1 Introduction

This document contains a short list of functions with a quick description. The complete documentation can be found at

Note: optional parameters are between square brackets [optional].

2 Connexion to CMM notebook server

In order to establish a connection to the CMM notebook server, launch a web browser (firefox, chrome...) and go to the following URL http://notebooks.cmm.mines-paristech.fr. Use the username and password that the organizers gave you.

3 import SMIL library

In order to import the required functions

from tp_init import *

In order to be able to display images in the notebook, you have to select the matplotlib backend:

- %matplotlib inline → just shows the image
- %matplotlib notebook → interactive mode

4 I/O images

- To read an image:
  
im = Image("images/toto.png")

- To write an image:
  
write(im,"filename.png")

- To create a new image based on the dimensions of a given image (but not to copy the contents):
  
im2 = Image(im)
• To create a new image based on the dimensions of a given image but with a different depth:
  \[ \text{im2} = \text{Image(im,"UINT16")} \]

• To display image:
  You have to define the matplotlib backend.
  - \%matplotlib inline - only shows the image
  - \%matplotlib notebook - interactive mode

  \text{disp(im)} : displays a single image
  \text{disp(im,True)} : displays a single image in false colors
  \text{disp([im1,im2],[bool1,bool2])} : displays a list of images (some of them can be in false colors).

5 Pixel-based functions

• To copy an image into another one:
  \text{copy(im,im2)}

• To set the image to zero:
  \text{im << 0}

• To invert the image
  \text{inv(imin,imout)}

• To add a constant (or an image):
  \text{add(imin,constant_or_image,imout)}

• To subtract a constant (or an image):
  \text{sub(imin,constant_or_image,imout)}

• The absolute difference between two images:
  \text{absDiff(im1,im2,imout)}

• Returns the maximum value of \text{im}:
  \text{maxVal(im)}

• Returns the maximum value of \text{im}:
  \text{minVal(im)}

• To compare an image to a constant or another image:
  \text{compare(imin,condition,a,b,c, imout)}
  \( a, b \) and \( c \) can be images or scalars. \( \text{imin} \) is compared to \( a \) according to the given condition. If result is true, parameter \( b \) is set in \( \text{imout} \). Otherwise the corresponding \( \text{imout} \) pixel is set to \( c \).
  For example
  \text{compare(im1,","}, im2, im1,im2,imomut)
  is equivalent to \text{sup(im1,im2,imout)}
- To compute the sup of two images:
  \[ \text{sup}(\text{im1}, \text{im2}, \text{imout}) \]

- To compute the inf of two images:
  \[ \text{inf}(\text{im1}, \text{im2}, \text{imout}) \]

- Threshold:
  \[ \text{threshold}(\text{im, minval, maxval, trueval, falseval, imout}) \]

- Otsu threshold:
  \[ \text{threshold}(\text{im, imout}) \]

- Scale: if im has the size \((W, H)\), the size of imout will be \((W \times \text{factor}_x, H \times \text{factor}_y)\). imout should be allocated first (imout = Image()).
  \[ \text{scale}(\text{im, factor}_x, \text{factor}_y, \text{imout}) \]

6 Structuring elements

6.1 Already defined structuring elements:

![CrossSE() SquSE() HexSE() HorizSE() VertSE()](image)

6.2 Default structuring element

If not specified, default SE is used in most morphological functions. As an optional parameter, SE is usually the last one.

- erode(\text{imin, imout})
- erode(\text{imin, imout, CrossSE()})
- erode(\text{imin, imout, CrossSE(size)})

6.3 Modify default SE

Get Default SE: \text{print Morpho.getDefaultSE()}. If not modified, HexSE() is used by default.

Set default SE: \text{Morpho.setDefaultSE(CrossSE())}. CrossSE() becomes default SE.

6.4 Define other SEs

Construct a structuring element with points defined by their indexes:

\[ \text{StrElt(HexFlag, PointList)} : \]

For example:

\[ \text{mySE1= StrElt(False,(0,1,5)) is equivalent to HorizSE().} \]
7 Morphological Erosion, Dilation, Opening, Closing

If not specified, default SE is used in most morphological functions. As an optional parameter, SE is usually the last one.

- Erosion with default SE:
  `erode(im,imout)`

- Erosion with other SE:
  `erode(im,imout,CrossSE())`

- Erosion with a homothetic SE:
  `erode(im,imout[,CrossSE(size)])`

- Dilation (idem for optional SE).
  `dilate(im,imout[,CrossSE(size)])`

- Morphological opening (idem for optional SE).
  `open(im,imout[,CrossSE(size)])`

- Morphological closing (idem for optional SE).
  `close(im,imout[,CrossSE(size)])`

8 Reconstruction

- Reconstruction by dilation
  `build(imMark,imRef,imOut[,nl])`

- Reconstruction by erosion
  `dualBuild(imMark,imRef,imOut[,nl])`

- hBuild: reconstruct by dilation from f-h:
  `hBuild(im,h,immout[,nl])`

- hDualBuild: reconstruct by erosion from f+h:
  `hDualBuild(im,h,immout[,nl])`

- hMaxima: regional maxima after hBuild (reconstruction from f-h),
  `hMaxima(im,h,immmax[,nl])`
• hMinima: regional minima after hDualBuild (reconstruction from f+h),
  \text{hMinima}(\text{im},h,\text{immin}[.,nl])
• hMaxima: regional maxima after razing function f constrained by f-h,
  \text{hMaxima}(\text{im},h,\text{immax}[.,nl])
• hMinima: regional minima after flooding function f constrained by f+h,
  \text{hMinima}(\text{im},h,\text{immin}[.,nl])

9 Filtering
Opening and closing are also morphological filters. Other filters are given in this section:
• Alternate filter: an opening of size \text{size} followed by a closing of the same size.
  \text{AF}(\text{im},\text{size},\text{imout}[.,nl])
• Alternate sequential filter: \phi_{\text{size}}\gamma_{\text{size}}...\phi_{2}\gamma_{2}\phi_{1}\gamma_{1}(\text{im})
  \text{ASF}(\text{im},\text{size},\text{imout}[.,nl])
• Opening by reconstruction: erosion of size \text{size} followed by reconstruction (by dilation):
  \text{buildOpen}(\text{imIn}, \text{imOut}, \text{se}(\text{size}))
• Closing by reconstruction: dilation of size \text{size} followed by reconstruction (by erosion):
  \text{buildClose}(\text{imIn}, \text{imOut}, \text{se}(\text{size}))
• Alternate filter combining buildOpen and buildClose of a give size:
  \text{buildAF}(\text{imIn}, \text{imOut}, \text{se}(\text{size}))
• Alternate sequential filter combining buildOpen and buildClose of increasing sizes (up to size \text{size}:
  \text{buildASF}(\text{imIn}, \text{imOut}, \text{se}(\text{size}))
• AreaOpen:
  \text{areaOpen}(\text{imIn}, \text{size}, \text{imOut}, \text{nl})
• heightOpen:
  \text{heightOpen}(\text{imIn}, \text{size}, \text{imOut}, \text{nl})
• widthOpen:
  \text{widthOpen}(\text{imIn}, \text{size}, \text{imOut}, \text{nl})
• Alternate levelings:
  \text{ASF Leveling}(\text{imIn}, \text{size}, \text{imOut}, \text{nl})
10 Connexity oriented functions

- Regional minima of imin:
  \texttt{minima}(imin,imout[, nl])

- Regional maxima of imin:
  \texttt{maxima}(imin,imout[, nl])

- \texttt{label}(im,iml[, nl]). Assign a different label (identifier) to each connected component.

11 Segmentation

- To compute the gradient:
  \texttt{gradient}(im,imout[,nl])

- Watershed of \texttt{imgra} into \texttt{imws}:
  \texttt{watershed}(imgra,imws[,nl])

- Watershed of \texttt{imgra} from markers \texttt{immark} into \texttt{imws}. Note: each marker should be identified with a different label.
  \texttt{watershed}(imgra,immark,imws[,nl])

- Watershed of \texttt{imgra} from markers \texttt{immark}. Two output parameters \texttt{imws} with the watershed line and \texttt{imbasins} the labelled mosaic without the watershed line:
  \texttt{watershed}(imgra,immark,imws,imbasins[,nl])

- Basins (labelled mosaic without watershed line) of \texttt{imgra}:
  \texttt{basins}(imgra,imbasins[,nl])

- Basins (labelled mosaic without watershed line) of \texttt{imgra} from markers \texttt{immark}:
  \texttt{basins}(imgra,immark,imbasins[,nl])

- Hierarchical segmentation based on extinction values: \texttt{imgra} is flooded and the computed extinction value is assigned to the minima in \texttt{imEV} image. \texttt{extinction_type} can be “d”, “a” or “v” for dynamics, area or volume respectively.
  \texttt{watershedExtinction}(imgra,imEV, extinction_type [,nl])

- A waterfall iteration of \texttt{imgra0} from \texttt{imws0}:
  \texttt{waterfall}(imgra0,imws0,imgra1,imws1[,nl])

- Iteration of waterfall, \texttt{level} times:
  \texttt{waterfall}(imgra,level,imwf[,nl])

- Extinction values based hierarchical segmentation:
  \texttt{imFineSeg,MST} = \texttt{watershedEV}(imgra,EVType[,nl])
• Get a partition imSeg with Nregions from the hierarchy stored in MST graph:
  \textit{getEVLevel}(imFineSeg, MST, Nregions, imSeg)

12 Color

• Extract channels from a color image:
  \textit{im1,im2,im3} = \textit{extractChannels}(colorim)

• Combine channels into a color image:
  colorout = \textit{combineChannels}(im1,im2,im3)

• Get luminance:
  \textit{im8} = \textit{Image}(colorim, "UINT8")
  \textit{RGBToLuminance}(colorim, im8)

• Color conversions:
  colorim2 = \textit{Image}(colorim)
  \begin{itemize}
    \item \textit{RTBToXYZ}(colorim, colorim2)
    \item \textit{RGBToLAB}(colorim, colorim2)
    \item \textit{RGBToHLS}(colorim, colorim2)
  \end{itemize}

• Color gradients:
  \begin{itemize}
    \item \textit{imgra} = \textit{Image}(colorim, "UINT8")
      \textit{gradient\_LAB}(colorim, imgra, nl)
    \item \textit{imgra} = \textit{Image}(colorim, "UINT8")
      \textit{gradient\_HLS}(colorim, imgra, nl)
  \end{itemize}