LibTIM Reference Manual

Generated by Doxygen 1.4.4

Fri Sep 8 12:19:22 2006

Contents

1 LibTIM Module Index **1**

1.1 LibTIM Modules 1

2 LibTIM Directory Hierarchy **3**

2.1 LibTIM Directories 3

3 LibTIM Namespace Index **5**

3.1 LibTIM Namespace List 5

4 LibTIM Hierarchical Index **7**

4.1 LibTIM Class Hierarchy 7

5 LibTIM Class Index **9**

5.1 LibTIM Class List 9

6 LibTIM File Index **11**

6.1 LibTIM File List 11

7 LibTIM Module Documentation **13**

7.1 Component-Tree Based Algorithms 13

7.2 Connected Components Labelling 16

7.3 Distance Transform 17

7.4 K-Means 18

7.5 Misc Functions 19

7.6 Morphological Operators 21

7.7 Basis Functions 22

7.8 Regional Extrema Extraction 26

7.9 Geodesic Reconstruction 28

7.10 Connected Operators 29

7.11 Interval Operators 30

7.12 Region Growing Algorithms . . . 33
7.13 Tarjan's Union Find Algorithms . . . 35
7.14 Template Matching Based Algorithms . . . 37
7.15 Image Processing Basis Functions . . . 38
7.16 Thresholding Functions . . . 39
7.17 Constrained Watershed Algorithms . . . 40
7.18 Watershed-based Algorithms . . . 41
7.19 Flat Structuring Elements . . . 42
7.20 Histogram . . . 43
7.21 Data Structures . . . 44
7.22 Image . . . 45
7.23 Non-flat structuring elements . . . 55
7.24 Point . . . 56

8 LibTIM Directory Documentation **57**
8.1 Algorithms/ Directory Reference . . . 57
8.2 Common/ Directory Reference . . . 58

9 LibTIM Namespace Documentation **59**
9.1 LibTIM Namespace Reference . . . 59
9.2 std Namespace Reference . . . 67

10 LibTIM Class Documentation **69**
10.1 LibTIM::FlatSE Class Reference . . . 69
10.2 LibTIM::Histogram< T > Class Template Reference . . . 74
10.3 LibTIM::Image< T > Class Template Reference . . . 75
10.4 LibTIM::ImageIterator< TImage, T > Class Template Reference . . . 80
10.5 LibTIM::ImageIteratorXYZ< TImage, T > Class Template Reference . . . 82
10.6 LibTIM::ImageRegionsInfos< T, T2 > Class Template Reference . . . 84
10.7 LibTIM::Node Struct Reference . . . 86
10.8 LibTIM::NonFlatSE< T > Class Template Reference . . . 87
10.9 LibTIM::OrderedQueue< T > Class Template Reference . . . 89
10.10 LibTIM::OrderedQueueDouble< T > Class Template Reference . . . 91
10.11 LibTIM::Point< T > Class Template Reference . . . 93
10.12 LibTIM::Queue< T > Class Template Reference . . . 95
10.13 Random Class Reference . . . 96
10.14 LibTIM::Region Struct Reference . . . 97
10.15 LibTIM::Table< T, N > Struct Template Reference . . . 98

11 LibTIM File Documentation **99**

11.1 Algorithms/AdaptativeSE.h File Reference 99
11.2 Algorithms/AdaptativeSE.hxx File Reference 100
11.3 Algorithms/ComponentTree.h File Reference 101
11.4 Algorithms/ComponentTree.hxx File Reference 102
11.5 Algorithms/ConnectedComponents.h File Reference 104
11.6 Algorithms/ConnectedComponents.hxx File Reference 105
11.7 Algorithms/DistanceTransform.h File Reference 106
11.8 Algorithms/DistanceTransform.hxx File Reference 107
11.9 Algorithms/KMeans.h File Reference 108
11.10Algorithms/KMeans.hxx File Reference 109
11.11Algorithms/Misc.h File Reference 110
11.12Algorithms/Misc.hxx File Reference 111
11.13Algorithms/Morphology.h File Reference 112
11.14Algorithms/Morphology.hxx File Reference 113
11.15Algorithms/random-singleton.cpp File Reference 116
11.16Algorithms/random-singleton.h File Reference 117
11.17Algorithms/RegionGrowing.h File Reference 118
11.18Algorithms/RegionGrowing.hxx File Reference 119
11.19Algorithms/Tarjan.h File Reference 120
11.20Algorithms/Tarjan.hxx File Reference 121
11.21Algorithms/TemplateMatching.h File Reference 122
11.22Algorithms/TemplateMatching.hxx File Reference 123
11.23Algorithms/Thresholding.h File Reference 124
11.24Algorithms/Thresholding.hxx File Reference 125
11.25Algorithms/ViscousWatershed.h File Reference 126
11.26Algorithms/ViscousWatershed.hxx File Reference 127
11.27Algorithms/Watershed.h File Reference 128
11.28Algorithms/Watershed.hxx File Reference 129
11.29Common/FlatSE.h File Reference 130
11.30Common/FlatSE.hxx File Reference 131
11.31Common/Histogram.h File Reference 132
11.32Common/Histogram.hxx File Reference 133
11.33Common/Image.h File Reference 134
11.34Common/Image.hxx File Reference 135
11.35Common/ImageIO.hxx File Reference 136

11.36Common/ImageIterators.h File Reference 137
11.37Common/NonFlatSE.h File Reference 138
11.38Common/NonFlatSE.hxx File Reference 139
11.39Common/OrderedQueue.h File Reference 140
11.40Common/Point.h File Reference 141
11.41Common/Types.h File Reference 142

Chapter 1

LibTIM Module Index

1.1 LibTIM Modules

Here is a list of all modules:

- Morphological Operators . . . 21
  - Component-Tree Based Algorithms . . . 13
  - Connected Components Labelling . . . 16
  - Basis Functions . . . 22
  - Regional Extrema Extraction . . . 26
  - Geodesic Reconstruction . . . 28
  - Connected Operators . . . 29
  - Interval Operators . . . 30
  - Region Growing Algorithms . . . 33
  - Constrained Watershed Algorithms . . . 40
  - Watershed-based Algorithms . . . 41
- Image Processing Basis Functions . . . 38
  - Distance Transform . . . 17
  - K-Means . . . 18
  - Misc Functions . . . 19
  - Tarjan's Union Find Algorithms . . . 35
  - Template Matching Based Algorithms . . . 37
  - Thresholding Functions . . . 39
- Data Structures . . . 44
  - Flat Structuring Elements . . . 42
  - Histogram . . . 43
  - Image . . . 45
  - Non-flat structuring elements . . . 55
  - Point . . . 56

Chapter 2

LibTIM Directory Hierarchy

2.1 LibTIM Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

Algorithms 57
Common 58

Chapter 3

LibTIM Namespace Index

3.1 LibTIM Namespace List

Here is a list of all namespaces with brief descriptions:

LibTIM (LibTIM library) 59
std 67

Chapter 4

LibTIM Hierarchical Index

4.1 LibTIM Class Hierarchy

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

LibTIM::FlatSE . . . 69
  LibTIM::NonFlatSE< T > . . . 87
LibTIM::Histogram< T > . . . 74
LibTIM::Image< T > . . . 75
LibTIM::ImageIterator< TImage, T > . . . 80
  LibTIM::ImageIteratorXYZ< TImage, T > . . . 82
LibTIM::ImageRegionsInfos< T, T2 > . . . 84
LibTIM::Node . . . 86
LibTIM::OrderedQueue< T > . . . 89
LibTIM::OrderedQueueDouble< T > . . . 91
LibTIM::Point< T > . . . 93
LibTIM::Queue< T > . . . 95
Random . . . 96
LibTIM::Region . . . 97
LibTIM::Table< T, N > . . . 98

Chapter 5

LibTIM Class Index

5.1 LibTIM Class List

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

LibTIM::FlatSE (Container base class for flat structuring elements (or binary masks)) 69
LibTIM::Histogram< T > (Container for histograms) 74
LibTIM::Image< T > (Container base for images of generic type T in **LibTIM**(p. 59)) 75
LibTIM::ImageIterator< TImage, T > 80
LibTIM::ImageIteratorXYZ< TImage, T > 82
LibTIM::ImageRegionsInfos< T, T2 > 84
LibTIM::Node 86
LibTIM::NonFlatSE< T > (Non-flat structuring elements (or ponderated masks)) . 87
LibTIM::OrderedQueue< T > (Ordered **Queue**(p. 95)) 89
LibTIM::OrderedQueueDouble< T > (Ordered **Queue**(p. 95) with double priority) 91
LibTIM::Point< T > (**Point**(p. 93) Structure) 93
LibTIM::Queue< T > 95
Random 96
LibTIM::Region 97
LibTIM::Table< T, N > 98

Chapter 6

LibTIM File Index

6.1 LibTIM File List

Here is a list of all files with brief descriptions:

Algorithms/**AdaptativeSE.h** . . . 99
Algorithms/**AdaptativeSE.hxx** . . . 100
Algorithms/**ComponentTree.h** . . . 101
Algorithms/**ComponentTree.hxx** . . . 102
Algorithms/**ConnectedComponents.h** . . . 104
Algorithms/**ConnectedComponents.hxx** . . . 105
Algorithms/**DistanceTransform.h** . . . 106
Algorithms/**DistanceTransform.hxx** . . . 107
Algorithms/**KMeans.h** . . . 108
Algorithms/**KMeans.hxx** . . . 109
Algorithms/**Misc.h** . . . 110
Algorithms/**Misc.hxx** . . . 111
Algorithms/**Morphology.h** . . . 112
Algorithms/**Morphology.hxx** . . . 113
Algorithms/**random-singleton.cpp** . . . 116
Algorithms/**random-singleton.h** . . . 117
Algorithms/**RegionGrowing.h** . . . 118
Algorithms/**RegionGrowing.hxx** . . . 119
Algorithms/**Tarjan.h** . . . 120
Algorithms/**Tarjan.hxx** . . . 121
Algorithms/**TemplateMatching.h** . . . 122
Algorithms/**TemplateMatching.hxx** . . . 123
Algorithms/**Thresholding.h** . . . 124
Algorithms/**Thresholding.hxx** . . . 125
Algorithms/**ViscousWatershed.h** . . . 126
Algorithms/**ViscousWatershed.hxx** . . . 127
Algorithms/**Watershed.h** . . . 128
Algorithms/**Watershed.hxx** . . . 129
Common/**FlatSE.h** . . . 130
Common/**FlatSE.hxx** . . . 131
Common/**Histogram.h** . . . 132
Common/**Histogram.hxx** . . . 133
Common/**Image.h** . . . 134

Common/**Image.hxx** 135
Common/**ImageIO.hxx** 136
Common/**ImageIterators.h** 137
Common/**NonFlatSE.h** 138
Common/**NonFlatSE.hxx** 139
Common/**OrderedQueue.h** 140
Common/**Point.h** 141
Common/**Types.h** 142

Chapter 7

LibTIM Module Documentation

7.1 Component-Tree Based Algorithms

Functions

- void **LibTIM::filterArea** (**tNode** *root, int area)
- void **LibTIM::printTree** (**tNode** *tree)
- void **LibTIM::make_father** (**tNode** ***index, int label1, int label2, int h1, int h2)
- **tNode** * **LibTIM::init_tree** (void)
- Image< **U8** > **LibTIM::reconstructImage** (**tNode** *tree, const **TSize** *size)

  Reconstruct image from tree.

- template<class T> **tNode** * **LibTIM::computeComponentTree** (Image< T > &im, FlatSE &se)
- void **LibTIM::father** (**tNode** *tree, **tNode** *child)
- **tNode** * **LibTIM::init_tree** (int h, int n)
- template<class T> **tNode** * **LibTIM::computeComponentTreeBensMethod** (Image< T > &im, FlatSE &se)
- int **LibTIM::computeArea** (**tNode** *tree)
- template<class T> int **LibTIM::flood** (Image< T > &im, std::map< int, std::queue< int > > &oq, int h, int hMin, vector< int > &STATUS, vector< int > &number_nodes, vector< bool > &node_at_level, FlatSE &se, std::map< T, std::map< **TLabel**, struct Node * > > &index)
- template<class T> int **LibTIM::flood2** (Image< T > &im, std::map< int, std::queue< int > > &oq, int h, int hMin, vector< int > &STATUS, vector< int > &number_nodes, vector< bool > &node_at_level, FlatSE &se, std::map< T, std::map< **TLabel**, struct Node * > > &index)

  New method to deal with neighbors.

- template<class T> **tNode** * **LibTIM::computeComponentTree2V1** (Image< T > &im, FlatSE &se)

  Following Salembier recursive implementation...

- template<class T> **tNode** * **LibTIM::computeComponentTree2** (Image< T > &im, FlatSE &se)

  Following Salembier recursive implementation...

7.1.1 Function Documentation

7.1.1.1 int LibTIM::computeArea (tNode * *tree*)

7.1.1.2 template<class T> tNode* LibTIM::computeComponentTree (Image< T > & *im*, FlatSE & *se*)

Build the component tree of image im Return a structure containing the image max-tree For now: trivial algorithm

7.1.1.3 template<class T> tNode* LibTIM::computeComponentTree2 (Image< T > & *im*, FlatSE & *se*)

Following Salembier recursive implementation...

7.1.1.4 template<class T> tNode* LibTIM::computeComponentTree2V1 (Image< T > & *im*, FlatSE & *se*)

Following Salembier recursive implementation...

7.1.1.5 template<class T> tNode* LibTIM::computeComponentTreeBensMethod (Image< T > & *im*, FlatSE & *se*)

7.1.1.6 void LibTIM::father (tNode * *tree*, tNode * *child*)

7.1.1.7 void LibTIM::filterArea (tNode * *root*, int *area*)

7.1.1.8 template<class T> int LibTIM::flood (Image< T > & *im*, std::map< int, std::queue< int > > & *oq*, int *h*, int *hMin*, vector< int > & *STATUS*, vector< int > & *number_nodes*, vector< bool > & *node_at_level*, FlatSE & *se*, std::map< T, std::map< TLabel, struct Node * > > & *index*)

7.1.1.9 template<class T> int LibTIM::flood2 (Image< T > & *im*, std::map< int, std::queue< int > > & *oq*, int *h*, int *hMin*, vector< int > & *STATUS*, vector< int > & *number_nodes*, vector< bool > & *node_at_level*, FlatSE & *se*, std::map< T, std::map< TLabel, struct Node * > > & *index*)

New method to deal with neighbors.

7.1.1.10 tNode* LibTIM::init_tree (int *h*, int *n*)

7.1.1.11 tNode* LibTIM::init_tree (void)

7.1.1.12 void LibTIM::make_father (tNode *** *index*, int *label1*, int *label2*, int *h1*, int *h2*)

7.1.1.13 void LibTIM::printTree (tNode * *tree*)

7.1.1.14 Image<U8> LibTIM::reconstructImage (tNode * *tree*, const TSize * *size*) [inline]

Reconstruct image from tree.

7.2 Connected Components Labelling

Functions

- template<class T> Image< **TLabel** > **LibTIM::labelConnectedComponents** (Image< T > &img, FlatSE &se)
- void **LibTIM::keepIestLargestComponent** (Image< **TLabel** > &img, FlatSE &se, int Iest)

7.2.1 Detailed Description

/**

7.2.2 Function Documentation

7.2.2.1 void LibTIM::keepIestLargestComponent (Image< TLabel > & *img*, FlatSE & *se*, int *Iest*) [inline]

Sort the connected components by their size and keep only the iest largest one(s) Largest one = 1 (not 0)

Map CC number to its size

Map CC size to its corresponding label Multiple CC can have the same size so we use multimap

7.2.2.2 template<class T> Image<TLabel> LibTIM::labelConnectedComponents (Image< T > & *img*, FlatSE & *se*)

Labelisation of connected components img is considered as a binary image with two values: foreground >0 and background = 0

7.3 Distance Transform

Functions

- template<class T, class T2> Image< **U16** > **LibTIM::chamferDistanceTransform** (Image< T > &im, NonFlatSE< T2 > &mask)

7.3.1 Function Documentation

7.3.1.1 template<class T, class T2> Image<U16> LibTIM::chamferDistanceTransform (Image< T > & *im*, NonFlatSE< T2 > & *mask*)

Distance transform Compute distance transform from non-zero pixels of im from the chamfer mask mask

Raster scan

Anti-raster scan

7.4 K-Means

Functions

- template<class T> Image< **TLabel** > **LibTIM::kMeansScalarImage** (const Image< T > &img, std::vector< double > &centroids)

7.4.1 Function Documentation

7.4.1.1 template<class T> Image<TLabel> LibTIM::kMeansScalarImage (const Image< T > & *img*, std::vector< double > & *centroids*)

K-means segmentation Take image and a vector containing centroids initialization (size of vector gives number of classes) Return classification result

7.5 Misc Functions

Functions

- template<class T> void **LibTIM::adjustContrast** (Image< T > &im)
- template<class T> void **LibTIM::adjustContrast** (Image< T > &im, T A, T B)

  Same thing but with A and B given in parameters.

- template<class T, class T2> Image< T > **LibTIM::computeMarkerMean** (Image< T > &src, Image< T2 > &marker)

  For each marker compute the mean of the points on original image.

- template<class T, class T2> Image< T > **LibTIM::computeMarkerMeanFast** (Image< T > &src, Image< T2 > &marker)

  For each marker compute the mean of the points on original image.

- template<class T> void **LibTIM::decimateTemplate** (Image< T > &im, int nx=1, int ny=1, int nz=1)

  Image(p. 75) *decimation by imposing a regular grid -> useful for simplifying structuring elements.*

- std::map< **TLabel**, Point< double > > **LibTIM::centroids** (Image< **TLabel** > &im)

  compute the centroids of labelled objects (first moments) in 2D images

- template<class T> void **LibTIM::drawContour** (Image< T > &im, const Image< **U8** > &mask, const T val)

7.5.1 Function Documentation

7.5.1.1 template<class T> void LibTIM::adjustContrast (Image< T > & *im*, T *A*, T *B*)

Same thing but with A and B given in parameters.

7.5.1.2 template<class T> void LibTIM::adjustContrast (Image< T > & *im*)

Scale image intensity according to the linear relation: x [a,b], f(x) [A,B] f(x)=A+(x-a)(B-A)/(b-a) In this version a=im.getMin(), b=im.getMax(), A=typeMin(), B=typeMax

7.5.1.3 std::map<TLabel,Point<double> > LibTIM::centroids (Image< TLabel > & *im*) [inline]

compute the centroids of labelled objects (first moments) in 2D images

7.5.1.4 template<class T, class T2> Image<T> LibTIM::computeMarkerMean (Image< T > & *src*, Image< T2 > & *marker*)

For each marker compute the mean of the points on original image.

7.5.1.5 template<class T, class T2> Image<T> LibTIM::computeMarkerMeanFast (Image< T > & *src*, Image< T2 > & *marker*)

For each marker compute the mean of the points on original image.

7.5.1.6 template<class T> void LibTIM::decimateTemplate (Image< T > & *im*, int *nx* = 1, int *ny* = 1, int *nz* = 1)

Image(p. 75) decimation by imposing a regular grid -> useful for simplifying structuring elements.

7.5.1.7 template<class T> void LibTIM::drawContour (Image< T > & *im*, const Image< U8 > & *mask*, const T *val*)

7.6 Morphological Operators

Modules

- **Component-Tree Based Algorithms**
- **Connected Components Labelling**
- **Basis Functions**
- **Regional Extrema Extraction**
- **Geodesic Reconstruction**
- **Connected Operators**
- **Interval Operators**
- **Region Growing Algorithms**
- **Constrained Watershed Algorithms**
- **Watershed-based Algorithms**

7.6.1 Detailed Description

Mathematical morphology operators

7.7 Basis Functions

Functions

- template<class T> void **LibTIM::addBorders** (Image< T > &im, **TCoord** *preWidth, **TCoord** *postWidth, T value)
- template<class T> void **LibTIM::addBorders** (Image< T > &im, FlatSE &se, T value)
- template<class T> Image< T > **LibTIM::dilation** (Image< T > im, FlatSE se)

  Basic flat-dilation algorithm.

- template<class T> Image< T > **LibTIM::erosion** (Image< T > im, FlatSE se)

  Basic flat-erosion algorithm.

- template<class T> Image< T > **LibTIM::dilationBorderMax** (Image< T > im, FlatSE se)

  Border max version of dilation.

- template<class T> Image< T > **LibTIM::erosionBorderMin** (Image< T > im, FlatSE se)

  Border min version of erosion.

- template<class T> Image< T > **LibTIM::opening** (Image< T > im, FlatSE se)

  Opening.

- template<class T> Image< T > **LibTIM::closing** (Image< T > im, FlatSE se)

  Closing.

- template<class T> Image< T > **LibTIM::morphologicalGradient** (Image< T > im, FlatSE se)

  Morphological gradient.

- template<class T> Image< T > **LibTIM::internalMorphologicalGradient** (Image< T > im, FlatSE se)

  Internal morphological gradient.

- template<class T> Image< T > **LibTIM::externalMorphologicalGradient** (Image< T > im, FlatSE se)

  External morphological gradient.

- template<class T> Image< T > **LibTIM::rankFilter** (Image< T > im, FlatSE se, int rank)

  Rank filter.

7.7.1 Function Documentation

7.7.1.1 template<class T> void LibTIM::addBorders (Image< T > & *im*, FlatSE & *se*, T *value*)

7.7.1.2 template<class T> void LibTIM::addBorders (Image< T > & *im*, TCoord * *preWidth*, TCoord * *postWidth*, T *value*)

7.7.1.3 template<class T> Image<T> LibTIM::closing (Image< T > *im*, FlatSE *se*)

Closing.

Computes the closing of im by se

7.7.1.4 template<class T> Image<T> LibTIM::dilation (Image< T > *im*, FlatSE *se*)

Basic flat-dilation algorithm.

Computes the dilation of im by flat structuring element se according to Heijman's definition (different from Soille)

7.7.1.5 template<class T> Image<T> LibTIM::dilationBorderMax (Image< T > *im*, FlatSE *se*)

Border max version of dilation.

Computes the dilation but with border set to the maximum possible value of the image type Useful for template matching, when one not want to detect something when hitting the border

7.7.1.6 template<class T> Image<T> LibTIM::erosion (Image< T > *im*, FlatSE *se*)

Basic flat-erosion algorithm.

Computes the erosion of im by flat structuring element se.

7.7.1.7 template<class T> Image<T> LibTIM::erosionBorderMin (Image< T > *im*, FlatSE *se*)

Border min version of erosion.

Computes the erosion but with border set to the minimum possible value of the image type Useful for template matching, when one not want to detect something when hitting the border

7.7.1.8 template<class T> Image<T> LibTIM::externalMorphologicalGradient (Image< T > *im*, FlatSE *se*)

External morphological gradient.

Computes the external morphological gradient

Parameters:

im The source image (not modified)

se The structuring element (not modified)

Returns:
The external morphological gradient of im

7.7.1.9 template<class T> Image<T> LibTIM::internalMorphologicalGradient (Image< T > *im*, FlatSE *se*)

Internal morphological gradient.

Computes the internal morphological gradient

Parameters:
im The source image (not modified)

se The structuring element (not modified)

Returns:
The internal morphological gradient of im

7.7.1.10 template<class T> Image<T> LibTIM::morphologicalGradient (Image< T > *im*, FlatSE *se*)

Morphological gradient.

Computes the morphological gradient (or Beucher gradient)

Parameters:
im The source image (not modified)

se The structuring element (not modified)

Returns:
The morphological gradient of im

7.7.1.11 template<class T> Image<T> LibTIM::opening (Image< T > *im*, FlatSE *se*)

Opening.

Computes the opening of im by se

7.7.1.12 template<class T> Image<T> LibTIM::rankFilter (Image< T > *im*, FlatSE *se*, int *rank*)

Rank filter.

Computes the rank filter.

Parameters:
im The source image

se The structuring element

rank The rank of the filter(rank=0 is equivalent to erosion; rank=se.getNbPoints()-1 is equivalent to dilation)

Returns:

The filtered image.

7.8 Regional Extrema Extraction

Functions

- template<class T> Image< **U8** > **LibTIM::regionalMinima** (Image< T > img, FlatSE se)

  Regional Minima Extraction.

- template<class T> Image< **U8** > **LibTIM::regionalMaxima** (Image< T > img, FlatSE se)

  Regional Maxima Extraction.

7.8.1 Function Documentation

7.8.1.1 template<class T> Image<U8> LibTIM::regionalMaxima (Image< T > *img*, FlatSE *se*)

Regional Maxima Extraction.

Compute the regional maxima of the source image. Returns a *binary* <U8> image with:

- 0 : not maxima
- 255: is maxima

Usually, you need to labelise the result with the **labelConnectedComponents()**(p. 16) function.

Parameters:

img Source **Image**(p. 75)

se Connexity used (use for example 4- or 8- connexity in 2D, see the **FlatSE**(p. 69) documentation)

Returns:

<U8> binary image (0=not maxima, 255=maxima)

Algorithm of Vincent

7.8.1.2 template<class T> Image<U8> LibTIM::regionalMinima (Image< T > *img*, FlatSE *se*)

Regional Minima Extraction.

Compute the regional minima of the source image. Returns a *binary* <U8> image with:

- 0 : not minima
- 255: is minima

Usually, you need to labelise the result with the **labelConnectedComponents()**(p. 16) function.

Parameters:

img Source **Image**(p. 75)

se Connexity used (use for example 4- or 8- connexity in 2D, see the **FlatSE**(p. 69) documentation)

Returns:

<U8> binary image (0=not minima, 255=minima)

Algorithm of Vincent

7.9 Geodesic Reconstruction

Functions

- template<class T> void **LibTIM::geodesicReconstructionByErosion** (Image< T > &marker, Image< T > mask, FlatSE &se)

  Geodesic reconstruction by erosion.

- template<class T> void **LibTIM::geodesicReconstructionByDilation** (Image< T > &marker, Image< T > mask, FlatSE &se)

  Geodesic reconstruction by dilation.

7.9.1 Function Documentation

7.9.1.1 template<class T> void LibTIM::geodesicReconstructionByDilation (Image< T > & *marker*, Image< T > *mask*, FlatSE & *se*)

Geodesic reconstruction by dilation.

Marker must be under the mask

Parameters:

marker The marker image. At the end of function marker is modified and contains the result of reconstruction.

mask The mask image (not modified).

Vincent's algorithm. In this implementation *all* points are inserted in the priority queue

7.9.1.2 template<class T> void LibTIM::geodesicReconstructionByErosion (Image< T > & *marker*, Image< T > *mask*, FlatSE & *se*)

Geodesic reconstruction by erosion.

Marker must be above the mask

Parameters:

marker The marker image. At the end of function marker is modified and contains the result of reconstruction.

mask The mask image (not modified).

Vincent's algorithm. In this implementation *all* points are inserted in the priority queue

7.10 Connected Operators

Functions

- template<class T> void **LibTIM::hMinFilter** (Image< T > &img, FlatSE &se, int h)
  h-Min filter
- template<class T> void **LibTIM::hMaxFilter** (Image< T > &img, FlatSE &se, int h)
  h-Max filter

7.10.1 Function Documentation

7.10.1.1 template<class T> void LibTIM::hMaxFilter (Image< T > & *img*, FlatSE & *se*, int *h*)

h-Max filter

Parameters:

img Source **Image**(p. 75). At the end of function, img is modified and contains the result

h h parameter of filter

Warning: potential overflow problem due to type limitation (when using U8).

7.10.1.2 template<class T> void LibTIM::hMinFilter (Image< T > & *img*, FlatSE & *se*, int *h*)

h-Min filter

Parameters:

img Source **Image**(p. 75). At the end of function, img is modified and contains the result

h h parameter of filter

Warning: potential overflow problem due to type limitation (when using U8).

7.11 Interval Operators

Functions

- template<class T> Image< int > **LibTIM::hitOrMissDifferenceImage** (Image< T > &im, FlatSE &seA, FlatSE &seB)

  Hit-or-miss difference **Image**(p. 75).

- template<class T> int **LibTIM::hitOrMissMaximumDifference** (Image< T > im, FlatSE &seA, FlatSE &seB)

  Maximum of the hitOrMissDifferenceImage.

- template<class T> Image< T > **LibTIM::hitOrMissIntegralK** (Image< T > &im, FlatSE &seA, FlatSE &seB)

  Grey-level hit-or-miss: Soille's version.

- template<class T> Image< T > **LibTIM::hitOrMissSupremalH** (Image< T > &im, FlatSE &seA, FlatSE &seB)

  Grey-level hit-or-miss: Ronse's version.

- template<class T> Image< T > **LibTIM::hitOrMissSupremalK** (Image< T > &im, FlatSE &seA, FlatSE &seB)

  Supremal K version of grey-level hit-or-miss.

- template<class T> Image< T > **LibTIM::hitOrMissIntegralKOpening** (Image< T > &im, FlatSE &seA, FlatSE &seB)

  Grey-level hit-or-miss opening: Soille's version.

- template<class T> Image< T > **LibTIM::hitOrMissSupremalKOpening** (Image< T > &im, FlatSE &seA, FlatSE &seB)

  Grey-level hit-or-miss opening: Supremal K version.

- template<class T> Image< T > **LibTIM::hitOrMissSupremalHOpening** (Image< T > &im, FlatSE &seA, FlatSE &seB)

  Grey-level hit-or-miss opening: Ronse's version.

7.11.1 Function Documentation

7.11.1.1 template<class T> Image<int> LibTIM::hitOrMissDifferenceImage (Image< T > & *im*, FlatSE & *seA*, FlatSE & *seB*)

Hit-or-miss difference **Image**(p. 75).

Parameters:

im The source image (not modified)

seA The first (foreground) structuring element

seB The second (background) structuring element

Returns:

An image of type <int> being the arithmetic difference: $I = (im \ominus seA) - (im \oplus seB)$

7.11.1.2 template<class T> Image<T> LibTIM::hitOrMissIntegralK (Image< T > & *im*, FlatSE & *seA*, FlatSE & *seB*)

Grey-level hit-or-miss: Soille's version.

This function implements the Soille's version of the grey-level hit-or-miss.

7.11.1.3 template<class T> Image<T> LibTIM::hitOrMissIntegralKOpening (Image< T > & *im*, FlatSE & *seA*, FlatSE & *seB*)

Grey-level hit-or-miss opening: Soille's version.

Note:
For details, see article: B.Naegel N.Passat C.Ronse.Grey-level hit-or-miss transforms - Part I: Unified theory Pattern Recognition, In Press.

7.11.1.4 template<class T> int LibTIM::hitOrMissMaximumDifference (Image< T > *im*, FlatSE & *seA*, FlatSE & *seB*)

Maximum of the hitOrMissDifferenceImage.

This function returns the maximum (scalar)of the hit-or-miss difference image: $I = (im \ominus seA) - (im \oplus seB)$

7.11.1.5 template<class T> Image<T> LibTIM::hitOrMissSupremalH (Image< T > & *im*, FlatSE & *seA*, FlatSE & *seB*)

Grey-level hit-or-miss: Ronse's version.

This function implements the Ronse's version of the grey-level hit-or-miss.

7.11.1.6 template<class T> Image<T> LibTIM::hitOrMissSupremalHOpening (Image< T > & *im*, FlatSE & *seA*, FlatSE & *seB*)

Grey-level hit-or-miss opening: Ronse's version.

Note:
For details, see article: B.Naegel N.Passat C.Ronse.Grey-level hit-or-miss transforms - Part I: Unified theory Pattern Recognition, In Press.

7.11.1.7 template<class T> Image<T> LibTIM::hitOrMissSupremalK (Image< T > & *im*, FlatSE & *seA*, FlatSE & *seB*)

Supremal K version of grey-level hit-or-miss.

Note:
For details, see article: B.Naegel N.Passat C.Ronse.Grey-level hit-or-miss transforms - Part I: Unified theory Pattern Recognition, In Press.

7.11.1.8 template<class T> Image<T> LibTIM::hitOrMissSupremalKOpening (Image< T > & *im*, FlatSE & *seA*, FlatSE & *seB*)

Grey-level hit-or-miss opening: Supremal K version.

Note:

For details, see article: B.Naegel N.Passat C.Ronse.Grey-level hit-or-miss transforms - Part I: Unified theory Pattern Recognition, In Press.

7.12 Region Growing Algorithms

Functions

- template<class T, class T2> void **LibTIM::RegionGrowingCriterion** (Image< T > &src, Image< T2 > &marker, FlatSE &se, bool observe=false)
- template<class T> void **LibTIM::seededRegionGrowingExactAlgorithm** (Image< T > &im, Image< **TLabel** > &marker, FlatSE &se, bool observe=false)

  Seeded region-growing algorithm: non-biased implementation.

- template<class T, class T2> void **LibTIM::seededRegionGrowing** (Image< T > &img, Image< T2 > &marker, FlatSE &se, bool observe=false)
- template<class T, class T2> void **LibTIM::seededRegionGrowing0** (Image< T > &img, Image< T2 > &marker, FlatSE &se, bool observe=false)

7.12.1 Function Documentation

7.12.1.1 template<class T, class T2> void LibTIM::RegionGrowingCriterion (Image< T > & *src*, Image< T2 > & *marker*, FlatSE & *se*, bool *observe* = `false`)

7.12.1.2 template<class T, class T2> void LibTIM::seededRegionGrowing (Image< T > & *img*, Image< T2 > & *marker*, FlatSE & *se*, bool *observe* = `false`)

Seeded region growing algorithm : work of Adams and Bischof with the following implementation: Points can be inserted several times in the queue Each time a point is inserted, its neighbors are scanned and their priority is recomputed and eventually reinserted in the queue if their priority has been lowered

Image(p. 75) containing the priority of points in the queue. Max if not in the queue BORDER=-1 if border point

7.12.1.3 template<class T, class T2> void LibTIM::seededRegionGrowing0 (Image< T > & *img*, Image< T2 > & *marker*, FlatSE & *se*, bool *observe* = `false`)

Same thing but each point is inserted with a fixed priority in the queue This gives slightly altered results

Image(p. 75) containing the priority of points in the queue. Max if not in the queue

7.12.1.4 template<class T> void LibTIM::seededRegionGrowingExactAlgorithm (Image< T > & *im*, Image< TLabel > & *marker*, FlatSE & *se*, bool *observe* = `false`)

Seeded region-growing algorithm: non-biased implementation.

Seeded region-growing (see works of Adams-Bischhof, Mehnert-Jackway, Salembier,...) is usually implemented by using hierarchical queues containing points to be processed. Each point is put in the queue with a priority given by some measure of distance. In the first implementation (Adams-Bischhof, and Salembier seems to have the same implementation) a point is put once in the queue with a fixed priority. That is to say that even if in the future these points are

found to have a lesser priority (because a neighbor having a similar grey-level is processed, for example), their priority is not updated. Mehnert-Jackway pointed out the two bias present in the SRG, and proposed an unbiased region-growing algorithms, (ISRG= improved seeded region growing). However, they don't talk of the bias induced by the use of hierarchical queues without the recomputation of priorities. The function **seededRegionGrowing()**(p. 33) implements SRG with the recomputation of priorities (see function for details) based on HQ. The function implemented here don't use anymore HQ: it is based on the trivial formulation of RG. The aim is to see if there is any difference with the RG based on HQ with recomputation of priorities. That is why the function is called "ExactAlgorithm", because the result should be the reference. It is based on this trivial algorithm:

- INIT: each region with the given seeds
- 1) For each region: process sequentially the neighbors, compute the distance and keep the neighbor having the least distance. STOP if there is no avalaible neighbors
- 2) Aggregate the point with the corresponding region and recompute the region characteristics
- 3) GOTO 1)

7.13 Tarjan's Union Find Algorithms

Typedefs

- typedef vector< int > **LibTIM::treeType**

Functions

- void **LibTIM::MakeSet** (**treeType** &tree, const int &offset)
- int **LibTIM::Find** (**treeType** &tree, const int &offset)
- int **LibTIM::FindSimple** (**treeType** &tree, const int &offset)
- int **LibTIM::Link** (**treeType** &tree, int &x, int &y)
- void **LibTIM::MakeSet** (int *tree, const int &offset)
- int **LibTIM::Find** (int *tree, const int &offset)
- int **LibTIM::Link** (int *tree, int &x, int &y)
- template<class T> Image< **TLabel** > **LibTIM::labelConnectedComponentsTarjan** (const Image< T > &im, const FlatSE &se)
- template<class T> Image< **TLabel** > **LibTIM::labelConnectedComponentsTarjan2** (const Image< T > &im, const FlatSE &se)

7.13.1 Detailed Description

Procedures implementing Tarjan's union-find algorithm

Tarjan's Union Find based routines, and algorithms. Mostly research code.

7.13.2 Typedef Documentation

7.13.2.1 typedef vector<int> LibTIM::treeType

7.13.3 Function Documentation

7.13.3.1 int LibTIM::Find (int * *tree*, const int & *offset*)

7.13.3.2 int LibTIM::Find (treeType & *tree*, const int & *offset*)

7.13.3.3 int LibTIM::FindSimple (treeType & *tree*, const int & *offset*)

7.13.3.4 template<class T> Image<TLabel> LibTIM::labelConnectedComponentsTarjan (const Image< T > & *im*, const FlatSE & *se*)

7.13.3.5 template<class T> Image<TLabel> LibTIM::labelConnectedComponentsTarjan2 (const Image< T > & *im*, const FlatSE & *se*)

Second version, using implementation described in ISMM'05 Tests showed that labeling with the method of breadth scan (propagation) is faster (in our implementation)

7.13.3.6 int LibTIM::Link (int * *tree*, int & *x*, int & *y*)

7.13.3.7 int LibTIM::Link (treeType & *tree*, int & *x*, int & *y*)

7.13.3.8 void LibTIM::MakeSet (int * *tree*, const int & *offset*)

7.13.3.9 void LibTIM::MakeSet (treeType & *tree*, const int & *offset*)

7.14 Template Matching Based Algorithms

Functions

- template<class T> Image< int > **LibTIM::templateMatchingL2** (const Image< T > &im, const NonFlatSE< **U8** > &mask)
- template<class T, class T2> Image< T > **LibTIM::printBestTemplate** (const Image< T2 > &resTM, const Image< T > &im, const FlatSE &A, T2 value)

  Same thing but with two templates: one for foreground (255), the other for background (0).

- template<class T> Image< double > **LibTIM::templateMatchingCorrelation** (const Image< T > &im, const NonFlatSE< **U8** > &mask)

7.14.1 Function Documentation

7.14.1.1 template<class T, class T2> Image<T> LibTIM::printBestTemplate (const Image< T2 > & *resTM*, const Image< T > & *im*, const FlatSE & *A*, T2 *value*)

Same thing but with two templates: one for foreground (255), the other for background (0).

7.14.1.2 template<class T> Image<double> LibTIM::templateMatchingCorrelation (const Image< T > & *im*, const NonFlatSE< U8 > & *mask*)

Correlation between template and image point by point Correlation score is regularized with respect to the product of the vector norms Here, we correlate a template of size L with a subimage of size L When template hits border, we set correlation score to 0 Regularized correlation score is comprised between -1 (anti-correlation) and 1 (correlation).

Mask size

7.14.1.3 template<class T> Image<int> LibTIM::templateMatchingL2 (const Image< T > & *im*, const NonFlatSE< U8 > & *mask*)

Compute point by point the mean euclidian distance (L2 norm) between the image and the template To avoid false detections we set to max the distance when the template hits the image border

7.15 Image Processing Basis Functions

Modules

- **Distance Transform**
- **K-Means**
- **Misc Functions**
- **Tarjan's Union Find Algorithms**
- **Template Matching Based Algorithms**
- **Thresholding Functions**

7.16 Thresholding Functions

Functions

- template<class T> Image< T > **LibTIM::threshold** (Image< T > &im, T tLow, T t-High)

  Thresholding.

- template<class T> Image< T > **LibTIM::threshold** (Image< T > &im, int tLow, int tHigh)

  Thresholding (overloaded).

- template<class T> Image< **U8** > **LibTIM::binarize** (Image< T > &im)

  Binarization: 0->0, !=0 -> 255.

7.16.1 Function Documentation

7.16.1.1 template<class T> Image<U8> LibTIM::binarize (Image< T > & *im*)

Binarization: 0->0, !=0 -> 255.

7.16.1.2 template<class T> Image<T> LibTIM::threshold (Image< T > & *im*, int *tLow*, int *tHigh*)

Thresholding (overloaded).

7.16.1.3 template<class T> Image<T> LibTIM::threshold (Image< T > & *im*, T *tLow*, T *tHigh*)

Thresholding.

7.17 Constrained Watershed Algorithms

Functions

- template<class T> void **LibTIM::viscousClosingMercuryBasic** (Image< T > &src, double r0)
- template<class T> void **LibTIM::viscousClosingMercury** (Image< T > &src, double r0)

  Version two: we try to optimize a little.

7.17.1 Function Documentation

7.17.1.1 template<class T> void LibTIM::viscousClosingMercury (Image< T > & *src*, double *r0*)

Version two: we try to optimize a little.

First we close the image src with all possible structuring elements We put each closing into a map referenced by the parameter r of the structuring element

element r is not yet in the map

7.17.1.2 template<class T> void LibTIM::viscousClosingMercuryBasic (Image< T > & *src*, double *r0*)

Viscous closing according to Vachier's definition. This function defines the mercury viscous closing on the gradient image src

First closing with maximal disk

7.18 Watershed-based Algorithms

Functions

- template<class T, class T2> void **LibTIM::watershedMeyer** (Image< T > &img, Image< T2 > &marker, FlatSE &se, bool observe=false)

7.18.1 Function Documentation

7.18.1.1 template<class T, class T2> void LibTIM::watershedMeyer (Image< T > & *img*, Image< T2 > & *marker*, FlatSE & *se*, bool *observe* = `false`)

7.19 Flat Structuring Elements

Classes

- class **LibTIM::FlatSE**

  Container base class for flat structuring elements (or binary masks).

7.20 Histogram

Classes

- class **LibTIM::Histogram< T >**
  Container for histograms.

7.21 Data Structures

Modules

- **Flat Structuring Elements**
- **Histogram**
- **Image**
- **Non-flat structuring elements**
- **Point**

7.22 Image

Classes

- class **LibTIM::Image**< **T** >

  Container base for images of generic type T in **LibTIM**(p. 59).

Typedefs

- typedef ImageIterator< Image, T > **LibTIM::Image::iterator**

  Iterators.

- typedef ImageIterator< const Image, const T > **LibTIM::Image::const_iterator**
- typedef ImageIteratorXYZ< Image, T > **LibTIM::Image::iteratorXYZ**
- typedef ImageIteratorXYZ< const Image, const T > **LibTIM::Image::const_iterator-XYZ**
- typedef std::reverse_iterator< const_iterator > **LibTIM::Image::const_reverse_-iterator**
- typedef std::reverse_iterator< iterator > **LibTIM::Image::reverse_iterator**

Functions

- static int **LibTIM::Image::load** (const char *filename, Image< T > &im)

  Image(p. 75) *file loader.*

- void **LibTIM::Image::save** (const char *filename)

  Save image file.

- **LibTIM::Image::Image** (const **TSize** *size)

  Constructors.

- **LibTIM::Image::Image** (const **TSize** xSize=1, const **TSize** ySize=1, const **TSize** z-Size=1)
- **LibTIM::Image::Image** (const **TSize** *size, const **TSpacing** *spacing, const T *data)
- **LibTIM::Image::~Image** ()

  Destructor (delete the buffer).

- **LibTIM::Image::Image** (const Image< T > &im)

  Copy constructor.

- Image< T > & **LibTIM::Image::operator=** (const Image< T > &im)

  Assignment operator.

- template<class T2> **LibTIM::Image::Image** (const Image< T2 > &im)

  Type conversion.

- **TSize** * **LibTIM::Image::getSize** () const
- **TSize LibTIM::Image::getSizeX** () const

- **TSize LibTIM::Image::getSizeY** () const
- **TSize LibTIM::Image::getSizeZ** () const
- void **LibTIM::Image::setSize** (**TSize** *size)
- void **LibTIM::Image::setSize** (**TSize** x, **TSize** y, **TSize** z)
- **TSpacing** * **LibTIM::Image::getSpacing** ()
- **TSpacing LibTIM::Image::getSpacingX** () const
- **TSpacing LibTIM::Image::getSpacingY** () const
- **TSpacing LibTIM::Image::getSpacingZ** () const
- void **LibTIM::Image::setSpacingX** (**TSpacing** vx)
- void **LibTIM::Image::setSpacingY** (**TSpacing** vy)
- void **LibTIM::Image::setSpacingZ** (**TSpacing** vz)
- **TOffset LibTIM::Image::getBufSize** () const
- T * **LibTIM::Image::getData** ()
- iterator **LibTIM::Image::begin** ()
- const_iterator **LibTIM::Image::begin** () const
- iterator **LibTIM::Image::end** ()
- const_iterator **LibTIM::Image::end** () const
- reverse_iterator **LibTIM::Image::rbegin** ()
- const_reverse_iterator **LibTIM::Image::rbegin** () const
- reverse_iterator **LibTIM::Image::rend** ()
- const_reverse_iterator **LibTIM::Image::rend** () const
- T & **LibTIM::Image::operator()** (**TCoord** x, **TCoord** y, **TCoord** z=0)

  Coordinates write version.

- T **LibTIM::Image::operator()** (**TCoord** x, **TCoord** y, **TCoord** z=0) const

  Coordinates read-only version.

- T & **LibTIM::Image::operator()** (**TOffset** offset)

  Offset write version.

- T **LibTIM::Image::operator()** (**TOffset** offset) const

  Offset read-only version.

- T & **LibTIM::Image::operator()** (Point< **TCoord** > p)

  Point(p. 93) *write version.*

- T **LibTIM::Image::operator()** (Point< **TCoord** > p) const

  Point(p. 93) *read-only version.*

- Image & **LibTIM::Image::operator**+= (Image< T > &op)

  Image(p. 75) *operators.*

- Image & **LibTIM::Image::operator-**= (Image< T > &op)
- Image & **LibTIM::Image::operator** *= (Image< T > &op)
- Image & **LibTIM::Image::operator**/= (Image< T > &op)
- Image & **LibTIM::Image::operator &**= (Image< T > &op)

  Pointwise minimum and maximum.

- Image & **LibTIM::Image::operator**|= (Image< T > &op)
- Image & **LibTIM::Image::operator!** (void)

Negative.

- bool **LibTIM::Image::operator==** (Image< T > &op)
- template<class T2> void **LibTIM::Image::setImageInfos** (Image< T2 > &im)
- T **LibTIM::Image::getMax** () const

  Min and max.

- T **LibTIM::Image::getMin** () const
- void **LibTIM::Image::fill** (const T value)

  Image(p. 75) *misc.*

- Image< T > **LibTIM::Image::crop** (const **TCoord** fromX=0, const **TCoord** toX=1, const **TCoord** fromY=0, const **TCoord** toY=1, const **TCoord** fromZ=0, const **TCoord** toZ=1)
- void **LibTIM::Image::copy** (Image< T > &im, **TCoord** x1, **TCoord** y1, **TCoord** z1, **TCoord** x2, **TCoord** y2, **TCoord** z2, **TCoord** px, **TCoord** py, **TCoord** pz)
- void **LibTIM::Image::copyFast** (Image< T > &im, int x1, int y1, int z1, int x2, int y2, int z2, int px, int py, int pz)
- void **LibTIM::Image::copyFast** (Image< T > &im, **TCoord** px, **TCoord** py, **TCoord** pz)
- void **LibTIM::Image::copy** (Image< T > &im, **TCoord** px, **TCoord** py, **TCoord** pz)
- void **LibTIM::Image::enlarge** ()
- int **LibTIM::Image::getOffset** (int x, int y, int z)
- Image< T > **LibTIM::Image::getReflection** ()
- void **LibTIM::Image::print** ()
- bool **LibTIM::Image::isPosValid** (**TCoord** x, **TCoord** y, **TCoord** z=0) const
- bool **LibTIM::Image::isPosValid** (**TOffset** offset) const
- bool **LibTIM::Image::isPosValid** (Point< **TCoord** > p) const
- template<class T> Image< T > **LibTIM::operator+** (Image< T > &a, Image< T > &b)

  Image(p. 75) *operators.*

- template<class T> Image< T > **LibTIM::operator-** (Image< T > &a, Image< T > &b)
- template<class T> Image< T > **LibTIM::operator** * (Image< T > &a, Image< T > &b)
- template<class T> Image< T > **LibTIM::operator+** (Image< T > &a, T s)

  Mixed mode arithmetic: operations with a scalar.

- template<class T> Image< T > **LibTIM::operator-** (Image< T > &a, T s)
- template<class T> Image< T > **LibTIM::operator** * (Image< T > &a, T s)

7.22.1 Typedef Documentation

7.22.1.1 template<class T> typedef ImageIterator<const Image, const T> LibTIM::Image< T >::const_iterator [inherited]

7.22.1.2 template<class T> typedef ImageIteratorXYZ<const Image, const T> LibTIM::Image< T >::const_iteratorXYZ [inherited]

7.22.1.3 template<class T> typedef std::reverse_iterator<const_iterator> LibTIM::Image< T >::const_reverse_iterator [inherited]

7.22.1.4 template<class T> typedef ImageIterator<Image,T> LibTIM::Image< T >::iterator [inherited]

Iterators.

7.22.1.5 template<class T> typedef ImageIteratorXYZ<Image,T> LibTIM::Image< T >::iteratorXYZ [inherited]

7.22.1.6 template<class T> typedef std::reverse_iterator<iterator> LibTIM::Image< T >::reverse_iterator [inherited]

7.22.2 Function Documentation

7.22.2.1 template<class T> const_iterator LibTIM::Image< T >::begin () const [inline, inherited]

7.22.2.2 template<class T> iterator LibTIM::Image< T >::begin () [inline, inherited]

7.22.2.3 template<class VoxelType> void LibTIM::Image< VoxelType >::copy (Image< T > & *im*, TCoord *px*, TCoord *py*, TCoord *pz*) [inherited]

7.22.2.4 template<class VoxelType> void LibTIM::Image< VoxelType >::copy (Image< T > & *im*, TCoord *x1*, TCoord *y1*, TCoord *z1*, TCoord *x2*, TCoord *y2*, TCoord *z2*, TCoord *px*, TCoord *py*, TCoord *pz*) [inherited]

7.22.2.5 template<class VoxelType> void LibTIM::Image< VoxelType >::copyFast (Image< T > & *im*, TCoord *px*, TCoord *py*, TCoord *pz*) [inherited]

7.22.2.6 template<class VoxelType> void LibTIM::Image< VoxelType >::copyFast (Image< T > & *im*, int *x1*, int *y1*, int *z1*, int *x2*, int *y2*, int *z2*, int *px*, int *py*, int *pz*) [inherited]

7.22.2.7 template<class T> Image< T > LibTIM::Image< T >::crop (const TCoord *fromX* = 0, const TCoord *toX* = 1, const TCoord *fromY* = 0, const TCoord *toY* = 1, const TCoord *fromZ* = 0, const TCoord *toZ* = 1) [inherited]

7.22.2.8 template<class T> const_iterator LibTIM::Image< T >::end () const [inline, inherited]

7.22.2.9 template<class T> iterator LibTIM::Image< T >::end () [inline, inherited]

7.22.2.10 template<class T> void LibTIM::Image< T >::enlarge () [inherited]

7.22.2.11 template<class T> void LibTIM::Image< T >::fill (const T *value*) [inherited]

Image(p. 75) misc.

7.22.2.12 template<class T> TOffset LibTIM::Image< T >::getBufSize () const [inline, inherited]

7.22.2.13 template<class T> T* LibTIM::Image< T >::getData () [inline, inherited]

7.22.2.14 template<class T> T LibTIM::Image< T >::getMax () const [inherited]

Min and max.

Return image min and max Warning: scan the image each time

7.22.2.15 template<class T> T LibTIM::Image< T >::getMin () const [inherited]

7.22.2.16 template<class T> int LibTIM::Image< T >::getOffset (int *x*, int *y*, int *z*) [inline, inherited]

7.22.2.17 template<class VoxelType> Image< VoxelType > LibTIM::Image< VoxelType >::getReflection () [inherited]

7.22.2.18 template<class T> TSize* LibTIM::Image< T >::getSize () const [inline, inherited]

7.22.2.19 template<class T> TSize LibTIM::Image< T >::getSizeX () const [inline, inherited]

7.22.2.20 template<class T> TSize LibTIM::Image< T >::getSizeY () const [inline, inherited]

7.22.2.21 template<class T> TSize LibTIM::Image< T >::getSizeZ () const [inline, inherited]

7.22.2.22 template<class T> TSpacing* LibTIM::Image< T >::getSpacing () [inline, inherited]

7.22.2.23 template<class T> TSpacing LibTIM::Image< T >::getSpacingX () const [inline, inherited]

7.22.2.24 template<class T> TSpacing LibTIM::Image< T >::getSpacingY () const [inline, inherited]

7.22.2.25 template<class T> TSpacing LibTIM::Image< T >::getSpacingZ () const [inline, inherited]

7.22.2.26 template<class T> template<class T2> LibTIM::Image< T >::Image (const Image< T2 > & *im*) [inherited]

Type conversion.

7.22.2.27 template<class T> LibTIM::Image< T >::Image (const Image< T > & *im*) [inherited]

Copy constructor.

7.22.2.28 template<class T> LibTIM::Image< T >::Image (const TSize * *size*, const TSpacing * *spacing*, const T * *data*) [inherited]

Construct an image from a buffer *data Tab data "must" be allocated and large enough (min bufSize)

7.22.2.29 template<class T> LibTIM::Image< T >::Image (const TSize *xSize* = 1, const TSize *ySize* = 1, const TSize *zSize* = 1) [inherited]

7.22.2.30 template<class T> LibTIM::Image< T >::Image (const TSize * *size*) [inherited]

Constructors.

7.22.2.31 template<class T> bool LibTIM::Image< T >::isPosValid (Point< TCoord > *p*) const [inline, inherited]

7.22.2.32 template<class T> bool LibTIM::Image< T >::isPosValid (TOffset *offset*) const [inline, inherited]

7.22.2.33 template<class VoxelType> bool LibTIM::Image< VoxelType >::isPosValid (TCoord *x*, TCoord *y*, TCoord *z* = 0) const [inherited]

7.22.2.34 template<class T> static int LibTIM::Image< T >::load (const char * *filename*, Image< T > & *im*) [static, inherited]

Image(p. 75) file loader.

Use as follows:

```
Image <U8> myIm;
Image <U8>::load("myFile.pgm", myIm);
```

7.22.2.35 template<class T> Image< T > & LibTIM::Image< T >::operator &= (Image< T > & *op*) [inherited]

Pointwise minimum and maximum.

7.22.2.36 template<class T> Image<T> LibTIM::operator * (Image< T > & *a*, T *s*)

7.22.2.37 template<class T> Image<T> LibTIM::operator * (Image< T > & *a*, Image< T > & *b*)

7.22.2.38 template<class T> Image< T > & LibTIM::Image< T >::operator *= (Image< T > & *op*) [inherited]

7.22.2.39 template<class T> Image< T > & LibTIM::Image< T >::operator! (void) [inherited]

Negative.

7.22.2.40 template<class T> T LibTIM::Image< T >::operator() (Point< TCoord > *p*) const [inline, inherited]

Point(p. 93) read-only version.

7.22.2.41 template<class T> T& LibTIM::Image< T >::operator() (Point< TCoord > *p*) [inline, inherited]

Point(p. 93) write version.

7.22.2.42 template<class T> T LibTIM::Image< T >::operator() (TOffset *offset*) const [inline, inherited]

Offset read-only version.

7.22.2.43 template<class T> T& LibTIM::Image< T >::operator() (TOffset *offset*) [inline, inherited]

Offset write version.

7.22.2.44 template<class T> T LibTIM::Image< T >::operator() (TCoord *x*, TCoord *y*, TCoord *z* = 0) const [inline, inherited]

Coordinates read-only version.

7.22.2.45 template<class T> T& LibTIM::Image< T >::operator() (TCoord *x*, TCoord *y*, TCoord *z* = 0) [inline, inherited]

Coordinates write version.

7.22.2.46 template<class T> Image<T> LibTIM::operator+ (Image< T > & *a*, T *s*)

Mixed mode arithmetic: operations with a scalar.

7.22.2.47 template<class T> Image<T> LibTIM::operator+ (Image< T > & *a*, Image< T > & *b*)

Image(p. 75) operators.

7.22.2.48 template<class T> Image< T > & LibTIM::Image< T >::operator+= (Image< T > & *op*) [inherited]

Image(p. 75) operators.

7.22.2.49 template<class T> Image<T> LibTIM::operator- (Image< T > & *a*, T *s*)

7.22.2.50 template<class T> Image<T> LibTIM::operator- (Image< T > & *a*, Image< T > & *b*)

7.22.2.51 template<class T> Image< T > & LibTIM::Image< T >::operator-= (Image< T > & *op*) [inherited]

7.22.2.52 template<class T> Image& LibTIM::Image< T >::operator/= (Image< T > & *op*) [inherited]

7.22.2.53 template<class T> Image< T > & LibTIM::Image< T >::operator= (const Image< T > & *im*) [inherited]

Assignment operator.

7.22.2.54 template<class T> bool LibTIM::Image< T >::operator== (Image< T > & *op*) [inline, inherited]

7.22.2.55 template<class T> Image< T > & LibTIM::Image< T >::operator|= (Image< T > & *op*) [inherited]

7.22.2.56 template<class T> void LibTIM::Image< T >::print () [inline, inherited]

7.22.2.57 template<class T> const_reverse_iterator LibTIM::Image< T >::rbegin () const [inline, inherited]

7.22.2.58 template<class T> reverse_iterator LibTIM::Image< T >::rbegin () [inline, inherited]

7.22.2.59 template<class T> const_reverse_iterator LibTIM::Image< T >::rend () const [inline, inherited]

7.22.2.60 template<class T> reverse_iterator LibTIM::Image< T >::rend () [inline, inherited]

7.22.2.61 template<class T> void LibTIM::Image< T >::save (const char * *filename*) [inherited]

Save image file.

7.22.2.62 **template<class VoxelType> template<class VoxelType2> void LibTIM::Image< VoxelType >::setImageInfos (Image< T2 > & *im*)** `[inherited]`

7.22.2.63 **template<class T> void LibTIM::Image< T >::setSize (TSize *x*, TSize *y*, TSize *z*)** `[inline, inherited]`

7.22.2.64 **template<class T> void LibTIM::Image< T >::setSize (TSize * *size*)** `[inline, inherited]`

7.22.2.65 **template<class T> void LibTIM::Image< T >::setSpacingX (TSpacing *vx*)** `[inline, inherited]`

7.22.2.66 **template<class T> void LibTIM::Image< T >::setSpacingY (TSpacing *vy*)** `[inline, inherited]`

7.22.2.67 **template<class T> void LibTIM::Image< T >::setSpacingZ (TSpacing *vz*)** `[inline, inherited]`

7.22.2.68 **template<class T> LibTIM::Image< T >::~Image ()** `[inline, inherited]`

Destructor (delete the buffer).

7.23 Non-flat structuring elements

Classes

- class **LibTIM::NonFlatSE< T >**

  Non-flat structuring elements (or ponderated masks).

7.24 Point

Classes

- class **LibTIM::Point< T >**
  Point(p. 93) *Structure.*

Functions

- template<class T> Point< T > **LibTIM::operator+** (Point< T > p, Point< T > q)
- template<class T> Point< T > **LibTIM::operator-** (Point< T > p, Point< T > q)

7.24.1 Function Documentation

7.24.1.1 template<class T> Point<T> LibTIM::operator+ (Point< T > *p*, Point< T > *q*)

7.24.1.2 template<class T> Point<T> LibTIM::operator- (Point< T > *p*, Point< T > *q*)

Chapter 8

LibTIM Directory Documentation

8.1 Algorithms/ Directory Reference

Files

- file **AdaptativeSE.h**
- file **AdaptativeSE.hxx**
- file **ComponentTree.h**
- file **ComponentTree.hxx**
- file **ConnectedComponents.h**
- file **ConnectedComponents.hxx**
- file **DistanceTransform.h**
- file **DistanceTransform.hxx**
- file **KMeans.h**
- file **KMeans.hxx**
- file **Misc.h**
- file **Misc.hxx**
- file **Morphology.h**
- file **Morphology.hxx**
- file **random-singleton.cpp**
- file **random-singleton.h**
- file **RegionGrowing.h**
- file **RegionGrowing.hxx**
- file **Tarjan.h**
- file **Tarjan.hxx**
- file **TemplateMatching.h**
- file **TemplateMatching.hxx**
- file **Thresholding.h**
- file **Thresholding.hxx**
- file **ViscousWatershed.h**
- file **ViscousWatershed.hxx**
- file **Watershed.h**
- file **Watershed.hxx**

8.2 Common/ Directory Reference

Files

- file **FlatSE.h**
- file **FlatSE.hxx**
- file **Histogram.h**
- file **Histogram.hxx**
- file **Image.h**
- file **Image.hxx**
- file **ImageIO.hxx**
- file **ImageIterators.h**
- file **NonFlatSE.h**
- file **NonFlatSE.hxx**
- file **OrderedQueue.h**
- file **Point.h**
- file **Types.h**

Chapter 9

LibTIM Namespace Documentation

9.1 LibTIM Namespace Reference

LibTIM library.

Classes

- struct **Node**
- class **ImageRegionsInfos**
- struct **Region**
- class **FlatSE**

  Container base class for flat structuring elements (or binary masks).

- class **Histogram**

  Container for histograms.

- class **Image**

  Container base for images of generic type T in **LibTIM**(p. 59).

- class **ImageIterator**
- class **ImageIteratorXYZ**
- class **NonFlatSE**

  Non-flat structuring elements (or ponderated masks).

- class **OrderedQueue**

  Ordered **Queue**(p. 95).

- class **OrderedQueueDouble**

  Ordered **Queue**(p. 95) *with double priority.*

- class **Queue**
- class **Point**

  Point(p. 93) *Structure.*

- struct **Table**

Typedefs

- typedef **Node tNode**
- typedef vector< int > **treeType**
- typedef unsigned char **U8**
- typedef signed char **S8**
- typedef unsigned short **U16**
- typedef signed short **S16**
- typedef unsigned long **U32**
- typedef signed long **S32**
- typedef **Table**< **U8**, 3 > **RGB**
- typedef unsigned short **TSize**
- typedef double **TSpacing**
- typedef int **TCoord**
- typedef unsigned long **TLabel**
- typedef long **TOffset**

Functions

- template<class T> void **dynamicSeNormL2** (**Image**< T > &img, const **Point**< **TCoord** > &p, const **FlatSE** &B, int param, **FlatSE** &se)
- template<class T> void **dynamicSeNormL2Rand** (**Image**< T > &img, const **Point**< **TCoord** > &p, const **FlatSE** &B, int param, **FlatSE** &se, int nbPoints)
- template<class T> void **dynamicSeNormL2NPoints** (**Image**< T > &img, const **Point**< **TCoord** > &p, const **FlatSE** &B, int NPoints, **FlatSE** &se)
- template<class T> **Image**< unsigned char > **computeNeighborhoodS1** (const **Image**< T > &im)
- template<class T> **Image**< vector< bool > > **computeNeighborhoodS1v2** (const **Image**< T > &im)
- template<class T> void **dynamicSeS1** (**Image**< T > &img, const **Point**< **TCoord** > &p, int param, **FlatSE** &se)
- void **dynamicSeS1v2** (**Image**< vector< bool > > &img, const **Point**< **TCoord** > &p, int param, **FlatSE** &se)
- template<class T> **Image**< T > **printSeAtPoint** (**Image**< T > &img, **FlatSE** &se, **Point**< **TCoord** > &p)
- vector< long int > **merge_pixels** (**tNode** *tree)

  Aggregate all subpixels from the node tree.

- void **filterArea** (**tNode** *root, int area)
- void **printTree** (**tNode** *tree)
- void **make_father** (**tNode** ***index, int label1, int label2, int h1, int h2)
- **tNode** * **init_tree** (void)
- **Image**< **U8** > **reconstructImage** (**tNode** *tree, const **TSize** *size)

  Reconstruct image from tree.

- template<class T> **tNode** * **computeComponentTree** (**Image**< T > &im, **FlatSE** &se)
- void **father** (**tNode** *tree, **tNode** *child)
- **tNode** * **init_tree** (int h, int n)
- template<class T> **tNode** * **computeComponentTreeBensMethod** (**Image**< T > &im, **FlatSE** &se)
- int **computeArea** (**tNode** *tree)

- template<class T> int **flood** (**Image**< T > &im, std::map< int, std::queue< int > > &oq, int h, int hMin, vector< int > &STATUS, vector< int > &number_nodes, vector< bool > &node_at_level, **FlatSE** &se, std::map< T, std::map< **TLabel**, struct **Node** * > > &index)
- template<class T> int **flood2** (**Image**< T > &im, std::map< int, std::queue< int > > &oq, int h, int hMin, vector< int > &STATUS, vector< int > &number_nodes, vector< bool > &node_at_level, **FlatSE** &se, std::map< T, std::map< **TLabel**, struct **Node** * > > &index)

  New method to deal with neighbors.

- template<class T> **tNode** * **computeComponentTree2V1** (**Image**< T > &im, **FlatSE** &se)

  Following Salembier recursive implementation...

- template<class T> **tNode** * **computeComponentTree2** (**Image**< T > &im, **FlatSE** &se)

  Following Salembier recursive implementation...

- template<class T> **Image**< **TLabel** > **labelConnectedComponents** (**Image**< T > &img, **FlatSE** &se)
- void **keepIestLargestComponent** (**Image**< **TLabel** > &img, **FlatSE** &se, int Iest)
- template<class T, class T2> **Image**< **U16** > **chamferDistanceTransform** (**Image**< T > &im, **NonFlatSE**< T2 > &mask)
- template<class T> **Image**< **TLabel** > **kMeansScalarImage** (const **Image**< T > &img, std::vector< double > &centroids)
- template<class T> void **adjustContrast** (**Image**< T > &im)
- template<class T> void **adjustContrast** (**Image**< T > &im, T A, T B)

  Same thing but with A and B given in parameters.

- template<class T, class T2> **Image**< T > **computeMarkerMean** (**Image**< T > &src, **Image**< T2 > &marker)

  For each marker compute the mean of the points on original image.

- template<class T, class T2> **Image**< T > **computeMarkerMeanFast** (**Image**< T > &src, **Image**< T2 > &marker)

  For each marker compute the mean of the points on original image.

- template<class T> void **decimateTemplate** (**Image**< T > &im, int nx=1, int ny=1, int nz=1)

  Image(p. 75) *decimation by imposing a regular grid -> useful for simplifying structuring elements.*

- std::map< **TLabel**, **Point**< double > > **centroids** (**Image**< **TLabel** > &im)

  compute the centroids of labelled objects (first moments) in 2D images

- template<class T> void **drawContour** (**Image**< T > &im, const **Image**< **U8** > &mask, const T val)
- template<class T> void **addBorders** (**Image**< T > &im, **TCoord** *preWidth, **TCoord** *postWidth, T value)
- template<class T> void **addBorders** (**Image**< T > &im, **FlatSE** &se, T value)
- template<class T> **Image**< T > **dilation** (**Image**< T > im, **FlatSE** se)

Basic flat-dilation algorithm.

- template<class T> **Image**< T > **erosion** (**Image**< T > im, **FlatSE** se)

  Basic flat-erosion algorithm.

- template<class T> **Image**< T > **dilationBorderMax** (**Image**< T > im, **FlatSE** se)

  Border max version of dilation.

- template<class T> **Image**< T > **erosionBorderMin** (**Image**< T > im, **FlatSE** se)

  Border min version of erosion.

- template<class T> **Image**< T > **opening** (**Image**< T > im, **FlatSE** se)

  Opening.

- template<class T> **Image**< T > **closing** (**Image**< T > im, **FlatSE** se)

  Closing.

- template<class T> **Image**< T > **morphologicalGradient** (**Image**< T > im, **FlatSE** se)

  Morphological gradient.

- template<class T> **Image**< T > **internalMorphologicalGradient** (**Image**< T > im, **FlatSE** se)

  Internal morphological gradient.

- template<class T> **Image**< T > **externalMorphologicalGradient** (**Image**< T > im, **FlatSE** se)

  External morphological gradient.

- template<class T> **Image**< T > **rankFilter** (**Image**< T > im, **FlatSE** se, int rank)

  Rank filter.

- template<class T> **Image**< **U8** > **regionalMinima** (**Image**< T > img, **FlatSE** se)

  Regional Minima Extraction.

- template<class T> **Image**< **U8** > **regionalMaxima** (**Image**< T > img, **FlatSE** se)

  Regional Maxima Extraction.

- template<class T> void **geodesicReconstructionByErosion** (**Image**< T > &marker, **Image**< T > mask, **FlatSE** &se)

  Geodesic reconstruction by erosion.

- template<class T> void **geodesicReconstructionByDilation** (**Image**< T > &marker, **Image**< T > mask, **FlatSE** &se)

  Geodesic reconstruction by dilation.

- template<class T> void **hMinFilter** (**Image**< T > &img, **FlatSE** &se, int h)

  h-Min filter

- template<class T> void **hMaxFilter** (**Image**< T > &img, **FlatSE** &se, int h)

h-Max filter

- template<class T> **Image**< int > **hitOrMissDifferenceImage** (**Image**< T > &im, **FlatSE** &seA, **FlatSE** &seB)

  Hit-or-miss difference **Image**(p. 75).

- template<class T> int **hitOrMissMaximumDifference** (**Image**< T > im, **FlatSE** &seA, **FlatSE** &seB)

  Maximum of the hitOrMissDifferenceImage.

- template<class T> **Image**< T > **hitOrMissIntegralK** (**Image**< T > &im, **FlatSE** &seA, **FlatSE** &seB)

  Grey-level hit-or-miss: Soille's version.

- template<class T> **Image**< T > **hitOrMissSupremalH** (**Image**< T > &im, **FlatSE** &seA, **FlatSE** &seB)

  Grey-level hit-or-miss: Ronse's version.

- template<class T> **Image**< T > **hitOrMissSupremalK** (**Image**< T > &im, **FlatSE** &seA, **FlatSE** &seB)

  Supremal K version of grey-level hit-or-miss.

- template<class T> **Image**< T > **hitOrMissIntegralKOpening** (**Image**< T > &im, **FlatSE** &seA, **FlatSE** &seB)

  Grey-level hit-or-miss opening: Soille's version.

- template<class T> **Image**< T > **hitOrMissSupremalKOpening** (**Image**< T > &im, **FlatSE** &seA, **FlatSE** &seB)

  Grey-level hit-or-miss opening: Supremal K version.

- template<class T> **Image**< T > **hitOrMissSupremalHOpening** (**Image**< T > &im, **FlatSE** &seA, **FlatSE** &seB)

  Grey-level hit-or-miss opening: Ronse's version.

- int **labelToOffset** (**TLabel** label)
- double **computePriority** (pair< long int, int > offset, **Image**< **U8** > &src, struct **Region** &region)
- template<class T, class T2> void **RegionGrowingCriterion** (**Image**< T > &src, **Image**< T2 > &marker, **FlatSE** &se, bool observe=false)
- template<class T> void **seededRegionGrowingExactAlgorithm** (**Image**< T > &im, **Image**< **TLabel** > &marker, **FlatSE** &se, bool observe=false)

  Seeded region-growing algorithm: non-biased implementation.

- template<class T, class T2> void **seededRegionGrowing** (**Image**< T > &img, **Image**< T2 > &marker, **FlatSE** &se, bool observe=false)
- template<class T, class T2> void **seededRegionGrowing0** (**Image**< T > &img, **Image**< T2 > &marker, **FlatSE** &se, bool observe=false)
- void **MakeSet** (**treeType** &tree, const int &offset)
- int **Find** (**treeType** &tree, const int &offset)
- int **FindSimple** (**treeType** &tree, const int &offset)
- int **Link** (**treeType** &tree, int &x, int &y)

- void **MakeSet** (int *tree, const int &offset)
- int **Find** (int *tree, const int &offset)
- int **Link** (int *tree, int &x, int &y)
- template<class T> **Image**< **TLabel** > **labelConnectedComponentsTarjan** (const **Image**< T > &im, const **FlatSE** &se)
- template<class T> **Image**< **TLabel** > **labelConnectedComponentsTarjan2** (const **Image**< T > &im, const **FlatSE** &se)
- template<class T> **Image**< int > **templateMatchingL2** (const **Image**< T > &im, const **NonFlatSE**< **U8** > &mask)
- template<class T, class T2> **Image**< T > **printBestTemplate** (const **Image**< T2 > &resTM, const **Image**< T > &im, const **FlatSE** &A, T2 value)

  Same thing but with two templates: one for foreground (255), the other for background (0).

- template<class T> **Image**< double > **templateMatchingCorrelation** (const **Image**< T > &im, const **NonFlatSE**< **U8** > &mask)
- template<class T> **Image**< T > **threshold** (**Image**< T > &im, T tLow, T tHigh)

  Thresholding.

- template<class T> **Image**< T > **threshold** (**Image**< T > &im, int tLow, int tHigh)

  Thresholding (overloaded).

- template<class T> **Image**< **U8** > **binarize** (**Image**< T > &im)

  Binarization: 0->0, !=0 -> 255.

- template<class T> int **functionR0** (double r0, T t)
- template<class T> void **viscousClosingMercuryBasic** (**Image**< T > &src, double r0)
- template<class T> void **viscousClosingMercury** (**Image**< T > &src, double r0)

  Version two: we try to optimize a little.

- template<class T, class T2> void **watershedMeyer** (**Image**< T > &img, **Image**< T2 > &marker, **FlatSE** &se, bool observe=false)
- template<class T> **Image**< T > **operator+** (**Image**< T > &a, **Image**< T > &b)

  Image(p. 75) *operators.*

- template<class T> **Image**< T > **operator-** (**Image**< T > &a, **Image**< T > &b)
- template<class T> **Image**< T > **operator** * (**Image**< T > &a, **Image**< T > &b)
- template<class T> **Image**< T > **operator+** (**Image**< T > &a, T s)

  Mixed mode arithmetic: operations with a scalar.

- template<class T> **Image**< T > **operator-** (**Image**< T > &a, T s)
- template<class T> **Image**< T > **operator** * (**Image**< T > &a, T s)
- std::string **GImageIO_NextLine** (std::ifstream &file)
- void **GImageIO_ReadPPMHeader** (std::ifstream &file, std::string &format, unsigned int &width, unsigned int &height, unsigned int &colormax)
- template<class T> **Point**< T > **operator+** (**Point**< T > p, **Point**< T > q)
- template<class T> **Point**< T > **operator-** (**Point**< T > p, **Point**< T > q)

Variables

- const float **FLOAT_EPSILON** = 0.0000000001f

9.1.1 Detailed Description

LibTIM library.

Philosophy is:

- reduced set of data structures (principal are: Image() FlatSE() NonFlatSE())
- user oriented, easy to use
- large choice of mathematical morphology functions (recent algorithms, research code,...)
- efficient implementation

9.1.2 Typedef Documentation

9.1.2.1 typedef Table<U8,3> LibTIM::RGB

9.1.2.2 typedef signed short LibTIM::S16

9.1.2.3 typedef signed long LibTIM::S32

9.1.2.4 typedef signed char LibTIM::S8

9.1.2.5 typedef int LibTIM::TCoord

9.1.2.6 typedef unsigned long LibTIM::TLabel

9.1.2.7 typedef struct Node LibTIM::tNode

9.1.2.8 typedef long LibTIM::TOffset

9.1.2.9 typedef unsigned short LibTIM::TSize

9.1.2.10 typedef double LibTIM::TSpacing

9.1.2.11 typedef unsigned short LibTIM::U16

9.1.2.12 typedef unsigned long LibTIM::U32

9.1.2.13 typedef unsigned char LibTIM::U8

9.1.3 Function Documentation

9.1.3.1 template<class T> Image<unsigned char> LibTIM::computeNeighborhood-S1 (const Image< T > & *im*)

Special function to compute the context given Compute an image giving on each point a uchar value resuming the context of the point (configuration of neighborhood) based on the order of the neighbors

9.1.3.2 template<class T> Image<vector <bool> > LibTIM::computeNeighborhoodS1v2 (const Image< T > & *im*)

The latter method seems to perform badly... Same thing as before but with vectors

9.1.3.3 double LibTIM::computePriority (pair< long int, int > *offset*, Image< U8 > & *src*, struct Region & *region*) [inline]

9.1.3.4 template<class T> void LibTIM::dynamicSeNormL2 (Image< T > & *img*, const Point< TCoord > & *p*, const FlatSE & *B*, int *param*, FlatSE & *se*)

9.1.3.5 template<class T> void LibTIM::dynamicSeNormL2NPoints (Image< T > & *img*, const Point< TCoord > & *p*, const FlatSE & *B*, int *NPoints*, FlatSE & *se*)

9.1.3.6 template<class T> void LibTIM::dynamicSeNormL2Rand (Image< T > & *img*, const Point< TCoord > & *p*, const FlatSE & *B*, int *param*, FlatSE & *se*, int *nbPoints*)

9.1.3.7 template<class T> void LibTIM::dynamicSeS1 (Image< T > & *img*, const Point< TCoord > & *p*, int *param*, FlatSE & *se*)

9.1.3.8 void LibTIM::dynamicSeS1v2 (Image< vector< bool > > & *img*, const Point< TCoord > & *p*, int *param*, FlatSE & *se*) [inline]

9.1.3.9 template<class T> int LibTIM::functionR0 (double *r0*, T *t*)

9.1.3.10 std::string LibTIM::GImageIO_NextLine (std::ifstream & *file*) [inline]

9.1.3.11 void LibTIM::GImageIO_ReadPPMHeader (std::ifstream & *file*, std::string & *format*, unsigned int & *width*, unsigned int & *height*, unsigned int & *colormax*) [inline]

9.1.3.12 int LibTIM::labelToOffset (TLabel *label*) [inline]

9.1.3.13 vector<long int> LibTIM::merge_pixels (tNode * *tree*)

Aggregate all subpixels from the node tree.

9.1.3.14 template<class T> Image<T> LibTIM::printSeAtPoint (Image< T > & *img*, FlatSE & *se*, Point< TCoord > & *p*)

9.1.4 Variable Documentation

9.1.4.1 const float LibTIM::FLOAT_EPSILON = 0.0000000001f

9.2 std Namespace Reference

Chapter 10

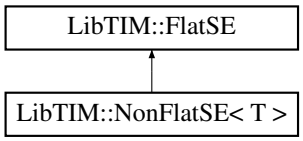
LibTIM Class Documentation

10.1 LibTIM::FlatSE Class Reference

Container base class for flat structuring elements (or binary masks).

`#include <FlatSE.h>`

Inheritance diagram for LibTIM::FlatSE::



Public Types

- typedef std::vector< **Point**< **TCoord** > >::**iterator iterator_point**
- typedef std::vector< **TOffset** >::**iterator iterator_offset**
- typedef **iterator_offset iterator**

Public Member Functions

- **FlatSE** ()
- **FlatSE** (const **Image**< **U8** > &im)
- **FlatSE** & **FlatSE::operator**= (const **FlatSE** &se)
- **FlatSE** (const **FlatSE** &se)
- int **getNbPoints** () const

  returns the number of points contained in the structuring element (cardinal of the set)

- void **setContext** (const **TSize** *size)

  computes the offset of each point, according to "size"

- **Point**< **TCoord** > **getPoint** (int i) const
- void **addPoint** (**Point**< **TCoord** > p)
- **TOffset getOffset** (int point)

returns the offset of "point"

- **TCoord** * **getNegativeOffsets** ()
- **TCoord** * **getPositiveOffsets** ()
- void **makeSymmetric** ()
- **Image**< **U8** > **FlatSE::toImage** ()
- **FlatSE** & **operator**+= (**FlatSE** &b)
- **iterator begin** ()
- **iterator end** ()
- **iterator_point begin_point** ()
- **iterator_point end_point** ()
- void **make2DN4** ()

  Basic neighborhoods in 2D N4 and N8.

- void **make2DN8** ()
- void **make2DN9** ()

  same as before but includes origin

- void **make3DN6** ()

  In 3D N6,18,26.

- void **make3DN18** ()
- void **make3DN26** ()
- template<class VoxelType> void **makeBallEuclidian2D** (**Image**< VoxelType > &img, double r)
- template<class VoxelType> void **makeBallChessboard2D** (**Image**< VoxelType > &img, double rx, double ry)
- template<class VoxelType> void **makeBallEuclidian3D** (**Image**< VoxelType > &img, double r)
- template<class VoxelType> void **makeCircle2D** (**Image**< VoxelType > &img, double r, double t)

  circle with specified thickness

- void **print** ()
- void **reserve** (size_t size)
- void **clear** ()

Protected Attributes

- std::vector< **Point**< **TCoord** > > **points**
- std::vector< **TOffset** > **offsets**

10.1.1 Detailed Description

Container base class for flat structuring elements (or binary masks).

Example:

```
FlatSE se;
se.make2DN9();
```

creates a 2D structuring element containing a 3x3 square. The origin is at the center.

```
FlatSE se;
se.makeBallEuclidian2D(r, im);
```

creates 2D structuring element containing a ball of radius r according to the voxels spacing of im. The origin is at the center.

WARNING: some algorithms require a *connexity* rather than a structuring element in parameters. To this end, use for example **make2DN8()**(p. 72) to compute a 8-neighborhood (now the center is *not* included in the structuring element) Adding a *connexity* structure is in project.

10.1.2 Member Typedef Documentation

10.1.2.1 typedef iterator_offset LibTIM::FlatSE::iterator

10.1.2.2 typedef std::vector<TOffset >::iterator LibTIM::FlatSE::iterator_offset

10.1.2.3 typedef std::vector<Point<TCoord> >::iterator LibTIM::FlatSE::iterator_point

10.1.3 Constructor & Destructor Documentation

10.1.3.1 LibTIM::FlatSE::FlatSE () [inline]

10.1.3.2 LibTIM::FlatSE::FlatSE (const Image< U8 > & *im*)

10.1.3.3 LibTIM::FlatSE::FlatSE (const FlatSE & *se*) [inline]

10.1.4 Member Function Documentation

10.1.4.1 void LibTIM::FlatSE::addPoint (Point< TCoord > *p*) [inline]

10.1.4.2 iterator LibTIM::FlatSE::begin () [inline]

10.1.4.3 iterator_point LibTIM::FlatSE::begin_point () [inline]

10.1.4.4 void LibTIM::FlatSE::clear () [inline]

Reimplemented in **LibTIM::NonFlatSE< T >** (p. 88).

10.1.4.5 iterator LibTIM::FlatSE::end () [inline]

10.1.4.6 iterator_point LibTIM::FlatSE::end_point () [inline]

10.1.4.7 FlatSE& LibTIM::FlatSE::FlatSE::operator= (const FlatSE & *se*)

10.1.4.8 Image<U8> LibTIM::FlatSE::FlatSE::toImage ()

10.1.4.9 int LibTIM::FlatSE::getNbPoints () const [inline]

returns the number of points contained in the structuring element (cardinal of the set)

10.1.4.10 TCoord * LibTIM::FlatSE::getNegativeOffsets () [inline]

10.1.4.11 TOffset LibTIM::FlatSE::getOffset (int *point*) [inline]

returns the offset of "point"

10.1.4.12 Point<TCoord> LibTIM::FlatSE::getPoint (int *i*) const [inline]

10.1.4.13 TCoord * LibTIM::FlatSE::getPositiveOffsets () [inline]

10.1.4.14 void LibTIM::FlatSE::make2DN4 () [inline]

Basic neighborhoods in 2D N4 and N8.
Basic neighborhood (4-neighborhood). Warning: do not contain the origin!

10.1.4.15 void LibTIM::FlatSE::make2DN8 () [inline]

Basic neighborhood (8-neighborhood). Warning: do not contain the origin!

10.1.4.16 void LibTIM::FlatSE::make2DN9 () [inline]

same as before but includes origin

10.1.4.17 void LibTIM::FlatSE::make3DN18 ()

10.1.4.18 void LibTIM::FlatSE::make3DN26 ()

10.1.4.19 void LibTIM::FlatSE::make3DN6 ()

In 3D N6,18,26.

10.1.4.20 template<class VoxelType> void LibTIM::FlatSE::makeBall-Chessboard2D (Image< VoxelType > & *img*, double *rx*, double *ry*)

10.1.4.21 template<class VoxelType> void LibTIM::FlatSE::makeBallEuclidian2D (Image< VoxelType > & *img*, double *r*)

10.1.4.22 template<class VoxelType> void LibTIM::FlatSE::makeBallEuclidian3D (Image< VoxelType > & *img*, double *r*)

10.1.4.23 template<class T> void LibTIM::FlatSE::makeCircle2D (Image< VoxelType > & *img*, double *r*, double *t*)

circle with specified thickness

10.1.4.24 void LibTIM::FlatSE::makeSymmetric () [inline]

10.1.4.25 FlatSE& LibTIM::FlatSE::operator+= (FlatSE & *b*) [inline]

10.1.4.26 void LibTIM::FlatSE::print () [inline]

Reimplemented in **LibTIM::NonFlatSE< T >** (p. 88).

10.1.4.27 void LibTIM::FlatSE::reserve (size_t *size*) [inline]

Reimplemented in **LibTIM::NonFlatSE< T >** (p. 88).

10.1.4.28 void LibTIM::FlatSE::setContext (const TSize * *size*) [inline]

computes the offset of each point, according to "size"

10.1.5 Member Data Documentation

10.1.5.1 std::vector<TOffset> LibTIM::FlatSE::offsets [protected]

10.1.5.2 std::vector<Point<TCoord> > LibTIM::FlatSE::points [protected]

The documentation for this class was generated from the following files:

- Common/**FlatSE.h**
- Common/**FlatSE.hxx**

10.2 LibTIM::Histogram< T > Class Template Reference

Container for histograms.

`#include <Histogram.h>`

Public Member Functions

- **Histogram** (**Image**< T > &im)
  Constructs an histogram from image im.
- void **write** (const char *filename)
  Write histogram into file.

10.2.1 Detailed Description

template<class T> class LibTIM::Histogram< T >

Container for histograms.

Structure describing an histogram. **Histogram**(p. 74) can be constructed from an **Image**(p. 75)

10.2.2 Constructor & Destructor Documentation

10.2.2.1 template<class T> LibTIM::Histogram< T >::Histogram (Image< T > & *im*)

Constructs an histogram from image im.

10.2.3 Member Function Documentation

10.2.3.1 template<class T> void LibTIM::Histogram< T >::write (const char * *filename*)

Write histogram into file.

Histogram(p. 74) is writed in a text file (xmgrace format)

The documentation for this class was generated from the following files:

- Common/**Histogram.h**
- Common/**Histogram.hxx**

10.3 LibTIM::Image< T > Class Template Reference

Container base for images of generic type T in **LibTIM**(p. 59).

`#include <Image.h>`

Public Types

- typedef **ImageIterator**< **Image**, T > **iterator**

  Iterators.

- typedef **ImageIterator**< const **Image**, const T > **const_iterator**
- typedef **ImageIteratorXYZ**< **Image**, T > **iteratorXYZ**
- typedef **ImageIteratorXYZ**< const **Image**, const T > **const_iteratorXYZ**
- typedef std::reverse_iterator< **const_iterator** > **const_reverse_iterator**
- typedef std::reverse_iterator< **iterator** > **reverse_iterator**

Public Member Functions

- void **save** (const char *filename)

  Save image file.

- **Image** (const **TSize** *size)

  Constructors.

- **Image** (const **TSize** xSize=1, const **TSize** ySize=1, const **TSize** zSize=1)
- **Image** (const **TSize** *size, const **TSpacing** *spacing, const T *data)
- ~**Image** ()

  Destructor (delete the buffer).

- **Image** (const **Image**< T > &im)

  Copy constructor.

- **Image**< T > & **operator**= (const **Image**< T > &im)

  Assignment operator.

- template<class T2> **Image** (const **Image**< T2 > &im)

  Type conversion.

- **TSize** * **getSize** () const
- **TSize getSizeX** () const
- **TSize getSizeY** () const
- **TSize getSizeZ** () const
- void **setSize** (**TSize** *size)
- void **setSize** (**TSize** x, **TSize** y, **TSize** z)
- **TSpacing** * **getSpacing** ()
- **TSpacing getSpacingX** () const
- **TSpacing getSpacingY** () const
- **TSpacing getSpacingZ** () const
- void **setSpacingX** (**TSpacing** vx)

- void **setSpacingY** (**TSpacing** vy)
- void **setSpacingZ** (**TSpacing** vz)
- **TOffset getBufSize** () const
- T * **getData** ()
- **iterator begin** ()
- **const_iterator begin** () const
- **iterator end** ()
- **const_iterator end** () const
- **reverse_iterator rbegin** ()
- **const_reverse_iterator rbegin** () const
- **reverse_iterator rend** ()
- **const_reverse_iterator rend** () const
- T & **operator()** (**TCoord** x, **TCoord** y, **TCoord** z=0)

  Coordinates write version.

- T **operator()** (**TCoord** x, **TCoord** y, **TCoord** z=0) const

  Coordinates read-only version.

- T & **operator()** (**TOffset** offset)

  Offset write version.

- T **operator()** (**TOffset** offset) const

  Offset read-only version.

- T & **operator()** (**Point**< **TCoord** > p)

  Point(p. 93) *write version.*

- T **operator()** (**Point**< **TCoord** > p) const

  Point(p. 93) *read-only version.*

- **Image** & **operator+=** (**Image**< T > &op)

  Image(p. 75) *operators.*

- **Image** & **operator-=** (**Image**< T > &op)
- **Image** & **operator** *= (**Image**< T > &op)
- **Image** & **operator/=** (**Image**< T > &op)
- **Image** & **operator &=** (**Image**< T > &op)

  Pointwise minimum and maximum.

- **Image** & **operator|=** (**Image**< T > &op)
- **Image** & **operator!** (void)

  Negative.

- bool **operator==** (**Image**< T > &op)
- template<class T2> void **setImageInfos** (**Image**< T2 > &im)
- T **getMax** () const

  Min and max.

- T **getMin** () const
- void **fill** (const T value)

Image(p. 75) *misc.*

- **Image**< T > **crop** (const **TCoord** fromX=0, const **TCoord** toX=1, const **TCoord** fromY=0, const **TCoord** toY=1, const **TCoord** fromZ=0, const **TCoord** toZ=1)
- void **copy** (**Image**< T > &im, **TCoord** x1, **TCoord** y1, **TCoord** z1, **TCoord** x2, **TCoord** y2, **TCoord** z2, **TCoord** px, **TCoord** py, **TCoord** pz)
- void **copyFast** (**Image**< T > &im, int x1, int y1, int z1, int x2, int y2, int z2, int px, int py, int pz)
- void **copyFast** (**Image**< T > &im, **TCoord** px, **TCoord** py, **TCoord** pz)
- void **copy** (**Image**< T > &im, **TCoord** px, **TCoord** py, **TCoord** pz)
- void **enlarge** ()
- int **getOffset** (int x, int y, int z)
- **Image**< T > **getReflection** ()
- void **print** ()
- bool **isPosValid** (**TCoord** x, **TCoord** y, **TCoord** z=0) const
- bool **isPosValid** (**TOffset** offset) const
- bool **isPosValid** (**Point**< **TCoord** > p) const
- template<> int **load** (const char *filename, **Image**< **U8** > &im)
- template<> int **load** (const char *filename, **Image**< **U16** > &im)
- template<> int **load** (const char *filename, **Image**< **RGB** > &im)
- template<> void **save** (const char *filename)
- template<> void **save** (const char *filename)
- template<> void **save** (const char *filename)

Static Public Member Functions

- static int **load** (const char *filename, **Image**< T > &im)

  Image(p. 75) *file loader.*

10.3.1 Detailed Description

template<class T> class LibTIM::Image< T >

Container base for images of generic type T in **LibTIM**(p. 59).

This structure represents the base image class of **LibTIM**(p. 59). It contains a buffer of elements of type T of size size[0]*size[1]*size[2] x,y and z voxels spacings are recorded in spacing[]

Accessing elements

To access element (x,y,z) or offset (q), use:

```
val=im(x,y,z)
```

```
val=im(q)
```

To write element (x,y,z) or offset (q), use:

```
im(x,y,z)=val
```

```
im(q)=val
```

Iterators

To iterate through elements, you should use iterators. There is two types of iterator:

- `iterator`
  to scan image in raster scan
- `reverse_iterator`
  to scan image in anti-raster scan
- `iteratorXYZ`
  to scan image in raster scan with the knowledge of coordinates
- `reverse_iteratorXYZ`
  to scan image in anti-raster scan with the knowledge of coordinates

begin()(p. 49) and **end()**(p. 49) methods return iterators on the beginning and the end of the **Image**(p. 75) **rbegin()**(p. 53) and **rend()**(p. 53) methods are used with reverse_iterators

Example:

This piece of code initialized all elements of **Image**(p. 75) to 0. (note that to do this you should better use the fill method)

```
Image <U8> im;
...
Image<U8>::iterator it;
Image<U8>::iterator end=im.end();
for(it=im.begin(); it!=end; ++it)
*it=0
```

Same thing with coordinates access (slower):

```
Image <U8> im;
...
Image<U8>::iteratorXYZ it;
Image<U8>::iteratorXYZ end=im.end();
for(it=im.begin(); it!=end; ++it)
im(it.x,it.y,it.z)=0
```

Note that (as in these examples) it is faster to put im.end() in a variable.

10.3.2 Member Function Documentation

10.3.2.1 template<> int LibTIM::Image< RGB >::load (const char * *filename*, Image< RGB > & *im*) [inline]

10.3.2.2 template<> int LibTIM::Image< U16 >::load (const char * *filename*, Image< U16 > & *im*) [inline]

10.3.2.3 template<> int LibTIM::Image< U8 >::load (const char * *filename*, Image< U8 > & *im*) [inline]

10.3.2.4 template<> void LibTIM::Image< RGB >::save (const char * *filename*) [inline]

10.3.2.5 template<> void LibTIM::Image< U16 >::save (const char * *filename*) [inline]

10.3.2.6 template<> void LibTIM::Image< U8 >::save (const char * *filename*) [inline]

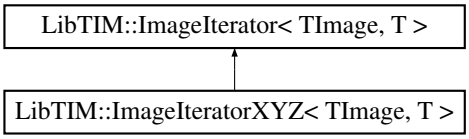
The documentation for this class was generated from the following files:

- Common/**Image.h**
- Common/**Image.hxx**

10.4 LibTIM::ImageIterator< TImage, T > Class Template Reference

`#include <ImageIterators.h>`

Inheritance diagram for LibTIM::ImageIterator< TImage, T >::



Public Member Functions

- **ImageIterator** ()
- **ImageIterator** (TImage *__im__, T *x)
- T & **operator** * ()
- T * **operator** → ()
- **ImageIterator**< TImage, T > & **operator**++ ()
- **ImageIterator**< TImage, T > **operator**++ (int)
- bool **operator**== (const **ImageIterator** &x)
- bool **operator!**= (const **ImageIterator** &x)

Public Attributes

- T * **ptr**
- TImage * **im**

template<class TImage, class T> class LibTIM::ImageIterator< TImage, T >

10.4.1 Constructor & Destructor Documentation

10.4.1.1 template<class TImage, class T> LibTIM::ImageIterator< TImage, T >::ImageIterator () [inline]

10.4.1.2 template<class TImage, class T> LibTIM::ImageIterator< TImage, T >::ImageIterator (TImage * *im*, T * *x*) [inline]

10.4.2 Member Function Documentation

10.4.2.1 template<class TImage, class T> T& LibTIM::ImageIterator< TImage, T >::operator * () [inline]

Reimplemented in **LibTIM::ImageIteratorXYZ< TImage, T >** (p. 82).

10.4.2.2 template<class TImage, class T> bool LibTIM::ImageIterator< TImage, T >::operator!= (const ImageIterator< TImage, T > & *x*) [inline]

10.4.2.3 template<class TImage, class T> ImageIterator<TImage, T> LibTIM::ImageIterator< TImage, T >::operator++ (int) [inline]

Reimplemented in **LibTIM::ImageIteratorXYZ< TImage, T >** (p. 83).

10.4.2.4 template<class TImage, class T> ImageIterator<TImage, T>& LibTIM::ImageIterator< TImage, T >::operator++ () [inline]

Reimplemented in **LibTIM::ImageIteratorXYZ< TImage, T >** (p. 83).

10.4.2.5 template<class TImage, class T> T* LibTIM::ImageIterator< TImage, T >::operator → () [inline]

Reimplemented in **LibTIM::ImageIteratorXYZ< TImage, T >** (p. 83).

10.4.2.6 template<class TImage, class T> bool LibTIM::ImageIterator< TImage, T >::operator== (const ImageIterator< TImage, T > & *x*) [inline]

10.4.3 Member Data Documentation

10.4.3.1 template<class TImage, class T> TImage* LibTIM::ImageIterator< TImage, T >::im

10.4.3.2 template<class TImage, class T> T* LibTIM::ImageIterator< TImage, T >::ptr

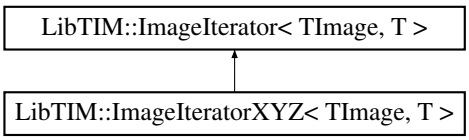
The documentation for this class was generated from the following file:

- Common/**ImageIterators.h**

10.5 LibTIM::ImageIteratorXYZ< TImage, T > Class Template Reference

`#include <ImageIterators.h>`

Inheritance diagram for LibTIM::ImageIteratorXYZ< TImage, T >::



Public Member Functions

- **ImageIteratorXYZ** ()
- **ImageIteratorXYZ** (T *x)
- void **operator=** (const **ImageIterator**< TImage, T > &other)
- **ImageIteratorXYZ** (const **ImageIterator**< TImage, T > &other)
- T & **operator** * ()
- T * **operator** -> ()
- **ImageIteratorXYZ**< TImage, T > & **operator**++ ()
- **ImageIteratorXYZ**< TImage, T > **operator**++ (int)

Public Attributes

- **TCoord x**
- **TCoord y**
- **TCoord z**

template<class TImage, class T> class LibTIM::ImageIteratorXYZ< TImage, T >

10.5.1 Constructor & Destructor Documentation

10.5.1.1 template<class TImage, class T> LibTIM::ImageIteratorXYZ< TImage, T >::ImageIteratorXYZ () [inline]

10.5.1.2 template<class TImage, class T> LibTIM::ImageIteratorXYZ< TImage, T >::ImageIteratorXYZ (T * *x*) [inline]

10.5.1.3 template<class TImage, class T> LibTIM::ImageIteratorXYZ< TImage, T >::ImageIteratorXYZ (const ImageIterator< TImage, T > & *other*) [inline]

10.5.2 Member Function Documentation

10.5.2.1 template<class TImage, class T> T& LibTIM::ImageIteratorXYZ< TImage, T >::operator * () [inline]

Reimplemented from **LibTIM::ImageIterator**< TImage, T > (p. 80).

10.5.2.2 template<class TImage, class T> ImageIteratorXYZ<TImage, T> LibTIM::ImageIteratorXYZ< TImage, T >::operator++ (int) [inline]

Reimplemented from **LibTIM::ImageIterator< TImage, T >** (p. 81).

10.5.2.3 template<class TImage, class T> ImageIteratorXYZ<TImage, T>& LibTIM::ImageIteratorXYZ< TImage, T >::operator++ () [inline]

Reimplemented from **LibTIM::ImageIterator< TImage, T >** (p. 81).

10.5.2.4 template<class TImage, class T> T* LibTIM::ImageIteratorXYZ< TImage, T >::operator → () [inline]

Reimplemented from **LibTIM::ImageIterator< TImage, T >** (p. 81).

10.5.2.5 template<class TImage, class T> void LibTIM::ImageIteratorXYZ< TImage, T >::operator= (const ImageIterator< TImage, T > & *other*) [inline]

10.5.3 Member Data Documentation

10.5.3.1 template<class TImage, class T> TCoord LibTIM::ImageIteratorXYZ< TImage, T >::x

10.5.3.2 template<class TImage, class T> TCoord LibTIM::ImageIteratorXYZ< TImage, T >::y

10.5.3.3 template<class TImage, class T> TCoord LibTIM::ImageIteratorXYZ< TImage, T >::z

The documentation for this class was generated from the following file:

- Common/**ImageIterators.h**

10.6 LibTIM::ImageRegionsInfos< T, T2 > Class Template Reference

`#include <RegionGrowing.h>`

Public Member Functions

- **ImageRegionsInfos** (**Image**< T > &img, **Image**< T2 > &seedRegions)
- double **computeDistance** (**Point**< **TCoord** > &p, **Point**< **TCoord** > &q)

  Return the distance between p and q. The return value can be interpreted as a priority.

- double **computeDistance** (**TOffset** &p, **TOffset** &q)
- void **setPoint** (**Point**< **TCoord** > &p, T2 label)
- void **setPoint** (**TOffset** &p, T2 label)
- void **distance** (**Point**< **TCoord** > &p, **Point**< **TCoord** > &q)
- void **fusion** (**Point**< **TCoord** > &p, T2 label)

template<class T, class T2> class LibTIM::ImageRegionsInfos< T, T2 >

10.6.1 Constructor & Destructor Documentation

10.6.1.1 template<class T, class T2> LibTIM::ImageRegionsInfos< T, T2 >::ImageRegionsInfos (Image< T > & *img*, Image< T2 > & *seedRegions*) [inline]

10.6.2 Member Function Documentation

10.6.2.1 template<class T, class T2> double LibTIM::ImageRegionsInfos< T, T2 >::computeDistance (TOffset & *p*, TOffset & *q*) [inline]

10.6.2.2 template<class T, class T2> double LibTIM::ImageRegionsInfos< T, T2 >::computeDistance (Point< TCoord > & *p*, Point< TCoord > & *q*) [inline]

Return the distance between p and q. The return value can be interpreted as a priority.

10.6.2.3 template<class T, class T2> void LibTIM::ImageRegionsInfos< T, T2 >::distance (Point< TCoord > & *p*, Point< TCoord > & *q*) [inline]

10.6.2.4 template<class T, class T2> void LibTIM::ImageRegionsInfos< T, T2 >::fusion (Point< TCoord > & *p*, T2 *label*) [inline]

10.6.2.5 template<class T, class T2> void LibTIM::ImageRegionsInfos< T, T2 >::setPoint (TOffset & *p*, T2 *label*) [inline]

10.6.2.6 template<class T, class T2> void LibTIM::ImageRegionsInfos< T, T2 >::setPoint (Point< TCoord > & *p*, T2 *label*) [inline]

The documentation for this class was generated from the following file:

- Algorithms/**RegionGrowing.h**

10.7 LibTIM::Node Struct Reference

`#include <ComponentTree.hxx>`

Public Attributes

- int **label**
- int **h**
- int **area**
- **Node** * **father**
- std::vector< long int > **pixels**
- std::vector< struct **Node** * > **childs**
- bool **active**

10.7.1 Member Data Documentation

10.7.1.1 bool LibTIM::Node::active

10.7.1.2 int LibTIM::Node::area

10.7.1.3 std::vector<struct Node *> LibTIM::Node::childs

10.7.1.4 struct Node* LibTIM::Node::father

10.7.1.5 int LibTIM::Node::h

10.7.1.6 int LibTIM::Node::label

10.7.1.7 std::vector<long int > LibTIM::Node::pixels

The documentation for this struct was generated from the following file:

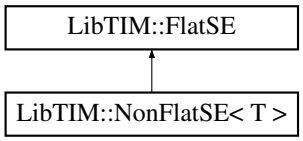
- Algorithms/**ComponentTree.hxx**

10.8 LibTIM::NonFlatSE< T > Class Template Reference

Non-flat structuring elements (or ponderated masks).

`#include <NonFlatSE.h>`

Inheritance diagram for LibTIM::NonFlatSE< T >::



Public Member Functions

- **NonFlatSE** ()
- ~**NonFlatSE** ()
- void **makeChamfer2D** ()
- **NonFlatSE rasterScan** ()
- **NonFlatSE antiRasterScan** ()
- double **getNorm** () const
- T **getValue** (int i) const
- void **addPoint** (**Point**< **TCoord** > p, T attribute)
- void **print** ()
- void **reserve** (size_t size)
- void **clear** ()
- template<> void **makeChamfer2D** ()

10.8.1 Detailed Description

template<class T> class LibTIM::NonFlatSE< T >

Non-flat structuring elements (or ponderated masks).

Can be used for convolution, chanfrein masks, or non-flat morphology

10.8.2 Constructor & Destructor Documentation

10.8.2.1 template<class T> LibTIM::NonFlatSE< T >::NonFlatSE () [inline]

10.8.2.2 template<class T> LibTIM::NonFlatSE< T >::~NonFlatSE () [inline]

10.8.3 Member Function Documentation

10.8.3.1 template<class T> void LibTIM::NonFlatSE< T >::addPoint (Point< TCoord > *p*, T *attribute*) [inline]

10.8.3.2 template<class T> NonFlatSE< T > LibTIM::NonFlatSE< T >::antiRasterScan ()

10.8.3.3 template<class T> void LibTIM::NonFlatSE< T >::clear () [inline]

Reimplemented from **LibTIM::FlatSE** (p. 71).

10.8.3.4 template<class T> double LibTIM::NonFlatSE< T >::getNorm () const

10.8.3.5 template<class T> T LibTIM::NonFlatSE< T >::getValue (int *i*) const [inline]

10.8.3.6 template<> void LibTIM::NonFlatSE< U8 >::makeChamfer2D () [inline]

10.8.3.7 template<class T> void LibTIM::NonFlatSE< T >::makeChamfer2D ()

10.8.3.8 template<class T> void LibTIM::NonFlatSE< T >::print () [inline]

Reimplemented from **LibTIM::FlatSE** (p. 73).

10.8.3.9 template<class T> NonFlatSE< T > LibTIM::NonFlatSE< T >::rasterScan ()

10.8.3.10 template<class T> void LibTIM::NonFlatSE< T >::reserve (size_t *size*) [inline]

Reimplemented from **LibTIM::FlatSE** (p. 73).

The documentation for this class was generated from the following files:

- Common/**NonFlatSE.h**
- Common/**NonFlatSE.hxx**

10.9 LibTIM::OrderedQueue< T > Class Template Reference

Ordered **Queue**(p. 95).

`#include <OrderedQueue.h>`

Public Member Functions

- **OrderedQueue** ()

  Creates an empty ordered queue.

- ~**OrderedQueue** ()
- void **put** (int order, T _val)

  add an element in OQ with specified order

- T **get** ()

  get a element in OQueue

- bool **empty** ()

  bool if OQueue is empty

10.9.1 Detailed Description

template<class T> class LibTIM::OrderedQueue< T >

Ordered **Queue**(p. 95).

This structure allow the use of ordered queue, it is templated to deal with any type. the order is integer and decreasing (order=0 have more priority than order=1)

10.9.2 Constructor & Destructor Documentation

10.9.2.1 template<class T> LibTIM::OrderedQueue< T >::OrderedQueue () [inline]

Creates an empty ordered queue.

10.9.2.2 template<class T> LibTIM::OrderedQueue< T >::~OrderedQueue () [inline]

10.9.3 Member Function Documentation

10.9.3.1 template<class T> bool LibTIM::OrderedQueue< T >::empty () [inline]

bool if OQueue is empty

10.9.3.2 template<class T> T LibTIM::OrderedQueue< T >::get () [inline]

get a element in OQueue

10.9.3.3 template<class T> void LibTIM::OrderedQueue< T >::put (int *order*, T *_val*) [inline]

add an element in OQ with specified order

The documentation for this class was generated from the following file:

- Common/**OrderedQueue.h**

10.10 LibTIM::OrderedQueueDouble< T > Class Template Reference

Ordered **Queue**(p. 95) with double priority.

`#include <OrderedQueue.h>`

Public Member Functions

- **OrderedQueueDouble** ()

  Creates an empty ordered queue.

- ~**OrderedQueueDouble** ()
- void **put** (double order, T _val)

  add an element in OQ with specified order

- T **get** ()

  get a element in OQueue

- bool **empty** ()

  bool if OQueue is empty

10.10.1 Detailed Description

template<class T> class LibTIM::OrderedQueueDouble< T >

Ordered **Queue**(p. 95) with double priority.

This structure allow the use of ordered queue, it is templated to deal with any type. the order is double and is decreasing (order=0 have more priority than order=1)

10.10.2 Constructor & Destructor Documentation

10.10.2.1 template<class T> LibTIM::OrderedQueueDouble< T >::OrderedQueueDouble () [inline]

Creates an empty ordered queue.

10.10.2.2 template<class T> LibTIM::OrderedQueueDouble< T >::~OrderedQueueDouble () [inline]

10.10.3 Member Function Documentation

10.10.3.1 template<class T> bool LibTIM::OrderedQueueDouble< T >::empty () [inline]

bool if OQueue is empty

10.10.3.2 template<class T> T LibTIM::OrderedQueueDouble< T >::get () [inline]

get a element in OQueue

10.10.3.3 template<class T> void LibTIM::OrderedQueueDouble< T >::put (double *order*, T _ *val*) [inline]

add an element in OQ with specified order

The documentation for this class was generated from the following file:

- Common/**OrderedQueue.h**

10.11 LibTIM::Point< T > Class Template Reference

Point(p. 93) Structure.

`#include <Point.h>`

Public Member Functions

- **Point** (**TCoord x**=0, **TCoord y**=0, **TCoord z**=0)
- **Point** & **operator+=** (**Point**< T > q)
- **Point** & **operator-=** (**Point**< T > q)
- bool **operator==** (**Point**< T > q)
- void **operator=** (const **Point**< T > &q)
- **Point** (const **Point** &q)
- void **operator()** (T **x**, T **y**, T **z**)
- void **print** ()

Public Attributes

- T **x**
- T **y**
- T **z**

10.11.1 Detailed Description

template<class T> class LibTIM::Point< T >

Point(p. 93) Structure.

Basic structure to manipulate 3D points of type T

10.11.2 Constructor & Destructor Documentation

10.11.2.1 template<class T> LibTIM::Point< T >::Point (TCoord *x* = 0, TCoord *y* = 0, TCoord *z* = 0) `[inline]`

10.11.2.2 template<class T> LibTIM::Point< T >::Point (const Point< T > & *q*) `[inline]`

10.11.3 Member Function Documentation

10.11.3.1 template<class T> void LibTIM::Point< T >::operator() (T *x*, T *y*, T *z*) `[inline]`

10.11.3.2 template<class T> Point& LibTIM::Point< T >::operator+= (Point< T > *q*) `[inline]`

10.11.3.3 template<class T> Point& LibTIM::Point< T >::operator-= (Point< T > *q*) `[inline]`

10.11.3.4 template<class T> void LibTIM::Point< T >::operator= (const Point< T > & *q*) `[inline]`

10.11.3.5 template<class T> bool LibTIM::Point< T >::operator== (Point< T > *q*) `[inline]`

10.11.3.6 template<class T> void LibTIM::Point< T >::print () `[inline]`

10.11.4 Member Data Documentation

10.11.4.1 template<class T> T LibTIM::Point< T >::x

10.11.4.2 template<class T> T LibTIM::Point< T >::y

10.11.4.3 template<class T> T LibTIM::Point< T >::z

The documentation for this class was generated from the following file:

- Common/**Point.h**

10.12 LibTIM::Queue< T > Class Template Reference

`#include <OrderedQueue.h>`

Public Member Functions

- **Queue** ()
- void **put** (T _t)
  add an element in **Queue**(p. 95)
- T **get** ()
  get a element in **Queue**(p. 95)
- bool **empty** ()
  bool if **Queue**(p. 95) *is empty*

template<class T> class LibTIM::Queue< T >

10.12.1 Constructor & Destructor Documentation

10.12.1.1 template<class T> LibTIM::Queue< T >::Queue () [inline]

10.12.2 Member Function Documentation

10.12.2.1 template<class T> bool LibTIM::Queue< T >::empty () [inline]

bool if **Queue**(p. 95) is empty

10.12.2.2 template<class T> T LibTIM::Queue< T >::get () [inline]

get a element in **Queue**(p. 95)

10.12.2.3 template<class T> void LibTIM::Queue< T >::put (T _ *t*) [inline]

add an element in **Queue**(p. 95)
The documentation for this class was generated from the following file:

- Common/**OrderedQueue.h**

10.13 Random Class Reference

`#include <random-singleton.h>`

Static Public Member Functions

- static void **Randomize** (long thatSeed=0)
- template<typename T> static T **Uniform** (T min, T max)
- static double **Uniform** (void)
- static double **Gaussian** (double mean=0, double standardDeviation=1)
- static double **Exponential** (double lambda)

10.13.1 Member Function Documentation

10.13.1.1 static double Random::Exponential (double *lambda*) [inline, static]

10.13.1.2 double Random::Gaussian (double *mean* = 0, double *standardDeviation* = 1) [static]

10.13.1.3 void Random::Randomize (long *thatSeed* = 0) [static]

10.13.1.4 static double Random::Uniform (void) [inline, static]

10.13.1.5 template<typename T> static T Random::Uniform (T *min*, T *max*) [inline, static]

The documentation for this class was generated from the following files:

- Algorithms/**random-singleton.h**
- Algorithms/**random-singleton.cpp**

10.14 LibTIM::Region Struct Reference

`#include <RegionGrowing.hxx>`

Public Attributes

- long int * **sumIntensity**
- int * **nbPoints**

10.14.1 Member Data Documentation

10.14.1.1 int* LibTIM::Region::nbPoints

10.14.1.2 long int* LibTIM::Region::sumIntensity

The documentation for this struct was generated from the following file:

- Algorithms/**RegionGrowing.hxx**

10.15 LibTIM::Table< T, N > Struct Template Reference

`#include <Types.h>`

Public Member Functions

- **Table** ()
- **Table** (const **Table** &v)
- **Table** (int p)
- **Table** (int *vect)
- T & **operator[]** (int i)

Public Attributes

- T **el** [N]

template<class T, int N> struct LibTIM::Table< T, N >

10.15.1 Constructor & Destructor Documentation

10.15.1.1 template<class T, int N> LibTIM::Table< T, N >::Table () [inline]

10.15.1.2 template<class T, int N> LibTIM::Table< T, N >::Table (const Table< T, N > & *v*) [inline]

10.15.1.3 template<class T, int N> LibTIM::Table< T, N >::Table (int *p*) [inline]

10.15.1.4 template<class T, int N> LibTIM::Table< T, N >::Table (int * *vect*) [inline]

10.15.2 Member Function Documentation

10.15.2.1 template<class T, int N> T& LibTIM::Table< T, N >::operator[] (int *i*) [inline]

10.15.3 Member Data Documentation

10.15.3.1 template<class T, int N> T LibTIM::Table< T, N >::el[N]

The documentation for this struct was generated from the following file:

- Common/**Types.h**

Chapter 11

LibTIM File Documentation

11.1 Algorithms/AdaptativeSE.h File Reference

```
#include "AdaptativeSE.hxx"
```

11.2 Algorithms/AdaptativeSE.hxx File Reference

```
#include <cmath>
```

Namespaces

- namespace **LibTIM**

Functions

- template<class T> void **LibTIM::dynamicSeNormL2** (Image< T > &img, const Point< **TCoord** > &p, const FlatSE &B, int param, FlatSE &se)
- template<class T> void **LibTIM::dynamicSeNormL2Rand** (Image< T > &img, const Point< **TCoord** > &p, const FlatSE &B, int param, FlatSE &se, int nbPoints)
- template<class T> void **LibTIM::dynamicSeNormL2NPoints** (Image< T > &img, const Point< **TCoord** > &p, const FlatSE &B, int NPoints, FlatSE &se)
- template<class T> Image< unsigned char > **LibTIM::computeNeighborhoodS1** (const Image< T > &im)
- template<class T> Image< vector< bool > > **LibTIM::computeNeighborhoodS1v2** (const Image< T > &im)
- template<class T> void **LibTIM::dynamicSeS1** (Image< T > &img, const Point< **TCoord** > &p, int param, FlatSE &se)
- void **LibTIM::dynamicSeS1v2** (Image< vector< bool > > &img, const Point< **TCoord** > &p, int param, FlatSE &se)
- template<class T> Image< T > **LibTIM::printSeAtPoint** (Image< T > &img, FlatSE &se, Point< **TCoord** > &p)

11.3 Algorithms/ComponentTree.h File Reference

```
#include "ComponentTree.hxx"
```

11.4 Algorithms/ComponentTree.hxx File Reference

```
#include "Common/OrderedQueue.h"
```

Namespaces

- namespace **LibTIM**

Classes

- struct **LibTIM::Node**

Typedefs

- typedef Node **LibTIM::tNode**

Functions

- vector< long int > **LibTIM::merge_pixels** (**tNode** *tree)

  Aggregate all subpixels from the node tree.

- void **LibTIM::filterArea** (**tNode** *root, int area)
- void **LibTIM::printTree** (**tNode** *tree)
- void **LibTIM::make_father** (**tNode** ***index, int label1, int label2, int h1, int h2)
- **tNode** * **LibTIM::init_tree** (void)
- Image< **U8** > **LibTIM::reconstructImage** (**tNode** *tree, const **TSize** *size)

  Reconstruct image from tree.

- template<class T> **tNode** * **LibTIM::computeComponentTree** (Image< T > &im, FlatSE &se)
- void **LibTIM::father** (**tNode** *tree, **tNode** *child)
- **tNode** * **LibTIM::init_tree** (int h, int n)
- template<class T> **tNode** * **LibTIM::computeComponentTreeBensMethod** (Image< T > &im, FlatSE &se)
- int **LibTIM::computeArea** (**tNode** *tree)
- template<class T> int **LibTIM::flood** (Image< T > &im, std::map< int, std::queue< int > > &oq, int h, int hMin, vector< int > &STATUS, vector< int > &number_nodes, vector< bool > &node_at_level, FlatSE &se, std::map< T, std::map< **TLabel**, struct Node * > > &index)
- template<class T> int **LibTIM::flood2** (Image< T > &im, std::map< int, std::queue< int > > &oq, int h, int hMin, vector< int > &STATUS, vector< int > &number_nodes, vector< bool > &node_at_level, FlatSE &se, std::map< T, std::map< **TLabel**, struct Node * > > &index)

  New method to deal with neighbors.

- template<class T> **tNode** * **LibTIM::computeComponentTree2V1** (Image< T > &im, FlatSE &se)

  Following Salembier recursive implementation...

- template<class T> **tNode** * **LibTIM::computeComponentTree2** (Image< T > &im, FlatSE &se)

  Following Salembier recursive implementation...

11.5 Algorithms/ConnectedComponents.h File Reference

```
#include "ConnectedComponents.hxx"
```

11.6 Algorithms/ConnectedComponents.hxx File Reference

```
#include <queue>
#include <map>
#include "Algorithms/Morphology.h"
```

Namespaces

- namespace **LibTIM**

Functions

- template<class T> Image< **TLabel** > **LibTIM::labelConnectedComponents** (Image< T > &img, FlatSE &se)
- void **LibTIM::keepIestLargestComponent** (Image< **TLabel** > &img, FlatSE &se, int Iest)

11.7 Algorithms/DistanceTransform.h File Reference

```
#include "DistanceTransform.hxx"
```

11.8 Algorithms/DistanceTransform.hxx File Reference

Namespaces

- namespace **LibTIM**

Functions

- template<class T, class T2> Image< **U16** > **LibTIM::chamferDistanceTransform** (Image< T > &im, NonFlatSE< T2 > &mask)

11.9 Algorithms/KMeans.h File Reference

```
#include "KMeans.hxx"
```

11.10 Algorithms/KMeans.hxx File Reference

```
#include <cmath>
#include <vector>
```

Namespaces

- namespace **LibTIM**

Defines

- #define **EPSILON** 0.0000001

Functions

- template<class T> Image< **TLabel** > **LibTIM::kMeansScalarImage** (const Image< T > &img, std::vector< double > &centroids)

11.10.1 Define Documentation

11.10.1.1 #define EPSILON 0.0000001

11.11 Algorithms/Misc.h File Reference

```
#include "Misc.hxx"
```

11.12 Algorithms/Misc.hxx File Reference

Namespaces

- namespace **LibTIM**

Functions

- template<class T> void **LibTIM::adjustContrast** (Image< T > &im)
- template<class T> void **LibTIM::adjustContrast** (Image< T > &im, T A, T B)

  Same thing but with A and B given in parameters.

- template<class T, class T2> Image< T > **LibTIM::computeMarkerMean** (Image< T > &src, Image< T2 > &marker)

  For each marker compute the mean of the points on original image.

- template<class T, class T2> Image< T > **LibTIM::computeMarkerMeanFast** (Image< T > &src, Image< T2 > &marker)

  For each marker compute the mean of the points on original image.

- template<class T> void **LibTIM::decimateTemplate** (Image< T > &im, int nx=1, int ny=1, int nz=1)

  Image(p. 75) *decimation by imposing a regular grid -> useful for simplifying structuring elements.*

- std::map< **TLabel**, Point< double > > **LibTIM::centroids** (Image< **TLabel** > &im)

  compute the centroids of labelled objects (first moments) in 2D images

- template<class T> void **LibTIM::drawContour** (Image< T > &im, const Image< **U8** > &mask, const T val)

11.13 Algorithms/Morphology.h File Reference

```
#include "Morphology.hxx"
```

11.14 Algorithms/Morphology.hxx File Reference

```
#include <queue>
#include "Common/FlatSE.h"
#include "Common/Image.h"
```

Namespaces

- namespace **LibTIM**

Functions

- template<class T> void **LibTIM::addBorders** (Image< T > &im, **TCoord** *preWidth, **TCoord** *postWidth, T value)
- template<class T> void **LibTIM::addBorders** (Image< T > &im, FlatSE &se, T value)
- template<class T> Image< T > **LibTIM::dilation** (Image< T > im, FlatSE se)

  Basic flat-dilation algorithm.

- template<class T> Image< T > **LibTIM::erosion** (Image< T > im, FlatSE se)

  Basic flat-erosion algorithm.

- template<class T> Image< T > **LibTIM::dilationBorderMax** (Image< T > im, FlatSE se)

  Border max version of dilation.

- template<class T> Image< T > **LibTIM::erosionBorderMin** (Image< T > im, FlatSE se)

  Border min version of erosion.

- template<class T> Image< T > **LibTIM::opening** (Image< T > im, FlatSE se)

  Opening.

- template<class T> Image< T > **LibTIM::closing** (Image< T > im, FlatSE se)

  Closing.

- template<class T> Image< T > **LibTIM::morphologicalGradient** (Image< T > im, FlatSE se)

  Morphological gradient.

- template<class T> Image< T > **LibTIM::internalMorphologicalGradient** (Image< T > im, FlatSE se)

  Internal morphological gradient.

- template<class T> Image< T > **LibTIM::externalMorphologicalGradient** (Image< T > im, FlatSE se)

  External morphological gradient.

- template<class T> Image< T > **LibTIM::rankFilter** (Image< T > im, FlatSE se, int rank)

Rank filter.

- template<class T> Image< **U8** > **LibTIM::regionalMinima** (Image< T > img, FlatSE se)

  Regional Minima Extraction.

- template<class T> Image< **U8** > **LibTIM::regionalMaxima** (Image< T > img, FlatSE se)

  Regional Maxima Extraction.

- template<class T> void **LibTIM::geodesicReconstructionByErosion** (Image< T > &marker, Image< T > mask, FlatSE &se)

  Geodesic reconstruction by erosion.

- template<class T> void **LibTIM::geodesicReconstructionByDilation** (Image< T > &marker, Image< T > mask, FlatSE &se)

  Geodesic reconstruction by dilation.

- template<class T> void **LibTIM::hMinFilter** (Image< T > &img, FlatSE &se, int h)

  h-Min filter

- template<class T> void **LibTIM::hMaxFilter** (Image< T > &img, FlatSE &se, int h)

  h-Max filter

- template<class T> Image< int > **LibTIM::hitOrMissDifferenceImage** (Image< T > &im, FlatSE &seA, FlatSE &seB)

  Hit-or-miss difference **Image**(p. 75).

- template<class T> int **LibTIM::hitOrMissMaximumDifference** (Image< T > im, FlatSE &seA, FlatSE &seB)

  Maximum of the hitOrMissDifferenceImage.

- template<class T> Image< T > **LibTIM::hitOrMissIntegralK** (Image< T > &im, FlatSE &seA, FlatSE &seB)

  Grey-level hit-or-miss: Soille's version.

- template<class T> Image< T > **LibTIM::hitOrMissSupremalH** (Image< T > &im, FlatSE &seA, FlatSE &seB)

  Grey-level hit-or-miss: Ronse's version.

- template<class T> Image< T > **LibTIM::hitOrMissSupremalK** (Image< T > &im, FlatSE &seA, FlatSE &seB)

  Supremal K version of grey-level hit-or-miss.

- template<class T> Image< T > **LibTIM::hitOrMissIntegralKOpening** (Image< T > &im, FlatSE &seA, FlatSE &seB)

  Grey-level hit-or-miss opening: Soille's version.

- template<class T> Image< T > **LibTIM::hitOrMissSupremalKOpening** (Image< T > &im, FlatSE &seA, FlatSE &seB)

  Grey-level hit-or-miss opening: Supremal K version.

- template<class T> Image< T > **LibTIM::hitOrMissSupremalHOpening** (Image< T > &im, FlatSE &seA, FlatSE &seB)

  Grey-level hit-or-miss opening: Ronse's version.

11.15 Algorithms/random-singleton.cpp File Reference

```
#include "random-singleton.h"
#include <cfloat>
```

Namespaces

- namespace **std**

11.16 Algorithms/random-singleton.h File Reference

```
#include <cmath>
#include <limits>
```

Classes

- class **Random**

11.17 Algorithms/RegionGrowing.h File Reference

```
#include "Common/OrderedQueue.h"
#include "Common/Image.h"
#include <cmath>
#include <map>
#include "RegionGrowing.hxx"
```

Namespaces

- namespace **LibTIM**

Classes

- class **LibTIM::ImageRegionsInfos**< **T**, **T2** >

11.18 Algorithms/RegionGrowing.hxx File Reference

```
#include "Common/FlatSE.h"
#include "Common/Image.h"
#include "Algorithms/Misc.h"
#include <list>
```

Namespaces

- namespace **LibTIM**

Classes

- struct **LibTIM::Region**

Functions

- int **LibTIM::labelToOffset** (**TLabel** label)
- double **LibTIM::computePriority** (pair< long int, int > offset, Image< **U8** > &src, struct Region &region)
- template<class T, class T2> void **LibTIM::RegionGrowingCriterion** (Image< T > &src, Image< T2 > &marker, FlatSE &se, bool observe=false)
- template<class T> void **LibTIM::seededRegionGrowingExactAlgorithm** (Image< T > &im, Image< **TLabel** > &marker, FlatSE &se, bool observe=false)

  Seeded region-growing algorithm: non-biased implementation.

- template<class T, class T2> void **LibTIM::seededRegionGrowing** (Image< T > &img, Image< T2 > &marker, FlatSE &se, bool observe=false)
- template<class T, class T2> void **LibTIM::seededRegionGrowing0** (Image< T > &img, Image< T2 > &marker, FlatSE &se, bool observe=false)

11.19 Algorithms/Tarjan.h File Reference

```
#include "Tarjan.hxx"
```

11.20 Algorithms/Tarjan.hxx File Reference

```
#include <vector>
```

Namespaces

- namespace **LibTIM**

Typedefs

- typedef vector< int > **LibTIM::treeType**

Functions

- void **LibTIM::MakeSet** (**treeType** &tree, const int &offset)
- int **LibTIM::Find** (**treeType** &tree, const int &offset)
- int **LibTIM::FindSimple** (**treeType** &tree, const int &offset)
- int **LibTIM::Link** (**treeType** &tree, int &x, int &y)
- void **LibTIM::MakeSet** (int *tree, const int &offset)
- int **LibTIM::Find** (int *tree, const int &offset)
- int **LibTIM::Link** (int *tree, int &x, int &y)
- template<class T> Image< **TLabel** > **LibTIM::labelConnectedComponentsTarjan** (const Image< T > &im, const FlatSE &se)
- template<class T> Image< **TLabel** > **LibTIM::labelConnectedComponentsTarjan2** (const Image< T > &im, const FlatSE &se)

11.21 Algorithms/TemplateMatching.h File Reference

```
#include "TemplateMatching.hxx"
```

11.22 Algorithms/TemplateMatching.hxx File Reference

Namespaces

- namespace **LibTIM**

Functions

- template<class T> Image< int > **LibTIM::templateMatchingL2** (const Image< T > &im, const NonFlatSE< **U8** > &mask)
- template<class T, class T2> Image< T > **LibTIM::printBestTemplate** (const Image< T2 > &resTM, const Image< T > &im, const FlatSE &A, T2 value)

  Same thing but with two templates: one for foreground (255), the other for background (0).

- template<class T> Image< double > **LibTIM::templateMatchingCorrelation** (const Image< T > &im, const NonFlatSE< **U8** > &mask)

11.23 Algorithms/Thresholding.h File Reference

```
#include "Thresholding.hxx"
```

11.24 Algorithms/Thresholding.hxx File Reference

Namespaces

- namespace **LibTIM**

Functions

- template<class T> Image< T > **LibTIM::threshold** (Image< T > &im, T tLow, T tHigh)

  Thresholding.

- template<class T> Image< T > **LibTIM::threshold** (Image< T > &im, int tLow, int tHigh)

  Thresholding (overloaded).

- template<class T> Image< **U8** > **LibTIM::binarize** (Image< T > &im)

  Binarization: 0->0, !=0 -> 255.

11.25 Algorithms/ViscousWatershed.h File Reference

```
#include "ViscousWatershed.hxx"
```

11.26 Algorithms/ViscousWatershed.hxx File Reference

```
#include <cmath>
```

```
#include <map>
```

Namespaces

- namespace **LibTIM**

Functions

- template<class T> int **LibTIM::functionR0** (double r0, T t)
- template<class T> void **LibTIM::viscousClosingMercuryBasic** (Image< T > &src, double r0)
- template<class T> void **LibTIM::viscousClosingMercury** (Image< T > &src, double r0)

  Version two: we try to optimize a little.

11.27 Algorithms/Watershed.h File Reference

```
#include "Watershed.hxx"
```

11.28 Algorithms/Watershed.hxx File Reference

```
#include "Common/OrderedQueue.h"
```

Namespaces

- namespace **LibTIM**

Functions

- template<class T, class T2> void **LibTIM::watershedMeyer** (Image< T > &img, Image< T2 > &marker, FlatSE &se, bool observe=false)

11.29 Common/FlatSE.h File Reference

```
#include <iostream>
#include <vector>
#include <limits>
#include "Image.h"
#include "Point.h"
#include "FlatSE.hxx"
```

Namespaces

- namespace **LibTIM**

Classes

- class **LibTIM::FlatSE**

  Container base class for flat structuring elements (or binary masks).

11.30 Common/FlatSE.hxx File Reference

Namespaces

- namespace **LibTIM**

11.31 Common/Histogram.h File Reference

```
#include "Image.h"
#include <map>
#include "Histogram.hxx"
```

Namespaces

- namespace **LibTIM**

Classes

- class **LibTIM::Histogram**< **T** >

  Container for histograms.

11.32 Common/Histogram.hxx File Reference

```
#include <fstream>
```

Namespaces

- namespace **LibTIM**

11.33 Common/Image.h File Reference

```
#include <iostream>
#include <limits>
#include <vector>
#include "Types.h"
#include "Point.h"
#include "OrderedQueue.h"
#include "ImageIterators.h"
#include "Image.hxx"
#include "ImageIO.hxx"
```

Namespaces

- namespace **LibTIM**

Classes

- class **LibTIM::Image**< **T** >

  Container base for images of generic type T in **LibTIM**(p. 59).

Defines

- #define **Image_internal_h**

Functions

- template<class T> Image< T > **LibTIM::operator**+ (Image< T > &a, Image< T > &b)

  Image(p. 75) *operators.*

- template<class T> Image< T > **LibTIM::operator-** (Image< T > &a, Image< T > &b)
- template<class T> Image< T > **LibTIM::operator** * (Image< T > &a, Image< T > &b)
- template<class T> Image< T > **LibTIM::operator**+ (Image< T > &a, T s)

  Mixed mode arithmetic: operations with a scalar.

- template<class T> Image< T > **LibTIM::operator-** (Image< T > &a, T s)
- template<class T> Image< T > **LibTIM::operator** * (Image< T > &a, T s)

11.33.1 Define Documentation

11.33.1.1 #define Image_internal_h

11.34 Common/Image.hxx File Reference

```
#include <assert.h>
```

Namespaces

- namespace **LibTIM**

11.35 Common/ImageIO.hxx File Reference

```
#include <fstream>
#include <string>
#include <sstream>
```

Namespaces

- namespace **LibTIM**

Functions

- std::string **LibTIM::GImageIO_NextLine** (std::ifstream &file)
- void **LibTIM::GImageIO_ReadPPMHeader** (std::ifstream &file, std::string &format, unsigned int &width, unsigned int &height, unsigned int &colormax)

11.36 Common/ImageIterators.h File Reference

```
#include "Image.h"
```

Namespaces

- namespace **LibTIM**

Classes

- class **LibTIM::ImageIterator**< **TImage**, **T** >
- class **LibTIM::ImageIteratorXYZ**< **TImage**, **T** >

11.37 Common/NonFlatSE.h File Reference

```
#include <cmath>
#include <limits>
#include "Point.h"
#include "FlatSE.h"
#include "NonFlatSE.hxx"
```

Namespaces

- namespace **LibTIM**

Classes

- class **LibTIM::NonFlatSE< T >**

  Non-flat structuring elements (or ponderated masks).

11.38 Common/NonFlatSE.hxx File Reference

Namespaces

- namespace **LibTIM**

11.39 Common/OrderedQueue.h File Reference

```
#include <utility>
#include <functional>
#include <queue>
#include <vector>
#include <set>
#include <map>
```

Namespaces

- namespace **LibTIM**

Classes

- class **LibTIM::OrderedQueue**< **T** >
  Ordered **Queue**(p. 95).
- class **LibTIM::OrderedQueueDouble**< **T** >
  Ordered **Queue**(p. 95) *with double priority.*
- class **LibTIM::Queue**< **T** >

11.40 Common/Point.h File Reference

Namespaces

- namespace **LibTIM**

Classes

- class **LibTIM::Point**< **T** >
  Point(p. 93) *Structure.*

Functions

- template<class T> Point< T > **LibTIM::operator**+ (Point< T > p, Point< T > q)
- template<class T> Point< T > **LibTIM::operator-** (Point< T > p, Point< T > q)

11.41 Common/Types.h File Reference

Namespaces

- namespace **LibTIM**

Classes

- struct **LibTIM::Table**< **T**, **N** >

Typedefs

- typedef unsigned char **LibTIM::U8**
- typedef signed char **LibTIM::S8**
- typedef unsigned short **LibTIM::U16**
- typedef signed short **LibTIM::S16**
- typedef unsigned long **LibTIM::U32**
- typedef signed long **LibTIM::S32**
- typedef Table< **U8**, 3 > **LibTIM::RGB**
- typedef unsigned short **LibTIM::TSize**
- typedef double **LibTIM::TSpacing**
- typedef int **LibTIM::TCoord**
- typedef unsigned long **LibTIM::TLabel**
- typedef long **LibTIM::TOffset**

Variables

- const float **LibTIM::FLOAT_EPSILON** = 0.0000000001f

Index

~Image
    Image, 54
~NonFlatSE
    LibTIM::NonFlatSE, 88
~OrderedQueue
    LibTIM::OrderedQueue, 89
~OrderedQueueDouble
    LibTIM::OrderedQueueDouble, 91

active
    LibTIM::Node, 86
addBorders
    basisFunctions, 23
addPoint
    LibTIM::FlatSE, 71
    LibTIM::NonFlatSE, 88
adjustContrast
    misc, 19
Algorithms/ Directory Reference, 57
Algorithms/AdaptativeSE.h, 99
Algorithms/AdaptativeSE.hxx, 100
Algorithms/ComponentTree.h, 101
Algorithms/ComponentTree.hxx, 102
Algorithms/ConnectedComponents.h, 104
Algorithms/ConnectedComponents.hxx, 105
Algorithms/DistanceTransform.h, 106
Algorithms/DistanceTransform.hxx, 107
Algorithms/KMeans.h, 108
Algorithms/KMeans.hxx, 109
Algorithms/Misc.h, 110
Algorithms/Misc.hxx, 111
Algorithms/Morphology.h, 112
Algorithms/Morphology.hxx, 113
Algorithms/random-singleton.cpp, 116
Algorithms/random-singleton.h, 117
Algorithms/RegionGrowing.h, 118
Algorithms/RegionGrowing.hxx, 119
Algorithms/Tarjan.h, 120
Algorithms/Tarjan.hxx, 121
Algorithms/TemplateMatching.h, 122
Algorithms/TemplateMatching.hxx, 123
Algorithms/Thresholding.h, 124
Algorithms/Thresholding.hxx, 125
Algorithms/ViscousWatershed.h, 126
Algorithms/ViscousWatershed.hxx, 127
Algorithms/Watershed.h, 128
Algorithms/Watershed.hxx, 129
antiRasterScan
    LibTIM::NonFlatSE, 88
area
    LibTIM::Node, 86

Basis Functions, 22
basisFunctions
    addBorders, 23
    closing, 23
    dilation, 23
    dilationBorderMax, 23
    erosion, 23
    erosionBorderMin, 23
    externalMorphologicalGradient, 23
    internalMorphologicalGradient, 24
    morphologicalGradient, 24
    opening, 24
    rankFilter, 24
begin
    Image, 49
    LibTIM::FlatSE, 71
begin_point
    LibTIM::FlatSE, 71
binarize
    thresholding, 39

ccLabelling
    keepIestLargestComponent, 16
    labelConnectedComponents, 16
ccTree
    computeArea, 14
    computeComponentTree, 14
    computeComponentTree2, 14
    computeComponentTree2V1, 14
    computeComponentTreeBensMethod, 14
    father, 14
    filterArea, 14
    flood, 14
    flood2, 14
    init_tree, 14, 15
    make_father, 15
    printTree, 15
    reconstructImage, 15

centroids
misc, 19
chamferDistanceTransform
DistanceTransform, 17
childs
LibTIM::Node, 86
clear
LibTIM::FlatSE, 71
LibTIM::NonFlatSE, 88
closing
basisFunctions, 23
Common/ Directory Reference, 58
Common/FlatSE.h, 130
Common/FlatSE.hxx, 131
Common/Histogram.h, 132
Common/Histogram.hxx, 133
Common/Image.h, 134
Common/Image.hxx, 135
Common/ImageIO.hxx, 136
Common/ImageIterators.h, 137
Common/NonFlatSE.h, 138
Common/NonFlatSE.hxx, 139
Common/OrderedQueue.h, 140
Common/Point.h, 141
Common/Types.h, 142
Component-Tree Based Algorithms, 13
computeArea
ccTree, 14
computeComponentTree
ccTree, 14
computeComponentTree2
ccTree, 14
computeComponentTree2V1
ccTree, 14
computeComponentTreeBensMethod
ccTree, 14
computeDistance
LibTIM::ImageRegionsInfos, 84
computeMarkerMean
misc, 19
computeMarkerMeanFast
misc, 19
computeNeighborhoodS1
LibTIM, 65
computeNeighborhoodS1v2
LibTIM, 65
computePriority
LibTIM, 66
Connected Components Labelling, 16
Connected Operators, 29
connectedOperators
hMaxFilter, 29
hMinFilter, 29
const_iterator
Image, 48
const_iteratorXYZ
Image, 48
const_reverse_iterator
Image, 48
Constrained Watershed Algorithms, 40
constrainedWatershed
viscousClosingMercury, 40
viscousClosingMercuryBasic, 40
copy
Image, 49
copyFast
Image, 49
crop
Image, 49

Data Structures, 44
decimateTemplate
misc, 20
dilation
basisFunctions, 23
dilationBorderMax
basisFunctions, 23
distance
LibTIM::ImageRegionsInfos, 84
Distance Transform, 17
DistanceTransform
chamferDistanceTransform, 17
drawContour
misc, 20
dynamicSeNormL2
LibTIM, 66
dynamicSeNormL2NPoints
LibTIM, 66
dynamicSeNormL2Rand
LibTIM, 66
dynamicSeS1
LibTIM, 66
dynamicSeS1v2
LibTIM, 66

el
LibTIM::Table, 98
empty
LibTIM::OrderedQueue, 89
LibTIM::OrderedQueueDouble, 91
LibTIM::Queue, 95
end
Image, 49
LibTIM::FlatSE, 71
end_point
LibTIM::FlatSE, 71
enlarge
Image, 49

EPSILON
KMeans.hxx, 109
erosion
basisFunctions, 23
erosionBorderMin
basisFunctions, 23
Exponential
Random, 96
externalMorphologicalGradient
basisFunctions, 23
extremaExtraction
regionalMaxima, 26
regionalMinima, 26

father
ccTree, 14
LibTIM::Node, 86
fill
Image, 49
filterArea
ccTree, 14
Find
Tarjan, 35
FindSimple
Tarjan, 35
Flat Structuring Elements, 42
FlatSE
LibTIM::FlatSE, 71
FlatSE::operator=
LibTIM::FlatSE, 71
FlatSE::toImage
LibTIM::FlatSE, 71
FLOAT_EPSILON
LibTIM, 66
flood
ccTree, 14
flood2
ccTree, 14
functionR0
LibTIM, 66
fusion
LibTIM::ImageRegionsInfos, 84

Gaussian
Random, 96
Geodesic Reconstruction, 28
geodesicReconstructionByDilation
reconstruction, 28
geodesicReconstructionByErosion
reconstruction, 28
get
LibTIM::OrderedQueue, 89
LibTIM::OrderedQueueDouble, 91
LibTIM::Queue, 95
getBufSize
Image, 49
getData
Image, 50
getMax
Image, 50
getMin
Image, 50
getNbPoints
LibTIM::FlatSE, 71
getNegativeOffsets
LibTIM::FlatSE, 71
getNorm
LibTIM::NonFlatSE, 88
getOffset
Image, 50
LibTIM::FlatSE, 72
getPoint
LibTIM::FlatSE, 72
getPositiveOffsets
LibTIM::FlatSE, 72
getReflection
Image, 50
getSize
Image, 50
getSizeX
Image, 50
getSizeY
Image, 50
getSizeZ
Image, 50
getSpacing
Image, 50
getSpacingX
Image, 50
getSpacingY
Image, 50
getSpacingZ
Image, 50
getValue
LibTIM::NonFlatSE, 88
GImageIO_NextLine
LibTIM, 66
GImageIO_ReadPPMHeader
LibTIM, 66

h
LibTIM::Node, 86
Histogram, 43
LibTIM::Histogram, 74
hitOrMissDifferenceImage
interval, 30
hitOrMissIntegralK
interval, 30

hitOrMissIntegralKOpening
interval, 31
hitOrMissMaximumDifference
interval, 31
hitOrMissSupremalH
interval, 31
hitOrMissSupremalHOpening
interval, 31
hitOrMissSupremalK
interval, 31
hitOrMissSupremalKOpening
interval, 31
hMaxFilter
connectedOperators, 29
hMinFilter
connectedOperators, 29

im
LibTIM::ImageIterator, 81
Image, 45
~Image, 54
begin, 49
const_iterator, 48
const_iteratorXYZ, 48
const_reverse_iterator, 48
copy, 49
copyFast, 49
crop, 49
end, 49
enlarge, 49
fill, 49
getBufSize, 49
getData, 50
getMax, 50
getMin, 50
getOffset, 50
getReflection, 50
getSize, 50
getSizeX, 50
getSizeY, 50
getSizeZ, 50
getSpacing, 50
getSpacingX, 50
getSpacingY, 50
getSpacingZ, 50
Image, 50, 51
isPosValid, 51
iterator, 48
iteratorXYZ, 48
load, 51
operator &=, 51
operator *, 51
operator *=, 51
operator!, 51
operator(), 51, 52
operator+, 52
operator+=, 52
operator-, 52, 53
operator-=, 53
operator/=, 53
operator=, 53
operator==, 53
operator|=, 53
print, 53
rbegin, 53
rend, 53
reverse_iterator, 49
save, 53
setImageInfos, 53
setSize, 54
setSpacingX, 54
setSpacingY, 54
setSpacingZ, 54
Image Processing Basis Functions, 38
Image.h
Image_internal_h, 134
Image_internal_h
Image.h, 134
ImageIterator
LibTIM::ImageIterator, 80
ImageIteratorXYZ
LibTIM::ImageIteratorXYZ, 82
ImageRegionsInfos
LibTIM::ImageRegionsInfos, 84
init_tree
ccTree, 14, 15
internalMorphologicalGradient
basisFunctions, 24
interval
hitOrMissDifferenceImage, 30
hitOrMissIntegralK, 30
hitOrMissIntegralKOpening, 31
hitOrMissMaximumDifference, 31
hitOrMissSupremalH, 31
hitOrMissSupremalHOpening, 31
hitOrMissSupremalK, 31
hitOrMissSupremalKOpening, 31
Interval Operators, 30
isPosValid
Image, 51
iterator
Image, 48
LibTIM::FlatSE, 71
iterator_offset
LibTIM::FlatSE, 71
iterator_point
LibTIM::FlatSE, 71
iteratorXYZ

Image, 48

K-Means, 18
keepIestLargestComponent
ccLabelling, 16
kMeans
kMeansScalarImage, 18
KMeans.hxx
EPSILON, 109
kMeansScalarImage
kMeans, 18

label
LibTIM::Node, 86
labelConnectedComponents
ccLabelling, 16
labelConnectedComponentsTarjan
Tarjan, 35
labelConnectedComponentsTarjan2
Tarjan, 35
labelToOffset
LibTIM, 66
LibTIM, 59
LibTIM
computeNeighborhoodS1, 65
computeNeighborhoodS1v2, 65
computePriority, 66
dynamicSeNormL2, 66
dynamicSeNormL2NPoints, 66
dynamicSeNormL2Rand, 66
dynamicSeS1, 66
dynamicSeS1v2, 66
FLOAT_EPSILON, 66
functionR0, 66
GImageIO_NextLine, 66
GImageIO_ReadPPMHeader, 66
labelToOffset, 66
merge_pixels, 66
printSeAtPoint, 66
RGB, 65
S16, 65
S32, 65
S8, 65
TCoord, 65
TLabel, 65
tNode, 65
TOffset, 65
TSize, 65
TSpacing, 65
U16, 65
U32, 65
U8, 65
LibTIM::FlatSE, 69
LibTIM::FlatSE
addPoint, 71
begin, 71
begin_point, 71
clear, 71
end, 71
end_point, 71
FlatSE, 71
FlatSE::operator=, 71
FlatSE::toImage, 71
getNbPoints, 71
getNegativeOffsets, 71
getOffset, 72
getPoint, 72
getPositiveOffsets, 72
iterator, 71
iterator_offset, 71
iterator_point, 71
make2DN4, 72
make2DN8, 72
make2DN9, 72
make3DN18, 72
make3DN26, 72
make3DN6, 72
makeBallChessboard2D, 72
makeBallEuclidian2D, 72
makeBallEuclidian3D, 72
makeCircle2D, 72
makeSymmetric, 72
offsets, 73
operator+=, 73
points, 73
print, 73
reserve, 73
setContext, 73
LibTIM::Histogram, 74
LibTIM::Histogram
Histogram, 74
write, 74
LibTIM::Image, 75
LibTIM::Image
load, 78
save, 78
LibTIM::ImageIterator, 80
LibTIM::ImageIterator
im, 81
ImageIterator, 80
operator *, 80
operator!=, 80
operator++, 81
operator->, 81
operator==, 81
ptr, 81
LibTIM::ImageIteratorXYZ, 82
LibTIM::ImageIteratorXYZ

ImageIteratorXYZ, 82
operator *, 82
operator++, 82, 83
operator->, 83
operator=, 83
x, 83
y, 83
z, 83
LibTIM::ImageRegionsInfos, 84
LibTIM::ImageRegionsInfos
computeDistance, 84
distance, 84
fusion, 84
ImageRegionsInfos, 84
setPoint, 84
LibTIM::Node, 86
LibTIM::Node
active, 86
area, 86
childs, 86
father, 86
h, 86
label, 86
pixels, 86
LibTIM::NonFlatSE, 87
LibTIM::NonFlatSE
~NonFlatSE, 88
addPoint, 88
antiRasterScan, 88
clear, 88
getNorm, 88
getValue, 88
makeChamfer2D, 88
NonFlatSE, 88
print, 88
rasterScan, 88
reserve, 88
LibTIM::OrderedQueue, 89
LibTIM::OrderedQueue
~OrderedQueue, 89
empty, 89
get, 89
OrderedQueue, 89
put, 90
LibTIM::OrderedQueueDouble, 91
LibTIM::OrderedQueueDouble
~OrderedQueueDouble, 91
empty, 91
get, 91
OrderedQueueDouble, 91
put, 92
LibTIM::Point, 93
LibTIM::Point
operator(), 94
operator+=, 94
operator-=, 94
operator=, 94
operator==, 94
Point, 94
print, 94
x, 94
y, 94
z, 94
LibTIM::Queue, 95
LibTIM::Queue
empty, 95
get, 95
put, 95
Queue, 95
LibTIM::Region, 97
LibTIM::Region
nbPoints, 97
sumIntensity, 97
LibTIM::Table, 98
LibTIM::Table
el, 98
operator[], 98
Table, 98
Link
Tarjan, 35, 36
load
Image, 51
LibTIM::Image, 78

make2DN4
LibTIM::FlatSE, 72
make2DN8
LibTIM::FlatSE, 72
make2DN9
LibTIM::FlatSE, 72
make3DN18
LibTIM::FlatSE, 72
make3DN26
LibTIM::FlatSE, 72
make3DN6
LibTIM::FlatSE, 72
make_father
ccTree, 15
makeBallChessboard2D
LibTIM::FlatSE, 72
makeBallEuclidian2D
LibTIM::FlatSE, 72
makeBallEuclidian3D
LibTIM::FlatSE, 72
makeChamfer2D
LibTIM::NonFlatSE, 88
makeCircle2D
LibTIM::FlatSE, 72

MakeSet
    Tarjan, 36
makeSymmetric
    LibTIM::FlatSE, 72
merge_pixels
    LibTIM, 66
misc
    adjustContrast, 19
    centroids, 19
    computeMarkerMean, 19
    computeMarkerMeanFast, 19
    decimateTemplate, 20
    drawContour, 20
Misc Functions, 19
Morphological Operators, 21
morphologicalGradient
    basisFunctions, 24

nbPoints
    LibTIM::Region, 97
Non-flat structuring elements, 55
NonFlatSE
    LibTIM::NonFlatSE, 88

offsets
    LibTIM::FlatSE, 73
opening
    basisFunctions, 24
operator &=
    Image, 51
operator *
    Image, 51
    LibTIM::ImageIterator, 80
    LibTIM::ImageIteratorXYZ, 82
operator *=
    Image, 51
operator!
    Image, 51
operator!=
    LibTIM::ImageIterator, 80
operator()
    Image, 51, 52
    LibTIM::Point, 94
operator+
    Image, 52
    Point, 56
operator++
    LibTIM::ImageIterator, 81
    LibTIM::ImageIteratorXYZ, 82, 83
operator+=
    Image, 52
    LibTIM::FlatSE, 73
    LibTIM::Point, 94
operator-
    Image, 52, 53
    Point, 56
operator-=
    Image, 53
    LibTIM::Point, 94
operator->
    LibTIM::ImageIterator, 81
    LibTIM::ImageIteratorXYZ, 83
operator/=
    Image, 53
operator=
    Image, 53
    LibTIM::ImageIteratorXYZ, 83
    LibTIM::Point, 94
operator==
    Image, 53
    LibTIM::ImageIterator, 81
    LibTIM::Point, 94
operator[]
    LibTIM::Table, 98
operator|=
    Image, 53
OrderedQueue
    LibTIM::OrderedQueue, 89
OrderedQueueDouble
    LibTIM::OrderedQueueDouble, 91

pixels
    LibTIM::Node, 86
Point, 56
    LibTIM::Point, 94
    operator+, 56
    operator-, 56
points
    LibTIM::FlatSE, 73
print
    Image, 53
    LibTIM::FlatSE, 73
    LibTIM::NonFlatSE, 88
    LibTIM::Point, 94
printBestTemplate
    templateMatching, 37
printSeAtPoint
    LibTIM, 66
printTree
    ccTree, 15
ptr
    LibTIM::ImageIterator, 81
put
    LibTIM::OrderedQueue, 90
    LibTIM::OrderedQueueDouble, 92
    LibTIM::Queue, 95

Queue

LibTIM::Queue, 95

Random, 96
Exponential, 96
Gaussian, 96
Randomize, 96
Uniform, 96
Randomize
Random, 96
rankFilter
basisFunctions, 24
rasterScan
LibTIM::NonFlatSE, 88
rbegin
Image, 53
reconstructImage
ccTree, 15
reconstruction
geodesicReconstructionByDilation, 28
geodesicReconstructionByErosion, 28
Region Growing Algorithms, 33
Regional Extrema Extraction, 26
regionalMaxima
extremaExtraction, 26
regionalMinima
extremaExtraction, 26
regionGrowing
RegionGrowingCriterion, 33
seededRegionGrowing, 33
seededRegionGrowing0, 33
seededRegionGrowingExactAlgorithm, 33
RegionGrowingCriterion
regionGrowing, 33
rend
Image, 53
reserve
LibTIM::FlatSE, 73
LibTIM::NonFlatSE, 88
reverse_iterator
Image, 49
RGB
LibTIM, 65

S16
LibTIM, 65
S32
LibTIM, 65
S8
LibTIM, 65
save
Image, 53
LibTIM::Image, 78
seededRegionGrowing
regionGrowing, 33
seededRegionGrowing0
regionGrowing, 33
seededRegionGrowingExactAlgorithm
regionGrowing, 33
setContext
LibTIM::FlatSE, 73
setImageInfos
Image, 53
setPoint
LibTIM::ImageRegionsInfos, 84
setSize
Image, 54
setSpacingX
Image, 54
setSpacingY
Image, 54
setSpacingZ
Image, 54
std, 67
sumIntensity
LibTIM::Region, 97

Table
LibTIM::Table, 98
Tarjan
Find, 35
FindSimple, 35
labelConnectedComponentsTarjan, 35
labelConnectedComponentsTarjan2, 35
Link, 35, 36
MakeSet, 36
treeType, 35
Tarjan's Union Find Algorithms, 35
TCoord
LibTIM, 65
Template Matching Based Algorithms, 37
templateMatching
printBestTemplate, 37
templateMatchingCorrelation, 37
templateMatchingL2, 37
templateMatchingCorrelation
templateMatching, 37
templateMatchingL2
templateMatching, 37
threshold
thresholding, 39
thresholding
binarize, 39
threshold, 39
Thresholding Functions, 39
TLabel
LibTIM, 65
tNode
LibTIM, 65

TOffset
    LibTIM, 65
treeType
    Tarjan, 35
TSize
    LibTIM, 65
TSpacing
    LibTIM, 65

U16
    LibTIM, 65
U32
    LibTIM, 65
U8
    LibTIM, 65
Uniform
    Random, 96

viscousClosingMercury
    constrainedWatershed, 40
viscousClosingMercuryBasic
    constrainedWatershed, 40

watershed
    watershedMeyer, 41
Watershed-based Algorithms, 41
watershedMeyer
    watershed, 41
write
    LibTIM::Histogram, 74

x
    LibTIM::ImageIteratorXYZ, 83
    LibTIM::Point, 94

y
    LibTIM::ImageIteratorXYZ, 83
    LibTIM::Point, 94

z
    LibTIM::ImageIteratorXYZ, 83
    LibTIM::Point, 94