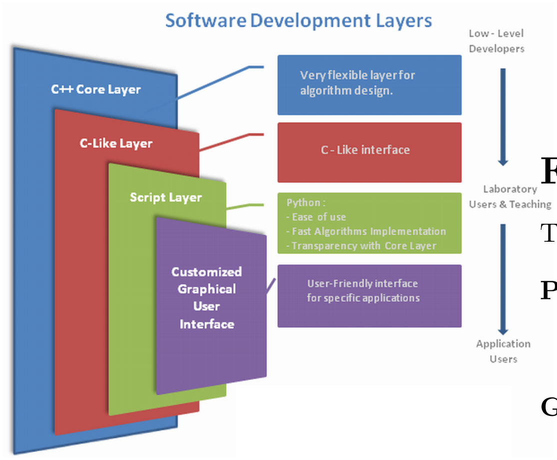
Serge Koudoro[*], Matthieu Faessel[*], and Michel Bilodeau[*]

[*]CMM-Centre de Morphologie Mathématique, MINES Paristech, France

The aim of Morph-M is to provide a rich environment to develop morphological algorithms for image processing. The main component of Morph-M is an image processing library implemented on C++ following the principles of generic programming. Morph-M focuses on Mathematical Morphology so this library contains most of the operators offered by mathematical morphology, from basic operations, such as dilations and erosions, up to the most powerful operators, such as the hierarchical watershed.

Modular Architecture

The modular architecture of Morph-M allows easy adaption of software to different levels of user experience:


Software Development Layers
C++ Core Layer
C-Like Layer
Script Layer
Customized Graphical User Interface
Very flexible layer for algorithm design.
C - Like interface
Python :
- Ease of use
- Fast Algorithms Implementation
- Transparency with Core Layer
User-Friendly interface for specific applications
Low - Level Developers
Laboratory Users & Teaching
Application Users

C++ template layer The lowest level with direct access to the raw image data. It's offers very flexible tools for algorithm design, at the cost of complexity. Common algorithms are well factored and can be easily extended, exotic data types can be used, etc.

C-style interface layer The aim of this layer is to make the most commonly used C++ template algorithms available to non template-aware tools. In order to do this, it instanciates templates for the most common types and uses

Interpreted language layer This is currently the uppermost layer of Morph-M. At the moment, Python is the only supported language. The aim of this layer is to provide a convenient environment for developers to prototype algorithms and for users to bind common algorithms together.

Features

The main features of Morph-M are :

Portability Morph-M can be used on differents plateforms (32bits or 64bits) : Windows, Linux and Mac.

Genericity Morph-M offers a large choice of processing regarding the image type and the structuring element. Morph-M is not designed to be a fast library. Morph-M is designed to be a powerful library in which developers can quickly develop algorithms by encouraging code re-use and generic components.However, there are some elegant ways in which some optimized functions can be seamlessly provided.

Robustness Nightly regression tests assure the correct functionning of each procedure.

Extensible Myriad of addons, provide connections with several library (vtk,opencv,numpy,....)

Licence Proprietary for exterior. Python interface Free for Mines Paristech personnal.

Content

- Images Structure
  - multi-dimensional image data for desired dimensions
  - templated image data structures for pixel type abstraction.
  - several image file formats avalaible: PNG,TIFF,BMP,JPEG,VTK,...
- Structuring Element
  - Myriad of predefined Structuring element
  - Easy use and easy manipulation of SE Iterator.
  - Multi-dimentionnal structuring element.
  - Dynamic Structuring Element
  - Image-based Structing Element
  - Structuring Elements following motion
  - Neighborhood based Generic operations
- Morphological Operation
  - Criteria based morphology (AreaClosing,...)
  - Basic morphological operators(Erode,...)
  - Distance functions and Geodesic operators
  - Lexicographical morphology
  - Morphological filters and measures
  - Labelling and Leveling
  - Morphological Segmentation
- Image Processing
  - Arithmetics and logics
  - Color conversion and manipulations
  - Geometrics transformations (Drawing,rotation,...)
  - Pixel-wise generic operatior
- Filters
  - Convolution filters
  - Diffusion filters
  - Noisifying filters
- Statistics Tools
  - Kriging
  - Linear algebra
  - Morphological Measures (Granulometry,...)
  - Usual statistics (mean, variance, ...)
  - Histograms and Counting (theshold inter variance class, ...
- Graphs and Addons
  - Morphology based on graph and Tree
  - Graphs Cuts and Graph Manipulation
  - FFT, Skeleton, ...

Example

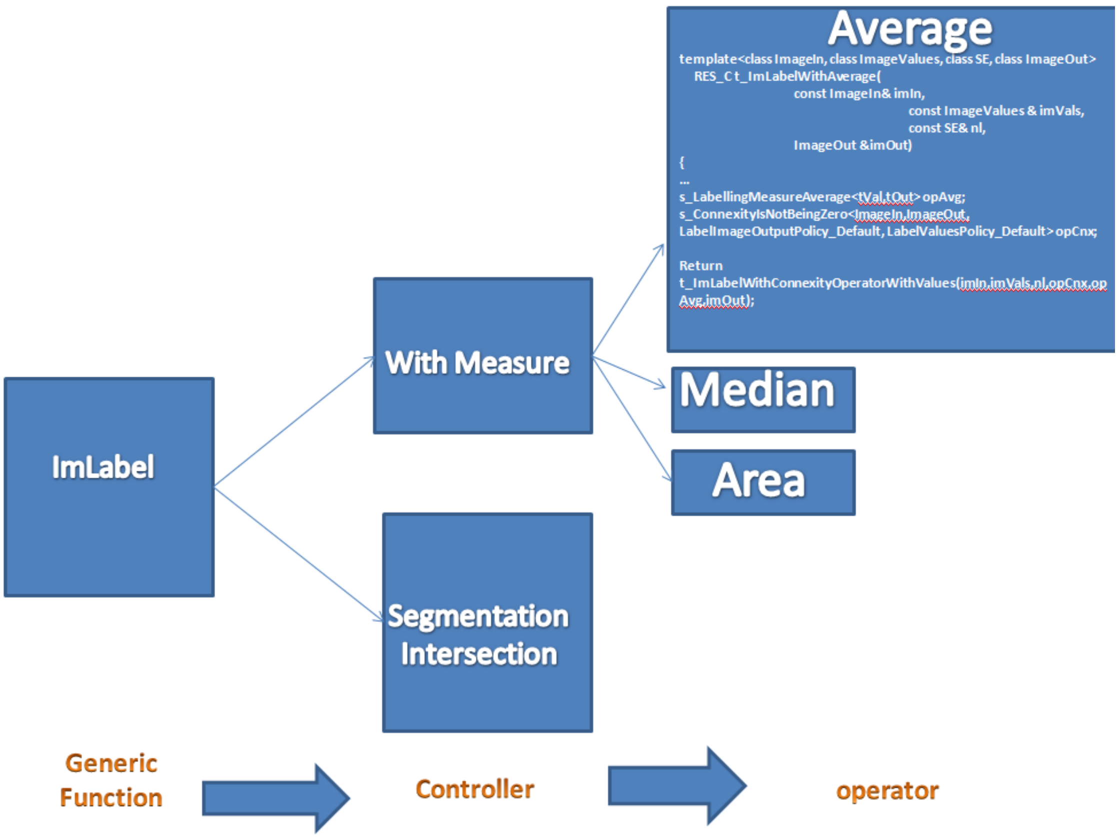
C++

Some Basics example of different Morph-M's element:

```
//Creating images
Image <INT16> im1( x, y, z);
im1.allocateImage();
//Simple iterator
typename Image<T>::iterator it,iend;
for(it=im.begin(), iend = im.end() ; it != iend ; ++it )
  *it=value;
// Get Raw Pointer
const INT16 *bufferIn = im1.rawPointer()
```

```
//Call a simple function
ImErode(&im1, morphee::selement::neighborsCross2D,&imout)
```

Algorithm Design Example



```
template<class ImageIn, class ImageValues, class SE, class ImageOut>
   RES_C t_ImLabelWithAverage(
                        const ImageIn& imIn,
                                           const ImageValues & imVals,
                                           const SE& nl,
                        ImageOut &imOut)
{
...
s_LabellingMeasureAverage<tVal,tOut> opAvg;
s_ConnexityIsNotBeingZero<ImageIn,ImageOut,
LabelImageOutputPolicy_Default, LabelValuesPolicy_Default> opCnx;

Return
t_ImLabelWithConnexityOperatorWithValues(imIn,imVals,nl,opCnx,op
Avg,imOut);
```

Python

This example show 3 different ways to make simple Erosion:

- The first way using generic :

  ```python
  def MyErode1(imIn, nl, imOut):
    # This generic function create neighborhood list
    # send it to operator and put the return
    # value on the centered pixel
    # lambda version:
    morphee.ImNeighborhoodUnaryOperation(imIn, nl, lambda l:min(l), imOut)
    # version using 'min' function:
    morphee.ImNeighborhoodUnaryOperation(imIn, nl, min, imOut)
  ```

- The second way using image and neighborhood Iterator. In spite of slowness of this method, it is a good way to quickly prototyting new algorithms. Once validated, it can be implemented on c++ core.

  ```python
  def MyErode2(imIn,nl,imOut):
    #Get image iterator
    itIn  = imIn.imageData()
    itOut = imOut.imageData()
  ```

```
#Create a neighborhood with specific SE
neighb = createNeighborhood( imIn, nl )

while itIn.isNotFinished() and itOut.isNotFinished():
    neighb.setCenter( itIn )
    # neighb.imageData return a list of neighborhodd pixel
    #then we compute the min
    itOut.setPixel( min( neighb.imageData() ) )
    itIn.next()
    itOut.next()
```

- Main function to check coherence between all developed version.

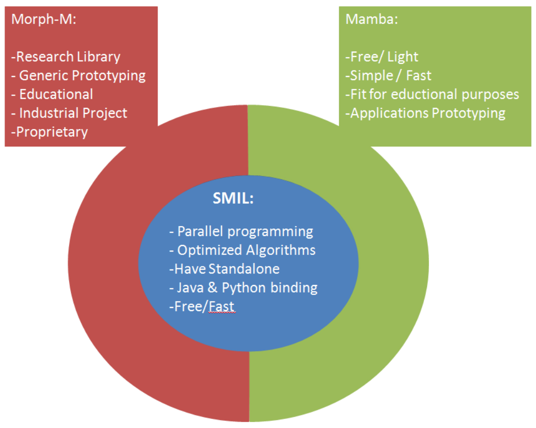
```
def main():
  im = fileRead("./Gray/foreman.png")
  imEro = getSame(im)
  imEroRef = getSame(im)
  #Structuring Element
  nl = NeighborList.neighborsSquare2D

  MyErode1( im, nl, imDil)
  # C++ function
  ImErode( im, nl, imDilRef)
  assert(isEqual(imDil,imDilRef))
  MyErode2( im, nl, imDil)
  assert(isEqual(imDil,imDilRef))
```

Evolution: Smil

The goal is to create new core with the following features:

- Light and work only on 2d and 3d images
- Fast processing (optimized algorithms and parellel programming)
- Real Time processing for industrial application



More Information

- MorphM Website : http://cmm.ensmp.fr/Morph-M
- Documentation : http://morphm.ensmp.fr
- Contact:
  - serge.koudoro@mines-paritech.fr
  - michel.bilodeau@mines-paristech.fr
  - matthieu.faessel@mines-paristech.fr