Scribo

User documentation 2.1

Generated by Doxygen 1.8.1.2

Tue Jul 8 2014 17:16:02

Contents

Chapter 1

Documentation of Scribo

1.1 Introduction

Scribo, a framework for Document Image Analysis.

Its initial release has been developed in the context of the SCRIBO project of the Free Software Thematic Group, part of the "Systematic Paris-Région" Cluster (France). This project was partially funded by the French government, its economic development agencies and by the Paris-Région institutions.

It aims at providing the following features:

- routines for DIA,
- DIA toolchains,
- text detection in document,
- document layout analysis,
- high-level data structures,
- novel algorithms and techniques,
- standard I/O,
- graphical user interfaces,
- and command Line Interfaces.

Its code is based on Milena, the generic and efficient c++ image processing library of Olena.

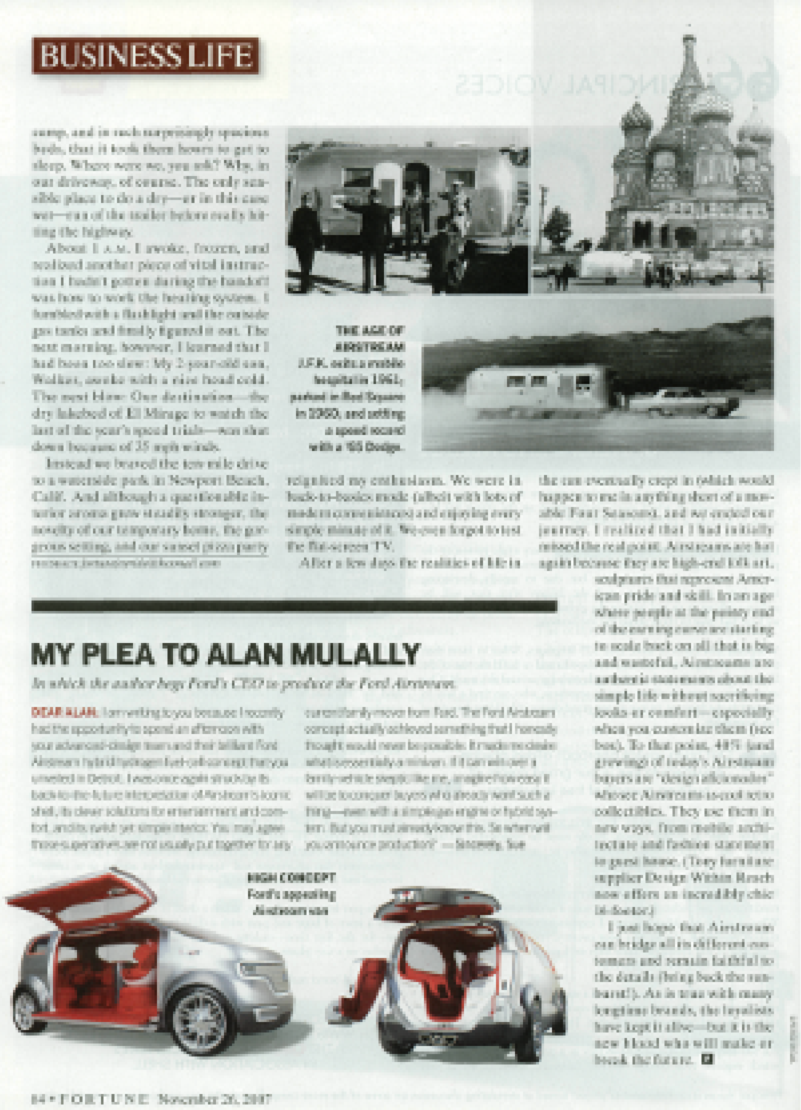
It is integrated in the `KDE` semantic desktop, `Nepomuk`, in the current `Mandriva Linux` distribution.

1.2 Quickstart

No tutorial nor quick reference guide are written for now. But a good start is to read the code of existing processing chain. You can take a look at the following files :

- text_in_doc_preprocess.hh
- content_in_doc_functor.hh
- text_in_picture.hh

Take also a look at the `Modules section`.



BUSINESS LIFE

camp, and in such surprisingly spacious beds, that it took them hours to get to sleep. Where were we, you ask? Why, in our driveway, of course. The only sensible place to do a dry—or in this case wet—run of the trailer before really hitting the highway.

About 1 A.M. I awoke, frozen, and realized another piece of vital instruction I hadn't gotten during the handoff was how to work the heating system. I fumbled with a flashlight and the outside gas tanks and finally figured it out. The next morning, however, I learned that I had been too slow: My 2-year-old son, Walker, awoke with a nice head cold. The next blow: Our destination—the dry lakebed of El Mirage to watch the last of the year's speed trials—was shut down because of 35 mph winds.

Instead we braved the ten-mile drive to a waterside park in Newport Beach, Calif. And although a questionable interior aroma grew steadily stronger, the novelty of our temporary home, the gorgeous setting, and our sunset pizza party reignited my enthusiasm. We were in back-to-basics mode (albeit with lots of modern conveniences) and enjoying every simple minute of it. We even forgot to test the flat-screen TV.

After a few days the realities of life in the can eventually crept in (which would happen to me in anything short of a movable Four Seasons), and we ended our journey. I realized that I had initially missed the real point. Airstreams are hot again because they are high-end folk art, sculptures that represent American pride and skill. In an age where people at the pointy end of the earning curve are starting to scale back on all that is big and wasteful, Airstreams are authentic statements about the simple life without sacrificing looks or comfort—especially when you customize them (see box). To that point, 40% (and growing) of today's Airstream buyers are "design aficionados" whose Airstreams-as-cool retro collectibles. They use them in new ways, from mobile architecture and fashion statement to guest house. (Tony furniture supplier Design Within Reach now offers an incredibly chic tin-dorator.)

I just hope that Airstream can bridge all its different customers and remain faithful to the details (bring back the sunburst!). As is true with many longtime brands, the loyalists have kept it alive—but it is the new blood who will make or break the future. ■

FEEDBACK [illegible]

THE AGE OF AIRSTREAM J.F.K. exits a mobile hospital in 1961; parked in Red Square in 1960, and setting a speed record with a '63 Dodge.

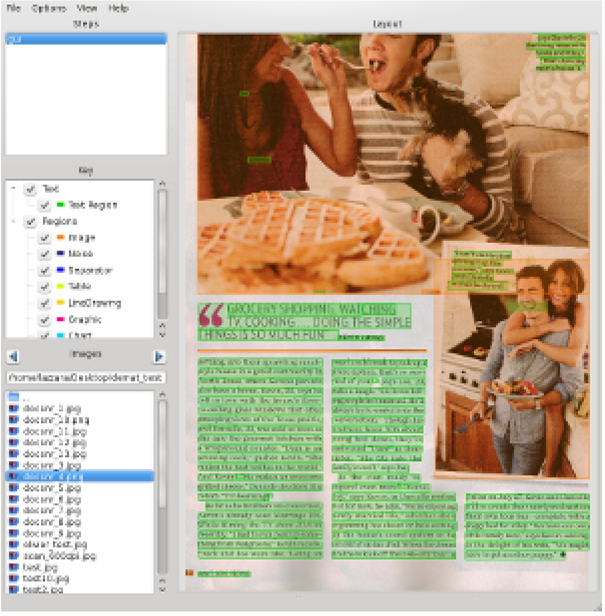
MY PLEA TO ALAN MULALLY

In which the author begs Ford's CEO to produce the Ford Airstream.

DEAR ALAN: I am writing to you because I recently had the opportunity to spend an afternoon with your advanced-design team and their brilliant Ford Airstream hybrid hydrogen fuel-cell concept that you unveiled in Detroit. I was once again struck by its back-to-the-future interpretation of Airstream's iconic shell, its clever solutions for entertainment and comfort, and its sweet yet simple interior. You may agree those superlatives are not usually put together for any current family mover from Ford. The Ford Airstream concept actually achieved something that I honestly thought would never be possible: It made me desire what is essentially a minivan. If it can win over a family-vehicle skeptic like me, imagine how easy it will be to convert buyers who already want such a thing—even with a simple gas engine or hybrid system. But you must already know this. So when will you announce production?" —Sincerely, Sue

HIGH CONCEPT Ford's appealing Airstream van

84 • FORTUNE November 26, 2007



BUSINESS LIFE

[illegible]

MY PLEA TO ALAN MULALLY

In which the author begs Ford's CEO to produce the Ford Airstream.

[illegible]

64 | FORTUNE November 26, 2007

1.3 Copyright and License.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 4

Class Index

4.1 Class Hierarchy

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

- cluster_stats< T > 139
- compare_values< T > 139
- mln::Function_v2b< components_large_filter< L > > [external]
  - scribo::fun::v2b::components_large_filter< L > 160
- mln::Function_v2b< components_on_border_filter< L > > [external]
  - scribo::fun::v2b::components_on_border_filter< L > 162
- mln::Function_v2b< components_small_filter< L > > [external]
  - scribo::fun::v2b::components_small_filter< L > 164
- mln::Function_v2b< label_to_bool< L > > [external]
  - scribo::fun::v2b::label_to_bool< L > 165
- mln::Function_v2v< highlight< R > > [external]
  - scribo::fun::v2v::highlight< R > 166
- mln::info 140
- mln::Object< E > [external]
  - scribo::DMax_Functor< E > 158
    - scribo::primitive::link::internal::dmax_functor_base< E > 190
  - scribo::Link_Functor< E > 183
    - scribo::primitive::link::internal::link_functor_base< L, E > 192
      - scribo::primitive::link::internal::link_several_dmax_base< L, E > 194
      - scribo::primitive::link::internal::link_single_dmax_base< L, E > 195
      - scribo::primitive::link::internal::link_single_dmax_ratio_base< L, F, E > 200
        - scribo::primitive::link::internal::link_single_dmax_ratio_aligned_base< L, F, E > 197
        - scribo::primitive::link::internal::link_single_dmax_ratio_aligned_delta_base< L, F, E > 199
  - scribo::Serializable< E > 202
  - scribo::SerializeVisitor< E > 203
    - scribo::doc_serializer< E > 158
- mln::Object< component_info< L > > [external]
  - scribo::Serializable< component_info< L > > 202
    - scribo::component_info< L > 149
- mln::Object< component_set< L > > [external]
  - scribo::Serializable< component_set< L > > 202
    - scribo::component_set< L > 150
- mln::Object< debug_img_visitor< L > > [external]
  - scribo::SerializeVisitor< debug_img_visitor< L > > 203
    - scribo::doc_serializer< debug_img_visitor< L > > 158
      - scribo::io::img::internal::debug_img_visitor< L > 172
- mln::Object< dmax_default > [external]

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

Module Documentation

6.1 Binarization

Modules

- Kim
- Niblack
- Otsu
- Sauvola
- Wolf

6.1.1 Detailed Description

Binarization algorithms.

6.2 Kim

Functions

- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::kim (const **Image**< I > &input, **unsigned** window_size, **double** k)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::kim (const **Image**< I > &input, **unsigned** window_size)

6.2.1 Detailed Description

Kim's Binarization implementations. This algorithms performs a first rough binarization on the input (here we use Sauvola's method). Then text lines are roughly detected by grouping connected components. For each text lines, character thickness and x height is computed. Finally, each lines is binarized again, using text features as parameters for adjusting the final threshold.

This algorithms considers that global and local statistics should be used to compute a threshold in text areas.

This is an improvement of Sauvola's method.

This implementation is based on the paper "Multi-Window Binarization of Camera Image for Document Recognition", In-Jung Kim, Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition (IWFHR-9 2004)

6.2.2 Function Documentation

6.2.2.1 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::kim (const Image< I > & *input,* unsigned *window_size,* double *k*)

Kim's binarization.

This algorithms performs a first rough binarization on the input (here we use Sauvola's method). Then text lines are roughly detected by grouping connected components. For each text lines, character thickness and x height is computed. Finally, each lines is binarized again, using text features as parameters for adjusting the final threshold.

This algorithms considers that global and local statistics should be used to compute a threshold in text areas.

This is an improvement of Sauvola's method.

This implementation is based on the paper "Multi-Window Binarization of Camera Image for Document Recognition", In-Jung Kim, Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition (IWFHR-9 2004)

Parameters

`in`	*input*	A gray-level image.
`in`	*window_size*	the window size to be used for the first binarization.
`in`	*k*	Sauvola's formula parameter.

6.2.2.2 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::kim (const Image< I > & *input,* unsigned *window_size*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts. k is set to SCRIBO_DEFAULT_SAUVOLA_K.

6.3 Niblack

Functions

- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::niblack (const **Image**< I > &input, **unsigned** window_size, **double** K)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::niblack (const **Image**< I > &input, **unsigned** window_size)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::niblack (const **Image**< I > &input)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::niblack_fast (const **Image**< I > &input, **unsigned** window_size, **double** K)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::niblack_fast (const **Image**< I > &input, **unsigned** window_size)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::niblack_fast (const **Image**< I > &input)
- template<typename I , typename J >
  mln::trait::ch_value< I,
  value::int_u8 >::**ret** scribo::binarization::niblack_threshold (const **Image**< I > &input, **unsigned** window_size, **double** K)
- template<typename I >
  mln::trait::ch_value< I,
  value::int_u8 >::**ret** scribo::binarization::niblack_threshold (const **Image**< I > &input, **unsigned** window_size)
- template<typename I >
  mln::trait::ch_value< I,
  value::int_u8 >::**ret** scribo::binarization::niblack_threshold (const **Image**< I > &input)

6.3.1 Detailed Description

Niblack's Binarization implementations.

6.3.2 Function Documentation

6.3.2.1 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::niblack (const Image< I > & *input*, unsigned *window_size*, double *K*)

Convert an image into a binary image.

Parameters

`in`	*input*	An image.
`in`	*window_size*	The window size.
`in`	*K*	Niblack's formulae constant.

Returns

A binary image.

6.3.2.2 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::niblack (const Image< I > & *input,* unsigned *window_size*)

Convert an image into a binary image.

Sauvola's formulae constant K is set to SCRIBO_DEFAULT_NIBLACK_K.

Parameters

in	*input*	An image.
in	*window_size*	The window size.

Returns

A binary image.

6.3.2.3 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::niblack (const Image< I > & *input*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.The window size is set to 11.

6.3.2.4 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::niblack_fast (const Image< I > & *input,* unsigned *window_size,* double *K*)

Convert an image into a binary image.

This implementation gives an approximation of the results. **It** is faster than the original implementation thanks to the use of integral images.

Parameters

in	*input*	An image.
in	*window_size*	The window size.
in	*K*	Sauvola's formulae constant.

Returns

A binary image.

6.3.2.5 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::niblack_fast (const Image< I > & *input,* unsigned *window_size*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts. Sauvola's formulae constant K is set to SCRIBO_DEFAULT_NIBLACK_K.

6.3.2.6 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::niblack_fast (const Image< I > & *input*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts. The window size is set to 11.

6.3.2.7 template< typename I , typename J > mln::trait::ch_value< I , value::int_u8 >::ret scribo::binarization::niblack_threshold (const Image< I > & *input*, unsigned *window_size*, double *K*)

Compute an image of local threshold using Niblack algorithm.

Parameters

in	*input*	A gray level image.
in	*window_size*	The window size.
in	*K*	Controls the threshold value in the local window.

Returns

An image of local thresholds.

6.3.2.8 template< typename I > mln::trait::ch_value< I , value::int_u8 >::ret scribo::binarization::niblack_threshold (const Image< I > & *input*, unsigned *window_size*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.K is set to 0.34.

6.3.2.9 template< typename I > mln::trait::ch_value< I , value::int_u8 >::ret scribo::binarization::niblack_threshold (const Image< I > & *input*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.The window size is set to 11.

6.4 Otsu

Functions

- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::otsu (const **Image**< I > &input)
- template<typename I >
  I::value scribo::binarization::otsu_threshold (const **Image**< I > &input)

6.4.1 Detailed Description

Otsu's Binarization implementations. This implementation is based on Ocropus's implementation. `http://code.google.com/p/ocropus/`

6.4.2 Function Documentation

6.4.2.1 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::otsu (const Image< I > & *input*)

An implementation of Otsu's binarization algorithm.

This implementation is based on Ocropus's implementation. `http://code.google.com/p/ocropus/`

Parameters

`in`	*input*	A gray-scale image.

Returns

A binary image. True for foreground, False for background.

6.4.2.2 template<typename I > I ::value scribo::binarization::otsu_threshold (const Image< I > & *input*)

Compute a global binarization threshold using Otsu's algorithm.

This implementation is based on Ocropus's implementation. `http://code.google.com/p/ocropus/`

Parameters

`in`	*input*	A gray-scale image.

Returns

A global threshold value.

6.5 Sauvola

Functions

- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::sauvola (const **Image**< I > &input, **unsigned** window_size, **double** K)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::sauvola (const **Image**< I > &input, **unsigned** window_size)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::sauvola (const **Image**< I > &input)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::sauvola_ms (const **Image**< I > &input_1, **unsigned** w_1, **unsigned** s, **image2d**< **mln::util::couple**< **double**, **double** > > &integral_sum_sum_2)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::sauvola_ms (const **Image**< I > &input_1, **unsigned** w_1, **unsigned** s)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::sauvola_ms (const **Image**< I > &input_1, **unsigned** w_1)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::sauvola_ms_split (const **Image**< I > &input_1, **unsigned** w_1, **unsigned** s, **unsigned** min_ntrue, **double** k2, **double** k3, **double** k4)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::sauvola_ms_split (const **Image**< I > &input_1, **unsigned** w_1, **unsigned** s, **unsigned** min_ntrue, **double** K)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::sauvola_ms_split (const **Image**< I > &input_1, **unsigned** w_1, **unsigned** s, **unsigned** min_ntrue)
- template<typename I , typename J >
  mln::trait::ch_value< I,
  value::int_u8 >::**ret** scribo::binarization::sauvola_threshold (const **Image**< I > &input, **unsigned** window_size, **double** K)
- template<typename I >
  mln::trait::ch_value< I,
  value::int_u8 >::**ret** scribo::binarization::sauvola_threshold (const **Image**< I > &input, **unsigned** window_size)
- template<typename I >
  mln::trait::ch_value< I,
  value::int_u8 >::**ret** scribo::binarization::sauvola_threshold (const **Image**< I > &input)

6.5.1 Detailed Description

Sauvola's Binarization implementations. **It** is well suited for classical document image binarization.

Scribo provides three variants of Sauvola's binarization algorithm :

- scribo::binarization::sauvola , the standard implementation.
- scribo::binarization::sauvola_ms , a multi-scale implementation.

- scribo::binarization::sauvola_ms_split , a multi-scale implementation running on each channel of a RGB image and merging the results into a single binary image.

These algorithms are based on the following article: [5] . All of them are implemented using integral images for best performance [6].

Sauvola's algorithm is a local thresholding method. Based on the mean and the standard deviation of a sliding centered window, it computes a local threshold for each pixel.

The following formula is used:

$$T(p) = m(p) + [1 + k * (\frac{s(p)}{R} - 1)]$$

Where *p* is the central window point, *m(p)* and *s(p)* are, respectively, the mean and the standard deviation of the window centered to point *p*.

k is user-defined parameter. According to [1], this parameter is set to *0.34*. However, it is possible to pass a different value and change the object sharpness.

The window size is also a user-defined parameter. In scribo::binarization::sauvola, it has to be adapted to the size of the binarized objects. In case of a document image with mixed objects of various size, we advice you to use the multi-scale version scribo::binarization::sauvola_ms. In this version, the window size parameter is adapted at each scale and both large and small objects are correctly retrieved at the same time.

scribo::binarization::sauvola_threshold variants return an image of thresholds. The latter can be used afterwards to binarize the input image.

See Also

src/binarization/sauvola.cc src/binarization/sauvola_ms.cc sauvola_ms_split.cc

6.5.2 comparison

Compare results between all Sauvola's implementations. The window size is set to *51* and *k* is set to *0.34*.

6.5.3 Function Documentation

6.5.3.1 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::sauvola (const Image< I > & *input,* unsigned *window_size,* double *K*)

Convert an image into a binary image.

Parameters

in	*input*	A greyscale image.
in	*window_size*	The window size.
in	*K*	Sauvola's formulae constant.

Returns

A binary image.

6.5.3.2 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::sauvola (const Image< I > & *input,* unsigned *window_size*)

Convert an image into a binary image.

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts. Sauvola's formulae constant K is set to 0.34.

Parameters

`in`	*input*	A grayscale image.
`in`	*window_size*	The window size.

Returns

A binary image.

6.5.3.3 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::sauvola (const Image< I > & *input*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts. The window size is set to 11.

6.5.3.4 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::sauvola_ms (const Image< I > & *input_1,* unsigned *w_1,* unsigned *s,* image2d< mln::util::couple< double, double > > & *integral_sum_sum_2*)

Binarize an image using a multi-scale implementation of Sauvola's algoritm.

Parameters

`in`	*input_1*	A grayscale image.
`in`	*w_1*	The window size used to compute stats.
`in`	*s*	The scale factor used for the first subscaling (usually 2 or 3 is enough).
`out`	*integral_sum_sum_2*	Integral image of sum and squared sum.

Sauvola's formula parameter K is set to 0.34. `w_1` and `lambda_min_1` are expressed according to the image at scale 0, i.e. the original size.

Returns

A Boolean image.

In this implementation, the input image is subscaled at three different scales. For each subscale image, Sauvola's is applied and relevant objects are preserved. The results of the three scales is merged and areas of the input image are marked with the appropriate scale to use for final binarization. Finally, the input image is binarized, using, for each pixel, the correct window size of the scale where it has been retrieved.

This implementation is very useful for document image with heterogeneous object sizes.

Its implementation is meant to be the fastest as possible. For instance, statistics computation is based on integral images [6].

6.5.3.5 template< typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::sauvola_ms (const Image< I > & *input_1*, unsigned *w_1*, unsigned *s*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.The integral image is not returned.

K is set to 0.34.

6.5.3.6 template< typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::sauvola_ms (const Image< I > & *input_1*, unsigned *w_1*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.The integral image is not returned.

K is set to 0.34. s is set to 3.

6.5.3.7 template< typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::sauvola_ms_split (const Image< I > & *input_1*, unsigned *w_1*, unsigned *s*, unsigned *min_ntrue*, double *k2*, double *k3*, double *k4*)

Binarize a color image merging the binarization of each component using Sauvola's algorithm.

Parameters

`in`	*input_1*	A color image.
`in`	*w_1*	The window size used to compute stats.
`in`	*s*	The scale factor used for the first subscaling.
`in`	*min_ntrue*	A site is set to 'True' in the output if it is set to 'True' at least `min_ntrue` components. Possible values: 1, 2, 3.
`in`	*k2*	Sauvola's formula parameter.
`in`	*k3*	Sauvola's formula parameter.
`in`	*k4*	Sauvola's formula parameter.

`w_1` is expressed according to the image at scale 0, i.e. the original size.

Returns

A Boolean image.

6.5.3.8 template< typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::sauvola_ms_split (const Image< I > & *input_1*, unsigned *w_1*, unsigned *s*, unsigned *min_ntrue*, double *K*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts. k2, k3 and k4 are set to `K`.

6.5.3.9 template< typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::sauvola_ms_split (const Image< I > & *input_1*, unsigned *w_1*, unsigned *s*, unsigned *min_ntrue*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts. k2, k3 and k4 are set to 0.34.

6.5.3.10 template< typename I , typename J > mln::trait::ch_value< I , value::int_u8 >::ret scribo::binarization::sauvola_threshold (const Image< I > & *input*, unsigned *window_size*, double *K*)

Compute an image of local threshold using Sauvola algorithm.

Parameters

in	*input*	A gray level image.
in	*window_size*	The window size.
in	*K*	Controls the threshold value in the local window.

Returns

An image of local thresholds.

6.5.3.11 template< typename I > mln::trait::ch_value< I , value::int_u8 >::ret scribo::binarization::sauvola_threshold (const Image< I > & *input*, unsigned *window_size*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.K is set to 0.34.

6.5.3.12 template< typename I > mln::trait::ch_value< I , value::int_u8 >::ret scribo::binarization::sauvola_threshold (const Image< I > & *input*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.The window size is set to 11.

6.6 Wolf

Functions

- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::wolf (const **Image**< I > &input, **unsigned** window_size, **double** K)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::wolf (const **Image**< I > &input, **unsigned** window_size)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::wolf (const **Image**< I > &input)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::wolf_fast (const **Image**< I > &input, **unsigned** window_size, **double** K)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::wolf_fast (const **Image**< I > &input, **unsigned** window_size)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::binarization::wolf_fast (const **Image**< I > &input)

6.6.1 Detailed Description

Wolf's Binarization implementations.

6.6.2 Function Documentation

6.6.2.1 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::wolf (const Image< I > & *input,* unsigned *window_size,* double *K*)

Convert an image into a binary image.

Parameters

in	*input*	An image.
in	*window_size*	The window size.
in	*K*	Wolf's formulae constant.

Returns

A binary image.

This implementation is based on article "Text Localization, Enhancement and Binarization in Multimedia Documents", Christian Wolf, Jean-Michel Jolion, Françoise Chassaing, ICPR 2002.

6.6.2.2 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::wolf (const Image< I > & *input,* unsigned *window_size*)

Convert an image into a binary image.

Wolf's formulae constant K is set to 0.34.

Parameters

in	*input*	An image.
in	*window_size*	The window size.

Returns

A binary image.

6.6.2.3 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::wolf (const Image< I > & *input*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts. The window size is set to 11.

6.6.2.4 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::wolf_fast (const Image< I > & *input,* unsigned *window_size,* double *K*)

Convert an image into a binary image.

Parameters

in	*input*	An image.
in	*window_size*	The window size.
in	*K*	Wolf's formulae constant.

Returns

A binary image.

This implementation is based on article "Text Localization, Enhancement and Binarization in Multimedia Documents", Christian Wolf, Jean-Michel Jolion, Françoise Chassaing, ICPR 2002.

This implementation gives an approximation of the results. **It** is faster than the original implementation thanks to the use of integral images.

6.6.2.5 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::wolf_fast (const Image< I > & *input,* unsigned *window_size*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts. Wolf's formulae constant K is set to 0.34.

6.6.2.6 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::wolf_fast (const Image< I > & *input*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts. The window size is set to 11.

6.7 Data structures

Classes

- class scribo::component_set< L >

  Represents all the components in a document image.
- class scribo::document< L >

  Represent document data and structure.
- class scribo::line_links< L >

  Line links representation.
- class scribo::line_set< L >

  Lines container.
- class scribo::object_groups< L >

  Object group representation.
- class scribo::object_links< L >

  Object links representation.
- class scribo::paragraph_set< L >

  Paragraph container.

6.7.1 Detailed Description

Scribo high level data structures. For the moment, Scribo provides bottom-up approaches to extract data from a document. From connected components, thanks to different algorithms, lines and paragraphs are reconstructed. **It** implies the following workflow (see figure below).

At each level, it is usually possible to "filter" the current results using routines in namespace scribo::filter. False positive can then be invalidated.

Data structures at a specific step, usually keep track of previous parent structures. For instance, object_links keeps a reference to the component_set it is based on.

These structures are iterable and usually store several attributes for their corresponding elements.

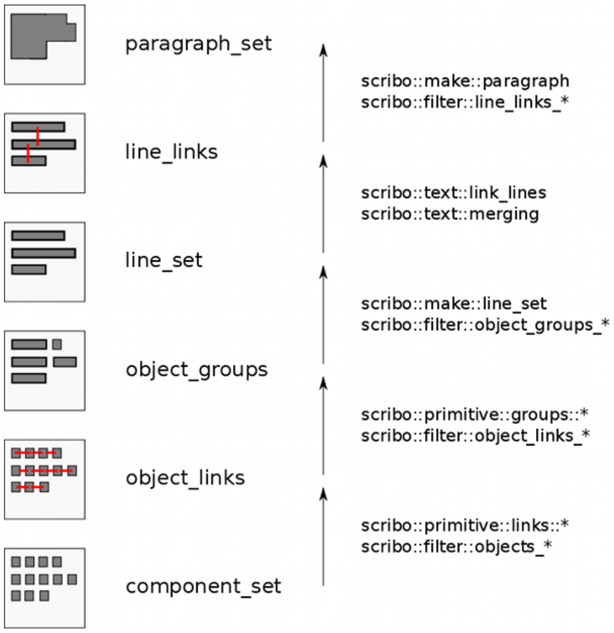

paragraph_set
line_links
line_set
object_groups
object_links
component_set
scribo::make::paragraph
scribo::filter::line_links_*
scribo::text::link_lines
scribo::text::merging
scribo::make::line_set
scribo::filter::object_groups_*
scribo::primitive::groups::*
scribo::filter::object_links_*
scribo::primitive::links::*
scribo::filter::objects_*

Figure 6.1: Data structure construction workflow and useful routines/namespaces for text extraction.

6.8 Debug

Functions

- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::alignment_decision_image (const **Image**< I > &input, const object_links< L > &links, const object_links< L > &filtered_links, const anchor::Type &anchor)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::bboxes_enlarged_image (const **Image**< I > &input, const line_set< L > &lines, const value::rgb8 &text_value, const value::rgb8 &non_text_value)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::bboxes_enlarged_image (const **Image**< I > &input, const line_set< L > &lines)
- template<typename I >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::bboxes_image (const **Image**< I > &input, const **mln::util::array**< **box**< typename I::site > > &bboxes, const value::rgb8 &**value**)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::bboxes_image (const **Image**< I > &input, const line_set< L > &lines, const value::rgb8 &**value**)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::bboxes_image (const **Image**< I > &input, const line_set< L > &lines)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::bboxes_image (const **Image**< I > &input, const component_set< L > &comps, const value::rgb8 &**value**)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::char_space_image (const **Image**< I > &input, const line_set< L > &**line**, const value::rgb8 &v=literal::cyan)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::decision_image (const **Image**< I > &input, const object_groups< L > &groups, const object_groups< L > &filtered_groups, anchor::Type anchor)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::decision_image (const **Image**< I > &input, const object_links< L > &links, const object_links< L > &filtered_links, anchor::Type anchor)
- template<typename I >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::highlight_text_area (const **Image**< I > &input, const **mln::util::array**< **box**< typename I::site > > &**bbox**)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::highlight_text_area (const **Image**< I > &input, const line_set< L > &lines)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::highlight_text_area (const **Image**< I > &input, const scribo::component_set< L > &components)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::line_info_image (const **Image**< I > &input, const line_set< L > &**line**)

- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::linked_bboxes_image (const **Image**< I > &input, const object_links< L > &array, const value::rgb8 &box_value, const value::rgb8 &link_value, anchor::Type anchor)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::linked_bboxes_image (const **Image**< I > &input, const object_links< L > &array, const value::rgb8 &box_value, const value::rgb8 &link_value)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::linked_bboxes_image (const **Image**< I > &input, const object_links< L > &left_link, const object_links< L > &right_link, const value::rgb8 &box_value, const value::rgb8 &link_value, anchor::Type anchor)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::linked_bboxes_image (const **Image**< I > &input, const object_links< L > &left_link, const object_links< L > &right_link, const value::rgb8 &box_value, const value::rgb8 &left_link_value, const value::rgb8 &right_link_value, const value::rgb8 &validated_link_value, anchor::Type anchor)
- template<typename I , typename L , typename G >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::linked_bboxes_image (const **Image**< I > &input, const **Graph**< G > &g, const value::rgb8 &box_value, const value::rgb8 &link_value, anchor::Type anchor)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::links_decision_image (const **Image**< I > &input_, const object_links< L > &links, const object_links< L > &filtered_links)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::links_image (const **Image**< I > &input_, const object_links< L > &links, anchor::Type anchor, **bool** draw_bboxes=true)
- scribo::debug::internal::logger_ & scribo::debug::logger ()
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::looks_like_a_text_line_image (const **Image**< I > &input, const line_set< L > &lines, const value::rgb8 &text_value, const value::rgb8 &non_text_value)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::looks_like_a_text_line_image (const **Image**< I > &input, const line_set< L > &lines)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::mean_and_base_lines_image (const **Image**< I > &input, const line_set< L > &lines, const value::rgb8 &bbox_value, const value::rgb8 &meanline_value, const value::rgb8 &baseline_value)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** scribo::debug::mean_and_base_lines_image (const **Image**< I > &input, const line_set< L > &lines)
- template<typename L , typename L2 >
  void scribo::debug::save_comp_diff (const component_set< L > &comps_ref, const component_set< L2 > &comps_new, const std::string &**filename**)
- template<typename I >
  void scribo::debug::save_label_image (const **Image**< I > &lbl, const typename **I::value** &nlabels, const char ∗**filename**)
- template<typename I >
  void scribo::debug::save_table_image (const **Image**< I > &input_, **mln::util::couple**< **mln::util::array**< **box**< typename I::site > >, **mln::util::array**< **box**< typename I::site > > > tableboxes, const value::rgb8 &bbox_color, const std::string &**filename**)

- template<typename S >
  void scribo::debug::save_table_image (const **Site_Set**< S > &input_domain, **mln::util::couple**< **mln::util::array**< **box**< typename S::site > >, **mln::util::array**< **box**< typename S::site > > > tableboxes, const value::rgb8 &bg_color, const value::rgb8 &bbox_color, const std::string &**filename**)
- template<typename I , typename L >
  mln::trait::concrete< I >::**ret** scribo::debug::text_areas_image (const **Image**< I > &input_rgb, const scribo::component_set< L > &comps)
- template<typename L >
  image2d< value::rgb8 > scribo::debug::text_color_image (const document< L > &doc)

6.8.1 Detailed Description

Debug routines.

6.8.2 Function Documentation

6.8.2.1 template<typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::alignment_decision_image (const Image< I > & *input,* const object_links< L > & *links,* const object_links< L > & *filtered_links,* const anchor::Type & *anchor*)

Save a color image showing the difference between to object links.

Parameters

in	*input*	An image. **It**'s value type must be convertible towards rgb8.
in	*links*	Object links information.
in	*filtered_links*	A copy of `links` which have been filtered.
in	*anchor*	Anchor from where the links are drawn.

Returns

A color image. Non filtered links are drawn in green. Others are drawn in red.

6.8.2.2 template<typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::bboxes_enlarged_image (const Image< I > & *input,* const line_set< L > & *lines,* const value::rgb8 & *text_value,* const value::rgb8 & *non_text_value*)

Compute an image of enlarged component bounding boxes.

This check whether each line looks like a text line. If it is a text line, its extended bounding box is drawn, otherwise, it is normal bounding box.

This rountine uses scribo::internal::looks_like_a_text_line to check if a component looks like a text line.

Parameters

in	*input*	An image convertible towards a color image.
in	*lines*	A line set.
in	*text_value*	The color used to draw bounding boxes of components looking like a text line.
in	*non_text_value*	The color used to draw bounding boxes of components NOT looking like a text line.

Returns

A color image.

6.8.2.3 template<typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::bboxes_enlarged_image (const Image< I > & *input*, const line_set< L > & *lines*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.text_value is set to literal::green.

non_text_value is set to literal::red.

6.8.2.4 template<typename I > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::bboxes_image (const Image< I > & *input*, const mln::util::array< box< typename I::site > > & *bboxes*, const value::rgb8 & *value*)

Draw a list of bounding boxes.

6.8.2.5 template<typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::bboxes_image (const Image< I > & *input*, const line_set< L > & *lines*, const value::rgb8 & *value*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.

6.8.2.6 template<typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::bboxes_image (const Image< I > & *input*, const line_set< L > & *lines*) `[inline]`

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.value is set to literal::red.

6.8.2.7 template<typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::bboxes_image (const Image< I > & *input*, const component_set< L > & *comps*, const value::rgb8 & *value*) `[inline]`

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.

6.8.2.8 template<typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::char_space_image (const Image< I > & *input*, const line_set< L > & *line*, const value::rgb8 & *v* = `literal::cyan`)

Draw inter character space.

6.8.2.9 template<typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::decision_image (const Image< I > & *input*, const object_groups< L > & *groups*, const object_groups< L > & *filtered_groups*, anchor::Type *anchor*)

Save a color image showing the difference between to object groups.

Parameters

`in`	*input*	An image. **It**'s value type must be convertible towards rgb8.
`in`	*groups*	Object groups information.
`in`	*filtered_groups*	A copy of `groups` which have been filtered.
`in`	*anchor*	Anchor from where the links are drawn.

Returns

A color image. Components part of a validated group are drawn in green with their bounding box. Otherwise, they are drawn in red.

6.8.2.10 template< typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::decision_image (const Image< I > & *input,* const object_links< L > & *links,* const object_links< L > & *filtered_links,* anchor::Type *anchor*)

Save a color image showing the difference between to object links.

Parameters

in	*input*	An image. **It**'s value type must be convertible towards rgb8.
in	*links*	Object links information.
in	*filtered_links*	A copy of `links` which have been filtered.
in	*anchor*	Anchor from where the links are drawn.

Returns

A color image. Non filtered links are drawn in green. Others are drawn in red.

6.8.2.11 template< typename I > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::highlight_text_area (const Image< I > & *input,* const mln::util::array< box< typename I::site > > & *bbox*)

Darken an image and highlight valid lines.

Parameters

in	*input*	An image.
in	*bbox*	An array of text area bounding boxes.

Returns

a color image with highlighted text areas.

6.8.2.12 template< typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::highlight_text_area (const Image< I > & *input,* const line_set< L > & *lines*)

Darken an image and highlight valid lines.

Parameters

in	*input*	An image.
in	*lines*	A line set.

Returns

a color image with highlighted text areas.

6.8.2.13 template< typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::highlight_text_area (const Image< I > & *input,* const scribo::component_set< L > & *components*)

Darken an image and highlight valid lines.

This overload is useful if you do know that all the components are text.

Parameters

`in`	*input*	An image.
`in`	*components*	A component set.

Returns

a color image with highlighted text areas.

6.8.2.14 template<typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::line_info_image (const Image< I > & *input*, const line_set< L > & *line*)

Draw typographic information from lines.

6.8.2.15 template<typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::linked_bboxes_image (const Image< I > & *input*, const object_links< L > & *array*, const value::rgb8 & *box_value*, const value::rgb8 & *link_value*, anchor::Type *anchor*)

Compute the line of components links image.

Parameters

`in,out`	*input*	The binary from where the components are extracted.
`in`	*array*	Components links.
`in`	*box_value*	Value used to draw line bounding boxes.
`in`	*link_value*	Value used to draw line links.
`in`	*anchor*	Anchor from where the links are drawn.

6.8.2.16 template<typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::linked_bboxes_image (const Image< I > & *input*, const object_links< L > & *array*, const value::rgb8 & *box_value*, const value::rgb8 & *link_value*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.The default anchor type is set to anchor::Center.

6.8.2.17 template<typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::linked_bboxes_image (const Image< I > & *input*, const object_links< L > & *left_link*, const object_links< L > & *right_link*, const value::rgb8 & *box_value*, const value::rgb8 & *link_value*, anchor::Type *anchor*)

Compute the line of components left and right links image.

Parameters

`in,out`	*input*	The binary from where the components are extracted.
`in`	*left_link*	Components left links.
`in`	*right_link*	Components right links.
`in`	*box_value*	Value used to draw line bounding boxes.
`in`	*link_value*	Value used to draw line links.
`in`	*anchor*	Anchor from where the links are drawn.

6.8.2.18 template<typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::linked_bboxes_image (const Image< I > & *input,* const object_links< L > & *left_link,* const object_links< L > & *right_link,* const value::rgb8 & *box_value,* const value::rgb8 & *left_link_value,* const value::rgb8 & *right_link_value,* const value::rgb8 & *validated_link_value,* anchor::Type *anchor*) [inline]

Compute the line of components left and right links image.

Draw also validated links.

Parameters

in,out	*input*	The binary from where the components are extracted.
in	*left_link*	Components left links.
in	*right_link*	Components right links.
in	*box_value*	Value used to draw line bounding boxes.
in	*left_link_value*	Value used to draw line left links.
in	*right_link_value*	Value used to draw line left links.
in	*validated_link_value*	Value used to draw line validated links.
in	*anchor*	Anchor from where the links are drawn.

6.8.2.19 template<typename I , typename L , typename G > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::linked_bboxes_image (const Image< I > & *input,* const Graph< G > & *g,* const value::rgb8 & *box_value,* const value::rgb8 & *link_value,* anchor::Type *anchor*)

Compute the line link graph image.

Parameters

in,out	*input*	The binary from where the components are extracted.
in	*g*	The link graph.
in	*box_value*	Value used to draw line bounding boxes.
in	*link_value*	Value used to draw line links.
in	*anchor*	Anchor from where the links are drawn.

6.8.2.20 template<typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::links_decision_image (const Image< I > & *input_,* const object_links< L > & *links,* const object_links< L > & *filtered_links*)

Save a color image showing the difference between to object groups.

6.8.2.21 template<typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::links_image (const Image< I > & *input_,* const object_links< L > & *links,* anchor::Type *anchor,* bool *draw_bboxes =* true)

Save a color image showing components links.

6.8.2.22 scribo::debug::internal::logger_& scribo::debug::logger ( )

Return a reference to the logger.

6.8.2.23 template<typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::looks_like_a_text_line_image (const Image< I > & *input*, const line_set< L > & *lines*, const value::rgb8 & *text_value*, const value::rgb8 & *non_text_value*)

Compute an image where components are drawn differently whether they look like a line or not.

Parameters

`in`	*input*	An image convertible towards a color image.
`in`	*lines*	A line set.
`in`	*text_value*	The color used to draw bounding boxes of components looking like a text line.
`in`	*non_text_value*	The color used to draw bounding boxes of components NOT looking like a text line.

Returns

A color image.

6.8.2.24 template<typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::looks_like_a_text_line_image (const Image< I > & *input*, const line_set< L > & *lines*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.text_value is set to literal::green.

non_text_value is set to literal::red.

6.8.2.25 template<typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::mean_and_base_lines_image (const Image< I > & *input*, const line_set< L > & *lines*, const value::rgb8 & *bbox_value*, const value::rgb8 & *meanline_value*, const value::rgb8 & *baseline_value*)

Compute a color image showing the mean and the base lines of the text lines.

The mean line is drawn with a dashed line. The base line is drawn with a plain line.

Parameters

`in`	*input*	An image convertible towards a color image.
`in`	*lines*	A line set.
`in`	*bbox_value*	Value used to draw lines bounding boxes.
`in`	*meanline_value*	Value used to draw mean lines.
`in`	*baseline_value*	Value used to draw base lines.

Returns

A color image.

6.8.2.26 template<typename I , typename L > mln::trait::ch_value< I , value::rgb8 >::ret scribo::debug::mean_and_base_lines_image (const Image< I > & *input*, const line_set< L > & *lines*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.text_value is set to literal::green.

non_text_value is set to literal::red.

6.8.2.27 template< typename L , typename L2 > void scribo::debug::save_comp_diff (const component_set< L > & *comps_ref,* const component_set< L2 > & *comps_new,* const std::string & *filename*)

Show the difference between two object images.

Parameters

in	*comps_ref*	A component set.
in	*comps_new*	Another component set.
in	*filename*	The output filename.

6.8.2.28 template< typename I > void scribo::debug::save_label_image (const Image< I > & *lbl,* const typename I::value & *nlabels,* const char * *filename*)

Save a labeled image in a color image.

Parameters

in	*lbl*	A label image.
in	*nlabels*	The number of labels.
in	*filename*	The output file name.

6.8.2.29 template< typename I > void scribo::debug::save_table_image (const Image< I > & *input_,* mln::util::couple< mln::util::array< box< typename I::site > >, mln::util::array< box< typename I::site > > > *tableboxes,* const value::rgb8 & *bbox_color,* const std::string & *filename*)

Save lines bounding boxes in a copy of `input_`.

Bounding boxes are displayed with `bbox_color`.

6.8.2.30 template< typename S > void scribo::debug::save_table_image (const Site_Set< S > & *input_domain,* mln::util::couple< mln::util::array< box< typename S::site > >, mln::util::array< box< typename S::site > > > *tableboxes,* const value::rgb8 & *bg_color,* const value::rgb8 & *bbox_color,* const std::string & *filename*)

Save lines bounding boxes in an image defined on `input_domain` filled with `bg_color`.

Bounding boxes are displayed with `bbox_color`.

6.8.2.31 template< typename I , typename L > mln::trait::concrete< I >::ret scribo::debug::text_areas_image (const Image< I > & *input_rgb,* const scribo::component_set< L > & *comps*)

Compute an image including detected text areas only.

6.8.2.32 template< typename L > image2d<value::rgb8> scribo::debug::text_color_image (const document< L > & *doc*)

Draw text components with their respective colors.

6.9 Component Group Filtering

Functions

- template<typename L >
  object_groups< L > scribo::filter::object_groups_mean_width (const object_groups< L > &groups, **float** width)
- template<typename L >
  object_groups< L > scribo::filter::object_groups_size_ratio (const object_groups< L > &groups, **float** max_size_ratio, **float** max_invalid_ratio_per_group)
- template<typename L >
  object_groups< L > scribo::filter::object_groups_small (const object_groups< L > &groups, **unsigned** n_links)
- template<typename L >
  object_groups< L > scribo::filter::object_groups_with_holes (const object_groups< L > &components, **unsigned** min_size)

6.9.1 Detailed Description

Component groups filtering routines.

6.9.2 Function Documentation

6.9.2.1 template<typename L > object_groups<L> scribo::filter::object_groups_mean_width (const object_groups< L > & *groups,* float *width*)

Filter groups having their object mean width too low.

Parameters

in	*groups*	Object group information.
in	*width*	Object group mean width must be greater or equal to this value.

Returns

Filtered object group information.

6.9.2.2 template<typename L > object_groups<L> scribo::filter::object_groups_size_ratio (const object_groups< L > & *groups,* float *max_size_ratio,* float *max_invalid_ratio_per_group*)

Invalidates groups with too much thin and high components.

For each components in the group, it computes the height/width ratio. If it is higher or equal to `max_size_ratio`, the component is counted as invalid.

If there are `max_invalid_ratio_per_group` invalid components, the group is invalidated.

Returns

An object_group structure potentially with invalided groups.

6.9.2.3 template<typename L > object_groups<L> scribo::filter::object_groups_small (const object_groups< L > & *groups,* unsigned *n_links*)

Invalidate groups with few components.

Parameters

in	*groups*	Information about object groups.
in	*n_links*	The minimum number of links per group.

Returns

A copy of object group in which small groups have been invalidated.

6.9.2.4 template<typename L > object_groups<L> scribo::filter::object_groups_with_holes (const object_groups< L > & *components,* unsigned *min_size*) [inline]

Remove groups not having at least two background components of `min_size` pixels.

In order to verify the property of "having two background components", a component group must have at least one component with a hole.

Example: the letter 'o' has two background components: outside and inside the letter.

6.10 Component Link Filtering

Functions

- template<typename L >
  object_links< L > scribo::filter::object_links_aligned (const object_links< L > &links, **float** max_alpha, anchor::Type anchor)
- template<typename L >
  object_links< L > scribo::filter::object_links_bbox_h_ratio (const object_links< L > &links, **float** max_h_ratio)
- template<typename L >
  object_links< L > scribo::filter::object_links_bbox_overlap (const object_links< L > &links, **float** max_overlap_ratio)
- template<typename L >
  object_links< L > scribo::filter::object_links_bbox_ratio (const object_links< L > &links, **unsigned** dim, **float** max_ratio)
- template<typename L >
  object_links< L > scribo::filter::object_links_bbox_w_ratio (const object_links< L > &links, **float** max_w_ratio)
- template<typename L >
  object_links< L > scribo::filter::object_links_bottom_aligned (const object_links< L > &links, **float** max_alpha)
- template<typename L >
  object_links< L > scribo::filter::object_links_center_aligned (const object_links< L > &links, **float** max_alpha)
- template<typename L >
  object_links< L > scribo::filter::object_links_left_aligned (const object_links< L > &links, **float** max_alpha)
- template<typename L >
  object_links< L > scribo::filter::object_links_non_aligned_simple (const object_links< L > &links, anchor::Type anchor, **float** max_alpha)
- template<typename L >
  object_links< L > scribo::filter::object_links_right_aligned (const object_links< L > &links, **float** max_alpha)
- template<typename L >
  object_links< L > scribo::filter::object_links_top_aligned (const object_links< L > &links, **float** max_alpha)

6.10.1 Detailed Description

Component links filtering routines.

6.10.2 Function Documentation

6.10.2.1 template<typename L > object_links<L> scribo::filter::object_links_aligned (const object_links< L > & *links*, float *max_alpha*, anchor::Type *anchor*)

Invalidate links between two components according to a specific anchor.

Parameters

in	*links*	Object links information.
in	*max_alpha*	Maximum angle value (degrees).
in	*anchor*	Anchor used to compute angles.

```
                      ------
                      |    |
         ------       |    |
         |    |       |    |
         | x------------x  |
         |    |       |    |
         ------~ ~ ~ ~| ~ ~| ~ ~
        object1  ~    |    |   ^
                   ~  |    |   |
              ~------   |
                     object2   | Alpha
                       ~       |
                  ~     |
                    ~   |
                      ~ v
                        ~
```

Example with an anchor set to the bottom right : The angle between the two bottoms must be lower than max_alpha.

6.10.2.2 template<typename L > object_links<L> scribo::filter::object_links_bbox_h_ratio (const object_links< L > & *links,* float *max_h_ratio*)

Invalidate links between two components with too different height.

Parameters

in	*links*	Link objects information.
in	*max_h_ratio*	The maximum height ratio of two linked bounding boxes.

Returns

A filtered object link information.

6.10.2.3 template<typename L > object_links<L> scribo::filter::object_links_bbox_overlap (const object_links< L > & *links,* float *max_overlap_ratio*)

Invalidate links between two components having their bounding box overlapping too much.

Parameters

in	*links*	Link components information.
in	*max_overlap_ratio*	The maximum ratio of the overlapping areas.

Returns

A filtered object link information.

6.10.2.4 template<typename L > object_links<L> scribo::filter::object_links_bbox_ratio (const object_links< L > & *links,* unsigned *dim,* float *max_ratio*)

Invalidate links between two components with too different height or width.

Parameters

in	*links*	Link components information.
in	*dim*	The dimension to use to compare bbox length.
in	*max_ratio*	The maximum length ratio of two linked bounding boxes.

Returns

A filtered object link information.

6.10.2.5 template<typename L > object_links<L> scribo::filter::object_links_bbox_w_ratio (const object_links< L > & *links,* float *max_w_ratio*)

Invalidate links between two objects with too different width.

Parameters

in	*links*	Link objects information.
in	*max_w_ratio*	The minimum width ratio of two linked bounding boxes.

Returns

A filtered object link information.

6.10.2.6 template<typename L > object_links<L> scribo::filter::object_links_bottom_aligned (const object_links< L > & *links,* float *max_alpha*)

Invalidate links between two components if their bottom are not aligned.

Parameters

in	*links*	Object links information.
in	*max_alpha*	Maximum angle value (degrees).

```
              ------
              |    |
 ------       |    |
 |    |       |    |
 | x------------x  |
 |    |       |    |
 ------~ ~ ~ ~| ~ ~| ~ ~
object1  ~    |    |   ^
           ~  |    |   |
      ~------   |
             object2   | Alpha
               ~       |
          ~     |
            ~   |
              ~ v
                ~
```

The angle between the two bottoms must be lower than `max_alpha`.

6.10.2.7 template<typename L > object_links<L> scribo::filter::object_links_center_aligned (const object_links< L > & *links,* float *max_alpha*)

Invalidate links between two objects if their center are not aligned.

Parameters

in	*links*	Object links information.
in	*max_alpha*	Maximum angle value (degrees).

Returns

New link data.

```
------       ------
|    |       |    |  v
| x ~| ~ ~ ~ | ~ ~|~ ~
|   ~| ~     |    |  | => Alpha
|    |   ~ ~ | x ~|~ ~
------       |    |  ^
object1      |    |
      ------
            object2
```

The angle between the two bottoms must be lower than `max_alpha`.

6.10.2.8 template<typename L > object_links<L> scribo::filter::object_links_left_aligned (const object_links< L > & *links,* float *max_alpha*)

Invalidate links between two objects if their left are not aligned.

Parameters

in	*links*	Object links information.
in	*max_alpha*	Maximum angle value (degrees).

```
                   ~ ^
                ~    |
             ~------ | Alpha
         ~    |    | |
      ~       |    | |
   ~          |    | v
 ~----- ~ ~ ~ | ~ ~| ~
 |    |       |    |
 | x------------x  |
 |    |       |    |
 ------       |    |
object1       |    |
       ------
             object2
```

The angle between the two lefts must be lower than `max_alpha`.

6.10.2.9 template<typename L > object_links<L> scribo::filter::object_links_non_aligned_simple (const object_links< L > & *links,* anchor::Type *anchor,* float *max_alpha*)

Invalidate links between two non aligned components. Alignment is based on a given anchor of object bounding boxes.

Parameters

in	*links*	Object links information.
in	*anchor*	Anchor from where the links are made.
in	*max_alpha*	Maximum angle value (degrees).

Exemple with anchor == 1 (bottom horizontal filter):

```
             ------
             |    |
------       |    |
|    |       |    |
| x------------x  |
|    |       |    |
 ------~ ~ ~ ~| ~ ~| ~ ~
object1  ~    |    |   ^
           ~  |    |   |
      ~------   |
             object2   | Alpha
               ~       |
          ~     |
            ~   |
              ~ v
                ~
```

The angle between the two bottoms must be lower than `alpha`.

6.10.2.10 template<typename L > object_links<L> scribo::filter::object_links_right_aligned (const object_links< L > & *links*, float *max_alpha*)

Invalidate links between two objects if their right are not aligned.

Parameters

`in`	*links*	Object links information.
`in`	*max_alpha*	Maximum angle value (degrees).

```
                     ~
              -----~ ^
              |  ~ | | Alpha
             ~|    | |
          ~   |    | v
 ------ ~ ~ ~ | ~ ~| ~ ~
 |    |       |    |
 | x------------x  |
 |    |       |    |
 ------       |    |
object1       |    |
       ------
             object2
```

The angle between the two rights must be lower than `max_alpha`.

6.10.2.11 template<typename L > object_links<L> scribo::filter::object_links_top_aligned (const object_links< L > & *links*, float *max_alpha*)

Invalidate links between two objects if their top are not aligned.

Parameters

`in`	*links*	Object links information.
`in`	*max_alpha*	Maximum angle value (degrees).

```
                     ~
                   ~ ^
                 ~   |
               ~     |
             ~------ | Alpha
           ~  |    | |
         ~    |    | |
       ~      |    | v
 ------ ~ ~ ~ | ~ ~| ~
 |    |       |    |
 | x------------x  |
 |    |       |    |
 ------       |    |
object1       |    |
       ------
```

```
                      object2
```

The angle between the two tops must be lower than `max_alpha`.

6.11 Component Filtering

Functions

- template<typename I , typename N , typename V >
  mln::trait::concrete< I >::**ret** scribo::filter::components_large (const **Image**< I > &input_, const **Neighborhood**< N > &nbh_, const V &label_type, **unsigned** max_size)
- template<typename L >
  component_set< L > scribo::filter::components_large (const component_set< L > &components, **unsigned** max_size)
- template<typename L >
  component_set< L > scribo::filter::components_on_border (const component_set< L > &components)
- template<typename I , typename N , typename V >
  mln::trait::concrete< I >::**ret** scribo::filter::components_small (const **Image**< I > &input, const **Neighborhood**< N > &nbh, V &nlabels, **unsigned** min_size)
- template<typename L >
  component_set< L > scribo::filter::components_small (const component_set< L > &components, **unsigned** min_size)
- template<typename I , typename N , typename V >
  mln::trait::concrete< I >::**ret** scribo::filter::components_thin (const **Image**< I > &input_, const **Neighborhood**< N > &nbh_, const V &label_type, **unsigned** min_thickness)
- template<typename L >
  component_set< L > scribo::filter::components_thin (const component_set< L > &components, **unsigned** min_thickness)
- template<typename L >
  component_set< L > scribo::filter::components_v_thin (const component_set< L > &comps, **unsigned** min_thinness)
- template<typename L >
  component_set< L > scribo::filter::components_with_two_holes (const component_set< L > &components, **unsigned** min_size)
- template<typename I , typename N , typename V >
  mln::trait::concrete< I >::**ret** scribo::filter::objects_h_thick (const **Image**< I > &input_, const **Neighborhood**< N > &nbh_, const V &label_type, **unsigned** max_thickness)
- template<typename L >
  component_set< L > scribo::filter::objects_h_thick (const component_set< L > &comps, **unsigned** max_thickness)
- template<typename I , typename N , typename V >
  mln::trait::concrete< I >::**ret** scribo::filter::objects_h_thin (const **Image**< I > &input_, const **Neighborhood**< N > &nbh_, const V &label_type, **unsigned** min_thinness)
- template<typename L >
  component_set< L > scribo::filter::objects_h_thin (const component_set< L > &comps, **unsigned** min_thinness)
- template<typename L >
  component_set< L > scribo::filter::objects_size_ratio (const component_set< L > &comps, **float** min_size_ratio)
- template<typename I , typename N , typename V >
  mln::trait::concrete< I >::**ret** scribo::filter::objects_thick (const **Image**< I > &input_, const **Neighborhood**< N > &nbh_, const V &label_type, **unsigned** max_thickness)
- template<typename L >
  component_set< L > scribo::filter::objects_thick (const component_set< L > &components, **unsigned** max_thickness)
- template<typename I , typename N , typename V >
  mln::trait::concrete< I >::**ret** scribo::filter::objects_v_thick (const **Image**< I > &input_, const **Neighborhood**< N > &nbh_, const V &label_type, **unsigned** max_thickness)
- template<typename L >
  component_set< L > scribo::filter::objects_v_thick (const component_set< L > &comps, **unsigned** max_thickness)

- template<typename I , typename N , typename V >
  mln::trait::concrete< I >::**ret** scribo::filter::objects_v_thin (const **Image**< I > &input, const **Neighborhood**< N > &nbh, const V &label_type, **unsigned** min_thinness)
- template<typename L >
  component_set< L > scribo::filter::objects_with_holes (const component_set< L > &components, **unsigned** min_holes_count, **unsigned** min_size)

6.11.1 Detailed Description

Component filtering routines.

6.11.2 Function Documentation

6.11.2.1 template<typename I , typename N , typename V > mln::trait::concrete< I >::ret scribo::filter::components_large (const Image< I > & *input_*, const Neighborhood< N > & *nbh_*, const V & *label_type*, unsigned *max_size*)

Remove large objects in a binary image.

Set to 'false' all the removed objects.

Parameters

in	*input_*	A binary image.
in	*nbh_*	A neighborhood used for labeling `input_`.
in	*label_type*	The label type used for labeling.
in	*max_size*	The minimum cardinality of an object.

Returns

A binary image without large objects.

6.11.2.2 template<typename L > component_set<L> scribo::filter::components_large (const component_set< L > & *components*, unsigned *max_size*) [inline]

Remove too large components.

Parameters

in	*components*	An object image.
in	*max_size*	The maximum cardinality of an object.

Returns

A component set with large components set to component::Ignored.

6.11.2.3 template<typename L > component_set<L> scribo::filter::components_on_border (const component_set< L > & *components*) [inline]

Remove components located on image borders.

Parameters

in	*components*	An object image.

Returns

A component set with large components set to component::Ignored.

6.11.2.4 template<typename I , typename N , typename V > mln::trait::concrete< I >::ret scribo::filter::components_small (const Image< I > & *input,* const Neighborhood< N > & *nbh,* V & *nlabels,* unsigned *min_size*)

Remove small components in a binary image.

Set to 'false' all the removed components.

Parameters

in	*input*	A binary image.
in	*nbh*	A neighborhood used for labeling `input_`.
out	*nlabels*	Return the number of components. Defines also the type used for labeling.
in	*min_size*	The minimum cardinality of an object.

Returns

A binary image without small components.

6.11.2.5 template<typename L > component_set<L> scribo::filter::components_small (const component_set< L > & *components,* unsigned *min_size*)

Remove too small components.

Parameters

in	*components*	An object image.
in	*min_size*	The minimum cardinality of an object.

Returns

A component set with small components set to component::Ignored.

6.11.2.6 template<typename I , typename N , typename V > mln::trait::concrete< I >::ret scribo::filter::components_thin (const Image< I > & *input_,* const Neighborhood< N > & *nbh_,* const V & *label_type,* unsigned *min_thickness*) [inline]

Remove components thinner or equal to `min_thickness`.

Parameters

in	*input_*	a binary image.
in	*nbh_*	a neighborhood used in labeling algorithms.
in	*label_type*	the label type used for labeling.
in	*min_thickness*	the minimum thickness value.

Returns

A binary image without thin components.

6.11.2.7 template<typename L > component_set<L> scribo::filter::components_thin (const component_set< L > & *components,* unsigned *min_thickness*) [inline]

Remove lines of text thinner or equal to `min_thickness`.

Parameters

in	*components*	An object image.
in	*min_thickness*	the minimum thickness value.

Returns

An object image without too thin components.

6.11.2.8 template<typename L > component_set<L> scribo::filter::components_v_thin (const component_set< L > & *comps,* unsigned *min_thinness*) [inline]

Remove lines of text thinner or equal to `min_thinness`.

Parameters

in	*comps*	A component set.
in	*min_thinness*	the minimum thinness value.

Returns

An object image without too thin vertical components.

6.11.2.9 template<typename L > component_set<L> scribo::filter::components_with_two_holes (const component_set< L > & *components,* unsigned *min_size*) [inline]

Remove components having at least two holes.

This is a fastest version since it is optimized for 2 holes detection.

Parameters

in	*components*	A component set.
in	*min_size*	The minimum hole area to take a hole into account.

Returns

A component where the component having at least two holes are invalidated.

6.11.2.10 template<typename I , typename N , typename V > mln::trait::concrete< I >::ret scribo::filter::objects_h_thick (const Image< I > & *input_,* const Neighborhood< N > & *nbh_,* const V & *label_type,* unsigned *max_thickness*) [inline]

Remove objects horizontaly thicker or equal to `max_thickness`.

Parameters

in	*input_*	A binary image.
in	*nbh_*	A neighborhood used in labeling algorithms.
in	*label_type*	The label type used for labeling.
in	*max_thickness*	The maximum thickness value.

Returns

A binary image without thick objects.

6.11.2.11 template<typename L > component_set<L> scribo::filter::objects_h_thick (const component_set< L > & *comps,* unsigned *max_thickness*) [inline]

Remove objects horizontaly thicker or equal to `max_thickness`.

Parameters

in	*comps*	Component data.
in	*max_thickness*	The minimum thickness value.

Returns

A component data set without too thick components.

6.11.2.12 template<typename I , typename N , typename V > mln::trait::concrete< I >::ret scribo::filter::objects_h_thin (const Image< I > & *input_,* const Neighborhood< N > & *nbh_,* const V & *label_type,* unsigned *min_thinness*) [inline]

Remove components thinner or equal to `min_thinness`.

Parameters

in	*input_*	a binary image.
in	*nbh_*	a neighborhood used in labeling algorithms.
in	*label_type*	the label type used for labeling.
in	*min_thinness*	the minimum thinness value.

Returns

A binary image without h_thin components.

6.11.2.13 template<typename L > component_set<L> scribo::filter::objects_h_thin (const component_set< L > & *comps,* unsigned *min_thinness*) [inline]

Remove lines of text thinner or equal to `min_thinness`.

Parameters

in	*comps*	A component set.
in	*min_thinness*	the minimum thinness value.

Returns

An object image without too thin vertical components.

6.11.2.14 template<typename L > component_set<L> scribo::filter::objects_size_ratio (const component_set< L > & *comps,* float *min_size_ratio*)

Invalidate components with a height/width ratio too low.

Compute the ratio height/width from the component bounding boxes and compare it to `size_ratio`.

If the height/width ratio is lower than min_size_ratio then the component is invalidated.

6.11.2.15 template<typename I , typename N , typename V > mln::trait::concrete< I >::ret scribo::filter::objects_thick (const Image< I > & *input_*, const Neighborhood< N > & *nbh_*, const V & *label_type*, unsigned *max_thickness*) [inline]

Remove components thicker or equal to `max_thickness`.

Parameters

in	*input_*	A binary image.
in	*nbh_*	A neighborhood used in labeling algorithms.
in	*label_type*	The label type used for labeling.
in	*max_thickness*	The maximum thickness value.

Returns

A binary image without thick components.

6.11.2.16 template<typename L > component_set<L> scribo::filter::objects_thick (const component_set< L > & *components*, unsigned *max_thickness*) [inline]

Remove components thicker or equal to `max_thickness`.

Parameters

in	*components*	An object image.
in	*max_thickness*	The maximum thickness value.

Returns

An object image without too thick components.

6.11.2.17 template<typename I , typename N , typename V > mln::trait::concrete< I >::ret scribo::filter::objects_v_thick (const Image< I > & *input_*, const Neighborhood< N > & *nbh_*, const V & *label_type*, unsigned *max_thickness*) [inline]

Remove components verticaly thicker or equal to `max_thickness`.

Parameters

in	*input_*	A binary image.
in	*nbh_*	A neighborhood used in labeling algorithms.
in	*label_type*	The label type used for labeling.
in	*max_thickness*	The maximum thickness value.

Returns

A binary image without thick components.

6.11.2.18 template< typename L > component_set<L> scribo::filter::objects_v_thick (const component_set< L > & *comps*, unsigned *max_thickness*) [inline]

Remove components verticaly thicker or equal to `max_thickness`.

Parameters

in	*comps*	A component set.
in	*max_thickness*	The maximum thickness value.

Returns

An object image without too thick components.

6.11.2.19 template< typename I , typename N , typename V > mln::trait::concrete< I >::ret scribo::filter::objects_v_thin (const Image< I > & *input*, const Neighborhood< N > & *nbh*, const V & *label_type*, unsigned *min_thinness*) [inline]

Remove components thinner or equal to `min_thinness`.

Parameters

in	*input*	a binary image.
in	*nbh*	a neighborhood used in labeling algorithms.
in	*label_type*	the label type used for labeling.
in	*min_thinness*	the minimum thinness value.

Returns

A binary image without v_thin components.

6.11.2.20 template< typename L > component_set<L> scribo::filter::objects_with_holes (const component_set< L > & *components*, unsigned *min_holes_count*, unsigned *min_size*)

Remove components having a minimum number of holes.

Parameters

in	*components*	A component set.
in	*min_holes_count*	If a component have at least `min_holes_count` holes it is invalidated.
in	*min_size*	The minimum hole area to take a hole into account.

Returns

A component where the component having too much holes are invalidated.

6.12 Element Filtering

Functions

- template<typename L >
  void scribo::filter::images_in_paragraph (document< L > &doc)
- template<typename L >
  void scribo::filter::objects_in_borders (component_set< L > &components, **float** vratio, **float** hratio)
- template<typename L >
  void scribo::filter::separators_in_borders (document< L > &doc, **float** vratio, **float** hratio)
- template<typename L >
  void scribo::filter::separators_in_element (document< L > &doc)
- template<typename L >
  void scribo::filter::separators_in_paragraph (document< L > &doc, **unsigned** hmin_size, **unsigned** vmin_size)
- template<typename L >
  void scribo::filter::separators_vert_in_borders (document< L > &doc)

6.12.1 Detailed Description

Elements filtering routines.

6.12.2 Function Documentation

6.12.2.1 template<typename L > void scribo::filter::images_in_paragraph (document< L > & *doc*)

Invalidate false positive images.

Parameters

`in,out`	*doc*	A document structure.

6.12.2.2 template<typename L > void scribo::filter::objects_in_borders (component_set< L > & *components,* float *vratio,* float *hratio*)

Invalidate components located close to the image borders.

Parameters

`in,out`	*components*	A component set.
`in`	*vratio*	Ratio to be used for evaluating the inner border size in which vertical separators will be invalidated.
`in`	*hratio*	Ratio to be used for evaluating the inner border size in which horizontal separators will be invalidated.

Warning

It only invalidates components in the given component_set.

```
-----------
|_!____!__|
| !    ! <--------- Components located in this area are
| !    !  |         invalidated.
| !    !  |
|_!____!__|
| !    !  |
-----------
```

6.12.2.3 template<typename L > void scribo::filter::separators_in_borders (document< L > & *doc*, float *vratio*, float *hratio*)

Invalidate separators located close to the image borders.

Parameters

in,out	*doc*	A document structure.
in	*vratio*	Ratio to be used for evaluating the inner border size in which vertical separators will be invalidated.
in	*hratio*	Ratio to be used for evaluating the inner border size in which horizontal separators will be invalidated.

Warning

it does not remove separators from separator image. **It** only invalidate separator components in their respective component_set.

```
-----------
|_!_____!__|
| !     ! <--------- Separators located in this area are
| !     !  |         invalidated.
| !     !  |
|_!_____!__|
| !     !  |
-----------
```

6.12.2.4 template<typename L > void scribo::filter::separators_in_element (document< L > & *doc*)

Invalidate false positive separators.

Parameters

in,out	*doc*	A document structure.

Warning: it does not remove separators from separator image. **It** only invalidate separator components in their respective component_set.

6.12.2.5 template<typename L > void scribo::filter::separators_in_paragraph (document< L > & *doc*, unsigned *hmin_size*, unsigned *vmin_size*)

Invalidate false positive separators.

Parameters

in,out	*doc*	A document structure.
in	*hmin_size*	Minimum width of a line to be considered as separators.
in	*vmin_size*	Minimum height of a line to be considered as separators.

Warning: it does not remove separators from separator image. **It** only invalidate separator components in their respective component_set.

6.12.2.6 template<typename L > void scribo::filter::separators_vert_in_borders (document< L > & *doc*)

Invalidate vertical separators located close to the image borders.

Parameters

in,out	*doc*	A document structure.

Warning: it does not remove separators from separator image. **It** only invalidate separator components in their respective component_set.

```
-----------
|_!____!__|
| !    ! <--------- Separators located in this area are
| !    !  |         invalidated.
| !    !  |
|_!____!__|
| !    !  |
-----------
```

6.13 Element Filtering

Modules

- Component Filtering
- Component Group Filtering
- Component Link Filtering
- Element Filtering
- Line Link Filtering
- Paragraph Filtering

6.13.1 Detailed Description

Element filtering routines.

6.14 Line Link Filtering

Functions

- template<typename L >
  line_links< L > scribo::filter::line_links_x_height (const line_links< L > &links)

6.14.1 Detailed Description

Line link filtering routines.

6.14.2 Function Documentation

6.14.2.1 template<typename L > line_links<L> scribo::filter::line_links_x_height (const line_links< L > & *links*)

Filter line links according to character x height.

Parameters

in	*links*	Line links information.

Returns

A new line_links structure where some links may have been invalidated.

6.15 Paragraph Filtering

Functions

- template<typename L >
  paragraph_set< L > scribo::filter::paragraphs_bbox_overlap (const paragraph_set< L > &parset)
- template<typename L >
  void scribo::filter::paragraphs_in_borders (document< L > &doc)
- template<typename L >
  void scribo::filter::paragraphs_in_image (document< L > &doc)

6.15.1 Detailed Description

Paragraph filtering routines.

6.15.2 Function Documentation

6.15.2.1 template<typename L > paragraph_set<L> scribo::filter::paragraphs_bbox_overlap (const paragraph_set< L > & *parset*)

Remove invalid paragraphs.

Parameters

in	*parset*	A paragraph set.

Returns

A paragraph set with invalid paragraphs tag set to Paragraph::Ignored.

6.15.2.2 template<typename L > void scribo::filter::paragraphs_in_borders (document< L > & *doc*)

Invalidate paragraphs located close to the image borders.

Parameters

in,out	*doc*	A document structure.

Warning: it does not remove paragraphs from separator image. **It** only invalidate separator components in their respective component_set.

```
-----------
|_!____!__|
| !    ! <--------- Paragraphs located in this area are
| !    ! |          invalidated.
| !    ! |
|_!____!__|
| !    ! |
-----------
```

6.15.2.3 template<typename L > void scribo::filter::paragraphs_in_image (document< L > & *doc*)

Remove invalid paragraphs.

Parameters

in,out	*doc*	A document structure.

Returns

A paragraph set with invalid paragraphs tag set to Paragraph::Ignored.

6.16 Layout Analysis

Functions

- template<typename I >
  mln::util::array< mln_box(I)> scribo::layout::xy_cut (const **Image**< I > &ima, **int** min_height, **int** min_width)

6.16.1 Detailed Description

Document Layout Analysis algorithms.

6.16.2 Function Documentation

6.16.2.1 template<typename I > mln::util::array<mln_box(I)> scribo::layout::xy_cut (const Image< I > & *ima,* int *min_height,* int *min_width*)

XY-Cut layout analysis algorithm.

This algorithm is an implementation inspired by [4], [2] and [3].

It recusively subdivides empty spaces in the document until a minimum division size is reached. The latter is defined with `min_height` and `min_width`.

Parameters

`in`	*ima*	A binary image.
`in`	*min_height*	The minimum height of a subdivision.
`in`	*min_width*	The minimum width of a subdivision.

Returns

An array of component group bounding boxes.

6.17 Components Extraction

Functions

- template<typename I , typename J , typename N , typename V >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > scribo::primitive::extract::components (const **Image**< I > &input, const **Image**< J > &binary_input, const **Neighborhood**< N > &nbh, V &ncomponents, component::Type type=component::Undefined)
- template<typename I , typename N , typename V >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > scribo::primitive::extract::components (const **Image**< I > &binary_input, const **Neighborhood**< N > &nbh, V &ncomponents, component::Type type=component::Undefined)

6.17.1 Detailed Description

All routines/algorithms to extract components.

6.17.2 Function Documentation

6.17.2.1 template<typename I , typename J , typename N , typename V > component_set<typename mln::trait::ch_value< I , V >::ret> scribo::primitive::extract::components (const Image< I > & *input*, const Image< J > & *binary_input*, const Neighborhood< N > & *nbh*, V & *ncomponents*, component::Type *type* = `component::Undefined`) `[inline]`

Extract components in a binary image.

Parameters

`in`	*input*	A RGB image.
`in`	*binary_input*	A binary image. Components must be set to 'True'. and background to 'false'.
`in`	*nbh*	A neighborhood to be used for labeling.
`in,out`	*ncomponents*	Will store the numbers of components found.
`in`	*type*	The default component type set to components.

Returns

An image of labeled components.

6.17.2.2 template<typename I , typename N , typename V > component_set<typename mln::trait::ch_value< I , V >::ret> scribo::primitive::extract::components (const Image< I > & *binary_input*, const Neighborhood< N > & *nbh*, V & *ncomponents*, component::Type *type* = `component::Undefined`) `[inline]`

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.

Parameters

`in`	*binary_input*	A binary image. Components must be set to 'True'. and background to 'false'.
`in`	*nbh*	A neighborhood to be used for labeling.
`in,out`	*ncomponents*	Will store the numbers of components found.
`in`	*type*	The default component type set to components.

Returns

A component_set.

6.18 Primitive Extraction

Modules

- Lines and Separators extraction

6.18.1 Detailed Description

All routines/algorithms to extract primitive objects.

6.19 Lines and Separators extraction

Functions

- template<typename I >
  mln::trait::concrete< I >::**ret** scribo::primitive::extract::horizontal_separators (const **Image**< I > &input, **unsigned** line_length)
- template<typename I , typename N , typename V , typename W >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > scribo::primitive::extract::lines_discontinued (const **Image**< I > &input_, const **Neighborhood**< N > &nbh_, V &nlines, const **Window**< W > &win_, **unsigned** rank_k)
- template<typename I , typename N , typename V >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > scribo::primitive::extract::lines_h_discontinued (const **Image**< I > &input, const **Neighborhood**< N > &nbh, V &nlines, **unsigned** line_length, **unsigned** rank_k)
- template<typename I >
  mln::trait::concrete< I >::**ret** scribo::primitive::extract::lines_h_pattern (const **Image**< I > &input, **unsigned** length, **unsigned delta**)
- template<typename I , typename N , typename V >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > scribo::primitive::extract::lines_h_single (const **Image**< I > &input, const **Neighborhood**< N > &nbh, const V &nlines, **unsigned** min_line_length, **float** w_h_ratio)
- template<typename L >
  component_set< L > scribo::primitive::extract::lines_h_single (const component_set< L > &components, **unsigned** min_line_length, **float** w_h_ratio)
- template<typename I , typename N , typename V >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > scribo::primitive::extract::lines_h_thick_and_single (const **Image**< I > &input, const **Neighborhood**< N > &nbh, V &nlines, **unsigned** min_line_length, **float** h_w_ratio)
- template<typename I >
  mln::trait::concrete< I >::**ret** scribo::primitive::extract::lines_h_thick_and_thin (const **Image**< I > &binary_image, **unsigned** length, **unsigned delta**, **float** p_few=0.2, **float** p_enough=0.6, **float** ratio=8)
- template<typename I , typename W >
  mln::trait::concrete< I >::**ret** scribo::primitive::extract::lines_pattern (const **Image**< I > &input_, **unsigned** length, **unsigned** dir, const **Window**< W > &win_)
- template<typename I , typename N , typename V , typename W >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > scribo::primitive::extract::lines_thick (const **Image**< I > &input_, const **Neighborhood**< N > &nbh_, V &nlines, **unsigned** line_length)
- template<typename I , typename N , typename V , typename W >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > scribo::primitive::extract::lines_thick (const **Image**< I > &input_, const **Neighborhood**< N > &nbh_, V &nlines, const **Window**< W > &win_)
- template<typename I , typename N , typename V >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > scribo::primitive::extract::lines_v_discontinued (const Image< I > &input, const Neighborhood< N > &nbh, V &nlines, **unsigned** line_length, **unsigned** rank_k)
- template<typename I >
  mln::trait::concrete< I >::**ret** scribo::primitive::extract::lines_v_pattern (const **Image**< I > &input, **unsigned** length, **unsigned delta**)

- template<typename I , typename N , typename V >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > scribo::primitive::extract::lines_v_single (const **Image**< I > &input, const **Neighborhood**< N > &nbh, const V &nlines, **unsigned** min_line_length, **float** h_w_ratio)
- template<typename L >
  component_set< L > scribo::primitive::extract::lines_v_single (const component_set< L > &components, **unsigned** min_line_length, **float** h_w_ratio)
- template<typename I , typename N , typename V >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > scribo::primitive::extract::lines_v_thick_and_single (const **Image**< I > &input, const **Neighborhood**< N > &nbh, V &nlines, **unsigned** min_line_length, **float** h_w_ratio)
- template<typename I >
  mln::trait::concrete< I >::**ret** scribo::primitive::extract::separators (const **Image**< I > &input, **unsigned** line_length)
- template<typename I >
  mln::trait::concrete< I >::**ret** scribo::primitive::extract::separators_nonvisible (const **Image**< I > &in_)
- template<typename I >
  mln::trait::concrete< I >::**ret** scribo::primitive::extract::vertical_separators (const **Image**< I > &input, **unsigned** line_length)

6.19.1 Detailed Description

All routines/algorithms to extract lines and separators.

6.19.2 Function Documentation

6.19.2.1 template<typename I > mln::trait::concrete< I >::ret scribo::primitive::extract::horizontal_separators (const Image< I > & *input,* unsigned *line_length*)

Extract horizontal separators.

Parameters

in	*input*	A binary image.
in	*line_length*	The minimum line length.

Returns

A binary image were separators are set to 'True'.

See Also

primitive::remove::separators

6.19.2.2 template<typename I , typename N , typename V , typename W > component_set<typename mln::trait::ch_value< I , V >::ret> scribo::primitive::extract::lines_discontinued (const Image< I > & *input_,* const Neighborhood< N > & *nbh_,* V & *nlines,* const Window< W > & *win_,* unsigned *rank_k*)

Extract discontinued lines in a binary image.

Based on a rank filter.

Parameters

in	*input_*	A binary image.
in	*nbh_*	The neighborhood used for labeling image the lines.
in,out	*nlines*	The label type used for labeling.
in	*win_*	A Window used to extract lines.
in	*rank_k*	Rank used for filtering.

Returns

An image in which lines are labeled.

Precondition

win.length() $>$ 2 $*$ (rank_k + 1)

6.19.2.3 template< typename I , typename N , typename V > component_set< typename mln::trait::ch_value< I , V >::ret> scribo::primitive::extract::lines_h_discontinued (const Image< I > & *input*, const Neighborhood< N > & *nbh*, V & *nlines*, unsigned *line_length*, unsigned *rank_k*)

Extract horizontal discontinued lines.

Parameters

in	*input*	A binary image.
in	*nbh*	A neighborhood used to label lines.
in,out	*nlines*	The number of lines found.
in	*line_length*	The minimum line length expected. (must be odd).
in	*rank_k*	Rank filter parameter.

Returns

An image in which lines are labeled with a value different from 0.

6.19.2.4 template< typename I > mln::trait::concrete< I >::ret scribo::primitive::extract::lines_h_pattern (const Image< I > & *input*, unsigned *length*, unsigned *delta*)

Extract horizontal lines matching a specific pattern.

Parameters

in	*input*	A binary image.
in	*length*	The minimum line length.
in	*delta*	Distance between the object pixel and the background pixel.

Returns

An image of horizontal lines.

```
o
|     ^
|     |  Delta
|     v
X
|     ^
|     |  Delta
|     v
o
```

Using a delta of 0 is equivalent to the use of a c2_row neighborhood.

6.19.2.5 template< typename I , typename N , typename V > component_set< typename mln::trait::ch_value< I , V >::ret> scribo::primitive::extract::lines_h_single (const Image< I > & *input,* const Neighborhood< N > & *nbh,* const V & *nlines,* unsigned *min_line_length,* float *w_h_ratio*)

Fast Extraction of single horizontal thick lines.

Only single non discontinued lines are correctly extracted with this routine.

Parameters

in	*input*	A binary image.
in	*nbh*	The neighborhood used for labeling image components.
in,out	*nlines*	Type used for labeling.
in	*min_line_length*	The minimum line length.
in	*w_h_ratio*	The minimum ratio width/height object bounding boxes to consider an object as a single line.

Returns

An image in which only horizontal single lines are labeled.

6.19.2.6 template< typename L > component_set<L> scribo::primitive::extract::lines_h_single (const component_set< L > & *components,* unsigned *min_line_length,* float *w_h_ratio*)

Fast Extraction of single horizontal thick lines.

Only single non discontinued lines are correctly extracted with this routine.

Parameters

in	*components*	A labeled image.
in	*min_line_length*	The minimum line length.
in	*w_h_ratio*	The minimum ratio width/height object bounding boxes to consider an object as a single line.

Returns

An image in which only horizontal single lines are labeled.

6.19.2.7 template< typename I , typename N , typename V > component_set< typename mln::trait::ch_value< I , V >::ret> scribo::primitive::extract::lines_h_thick_and_single (const Image< I > & *input,* const Neighborhood< N > & *nbh,* V & *nlines,* unsigned *min_line_length,* float *h_w_ratio*)

Extract horizontal thick lines in a binary image.

Only non discontinued lines are correctly extracted with this routine. Only lines matching the given criterions are kept in the result.

Parameters

in	*input*	A binary image.
in	*nbh*	The neighborhood used for labeling image components.
in,out	*nlines*	Type used for labeling.
in	*min_line_length*	The minimum line length.
in	*h_w_ratio*	The minimum ratio width/height object bounding boxes to consider an object as a single line.

Returns

An image in which lines are labeled.

6.19.2.8 template<typename I > mln::trait::concrete< I >::ret scribo::primitive::extract::lines_h_thick_and_thin (const Image< I > & *binary_image*, unsigned *length*, unsigned *delta*, float *p_few* = 0.2, float *p_enough* = 0.6, float *ratio* = 8)

Extract both thick and thin horizontal lines.

6.19.2.9 template<typename I , typename W > mln::trait::concrete< I >::ret scribo::primitive::extract::lines_pattern (const Image< I > & *input_*, unsigned *length*, unsigned *dir*, const Window< W > & *win_*)

Extract lines with a specific pattern.

Parameters

in	*input_*	A binary image.
in	*length*	The minimum line length.
in	*dir*	The direction of the lines.
in	*win_*	A window corresponding to the line pattern.

Returns

A image with lines of direction `dir`.

6.19.2.10 template<typename I , typename N , typename V , typename W > component_set< typename mln::trait::ch_value< I, V >::ret > scribo::primitive::extract::lines_thick (const Image< I > & *input_*, const Neighborhood< N > & *nbh_*, V & *nlines*, unsigned *line_length*)

Extract thick lines in a binary image.

Only non discontinued lines are correctly extracted with this routine.

Parameters

in	*input_*	A binary image.
in	*nbh_*	The neighborhood used for labeling image components.
in,out	*nlines*	Type used for labeling.
in	*line_length*	The minimum line length.

Returns

An image in which lines are labeled.

6.19.2.11 template<typename I , typename N , typename V , typename W > component_set<typename mln::trait::ch_value< I , V >::ret> scribo::primitive::extract::lines_thick (const Image< I > & *input_*, const Neighborhood< N > & *nbh_*, V & *nlines*, const Window< W > & *win_*)

Extract thick lines in a binary image. Only non discontinued lines are correctly extracted with this routine.

Parameters

in	*input_*	A binary image.
in	*nbh_*	The neighborhood used for labeling image components.

in,out	*nlines*	Type used for labeling.
in	*win_*	Window used to extract the lines

Returns

An image in which lines are labeled.

6.19.2.12 template<typename I , typename N , typename V > component_set<typename mln::trait::ch_value< I , V >::ret> scribo::primitive::extract::lines_v_discontinued (const Image< I > & *input,* const Neighborhood< N > & *nbh,* V & *nlines,* unsigned *line_length,* unsigned *rank_k*)

Extract vertical discontinued lines.

Parameters

in	*input*	A binary image.
in	*nbh*	A neighborhood used to label lines.
in,out	*nlines*	The number of lines found.
in	*line_length*	The minimum line length expected. (must be odd).
in	*rank_k*	Rank filter parameter.

Returns

An image in which lines are labeled with a value different from 0.

6.19.2.13 template<typename I > mln::trait::concrete< I >::ret scribo::primitive::extract::lines_v_pattern (const Image< I > & *input,* unsigned *length,* unsigned *delta*)

Extract vertical lines matching a specific pattern.

Parameters

in	*input*	A binary image.
in	*length*	The minimum line length.
in	*delta*	space between the first background pixels and the line pixels (usually 2 or 3).

Returns

An image of vertical lines.

6.19.2.14 template<typename I , typename N , typename V > component_set<typename mln::trait::ch_value< I , V >::ret> scribo::primitive::extract::lines_v_single (const Image< I > & *input,* const Neighborhood< N > & *nbh,* const V & *nlines,* unsigned *min_line_length,* float *h_w_ratio*)

Fast Extraction of single vertical thick lines.

Only single non discontinued lines are correctly extracted with this routine.

Parameters

in	*input*	A binary image.
in	*nbh*	The neighborhood used for labeling image components.
in	*nlines*	Type used for labeling.
in	*min_line_length*	The minimum line length.

in	*h_w_ratio*	The minimum ratio height/width object bounding boxes to consider an object as a single line.

Returns

An image in which only vertical single lines are labeled.

6.19.2.15 template<typename L > component_set<L> scribo::primitive::extract::lines_v_single (const component_set< L > & *components,* unsigned *min_line_length,* float *h_w_ratio*)

Fast Extraction of single vertical thick lines.

Only single non discontinued lines are correctly extracted with this routine.

Parameters

in	*components*	A labeled image.
in	*min_line_length*	The minimum line length.
in	*h_w_ratio*	The minimum ratio height/width object bounding boxes to consider an object as a single line.

Returns

An image in which only vertical single lines are labeled.

6.19.2.16 template<typename I , typename N , typename V > component_set<typename mln::trait::ch_value< I , V >::ret> scribo::primitive::extract::lines_v_thick_and_single (const Image< I > & *input,* const Neighborhood< N > & *nbh,* V & *nlines,* unsigned *min_line_length,* float *h_w_ratio*)

Extract vertical thick lines in a binary image.

Only non discontinued lines are correctly extracted with this routine. Only lines matching the given criterions are kept in the result.

Parameters

in	*input*	A binary image.
in	*nbh*	The neighborhood used for labeling image components.
in,out	*nlines*	Type used for labeling.
in	*min_line_length*	The minimum line length.
in	*h_w_ratio*	The minimum ratio height/width object bounding boxes to consider an object as a single line.

Returns

An image in which lines are labeled.

6.19.2.17 template<typename I > mln::trait::concrete< I >::ret scribo::primitive::extract::separators (const Image< I > & *input,* unsigned *line_length*)

Extract vertical and horizontal separators.

Parameters

in	*input*	A binary image.
in	*line_length*	The minimum line length.

Returns

A binary image were separators are set to 'True'.

See Also

primitive::remove::separators

6.19.2.18 template<typename I > mln::trait::concrete< I >::ret scribo::primitive::extract::separators_nonvisible (const Image< I > & *in_*)

Find non visible separators. Based on components alignments.

6.19.2.19 template<typename I > mln::trait::concrete< I >::ret scribo::primitive::extract::vertical_separators (const Image< I > & *input*, unsigned *line_length*)

Extract vertical separators.

Parameters

in	*input*	A binary image.
in	*line_length*	The minimum line length.

Returns

A binary image were separators are set to 'True'.

See Also

primitive::remove::separators

6.20 Routines

Modules

- Binarization
- Components Extraction
- Debug
- Element Filtering
- Layout Analysis
- Primitive Extraction
- Text Extraction
- Text Recognition
- Toolchains

6.20.1 Detailed Description

All routines/algorithms provided in Scribo.

6.21 Text Recognition

Functions

- template<typename L >
  void scribo::text::recognition (line_set< L > &lines, const char ∗language)
- template<typename I >
  void scribo::text::recognition (const **Image**< I > &**line**, const char ∗language, const std::string &output_file=std::string())

6.21.1 Detailed Description

All routines/algorithms to recognize text.

6.21.2 Function Documentation

6.21.2.1 template<typename L > void scribo::text::recognition (line_set< L > & *lines*, const char ∗ *language*)

Passes the text bboxes to Tesseract (OCR).

Parameters

`in`	*lines*	The lines of text.
`in`	*language*	The language which should be recognized by Tesseract. (fra, en, ...)

6.21.2.2 template<typename I > void scribo::text::recognition (const Image< I > & *line*, const char ∗ *language*, const std::string & *output_file* = `std::string()`)

Recognize text from an image.

Parameters

`in`	*line*	Image of text line.
`in`	*language*	The language which should be recognized by Tesseract. (fra, en, ...)
`in`	*output_file*	If set, store the recognized text in this file.

6.22 Text Extraction

Functions

- template<typename I , typename N >
  line_set< typename
  mln::trait::ch_value< I,
  scribo::def::lbl_type >::**ret** > scribo::text::extract_lines (const **Image**< I > &input, const **Neighborhood**< N > &nbh, const typename **mln::trait::ch_value**< I, **bool** >::**ret** &separators)
- template<typename I , typename N >
  line_set< typename
  mln::trait::ch_value< I,
  scribo::def::lbl_type >::**ret** > scribo::text::extract_lines (const **Image**< I > &input, const **Neighborhood**< N > &nbh)
- template<typename L >
  line_set< typename
  mln::trait::ch_value< L,
  scribo::def::lbl_type >::**ret** > scribo::text::extract_lines (const component_set< L > &input)
- template<typename I , typename J , typename N >
  line_set< typename
  mln::trait::ch_value< I,
  scribo::def::lbl_type >::**ret** > scribo::text::extract_lines_with_features (const **Image**< I > &input, const **Image**< J > &input_binary, const **Neighborhood**< N > &nbh, const typename **mln::trait::ch_value**< I, **bool** >::**ret** &separators)
- template<typename I , typename J , typename N >
  line_set< typename
  mln::trait::ch_value< I,
  scribo::def::lbl_type >::**ret** > scribo::text::extract_lines_with_features (const **Image**< I > &input, const **Image**< J > &input_binary, const **Neighborhood**< N > &nbh)
- template<typename I , typename N >
  line_set< typename
  mln::trait::ch_value< I,
  scribo::def::lbl_type >::**ret** > scribo::text::extract_lines_wo_merge (const **Image**< I > &input, const **Neighborhood**< N > &nbh, const typename **mln::trait::ch_value**< I, **bool** >::**ret** &separators)
- template<typename I , typename N >
  line_set< typename
  mln::trait::ch_value< I,
  scribo::def::lbl_type >::**ret** > scribo::text::extract_lines_wo_merge (const **Image**< I > &input, const **Neighborhood**< N > &nbh)
- template<typename L , typename N >
  line_set< typename
  mln::trait::ch_value< L,
  scribo::def::lbl_type >::**ret** > scribo::text::extract_lines_wo_merge (const document< L > &doc, const **Neighborhood**< N > &nbh)
- template<typename L , typename N >
  line_set< typename
  mln::trait::ch_value< L,
  scribo::def::lbl_type >::**ret** > scribo::text::extract_lines_wo_merge (const document< L > &doc, const **Neighborhood**< N > &nbh, const typename **mln::trait::ch_value**< L, **bool** >::**ret** &separators)
- template<typename L >
  paragraph_set< L > scribo::text::extract_paragraphs (line_set< L > &lines, const **image2d**< **bool** > &input)
- template<typename L >
  paragraph_set< L > scribo::text::extract_paragraphs_hdoc (line_set< L > &lines, const **image2d**< **bool** > &input)
- template<typename L >
  line_links< L > scribo::text::link_lines (const line_set< L > &lines)

- template<typename L >
  line_set< L > scribo::text::merging (const scribo::line_set< L > &lines)
- template<typename L >
  line_set< L > scribo::text::merging_hdoc (const scribo::line_set< L > &lines)

6.22.1 Detailed Description

All routines/algorithms to extract text.

6.22.2 Function Documentation

6.22.2.1 template<typename I , typename N > line_set<typename **mln::trait::ch_value**< I , **scribo::def::lbl_type** >::ret> scribo::text::extract_lines (const **Image**< I > & *input,* const **Neighborhood**< N > & *nbh,* const typename **mln::trait::ch_value**< I, **bool** >::ret & *separators*)

Extract lines of text in a binary image.

Parameters

in	*input*	A binary image.
in	*nbh*	A neighborhood used for labeling.
in	*separators*	A binary image with separator information.

Returns

A set of lines.

6.22.2.2 template<typename I , typename N > line_set<typename **mln::trait::ch_value**< I , **scribo::def::lbl_type** >::ret> scribo::text::extract_lines (const **Image**< I > & *input,* const **Neighborhood**< N > & *nbh*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.

6.22.2.3 template<typename L > line_set<typename **mln::trait::ch_value**< L , **scribo::def::lbl_type** >::ret> scribo::text::extract_lines (const component_set< L > & *input*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.

6.22.2.4 template<typename I , typename J , typename N > line_set<typename **mln::trait::ch_value**< I , **scribo::def::lbl_type** >::ret> scribo::text::extract_lines_with_features (const **Image**< I > & *input,* const **Image**< J > & *input_binary,* const **Neighborhood**< N > & *nbh,* const typename **mln::trait::ch_value**< I, **bool** >::ret & *separators*)

Extract lines of text in a binary image.

Parameters

in	*input*	Original color image.
in	*input_binary*	A binary image.
in	*nbh*	A neighborhood used for labeling.
in	*separators*	A binary image with separator information.

Returns

A set of lines.

6.22.2.5 template< typename I , typename J , typename N > line_set< typename mln::trait::ch_value< I , scribo::def::lbl_type >::ret> scribo::text::extract_lines_with_features (const Image< I > & *input,* const Image< J > & *input_binary,* const Neighborhood< N > & *nbh*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.

6.22.2.6 template< typename I , typename N > line_set< typename mln::trait::ch_value< I , scribo::def::lbl_type >::ret> scribo::text::extract_lines_wo_merge (const Image< I > & *input,* const Neighborhood< N > & *nbh,* const typename mln::trait::ch_value< I, bool >::ret & *separators*)

Extract lines of text in a binary image.

Parameters

in	*input*	A binary image.
in	*nbh*	A neighborhood used for labeling.
in	*separators*	A binary image with separator information.

Returns

A set of lines.

6.22.2.7 template< typename I , typename N > line_set< typename mln::trait::ch_value< I , scribo::def::lbl_type >::ret> scribo::text::extract_lines_wo_merge (const Image< I > & *input,* const Neighborhood< N > & *nbh*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.

6.22.2.8 template< typename L , typename N > line_set< typename mln::trait::ch_value< L , scribo::def::lbl_type >::ret> scribo::text::extract_lines_wo_merge (const document< L > & *doc,* const Neighborhood< N > & *nbh*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.

6.22.2.9 template< typename L , typename N > line_set< typename mln::trait::ch_value< L , scribo::def::lbl_type >::ret> scribo::text::extract_lines_wo_merge (const document< L > & *doc,* const Neighborhood< N > & *nbh,* const typename mln::trait::ch_value< L, bool >::ret & *separators*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.

6.22.2.10 template< typename L > paragraph_set< L> scribo::text::extract_paragraphs (line_set< L > & *lines,* const image2d< bool > & *input*) [inline]

Extract paragraphs.

6.22.2.11 template<typename L > paragraph_set<L> scribo::text::extract_paragraphs_hdoc (line_set< L > & *lines*, const image2d< bool > & *input*)

Extract paragraph in old documents.

6.22.2.12 template<typename L > line_links<L> scribo::text::link_lines (const line_set< L > & *lines*)

Link text lines to prepare paragraph construction.

Lines MUST be tagged as line::Text to be considered in this processing.

6.22.2.13 template<typename L > line_set<L> scribo::text::merging (const scribo::line_set< L > & *lines*)

Merge text component in order to reconstruct text lines.

Parameters

in	*lines*	A line set.

Returns

A new line set. Line ids are preserved and merged lines (not valid anymore) are tagged with line::Merged. The lines produced with this algorithm (valid lines) are tagged with line::None. Line type is also set either with line::Text or line::Punctuation.

6.22.2.14 template<typename L > line_set<L> scribo::text::merging_hdoc (const scribo::line_set< L > & *lines*)

Merge text component in order to reconstruct text lines in old documents.

Parameters

in	*lines*	A line set.

Returns

A new line set. Line ids are preserved and merged lines (not valid anymore) are tagged with line::Merged. The lines produced with this algorithm (valid lines) are tagged with line::None. Line type is also set either with line::Text or line::Punctuation.

6.23 Toolchains

Modules

- Documents
- Pictures

Functions

- QSet< QString > scribo::toolchain::nepomuk::text_extraction (const QImage &input, const QString &language)

6.23.1 Detailed Description

Full toolchains performing content analysis and extraction.

6.23.2 Function Documentation

6.23.2.1 QSet<QString> scribo::toolchain::nepomuk::text_extraction (const QImage & *input,* const QString & *language*)

Extract text from a document.

This is a convenient routine to be used in Nepomuk.

Parameters

in	*input*	A document image.
in	*language*	The main language used in the input document image. Improve text recognition quality if accurate.

Returns

A set of recognized words.

Don't forget to define NDEBUG for compilation to disable debug checks.

Depending on your version of Tesseract (OCR) you may define HAVE_TESSERACT_2 or HAVE_TESSERACT_3 .

6.24 Pictures

Modules

- Preprocessing
- Processing

6.24.1 Detailed Description

Full toolchains performing content analysis and extraction in pictures.

6.25 Documents

Modules

- Preprocessing
- Processing

6.25.1 Detailed Description

Full toolchains performing content analysis and extraction in document images.

6.26 Preprocessing

Classical preprocessing toolchains for pictures.

6.27 Preprocessing

Functions

- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::toolchain::text_in_doc_preprocess (const **Image**< I > &input, **bool** enable_fg_bg, **unsigned** lambda, **double** K, **bool** enable_deskew, **bool** verbose)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::toolchain::text_in_doc_preprocess (const **Image**< I > &input, **bool** enable_fg_bg, **bool** verbose)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::toolchain::text_in_doc_preprocess (const **Image**< I > &input, **unsigned** lambda, **bool** verbose)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::toolchain::text_in_doc_preprocess (const **Image**< I > &input, **unsigned** lambda, **double** k2, **double** k3, **double** k4, **bool** enable_fg_bg, **Image**< I > &fg, **bool** enable_deskew, **bool** verbose)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** scribo::toolchain::text_in_doc_preprocess (const **Image**< I > &input, **unsigned** lambda, **bool** enable_fg_bg, **Image**< I > &fg, **bool** enable_deskew, **bool** verbose)

6.27.1 Detailed Description

Classical preprocessing toolchains for document image.

6.27.2 Function Documentation

6.27.2.1 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::toolchain::text_in_doc_preprocess (const Image< I > & *input*, bool *enable_fg_bg*, unsigned *lambda*, double *K*, bool *enable_deskew*, bool *verbose*)

Preprocess a document before looking for its content.

Parameters

in	*input*	An image.
in	*enable_fg_bg*	Enable/Disable background removal.
in	*lambda*	Parameter to the background removal. Maximum area of foreground objects.
in	*K*	Binarization threshold parameter. Use the same value for all scales.(Default 0.34)
in	*enable_deskew*	Deskew document.
in	*verbose*	Enable/Disable debug output on std::cout.

If `enable_fg_bg` is set to 'True' then a background removal is performed. Its parameter lambda is automatically set according to the input image size.

6.27.2.2 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::toolchain::text_in_doc_preprocess (const Image< I > & *input*, bool *enable_fg_bg*, bool *verbose*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts. K is set to 0.34.

6.27.2.3 template< typename I > mln::trait::ch_value< I , bool >::ret scribo::toolchain::text_in_doc_preprocess (const Image< I > & *input*, unsigned *lambda*, bool *verbose*)

Preprocess a document before looking for its content.

Parameters

in	*input*	An image.
in	*lambda*	Parameter to the background removal.
in	*verbose*	Enable/Disable debug information printed on std::cout.

If lambda is set to '0' no background removal is performed. Otherwise, a background removal is performed with the given `lambda` value.

6.27.2.4 template< typename I > mln::trait::ch_value< I , bool >::ret scribo::toolchain::text_in_doc_preprocess (const Image< I > & *input*, unsigned *lambda*, double *k2*, double *k3*, double *k4*, bool *enable_fg_bg*, Image< I > & *fg*, bool *enable_deskew*, bool *verbose*)

Preprocess a document before looking for its content. This methods relies on a multi-scale implementation of Sauvola's binarization.

Parameters

in	*input*	An image.
in	*lambda*	Parameter to the background removal.
in	*k2*	Binarization threshold parameter for scale 2. (Default 0.34)
in	*k3*	Binarization threshold parameter for scale 3. (Default 0.34)
in	*k4*	Binarization threshold parameter for scale 4. (Default 0.34)
in	*enable_fg_bg*	If set to True tries to identify background components and remove them.
in,out	*fg*	The foreground layer of `input`.
in	*enable_deskew*	Deskew document.
in	*verbose*	Enable/Disable debug information printed on std::cout.

If lambda is set to '0' no background removal is performed. Otherwise, a background removal is performed with the given `lambda` value.

6.27.2.5 template< typename I > mln::trait::ch_value< I , bool >::ret scribo::toolchain::text_in_doc_preprocess (const Image< I > & *input*, unsigned *lambda*, bool *enable_fg_bg*, Image< I > & *fg*, bool *enable_deskew*, bool *verbose*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.

6.28 Processing

Functions

- template<typename I >
  component_set< typename **mln::trait::ch_value**< I, def::lbl_type >::**ret** > scribo::toolchain::text_in_picture (const **Image**< I > &input_rgb_orig, **bool** bg_removal, **bool** multi_scale_bin, **unsigned** max_dim_size=0, **unsigned** lambda=0, **bool** verbose=false)

6.28.1 Detailed Description

Classical processing toolchains for pictures.

6.28.2 Function Documentation

6.28.2.1 template<typename I > component_set<typename **mln::trait::ch_value**< I , def::lbl_type >::**ret**> scribo::toolchain::text_in_picture (const **Image**< I > & *input_rgb_orig,* **bool** *bg_removal,* **bool** *multi_scale_bin,* **unsigned** *max_dim_size =* `0`*,* **unsigned** *lambda =* `0`*,* **bool** *verbose =* `false`)

Localize text in pictures.

Parameters

`in`	*input_rgb_orig*	A RGB image.
`in`	*bg_removal*	If set to True tries to identify background components and remove them.
`in`	*multi_scale_bin*	Sauvola's binarization is used. Tells whether to use a multi-scale version or not.
`in`	*max_dim_size*	The maximum height or width allowed for the input image. If larger, it is resized.
`in`	*lambda*	Maximum area of components considered as foreground objects. If set to '0' (default), it is fixed automatically.
`in`	*verbose*	Enable/Disable debug output on std::cout.

Returns

A component set corresponding to the text components.

6.29 Processing

Functions

- template< typename I , typename J >
  document< typename
  mln::trait::ch_value< I,
  def::lbl_type >::**ret** > scribo::toolchain::content_in_doc (const **Image**< I > &input, const **Image**< J > &input_preproc, **bool** denoise, **bool** find_line_seps=true, **bool** find_whitespace_seps=true, **bool** enable_ocr=true, const std::string &language=std::string("eng"), bool verbose=false)
- template< typename I , typename J >
  document< typename
  mln::trait::ch_value< I,
  def::lbl_type >::**ret** > scribo::toolchain::content_in_hdoc (const **Image**< I > &input, const **Image**< J > &input_preproc, **bool** denoise, **bool** find_line_seps=true, **bool** find_whitespace_seps=true, **bool** enable_ocr=true, const std::string &language=std::string("eng"))
- template< typename I >
  line_set< typename
  mln::trait::ch_value< I,
  def::lbl_type >::**ret** > scribo::toolchain::text_in_doc (const **Image**< I > &input, **bool** denoise, const std::string &language=std::string("eng"), bool find_line_seps=true, **bool** find_whitespace_seps=true, **bool** verbose=false)

6.29.1 Detailed Description

Classical processing toolchains for document image.

6.29.2 Function Documentation

6.29.2.1 template< typename I , typename J > document< typename mln::trait::ch_value< I , def::lbl_type >::ret> scribo::toolchain::content_in_doc (const Image< I > & *input,* const Image< J > & *input_preproc,* bool *denoise,* bool *find_line_seps* = true, bool *find_whitespace_seps* = true, bool *enable_ocr* = true, const std::string & *language* = std::string("eng"), bool *verbose* = false)

Analyse and extract document image content.

Parameters

in	*input*	A RGB image.
in	*input_preproc*	A Binary image.
in	*denoise*	Remove too small components (<= 2 pixels).
in	*find_line_seps*	Enable/Disable lookup for vertical/horizontal separators.
in	*find_whitespace_seps*	Enable/Disable lookup for vertical alignments and whitespaces. Helps finding text layout.
in	*enable_ocr*	Enable/Disable text recognition.
in	*language*	Main language used in the input document. (OCR settings)
in	*verbose*	Enable/Disable debug information printed on std::cout.

Returns

A document structure.

6.29.2.2 template< typename I , typename J > document< typename mln::trait::ch_value< I , def::lbl_type >::ret > scribo::toolchain::content_in_hdoc (const Image< I > & *input*, const Image< J > & *input_preproc*, bool *denoise*, bool *find_line_seps* = `true`, bool *find_whitespace_seps* = `true`, bool *enable_ocr* = `true`, const std::string & *language* = `std::string("eng")`)

Analyse and extract content in a historical/degraded document.

Parameters

`in`	*input*	A RGB image.
`in`	*input_preproc*	A Binary image.
`in`	*denoise*	Remove too small components (<= 2 pixels).
`in`	*find_line_seps*	Enable/Disable lookup for vertical/horizontal separators.
`in`	*find_whitespace-_seps*	Enable/Disable lookup for vertical alignments and whitespaces. Helps finding text layout.
`in`	*enable_ocr*	Enable/Disable text recognition.
`in`	*language*	Main language used in the input document. (OCR settings)

Returns

A document structure.

6.29.2.3 template< typename I > line_set< typename mln::trait::ch_value< I , def::lbl_type >::ret > scribo::toolchain::text_in_doc (const Image< I > & *input*, bool *denoise*, const std::string & *language* = `std::string("eng")`, bool *find_line_seps* = `true`, bool *find_whitespace_seps* = `true`, bool *verbose* = `false`)

Extract text lines from a document image.

Parameters

`in`	*input*	A RGB image.
`in`	*denoise*	Remove too small components (<= 2 pixels).
`in`	*language*	Main language used in the input document. (OCR settings)
`in`	*find_line_seps*	Enable/Disable lookup for vertical/horizontal separators.
`in`	*find_whitespace-_seps*	Enable/Disable lookup for vertical alignments and whitespaces. Helps finding text layout.
`in`	*verbose*	Enable/Disable debug information printed on std::cout.

Returns

A line set including text information and recognized text.

Chapter 7

Namespace Documentation

7.1 scribo Namespace Reference

Namespaces

- namespace binarization
- namespace component
- namespace core
- namespace debug
- namespace draw
- namespace filter
- namespace make
- namespace postprocessing
- namespace preprocessing
- namespace primitive
- namespace table
- namespace text

Classes

- struct component_features_data
- class component_info

  Component information data structure.
- class component_set

  Represents all the components in a document image.
- class DMax_Functor

  Dmax functor concept.
- class doc_serializer

  Link functor concept.
- class document

  Represent document data and structure.
- class group_info
- class line_info
- class line_links

  Line links representation.
- class line_set

  Lines container.
- class Link_Functor

Link functor concept.

- class object_groups

  Object group representation.
- class object_links

  Object links representation.
- class paragraph_info

  Paragraph structure information.
- class paragraph_set

  Paragraph container.
- class Serializable

  Concept for objects that can be serialized.
- class SerializeVisitor

  Link functor concept.

Typedefs

- typedef **mln::util::object_id**
  < scribo::ComponentId,
  unsigned > component_id_t
- typedef **mln::util::object_id**
  < scribo::LineId, **unsigned** > **line_id_t**
- typedef **mln::util::object_id**
  < scribo::ParagraphId,
  unsigned > **paragraph_id_t**

Functions

- template<typename P >
  mln::util::couple< P, P > central_sites (const **box**< P > &b, **unsigned** dim)
- template<typename I , typename L >
  void erase_objects (**Image**< I > &input, const component_set< L > &comps)
- template<typename I , typename F >
  mln::trait::ch_value< I,
  double >::**ret init_integral_image** (const **Image**< I > &input_, F &func)
- std::ostream & **operator**<< (std::ostream &ostr, const component_features_data &**data**)
- template<typename L >
  std::ostream & **operator**<< (std::ostream &ostr, const paragraph_info< L > &info)
- template<typename L >
  std::ostream & **operator**<< (std::ostream &ostr, const line_links< L > &links)
- template<typename L >
  std::ostream & **operator**<< (std::ostream &ostr, const component_info< L > &info)
- template<typename L >
  std::ostream & **operator**<< (std::ostream &ostr, const object_groups< L > &groups)
- template<typename L >
  std::ostream & **operator**<< (std::ostream &ostr, const line_set< L > &lines)
- template<typename L >
  std::ostream & **operator**<< (std::ostream &ostr, const line_info< L > &info)
- **bool operator==** (const component_features_data &lhs, const component_features_data &rhs)
- **bool operator==** (const group_info &lhs, const group_info &rhs)
- template<typename L >
  bool operator== (const paragraph_set< L > &lhs, const paragraph_set< L > &rhs)
- template<typename L >
  bool operator== (const paragraph_info< L > &lhs, const paragraph_info< L > &rhs)

- template<typename L >
  bool operator== (const line_links< L > &lhs, const line_links< L > &rhs)
- template<typename L >
  bool operator== (const component_info< L > &lhs, const component_info< L > &rhs)
- template<typename L >
  bool operator== (const object_groups< L > &lhs, const object_groups< L > &rhs)
- template<typename L >
  bool operator== (const document< L > &lhs, const document< L > &rhs)
- template<typename L >
  bool operator== (const line_set< L > &lhs, const line_set< L > &rhs)
- template<typename L >
  bool operator== (const line_info< L > &lhs, const line_info< L > &rhs)

7.1.1 Detailed Description

The main namespace of the Scribo module. try to determine the type of a component.

FIXME: provide a version for binary images.

FIXME: share code with filter/object_groups_with_holes.hh FIXME: Merge the two following routines.

Function increasing values to highlight areas.

FIXME: return type too restrictive!

FIXME: result is not consistent with other binarization methods.

Namespace of the whole project.

The result is inverted because of the threshold functor: should we invert the test in this functor?

FIXME: Use a size ratio in both overloads.

7.1.2 Typedef Documentation

7.1.2.1 typedef mln::util::object_id<scribo::ComponentId, unsigned> scribo::component_id_t

The type of the component ids.

This id is mainly used in structures like scribo::component_set and scribo::component_info structures. **It** refers to the actual label used in the underlying labeled component image stored in scribo::component_set.

Definition at line 48 of file component_info.hh.

7.1.3 Function Documentation

7.1.3.1 template<typename P > mln::util::couple<P,P> scribo::central_sites (const box< P > & *b*, unsigned *dim*)

Returns the edge central sites of a box.

Parameters

in	*b*	the bbbox
in	*dim*	the dimension used to compute the site.

If dim == 0, returns the left and right central sites.

|-------|

X X

|-------|

If dim == 1, returns the top and bottom central sites.

|—X—|

| |

|—X—|

... And so on.

7.1.3.2 template<typename I , typename L > void scribo::erase_objects (Image< I > & *input*, const component_set< L > & *comps*)

Remove labeled components from a binary image.

Parameters

`in,out`	*input*	A binary image.
`in`	*comps*	A set of components to be erased.

7.2 scribo::binarization Namespace Reference

Functions

- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** global_threshold (const **Image**< I > &input, const typename **I::value** &**threshold**)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** global_threshold_auto (const **Image**< I > &input)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** kim (const **Image**< I > &input, **unsigned** window_size, **double** k)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** kim (const **Image**< I > &input, **unsigned** window_size)
- template<typename I , typename T >
  mln::trait::ch_value< I, **bool** >
  ::**ret** local_threshold (const **Image**< I > &input, const **Image**< **T** > &**threshold**)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** niblack (const **Image**< I > &input, **unsigned** window_size, **double** K)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** niblack (const **Image**< I > &input, **unsigned** window_size)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** niblack (const **Image**< I > &input)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** niblack_fast (const **Image**< I > &input, **unsigned** window_size, **double** K)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** niblack_fast (const **Image**< I > &input, **unsigned** window_size)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** niblack_fast (const **Image**< I > &input)

- template<typename I , typename J >
  mln::trait::ch_value< I,
  value::int_u8 >::**ret** niblack_threshold (const **Image**< I > &input, **unsigned** window_size, **double** K)
- template<typename I >
  mln::trait::ch_value< I,
  value::int_u8 >::**ret** niblack_threshold (const **Image**< I > &input, **unsigned** window_size)
- template<typename I >
  mln::trait::ch_value< I,
  value::int_u8 >::**ret** niblack_threshold (const **Image**< I > &input)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** otsu (const **Image**< I > &input)
- template<typename I >
  I::value otsu_threshold (const **Image**< I > &input)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** sauvola (const **Image**< I > &input, **unsigned** window_size, **double** K)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** sauvola (const **Image**< I > &input, **unsigned** window_size)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** sauvola (const **Image**< I > &input)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** sauvola_ms (const **Image**< I > &input_1, **unsigned** w_1, **unsigned** s, **image2d**< **mln::util::couple**< **double**, **double** > > &integral_sum_sum_2)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** sauvola_ms (const **Image**< I > &input_1, **unsigned** w_1, **unsigned** s)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** sauvola_ms (const **Image**< I > &input_1, **unsigned** w_1)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** sauvola_ms (const **Image**< I > &input_1, **unsigned** w_1, **unsigned** s, **double** k2, **double** k3, **double** k4)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** sauvola_ms (const **Image**< I > &input_1, **unsigned** w_1, **unsigned** s, **double** all_k)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** sauvola_ms_split (const **Image**< I > &input_1, **unsigned** w_1, **unsigned** s, **unsigned** min_ntrue, **double** k2, **double** k3, **double** k4)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** sauvola_ms_split (const **Image**< I > &input_1, **unsigned** w_1, **unsigned** s, **unsigned** min_ntrue, **double** K)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** sauvola_ms_split (const **Image**< I > &input_1, **unsigned** w_1, **unsigned** s, **unsigned** min_ntrue)
- template<typename I , typename J >
  mln::trait::ch_value< I,
  value::int_u8 >::**ret** sauvola_threshold (const **Image**< I > &input, **unsigned** window_size, **double** K)
- template<typename I >
  mln::trait::ch_value< I,
  value::int_u8 >::**ret** sauvola_threshold (const **Image**< I > &input, **unsigned** window_size)

- template<typename I >
  mln::trait::ch_value< I,
  value::int_u8 >::**ret** sauvola_threshold (const **Image**< I > &input)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** singh (const **Image**< I > &input, **unsigned** window_size, **double** K)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** singh (const **Image**< I > &input, **unsigned** window_size)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** singh (const **Image**< I > &input)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** wolf (const **Image**< I > &input, **unsigned** window_size, **double** K)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** wolf (const **Image**< I > &input, **unsigned** window_size)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** wolf (const **Image**< I > &input)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** wolf_fast (const **Image**< I > &input, **unsigned** window_size, **double** K)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** wolf_fast (const **Image**< I > &input, **unsigned** window_size)
- template<typename I >
  mln::trait::ch_value< I, **bool** >
  ::**ret** wolf_fast (const **Image**< I > &input)

7.2.1 Detailed Description

Namespace of binarization routines.

7.2.2 Function Documentation

7.2.2.1 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::global_threshold (const Image< I > & *input*, const typename I::value & *threshold*)

Binarize an image using a global threshold value.

For a site 'p' in `input` image:

output(p) = input(p) >= threshold

Parameters

in	*input*	A grayscale image.
in	*threshold*	A value.

Returns

A boolean image.

7.2.2.2 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::global_threshold_auto (const Image< I > & *input*)

Simple binarization of a gray-level document.

Automatically find a global threshold for the given image.

Parameters

in	*input*	A gray-level image.

Returns

A Boolean image.

7.2.2.3 template<typename I , typename T > mln::trait::ch_value< I , bool >::ret scribo::binarization::local_threshold (const Image< I > & *input,* const Image< T > & *threshold*)

Binarize an image using a threshold image.

The threshold image is used to specify a specific threshold for each site.

For a site 'p' in `input` image:

output(p) = input(p) >= threshold(p)

Precondition

input.domain() == threshold.domain()

Parameters

in	*input*	A grayscale image.
in	*threshold*	A grayscale image.

Returns

A boolean image.

7.2.2.4 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::sauvola_ms (const Image< I > & *input_1,* unsigned *w_1,* unsigned *s,* double *k2,* double *k3,* double *k4*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.Allow to specify a different k parameter for each scale.

7.2.2.5 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::sauvola_ms (const Image< I > & *input_1,* unsigned *w_1,* unsigned *s,* double *all_k*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.Allow to specify the same k parameter for all scales.

7.2.2.6 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::singh (const Image< I > & *input,* unsigned *window_size,* double *K*)

Convert an image into a binary image.

Parameters

in	*input*	An image.
in	*window_size*	The window size.
in	*K*	Singh's formulae constant.

Returns

A binary image.

7.2.2.7 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::singh (const Image< I > & *input,* unsigned *window_size*)

Convert an image into a binary image.

Singh's formulae constant K is set to 0.34.

Parameters

in	*input*	An image.
in	*window_size*	The window size.

Returns

A binary image.

7.2.2.8 template<typename I > mln::trait::ch_value< I , bool >::ret scribo::binarization::singh (const Image< I > & *input*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.The window size is set to 11.

7.3 scribo::component Namespace Reference

Enumerations

- enum Tag { None = 0, Ignored }
- enum Type {
  Undefined = 0, Character, VerticalLineSeparator, HorizontalLineSeparator,
  WhitespaceSeparator, Noise, Punctuation, Image,
  DropCapital }

Functions

- std::ostream & operator<< (std::ostream &ostr, const Tag &tag)
- std::ostream & operator<< (std::ostream &ostr, const Type &type)
- Tag str2tag (const std::string &str)
- Type str2type (const std::string &str)

7.3.1 Detailed Description

Namespace of classes related to components.

7.3.2 Enumeration Type Documentation

7.3.2.1 enum scribo::component::Tag

All possible tags that can be used for components.

component::Tag values can be converted to String using operator<<(std::ostream &ostr, const Tag &tag). String values can be converted to component::Tag using str2tag().

See Also

scribo::component_set scribo::component_info

Enumerator:

None No tag set.

Ignored To be ignored in further processing.

Definition at line 54 of file component.hh.

7.3.2.2 enum scribo::component::Type

All possible types of components.

component::Type values can be converted to String using operator<<(std::ostream &ostr, const Type &type). String values can be converted to component::Type using str2type().

See Also

scribo::component_set scribo::component_info str2type()

Enumerator:

Undefined No type defined.

Character Character.

VerticalLineSeparator Vertical line separator.

HorizontalLineSeparator Horizontal line separator.

WhitespaceSeparator whitespace separator.

Noise Noise.

Punctuation Punctuation.

Image Image.

DropCapital Drop capital.

Definition at line 69 of file component.hh.

7.3.3 Function Documentation

7.3.3.1 std::ostream& scribo::component::operator<< (std::ostream & *ostr*, const Tag & *tag*)

Operator allowing scribo::component::Tag to be printed out.

7.3.3.2 std::ostream& scribo::component::operator<< (std::ostream & *ostr*, const Type & *type*)

Operator allowing scribo::component::Type to be printed out.

7.3.3.3 Tag scribo::component::str2tag (const std::string & *str*)

Convert a std::string to a scribo::component::Tag.

7.3.3.4 Type scribo::component::str2type (const std::string & *str*)

Convert a std::string to a scribo::component::Type.

7.4 scribo::core Namespace Reference

7.4.1 Detailed Description

Namespace of core routines.

7.5 scribo::debug Namespace Reference

Classes

- struct arg_data
- struct opt_data
- class option_parser
- struct toggle_data

Enumerations

- enum Level {
  None = 0, **Special**, **Results**, **AuxiliaryResults**,
  All, **InvalidLevel** }
- enum VerboseMode {
  Mute = 0, **UserDebug**, **Time**, **Low**,
  Medium, **Full**, **Invalid** }

Functions

- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** alignment_decision_image (const **Image**< I > &input, const object_links< L > &links, const object_links< L > &filtered_links, const anchor::Type &anchor)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** bboxes_enlarged_image (const **Image**< I > &input, const line_set< L > &lines, const **value::rgb8** &text_value, const **value::rgb8** &non_text_value)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** bboxes_enlarged_image (const **Image**< I > &input, const line_set< L > &lines)
- template<typename I >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** bboxes_image (const **Image**< I > &input, const **mln::util::array**< **box**< typename I::site > > &bboxes, const **value::rgb8** &**value**)

- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** bboxes_image (const **Image**< I > &input, const line_set< L > &lines, const **value::rgb8** &**value**)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** bboxes_image (const **Image**< I > &input, const line_set< L > &lines)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** bboxes_image (const **Image**< I > &input, const component_set< L > &comps, const **value::rgb8** &**value**)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** char_space_image (const **Image**< I > &input, const line_set< L > &**line**, const **value::rgb8** &v=literal::cyan)
- **bool check_ocr_lang** (const std::vector< const char ∗ > &args)
- **bool check_sauvola_first_subsampling** (const std::vector< const char ∗ > &args)
- **bool check_sauvola_split_ntrue** (const std::vector< const char ∗ > &args)
- **bool check_verbose_mode** (const std::vector< const char ∗ > &args)
- **bool check_xml_format** (const std::vector< const char ∗ > &args)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** decision_image (const **Image**< I > &input, const object_groups< L > &groups, const object_groups< L > &filtered_groups, anchor::Type anchor)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** decision_image (const **Image**< I > &input, const object_links< L > &links, const object_links< L > &filtered_links, anchor::Type anchor)
- template<typename I >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** highlight_text_area (const **Image**< I > &input, const **mln::util::array**< **box**< typename I::site > > &**bbox**)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** highlight_text_area (const **Image**< I > &input, const line_set< L > &lines)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** highlight_text_area (const **Image**< I > &input, const scribo::component_set< L > &components)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** line_info_image (const **Image**< I > &input, const line_set< L > &**line**)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** linked_bboxes_image (const **Image**< I > &input, const object_links< L > &array, const **value::rgb8** &box_value, const **value::rgb8** &link_value, anchor::Type anchor)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** linked_bboxes_image (const **Image**< I > &input, const object_links< L > &array, const **value::rgb8** &box_value, const **value::rgb8** &link_value)
- template<typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** linked_bboxes_image (const **Image**< I > &input, const object_links< L > &left_link, const object_links< L > &right_link, const **value::rgb8** &box_value, const **value::rgb8** &link_value, anchor::Type anchor)

- template< typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** linked_bboxes_image (const **Image**< I > &input, const object_links< L > &left_link, const object_links< L > &right_link, const **value::rgb8** &box_value, const **value::rgb8** &left_link_value, const **value::rgb8** &right_link_value, const **value::rgb8** &validated_link_value, anchor::Type anchor)
- template< typename I , typename L , typename G >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** linked_bboxes_image (const **Image**< I > &input, const **Graph**< G > &g, const **value::rgb8** &box_value, const **value::rgb8** &link_value, anchor::Type anchor)
- template< typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** links_decision_image (const **Image**< I > &input_, const object_links< L > &links, const object_links< L > &filtered_links)
- template< typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** links_image (const **Image**< I > &input_, const object_links< L > &links, anchor::Type anchor, **bool** draw_bboxes=true)
- scribo::debug::internal::logger_ & logger ()
- template< typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** looks_like_a_text_line_image (const **Image**< I > &input, const line_set< L > &lines, const **value::rgb8** &text_value, const **value::rgb8** &non_text_value)
- template< typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** looks_like_a_text_line_image (const **Image**< I > &input, const line_set< L > &lines)
- template< typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** mean_and_base_lines_image (const **Image**< I > &input, const line_set< L > &lines, const **value::rgb8** &bbox_value, const **value::rgb8** &meanline_value, const **value::rgb8** &baseline_value)
- template< typename I , typename L >
  mln::trait::ch_value< I,
  value::rgb8 >::**ret** mean_and_base_lines_image (const **Image**< I > &input, const line_set< L > &lines)
- template< typename L , typename L2 >
  void save_comp_diff (const component_set< L > &comps_ref, const component_set< L2 > &comps_new, const std::string &**filename**)
- template< typename I >
  void save_label_image (const **Image**< I > &lbl, const typename **I::value** &nlabels, const char ∗**filename**)
- template< typename I >
  void save_table_image (const **Image**< I > &input_, **mln::util::couple**< **mln::util::array**< **box**< typename I::site > >, **mln::util::array**< **box**< typename I::site > > > tableboxes, const **value::rgb8** &bbox_color, const std::string &**filename**)
- template< typename S >
  void save_table_image (const **Site_Set**< S > &input_domain, **mln::util::couple**< **mln::util::array**< **box**< typename S::site > >, **mln::util::array**< **box**< typename S::site > > > tableboxes, const **value::rgb8** &bg_color, const **value::rgb8** &bbox_color, const std::string &**filename**)
- template< typename I , typename L >
  mln::trait::concrete< I >::**ret** text_areas_image (const **Image**< I > &input_rgb, const scribo::component_set< L > &comps)
- template< typename L >
  image2d< **value::rgb8** > text_color_image (const document< L > &doc)
- VerboseMode txt_to_verbose_mode (const std::string &name)
- **int** usage (char ∗argv[ ], const char ∗desc, const char ∗args, const char ∗args_desc[ ][2])

7.5.1 Detailed Description

Namespace of debug routines.

7.5.2 Enumeration Type Documentation

7.5.2.1 enum scribo::debug::Level

Enum defining different level of image logging.

According to the debug level set in scribo::debug::logger_ the image may be saved or not.

Definition at line 57 of file logger.hh.

7.5.2.2 enum scribo::debug::VerboseMode

Enum defining different modes of text logging.

According to the verbose mode set in scribo::debug::logger_ the text may be logged or not.

Definition at line 72 of file logger.hh.

7.5.3 Function Documentation

7.5.3.1 VerboseMode scribo::debug::txt_to_verbose_mode (const std::string & *name*)

returns the corresponding verboseMode from its name.

7.5.3.2 int scribo::debug::usage (char ∗ *argv[ ],* const char ∗ *desc,* const char ∗ *args,* const char ∗ *args_desc[ ][2]*)

Format a standard usage output.

Parameters

in	*argv*	Arguments passed to the program.
in	*desc*	Description of the program.
in	*args*	The expected arguments.
in	*args_desc*	The description of the expected arguments.

Returns

Return 1.

7.6 scribo::draw Namespace Reference

Functions

- template<typename I , typename L >
  void bounding_box_links (**Image**< I > &input, const object_links< L > &link, const typename **I::value** &**value**, anchor::Type anchor)
- template<typename I , typename L >
  void bounding_box_links (**Image**< I > &input, const object_links< L > &link, const typename **I::value** &**value**)
- template<typename I , typename L >
  void bounding_box_links (**Image**< I > &input, const object_links< L > &left_link, const object_links< L > &right_link, const typename **I::value** &left_link_value, const typename **I::value** &right_link_value, const typename **I::value** &validated_link_value, anchor::Type anchor)
- template<typename I , typename G >
  void bounding_box_links (**Image**< I > &input, const **Graph**< G > &g, const typename **I::value** &link_value)

- template<typename I >
  void bounding_boxes (**Image**< I > &input_, const **mln::util::array**< **box**< typename I::site > > &boxes, const typename **I::value** &**value**)
- template<typename I , typename L >
  void bounding_boxes (**Image**< I > &input_, const component_set< L > &components, const typename **I::value** &**value**)
- template<typename I , typename L >
  void groups_bboxes (**Image**< I > &input_, const object_groups< L > &groups, const typename **I::value** &**value**)
- template<typename L , typename I >
  void **line_components** (**Image**< I > &input_, const line_set< L > &lines, const line_info< L > &**line**, const typename **I::value** &**value**)

7.6.1 Detailed Description

Namespace of drawing routines.

7.6.2 Function Documentation

7.6.2.1 template<typename I , typename L > void scribo::draw::bounding_box_links (Image< I > & *input*, const object_links< L > & *link*, const typename I::value & *value*, anchor::Type *anchor*)

Draw a list of bounding box links.

Parameters

in,out	*input*	An image where to draw.
in	*link*	component links.
in	*value*	Value used to draw links.
in	*anchor*	Anchor from where the links are drawn.

7.6.2.2 template<typename I , typename L > void scribo::draw::bounding_box_links (Image< I > & *input*, const object_links< L > & *link*, const typename I::value & *value*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.The default anchor type is set to anchor::Center.

7.6.2.3 template<typename I , typename L > void scribo::draw::bounding_box_links (Image< I > & *input*, const object_links< L > & *left_link*, const object_links< L > & *right_link*, const typename I::value & *left_link_value*, const typename I::value & *right_link_value*, const typename I::value & *validated_link_value*, anchor::Type *anchor*) [inline]

Draw left, right and validated lists of bounding box links.

Parameters

in,out	*input*	An image where to draw.
in	*left_link*	Component's left links.
in	*right_link*	Component's right links.
in	*left_link_value*	Value used to draw left links.
in	*right_link_value*	Value used to draw right links.
in	*validated_link_value*	Value used to draw validated links.
in	*anchor*	Anchor from where the links are drawn.

7.6.2.4 template<typename I , typename G > void scribo::draw::bounding_box_links (Image< I > & *input*, const Graph< G > & *g*, const typename I::value & *link_value*) [inline]

Draw a graph of bounding box links.

Draw from bounding box centers.

Parameters

in,out	*input*	An image where to draw.
in	*g*	The link graph.
in	*link_value*	The value used to draw the links.

7.6.2.5 template<typename I > void scribo::draw::bounding_boxes (Image< I > & *input_*, const mln::util::array< box< typename I::site > > & *boxes*, const typename I::value & *value*)

Draw a list of bounding boxes.

7.6.2.6 template<typename I , typename L > void scribo::draw::bounding_boxes (Image< I > & *input_*, const component_set< L > & *components*, const typename I::value & *value*)

Draw object bounding boxes.

7.6.2.7 template<typename I , typename L > void scribo::draw::groups_bboxes (Image< I > & *input_*, const object_groups< L > & *groups*, const typename I::value & *value*) [inline]

Draw a list of bounding boxes.

7.7 scribo::filter Namespace Reference

Functions

- template<typename I , typename N , typename V >
  mln::trait::concrete< I >::**ret** components_large (const **Image**< I > &input_, const **Neighborhood**< N > &nbh_, const V &label_type, **unsigned** max_size)
- template<typename L >
  component_set< L > components_large (const component_set< L > &components, **unsigned** max_size)
- template<typename L >
  component_set< L > components_on_border (const component_set< L > &components)
- template<typename I , typename N , typename V >
  mln::trait::concrete< I >::**ret** components_small (const **Image**< I > &input, const **Neighborhood**< N > &nbh, V &nlabels, **unsigned** min_size)
- template<typename L >
  component_set< L > components_small (const component_set< L > &components, **unsigned** min_size)
- template<typename I , typename N , typename V >
  mln::trait::concrete< I >::**ret** components_thin (const **Image**< I > &input_, const **Neighborhood**< N > &nbh_, const V &label_type, **unsigned** min_thickness)
- template<typename L >
  component_set< L > components_thin (const component_set< L > &components, **unsigned** min_thickness)
- template<typename L >
  component_set< L > components_v_thin (const component_set< L > &comps, **unsigned** min_thinness)

- template<typename L >
  component_set< L > components_with_two_holes (const component_set< L > &components, **unsigned** min_size)
- template<typename P , typename V , typename G , typename F , typename FP >
  edge_image< void, **bool**, G > **graph_edges** (const **vertex_image**< P, V, G > &v_ima, const **Function**< F > &, const **Function**< FP > &)
- template<typename L >
  void images_in_paragraph (document< L > &doc)
- template<typename L >
  line_links< L > line_links_x_height (const line_links< L > &links)
- template<typename L >
  object_groups< L > object_groups_mean_width (const object_groups< L > &groups, **float** width)
- template<typename L >
  object_groups< L > object_groups_size_ratio (const object_groups< L > &groups, **float** max_size_ratio, **float** max_invalid_ratio_per_group)
- template<typename L >
  object_groups< L > object_groups_small (const object_groups< L > &groups, **unsigned** n_links)
- template<typename L >
  object_groups< L > object_groups_with_holes (const object_groups< L > &components, **unsigned** min_size)
- template<typename L >
  object_links< L > object_links_aligned (const object_links< L > &links, **float** max_alpha, anchor::Type anchor)
- template<typename L >
  object_links< L > object_links_bbox_h_ratio (const object_links< L > &links, **float** max_h_ratio)
- template<typename L >
  object_links< L > object_links_bbox_overlap (const object_links< L > &links, **float** max_overlap_ratio)
- template<typename L >
  object_links< L > object_links_bbox_ratio (const object_links< L > &links, **unsigned** dim, **float** max_ratio)
- template<typename L >
  object_links< L > object_links_bbox_w_ratio (const object_links< L > &links, **float** max_w_ratio)
- template<typename L >
  object_links< L > object_links_bottom_aligned (const object_links< L > &links, **float** max_alpha)
- template<typename L >
  object_links< L > object_links_center_aligned (const object_links< L > &links, **float** max_alpha)
- template<typename L >
  object_links< L > object_links_left_aligned (const object_links< L > &links, **float** max_alpha)
- template<typename L >
  object_links< L > object_links_non_aligned_simple (const object_links< L > &links, anchor::Type anchor, **float** max_alpha)
- template<typename L >
  object_links< L > object_links_right_aligned (const object_links< L > &links, **float** max_alpha)
- template<typename L >
  object_links< L > object_links_top_aligned (const object_links< L > &links, **float** max_alpha)
- template<typename I , typename N , typename V >
  mln::trait::concrete< I >::**ret** objects_h_thick (const **Image**< I > &input_, const **Neighborhood**< N > &nbh_, const V &label_type, **unsigned** max_thickness)
- template<typename L >
  component_set< L > objects_h_thick (const component_set< L > &comps, **unsigned** max_thickness)
- template<typename I , typename N , typename V >
  mln::trait::concrete< I >::**ret** objects_h_thin (const **Image**< I > &input_, const **Neighborhood**< N > &nbh_, const V &label_type, **unsigned** min_thinness)
- template<typename L >
  component_set< L > objects_h_thin (const component_set< L > &comps, **unsigned** min_thinness)
- template<typename L >
  void objects_in_borders (component_set< L > &components, **float** vratio, **float** hratio)

- template<typename L >
  component_set< L > objects_size_ratio (const component_set< L > &comps, **float** min_size_ratio)
- template<typename I , typename N , typename V >
  mln::trait::concrete< I >::**ret** objects_thick (const **Image**< I > &input_, const **Neighborhood**< N > &nbh_, const V &label_type, **unsigned** max_thickness)
- template<typename L >
  component_set< L > objects_thick (const component_set< L > &components, **unsigned** max_thickness)
- template<typename I , typename N , typename V >
  mln::trait::concrete< I >::**ret** objects_v_thick (const **Image**< I > &input_, const **Neighborhood**< N > &nbh_, const V &label_type, **unsigned** max_thickness)
- template<typename L >
  component_set< L > objects_v_thick (const component_set< L > &comps, **unsigned** max_thickness)
- template<typename I , typename N , typename V >
  mln::trait::concrete< I >::**ret** objects_v_thin (const **Image**< I > &input, const **Neighborhood**< N > &nbh, const V &label_type, **unsigned** min_thinness)
- template<typename L >
  component_set< L > objects_with_holes (const component_set< L > &components, **unsigned** min_holes_count, **unsigned** min_size)
- template<typename L >
  paragraph_set< L > paragraphs_bbox_overlap (const paragraph_set< L > &parset)
- template<typename L >
  void paragraphs_in_borders (document< L > &doc)
- template<typename L >
  void paragraphs_in_image (document< L > &doc)
- template<typename L >
  void separators_in_borders (document< L > &doc, **float** vratio, **float** hratio)
- template<typename L >
  void separators_in_element (document< L > &doc)
- template<typename L >
  void separators_in_paragraph (document< L > &doc, **unsigned** hmin_size, **unsigned** vmin_size)
- template<typename L >
  void separators_vert_in_borders (document< L > &doc)

7.7.1 Detailed Description

Namespace of filtering rountines.

7.8 scribo::make Namespace Reference

Functions

- std::string debug_filename (const std::string &name)
- template<typename I , typename N , typename V >
  mln::util::graph influence_zone_graph (const **Image**< I > &input_, const **Neighborhood**< N > &nbh_, const V &label_type, **unsigned** iz_dmax)
- template<typename L >
  scribo::line_set< L > **line_set** (const object_groups< L > &groups)
- template<typename L >
  scribo::paragraph_set< L > paragraph (const line_links< L > &llinks, const line_links< L > &rlinks)
- template<typename L >
  scribo::paragraph_set< L > paragraph (const scribo::line_set< L > &lines)
- template<typename L >
  scribo::paragraph_set< L > paragraph (const line_links< L > &llinks)

- template<typename L >
  mln::trait::ch_value< L, **bool** >
  ::**ret** text_blocks_image (const document< L > &doc, **unsigned** min_nlines)
- template<typename L >
  mln::trait::ch_value< L, **bool** >
  ::**ret** text_components_image (const document< L > &doc)

7.8.1 Detailed Description

Namespace of routines constructing objects.

7.8.2 Function Documentation

7.8.2.1 std::string scribo::make::debug_filename (const std::string & *name*)

Construct and returns a formated output file name:

```
'input_filename'_'id'_'name'
```

See Also

scribo::make::internal::debug_filename_prefix

7.8.2.2 template<typename I , typename N , typename V > mln::util::graph scribo::make::influence_zone_graph (const Image< I > & *input_,* const Neighborhood< N > & *nbh_,* const V & *label_type,* unsigned *iz_dmax*)

Compute a labeled image of input, then compute an influence zone image and make a graph from it.

Parameters

in	*input_*	a binary image.
in	*nbh_*	a neighbordhood.
in	*label_type*	The type of this argument is used as label type while labeling the image.
in	*iz_dmax*	Max distance of the influence zone.

Returns

a region adjacency graph.

7.8.2.3 template<typename L > scribo::paragraph_set<L> scribo::make::paragraph (const line_links< L > & *llinks,* const line_links< L > & *rlinks*)

Construct a paragraph set from line links information.

7.8.2.4 template<typename L > scribo::paragraph_set<L> scribo::make::paragraph (const scribo::line_set< L > & *lines*)

Construct a paragraph set from line set information.

7.8.2.5 template<typename L > scribo::paragraph_set<L> scribo::make::paragraph (const line_links< L > & *llinks*)

Construct a paragraph set from line links information.

7.8.2.6 template< typename L > mln::trait::ch_value< L , bool >::ret scribo::make::text_blocks_image (const document< L > & *doc*, unsigned *min_nlines*)

Create a mask of paragraph blocks.

Precondition

`doc` has_text() methods MUST return True.

7.8.2.7 template< typename L > mln::trait::ch_value< L , bool >::ret scribo::make::text_components_image (const document< L > & *doc*)

Create a binary image with text components only.

Precondition

`doc` has_text() methods MUST return True.

7.9 scribo::postprocessing Namespace Reference

Functions

- template< typename I >
  mln::trait::concrete< I >::**ret** fill_object_holes (const **Image**< I > &input, **float** ratio)
- template< typename L >
  object_groups< L > **fill_object_holes** (const object_groups< L > &groups, **unsigned** min_size)
- template< typename L >
  void images_to_drop_capital (document< L > &doc)

7.9.1 Detailed Description

Namespace of postprocessing routines.

7.9.2 Function Documentation

7.9.2.1 template< typename I > mln::trait::concrete< I >::ret scribo::postprocessing::fill_object_holes (const Image< I > & *input*, float *ratio*) [inline]

Fill-in object small holes.

7.9.2.2 template< typename L > void scribo::postprocessing::images_to_drop_capital (document< L > & *doc*)

Set type for specific images to Drop Capital component.

Parameters

in	*doc*	A document structure.

7.10 scribo::preprocessing Namespace Reference

Functions

- template<typename I >
  mln::trait::concrete< I >::**ret** crop (const **Image**< I > &input, const mln_box(I)&domain)
- template<typename I >
  mln::trait::concrete< I >::**ret** crop_without_localization (const **Image**< I > &input, const mln_box(I)&domain)
- template<typename I , typename N >
  mln::trait::concrete< I >::**ret** denoise (const **Image**< I > &input, const **Neighborhood**< N > &nbh, **unsigned** fg_min_card, **unsigned** bg_min_card)
- template<typename I , typename N >
  mln::trait::concrete< I >::**ret** denoise_bg (const **Image**< I > &input, const **Neighborhood**< N > &nbh, **unsigned** min_card)
- template<typename I , typename N >
  mln::trait::concrete< I >::**ret** denoise_fg (const **Image**< I > &input, const **Neighborhood**< N > &nbh, **unsigned** min_card)
- template<typename I , typename J >
  mln::trait::concrete< I >::**ret** deskew (const **Image**< I > &crop_gl, const **Image**< I > &input_gl)
- template<typename I >
  mln::trait::concrete< I >::**ret** homogeneous_contrast (const **Image**< I > &input, **unsigned** h)
- template<typename I >
  mln::trait::concrete< I >::**ret** homogeneous_contrast (const **Image**< I > &input)
- template<typename I >
  mln::trait::concrete< I >::**ret** rotate_90 (const **Image**< I > &input, **bool** positive)
- template<typename I >
  mln::trait::concrete< I >::**ret** rotate_90 (const **Image**< I > &input)
- template<typename I >
  mln::util::couple< typename
  mln::trait::concrete< I >::**ret**,
  typename **mln::trait::concrete**
  < I >::**ret** > split_bg_fg (const **Image**< I > &input, **unsigned** lambda, **unsigned delta**)

7.10.1 Detailed Description

Namespace of preprocessing routines.

7.10.2 Function Documentation

7.10.2.1 template<typename I > mln::trait::concrete< I >::ret scribo::preprocessing::crop (const Image< I > & *input,* const mln_box(I)& *domain*)

crop an image preserving the localization.

Parameters

in	*input*	An image.
in	*domain*	A region of interest.

Returns

An image defined on the domain `domain` with the corresponding data copied from `input`.

7.10.2.2 template<typename I > mln::trait::concrete< I >::ret scribo::preprocessing::crop_without_localization (const Image< I > & *input*, const mln_box(I)& *domain*)

crop an image without preserving the localization.

Parameters

in	*input*	An image.
in	*domain*	A region of interest.

Returns

An image defined on a domain starting from the origin (literal::origin) and having the same size as `domain`. Its data corresponds to the data copied from `input` in the domain `domain`.

7.10.2.3 template<typename I , typename N > mln::trait::concrete< I >::ret scribo::preprocessing::denoise (const Image< I > & *input*, const Neighborhood< N > & *nbh*, unsigned *fg_min_card*, unsigned *bg_min_card*)

Denoise an image.

Denoising is performed both on the foreground and the background.

Parameters

in	*input*	A binary image. True for objects, False for background.
in	*nbh*	Neighborhood to use for denoising.
in	*fg_min_card*	Minimum component cardinality to not be considered as noise in the foreground.
in	*bg_min_card*	Minimum component cardinality to not be considered as noise in the foreground.

Returns

A binary image with the same domain as `input`. All small components have been removed.

7.10.2.4 template<typename I , typename N > mln::trait::concrete< I >::ret scribo::preprocessing::denoise_bg (const Image< I > & *input*, const Neighborhood< N > & *nbh*, unsigned *min_card*)

Denoise image background.

Parameters

in	*input*	A binary image. True for objects, False for background.
in	*nbh*	Neighborhood to use for denoising.
in	*min_card*	Minimum component cardinality to not be considered as noise.

Returns

A binary image with the same domain as `input`. All small components have been removed and merged with the background.

7.10.2.5 template<typename I , typename N > mln::trait::concrete< I >::ret scribo::preprocessing::denoise_fg (const Image< I > & *input,* const Neighborhood< N > & *nbh,* unsigned *min_card*)

Denoise image foreground.

Parameters

in	*input*	A binary image. True for objects, False for background.
in	*nbh*	Neighborhood to use for denoising.
in	*min_card*	Minimum component cardinality to not be considered as noise.

Returns

A binary image with the same domain as `input`. All small components have been removed and merged with the background.

7.10.2.6 template<typename I , typename J > mln::trait::concrete< I >::ret scribo::preprocessing::deskew (const Image< I > & *crop_gl,* const Image< I > & *input_gl*)

Deskew a region of interest.

Parameters

in	*crop_gl*	A gray-level image.
in	*input_gl*	A gray-level image.

Returns

A deskewed binary image.

Handles skew angles from -25 to +25 degrees.

`crop_gl` and `input_gl` must be 2D images and must be identical (e.g. only the value differs).

This algorithm is designed for images created from a region of interest (e.g. Not a full document).

7.10.2.7 template<typename I > mln::trait::concrete< I >::ret scribo::preprocessing::homogeneous_contrast (const Image< I > & *input,* unsigned *h*)

Improve contrast homogeneity in an image.

Parameters

in	*input*	A gray-level image.
in	*h*	Height attribute value for leveling closing.

Returns

A gray-level image with better contrast homogeneity.

7.10.2.8 template<typename I > mln::trait::concrete< I >::ret scribo::preprocessing::homogeneous_contrast (const Image< I > & *input*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.Default height attribute value is set to 75.

7.10.2.9 template<typename I > mln::trait::concrete< I >::ret scribo::preprocessing::rotate_90 (const Image< I > & *input,* bool *positive*)

Perform a +90/-90 degree rotation.

Parameters

in	*input*	An image.
in	*positive*	If set to true, performs a +90 degree rotation, -90 degree otherwise.

Returns

A rotated image.

7.10.2.10 template<typename I > mln::trait::concrete< I >::ret scribo::preprocessing::rotate_90 (const Image< I > & *input*)

Performs a +90 degree rotation.

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.

7.10.2.11 template<typename I > mln::util::couple<typename mln::trait::concrete< I >::ret, typename mln::trait::concrete< I >::ret> scribo::preprocessing::split_bg_fg (const Image< I > & *input,* unsigned *lambda,* unsigned *delta*)

Split the background and the foreground.

Parameters

in	*input*	A color image.
in	*lambda*	Lambda used for morphological closing/opening.
in	*delta*	Max distance between values in closing and opening image.

Returns

A couple of color images. The first is the background and the second is the foreground.

7.11 scribo::primitive Namespace Reference

Namespaces

- namespace extract
- namespace group
- namespace internal
- namespace link

Functions

- template<typename L >
  component_set< L > **identify** (const component_set< L > comps)

7.11.1 Detailed Description

Namespace of primitive related routines.

7.12 scribo::primitive::extract Namespace Reference

Functions

- template<typename L >
  mln::util::couple
  < component_set< L >, typename
  mln::trait::ch_value< L, **bool** >
  ::**ret** > alignments (const document< L > &doc, **float** dmax_ratio, **unsigned** delta_pixel)
- template<typename I , typename V >
  mln::trait::ch_value< I, **bool** >
  ::**ret** canvas (const **Image**< I > &input_, const **mln::util::array**< **box**< typename I::site > > &hlines_, const **mln::util::array**< **box**< typename I::site > > &vlines_, **unsigned** max_dist_lines)
- template<typename I , typename N , typename V >
  mln::util::couple
  < **mln::util::array**< **box**
  < typename I::site >
  >, **mln::util::array**< **box**
  < typename I::site > > > cells (const **Image**< I > &input, const **Neighborhood**< N > &nbh, const V &label_type)
- template<typename I , typename J , typename N , typename V >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > components (const **Image**< I > &input, const **Image**< J > &binary_input, const **Neighborhood**< N > &nbh, V &ncomponents, component::Type type=component::Undefined)
- template<typename I , typename N , typename V >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > components (const **Image**< I > &binary_input, const **Neighborhood**< N > &nbh, V &ncomponents, component::Type type=component::Undefined)
- template<typename I >
  mln::trait::concrete< I >::**ret** horizontal_separators (const **Image**< I > &input, **unsigned** line_length)
- template<typename I , typename N , typename V , typename W >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > lines_discontinued (const **Image**< I > &input_, const **Neighborhood**< N > &nbh_, V &nlines, const **Window**< W > &win_, **unsigned** rank_k)
- template<typename I , typename N , typename V >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > lines_h_discontinued (const **Image**< I > &input, const **Neighborhood**< N > &nbh, V &nlines, **unsigned** line_length, **unsigned** rank_k)
- template<typename I >
  mln::trait::concrete< I >::**ret** lines_h_pattern (const **Image**< I > &input, **unsigned** length, **unsigned delta**)
- template<typename I , typename N , typename V >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > lines_h_single (const **Image**< I > &input, const **Neighborhood**< N > &nbh, const V &nlines, **unsigned** min_line_length, **float** w_h_ratio)
- template<typename L >
  component_set< L > lines_h_single (const component_set< L > &components, **unsigned** min_line_length, **float** w_h_ratio)

- template<typename I , typename N , typename V >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > lines_h_thick_and_single (const **Image**< I > &input, const **Neighborhood**< N > &nbh, V &nlines, **unsigned** min_line_length, **float** h_w_ratio)
- template<typename I >
  mln::trait::concrete< I >::**ret** lines_h_thick_and_thin (const **Image**< I > &binary_image, **unsigned** length, **unsigned delta**, **float** p_few=0.2, **float** p_enough=0.6, **float** ratio=8)
- template<typename I , typename W >
  mln::trait::concrete< I >::**ret** lines_pattern (const **Image**< I > &input_, **unsigned** length, **unsigned** dir, const **Window**< W > &win_)
- template<typename I , typename N , typename V , typename W >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > lines_thick (const **Image**< I > &input_, const **Neighborhood**< N > &nbh_, V &nlines, **unsigned** line_length)
- template<typename I , typename N , typename V , typename W >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > lines_thick (const **Image**< I > &input_, const **Neighborhood**< N > &nbh_, V &nlines, const **Window**< W > &win_)
- template<typename I , typename N , typename V >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > lines_v_discontinued (const **Image**< I > &input, const **Neighborhood**< N > &nbh, V &nlines, **unsigned** line_length, **unsigned** rank_k)
- template<typename I >
  mln::trait::concrete< I >::**ret** lines_v_pattern (const **Image**< I > &input, **unsigned** length, **unsigned delta**)
- template<typename I , typename N , typename V >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > lines_v_single (const **Image**< I > &input, const **Neighborhood**< N > &nbh, const V &nlines, **unsigned** min_line_length, **float** h_w_ratio)
- template<typename L >
  component_set< L > lines_v_single (const component_set< L > &components, **unsigned** min_line_length, **float** h_w_ratio)
- template<typename I , typename N , typename V >
  component_set< typename
  mln::trait::ch_value< I, V >
  ::**ret** > lines_v_thick_and_single (const **Image**< I > &input, const **Neighborhood**< N > &nbh, V &nlines, **unsigned** min_line_length, **float** h_w_ratio)
- template<typename L >
  component_set< L > non_text (const document< L > &doc, **unsigned** nlines)
- template<typename L >
  component_set< L > non_text_hdoc (const document< L > &doc, **unsigned** closing_size)
- template<typename L , typename I >
  component_set< L > **non_text_kmean** (const document< L > &doc, const **Image**< I > &input)
- template<typename I >
  mln::trait::concrete< I >::**ret** separators (const **Image**< I > &input, **unsigned** line_length)
- template<typename I >
  mln::trait::concrete< I >::**ret** separators_nonvisible (const **Image**< I > &in_)
- template<typename I >
  mln::trait::concrete< I >::**ret** vertical_separators (const **Image**< I > &input, **unsigned** line_length)

7.12.1 Detailed Description

Namespace of primitive extraction related routines.

7.12.2 Function Documentation

7.12.2.1 template<typename L > mln::util::couple<component_set<L>, typename mln::trait::ch_value< L , bool >::ret> scribo::primitive::extract::alignments (const document< L > & *doc*, float *dmax_ratio*, unsigned *delta_pixel*)

Find page delimitors from tabstops and whitespaces.

Precondition

Separators should be removed from input document image .
Text in `doc` must be constructed from components grouped by lines with a very strict criterion in order to keep spaces between words and paragraphs. This first grouping is necessary to avoid false positive (e.g. inside the text blocks).
`doc` must have text (doc.has_text() returns true).

Internal description:

1) Build an image of line bboxes 2) For TOP and BOTTOM 2.a) Link bboxes. Links are validated only if :

- Alignment difference is less than `delta_pixel`.
- Bboxes are not too far
- No component is located at 5 pixels along the aligned side. 2.b) Invalidates groups if there are less than 3 links 2.c) Invalidates groups if a component is located at a specific distance from the aligned side.

This method handles skewed alignments and draw skew lines if possible. Examples :

```
|x              |x
| x              \x
|  x      ->      \x
|   x              \x
```

```
|x
| x
|  x
| x
|x
```

Here, the 'x' are aligned pair by pair but globally they are not. Here we cannot draw skewed lines without processing every links and look for that pattern. More over, in step 2.c alignment is validated by looking for other components in the supposed "whitespace area" from the aligned side. With such an alignment there are more chances that a component intersect with that line. Here, we would like to split links/alignement in two groups in order to get something like that :

```
\x
 \x
  \x
 /x
/x
```

Parameters

`in`	*doc*	A document information with text lines.
`in`	*dmax_ratio*	The ratio used to compute the maximum lookup distance while linking up components.
`in`	*delta_pixel*	The maximum number of pixels allowed for alignment delta precision.

7.12.2.2 template< typename I , typename V > mln::trait::ch_value< I , bool >::ret scribo::primitive::extract::canvas (const Image< I > & *input_*, const mln::util::array< box< typename I::site > > & *hlines_*, const mln::util::array< box< typename I::site > > & *vlines_*, unsigned *max_dist_lines*)

Rebuild a table from its line bounding boxes.

Parameters

in	*input_*	A binary image.
in	*hlines_*	Vorizontal line bounding boxes.
in	*vlines_*	vertical line bounding boxes.
in	*max_dist_lines*	The maximum distance allowed between vertical and horizontal lines to connect them eachother.

Returns

The canvas as a binary image. canvas lines are set to true.

7.12.2.3 template< typename I , typename N , typename V > mln::util::couple<mln::util::array<box<typename I ::site> >, mln::util::array<box<typename I ::site> > > scribo::primitive::extract::cells (const Image< I > & *input*, const Neighborhood< N > & *nbh*, const V & *label_type*)

Extract canvas cells from a binary image.

Use arbitrary criterions.

Parameters

in	*input*	A binary image.
in	*nbh*	A neighborhood.
in,out	*label_type*	Type of the labeled image.

Returns

A list of cell bounding boxes.

7.12.2.4 template< typename L > component_set<L> scribo::primitive::extract::non_text (const document< L > & *doc*, unsigned *nlines*)

Extract non text components.

This method takes text localization into account and tries to learn the background colors to deduce the relevant non text components.

Parameters

in	*doc*	A document structure. Its must have paragraph information.
in	*nlines*	The number of lines needed in a paragraph to consider the latter during the background color learning.

Returns

A component set of non text components.

7.12.2.5 template<typename L > component_set<L> scribo::primitive::extract::non_text_hdoc (const document< L > & *doc,* unsigned *closing_size*)

Extract non text components.

Variant adapted for historical documents.

7.13 scribo::primitive::group Namespace Reference

Functions

- template<typename L >
  component_set< L > apply (const object_groups< L > &groups)
- template<typename L >
  component_set< L > apply (const object_links< L > &links)
- template<typename L >
  object_groups< L > from_double_link (const object_links< L > &left_link, const object_links< L > &right_link)
- template<typename L >
  object_groups< L > from_double_link_any (const object_links< L > &left_link, const object_links< L > &right_link)
- template<typename L , typename G >
  object_groups< L > from_graph (const component_set< L > &comps, const **Graph**< G > &g_)
- template<typename L >
  object_groups< L > from_single_link (const object_links< L > &links)
- template<typename L >
  object_groups< L > **regroup_left** (const component_set< L > &components, const object_groups< L > &groups, **unsigned** dmax)

7.13.1 Detailed Description

Namespace of primitive grouping related routines.

7.13.2 Function Documentation

7.13.2.1 template<typename L > component_set<L> scribo::primitive::group::apply (const object_groups< L > & *groups*)

Apply grouping in an object image.

Parameters

groups	An object group structure.

Returns

A copy of `components` with grouped components.

7.13.2.2 template<typename L > component_set<L> scribo::primitive::group::apply (const object_links< L > & *links*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.

7.13.2.3 template< typename L > object_groups<L> scribo::primitive::group::from_double_link (const object_links< L > & *left_link,* const object_links< L > & *right_link*)

Group components from left and right links information and validate These links. A link must exist in both ways to be validated.

Parameters

in	*left_link*	The left neighbor of each line of text.
in	*right_link*	The right neighbor of each line of text.

Returns

Return object groups information.

7.13.2.4 template< typename L > object_groups<L> scribo::primitive::group::from_double_link_any (const object_links< L > & *left_link,* const object_links< L > & *right_link*)

Group components from left and right links information and validate These links.

Parameters

in	*left_link*	The left neighbor of each line of text.
in	*right_link*	The right neighbor of each line of text.

Returns

Return object groups information.

7.13.2.5 template< typename L , typename G > object_groups<L> scribo::primitive::group::from_graph (const component_set< L > & *comps,* const Graph< G > & *g_*)

Group objects according to a graph of links.

Parameters

in	*comps*	A component set.
in	*g_*	The graph of object links.

Returns

Object groups information.

7.13.2.6 template< typename L > object_groups<L> scribo::primitive::group::from_single_link (const object_links< L > & *links*)

Link text components with their neighbor line if they have one.

Parameters

in	*links*	The neighbor line of each line.

Returns

Object groups information.

7.14 scribo::primitive::internal Namespace Reference

Functions

- template<typename G , typename L >
  void find_graph_link (**Graph**< G > &g_, const component_set< L > &comps, **unsigned** current_comp, **int** dmax, const typename L::site &c)
- template<typename F >
  mln::util::couple< **bool**,
  typename scribo_support_(F)::**site** > find_link (Link_Functor< F > &functor, **unsigned** current_object, anchor::Type anchor)
- template<typename L >
  unsigned find_root (object_groups< L > &parent, **unsigned** x)
- **bool** have_link_valid (const **mln::util::array**< **unsigned** > &left_link, const **mln::util::array**< **unsigned** > &right_link, **unsigned** i)
- template<typename L >
  bool is_link_valid (const object_links< L > &left_link, const object_links< L > &right_link, **unsigned** i)
- template<typename I , typename J >
  mln::trait::concrete< I >::**ret** rd (const **Image**< I > &ima, const **Image**< J > &constraint)
- template<typename I , typename G >
  void update_graph_link (const **Image**< I > &lbl_, **Graph**< G > &g_, const typename I::site &p, const typename I::site &c, **unsigned** i, **int** dmax)

7.14.1 Detailed Description

Namespace of internal routines grouping text components.

7.14.2 Function Documentation

7.14.2.1 template<typename G , typename L > void scribo::primitive::internal::find_graph_link (Graph< G > & *g_*, const component_set< L > & *comps*, unsigned *current_comp*, int *dmax*, const typename L::site & *c*)

Find a neighbor of a component in a specific range if it exists.

Parameters

*g_*	The link graph.
comps	A component set.
current_comp	The current line being processed.
dmax	The maximum lookup distance.
c	Start point of the neighbor lookup.

7.14.2.2 template<typename F > mln::util::couple<bool, typename scribo_support_(F) ::site> scribo::primitive::internal::find_link (Link_Functor< F > & *functor*, unsigned *current_object*, anchor::Type *anchor*)

Find the neighbor of a line of text if exists.

Parameters

in,out	*functor*	Functor used to compute the links. Stores the results.
in	*current_object*	Current object id.
in	*anchor*	The lookup anchor.

Returns

A couple. The first argument tells whether a valid link has been found, the second one is link anchor if exists.

7.14.2.3 template< typename L > unsigned scribo::primitive::internal::find_root (object_groups< L > & *parent*, unsigned *x*)

Find root in a parent array arrays.

7.14.2.4 bool scribo::primitive::internal::have_link_valid (const mln::util::array< unsigned > & *left_link*, const mln::util::array< unsigned > & *right_link*, unsigned *i*)

Tells whether a component have at least one valid link link.

Parameters

in	*left_link*	Left link of components.
in	*right_link*	Right link of components.
in	*i*	The component id.

Returns

True if the `i-th` component has at least one valid link.

7.14.2.5 template< typename L > bool scribo::primitive::internal::is_link_valid (const object_links< L > & *left_link*, const object_links< L > & *right_link*, unsigned *i*)

Validate a link from two different links.

Parameters

in	*left_link*	Left link of components.
in	*right_link*	Right link of components.
in	*i*	The component id.

Returns

True if the link is between the `i-th` component

7.14.2.6 template< typename I , typename J > mln::trait::concrete< I >::ret scribo::primitive::internal::rd (const Image< I > & *ima*, const Image< J > & *constraint*)

Tolerant constrained reconstruction algorithm.

7.14.2.7 template< typename I , typename G > void scribo::primitive::internal::update_graph_link (const Image< I > & *lbl_*, Graph< G > & *g_*, const typename I::site & *p*, const typename I::site & *c*, unsigned *i*, int *dmax*)

Update graph edges if a valid neighbor is found.

Parameters

in	*lbl_*	A label image.
in	*g_*	A graph.
in	*p*	A site of `lbl_`.
in	*c*	A site of `lbl_`.
in	*i*	A vertex id.
in	*dmax*	The maximum distance allowed to look for a neighbor.

7.15 scribo::primitive::link Namespace Reference

Functions

- template<typename F >
  object_links< scribo_support(F)> compute (Link_Functor< F > &functor, anchor::Type anchor)
- template<typename F >
  object_links< scribo_support(F)> compute (Link_Functor< F > &functor)
- template<typename F >
  object_links< scribo_support(F)> compute_several (Link_Functor< F > &functor)
- template<typename L >
  object_links< L > **left** (const component_set< L > &components, **unsigned** dmax)
- template<typename L >
  mln::util::couple
  < object_links< L >
  , object_links< L > > **left_right** (const component_set< L > &components)
- template<typename L >
  object_links< L > merge_double_link (const object_links< L > &left_link, const object_links< L > &right_link)
- template<typename L >
  object_links< L > merge_double_link_closest_aligned (const object_links< L > &left, const object_links< L > &right, anchor::Type anchor_angle)
- template<typename L >
  mln::util::graph with_graph (const component_set< L > &comps, **unsigned** neighb_max_distance)
- template<typename L , typename N >
  util::couple< **mln::util::graph**,
  typename **mln::trait::concrete**
  < L >::**ret** > with_rag (const component_set< L > &comps, const **Neighborhood**< N > &nbh)
- template<typename L >
  mln::util::graph with_several_graphes (const component_set< L > &comps, **unsigned** neighb_max_distance)
- template<typename L >
  object_links< L > with_several_left_links (const component_set< L > &objects, **unsigned** neighb_max_distance)
- template<typename L >
  object_links< L > with_several_left_links (const component_set< L > &comps)
- template<typename L >
  object_links< L > with_several_right_closest_links (const component_set< L > &comps, **unsigned** neighb_max_distance)
- template<typename L >
  object_links< L > with_several_right_closest_links (const component_set< L > &comps)
- template<typename L >
  object_links< L > with_several_right_links (const component_set< L > &comps, **unsigned** neighb_max_distance)
- template<typename L >
  object_links< L > with_several_right_links (const component_set< L > &comps)

- template<typename L >
  object_links< L > with_several_right_links_overlap (const object_image(L)&objects, **unsigned** neighb_max_distance)
- template<typename L >
  object_links< L > with_several_right_links_overlap (const object_image(L)&objects)
- template<typename L >
  object_links< L > with_single_down_link (const component_set< L > &comps, **unsigned** neighb_max_distance, anchor::Type anchor)
- template<typename L >
  object_links< L > with_single_down_link (const component_set< L > &comps, **unsigned** neighb_max_distance)
- template<typename L >
  object_links< L > with_single_down_link (const component_set< L > &comps)
- template<typename L >
  object_links< L > with_single_left_link (const component_set< L > &components, **unsigned** neighb_max_distance)
- template<typename L >
  object_links< L > with_single_left_link (const component_set< L > &components)
- template<typename L , typename F >
  object_links< L > with_single_left_link_dmax_ratio (const component_set< L > &components, const DMax_Functor< F > &dmax_f, anchor::Type anchor)
- template<typename L , typename F >
  object_links< L > with_single_left_link_dmax_ratio (const component_set< L > &components, **float** dmax_ratio, anchor::Type anchor)
- template<typename L >
  object_links< L > with_single_left_link_dmax_ratio (const component_set< L > &components, **float** dmax_ratio)
- template<typename L , typename F >
  object_links< L > with_single_left_link_dmax_ratio (const component_set< L > &components)
- template<typename L , typename F >
  object_links< L > with_single_left_link_dmax_ratio_aligned (const component_set< L > &components, const DMax_Functor< F > &dmax_f, **float** min_angle, **float** max_angle, anchor::Type anchor)
- template<typename L >
  object_links< L > with_single_left_link_dmax_ratio_aligned (const component_set< L > &components, **float** dmax_ratio, **float** min_angle, **float** max_angle)
- template<typename L >
  object_links< L > with_single_left_link_dmax_ratio_aligned (const component_set< L > &components)
- template<typename L >
  object_links< L > with_single_right_link (const component_set< L > &components, **unsigned** neighb_max_distance, anchor::Type anchor=anchor::MassCenter)
- template<typename L >
  object_links< L > with_single_right_link (const component_set< L > &components)
- template<typename L >
  object_links< L > with_single_right_link_bottom (const component_set< L > &components, **unsigned** neighb_max_distance)
- template<typename L >
  object_links< L > with_single_right_link_bottom (const component_set< L > &components)
- template<typename L , typename F >
  object_links< L > with_single_right_link_dmax_ratio (const component_set< L > &components, const DMax_Functor< F > &dmax_f, anchor::Type anchor)
- template<typename L >
  object_links< L > with_single_right_link_dmax_ratio (const component_set< L > &components, **float** dmax_ratio, anchor::Type anchor)
- template<typename L >
  object_links< L > with_single_right_link_dmax_ratio (const component_set< L > &components, **float** dmax_ratio)

- template<typename L >
  object_links< L > with_single_right_link_dmax_ratio (const component_set< L > &components)
- template<typename L >
  object_links< L > with_single_right_link_dmax_ratio_aligned (const component_set< L > &components, **float** dmax_ratio, **float** min_angle, **float** max_angle, anchor::Type anchor)
- template<typename L >
  object_links< L > with_single_right_link_dmax_ratio_aligned (const component_set< L > &components, **float** dmax_ratio, **float** min_angle, **float** max_angle)
- template<typename L >
  object_links< L > with_single_right_link_dmax_ratio_aligned (const component_set< L > &components)
- template<typename L >
  object_links< L > with_single_right_link_top (const component_set< L > &components, **unsigned** neighb_max_distance)
- template<typename L >
  object_links< L > with_single_right_link_top (const component_set< L > &components)
- template<typename L >
  object_links< L > with_single_up_link (const component_set< L > &comps, **unsigned** neighb_max_distance, anchor::Type anchor)
- template<typename L >
  object_links< L > with_single_up_link (const component_set< L > &comps, **unsigned** neighb_max_distance)
- template<typename L >
  object_links< L > with_single_up_link (const component_set< L > &comps)

7.15.1 Detailed Description

Namespace of primitive linking related routines.

7.15.2 Function Documentation

7.15.2.1 template<typename F > object_links<scribo_support(F)> scribo::primitive::link::compute (Link_Functor< F > & *functor,* anchor::Type *anchor*)

Compute links between objects according a given functor.

Parameters

in,out	*functor*	Linking policy.
in	*anchor*	Lookup anchor. Starts looking for neighbors from there.

Returns

Object links.

Functors must implement the following interface :

bool is_potential_link_(unsigned current_object, const P& start_point, const P& p) const

bool valid_link_(unsigned current_object, const P& start_point, const P& p)

bool verify_link_criterion_(unsigned current_object, const P& start_point, const P& p)

void validate_link_(unsigned current_object, const P& start_point, const P& p, unsigned anchor)

void invalidate_link_(unsigned current_object, const P& start_point, const P& p, unsigned anchor)

void compute_next_site_(P& p)

const mln_site(L)& start_point_(unsigned current_object, unsigned anchor)

void start_processing_object_(unsigned current_object)

7.15.2.2 template< typename F > object_links< scribo_support(F)> scribo::primitive::link::compute (Link_Functor< F > & *functor*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.The default anchor is set to anchor::MassCenter.

7.15.2.3 template< typename F > object_links< scribo_support(F)> scribo::primitive::link::compute_several (Link_Functor< F > & *functor*)

Compute_Several links between objects according a given functor.

Parameters

`in,out`	*functor*	Linking policy.

Returns

Object links.

Functors must implement the following interface :

bool verify_link_criterion_(unsigned current_object, const P& start_point, const P& p) const;

void start_processing_object_(unsigned current_object);

mln_site(L) start_point_(unsigned current_object, unsigned anchor);

void validate_link_(unsigned current_object, const P& start_point, const P& p, unsigned anchor); void invalidate_-link_(unsigned current_object, const P& start_point, const P& p, unsigned anchor);

void initialize_link_(unsigned current_object); void finalize_link_(unsigned current_object);

bool is_potential_link(unsigned current_object, const P& start_point, const P& p) const

void compute_next_site(P& p)

7.15.2.4 template< typename L > object_links< L> scribo::primitive::link::merge_double_link (const object_links< L > & *left_link,* const object_links< L > & *right_link*)

Validate and merge double link information. A link must exist in both ways to be validated.

Parameters

`in`	*left_link*	The left neighbor of each line of text.
`in`	*right_link*	The right neighbor of each line of text.

Returns

The merge of left_link and right_link.

7.15.2.5 template< typename L > object_links< L> scribo::primitive::link::merge_double_link_closest_aligned (const object_links< L > & *left,* const object_links< L > & *right,* anchor::Type *anchor_angle*)

Merge two object links data based on distance and angle.

Performs an 'OR' operation on the links according distance and angle criterion.

If a component has several incoming links, only the link for which the the object is the closest or the angle performed between the two bboxes is the lower, will be preserved.

7.15.2.6 template<typename L > mln::util::graph scribo::primitive::link::with_graph (const component_set< L > & *comps,* unsigned *neighb_max_distance*)

Construct the links between each line of text and store it as a graph. Look up for neighbors on the right of each box.

Parameters

`in`	*comps*	A component set.
`in`	*neighb_max_distance*	The maximum distance allowed to look for a neighbor.

Returns

A graph of relationship.

7.15.2.7 template<typename L , typename N > util::couple<mln::util::graph, typename mln::trait::concrete< L >::ret> scribo::primitive::link::with_rag (const component_set< L > & *comps,* const Neighborhood< N > & *nbh*)

Link components with a region adjacency graph.

7.15.2.8 template<typename L > mln::util::graph scribo::primitive::link::with_several_graphes (const component_set< L > & *comps,* unsigned *neighb_max_distance*)

Link character bounding boxes with several graphes.

Look up for neighbors on the left of each box.

7.15.2.9 template<typename L > object_links<L> scribo::primitive::link::with_several_left_links (const component_set< L > & *objects,* unsigned *neighb_max_distance*) [inline]

Map each character bounding box to its left bounding box neighbor if possible.

Iterate to the right but link boxes to the left.

Returns

object links data.

7.15.2.10 template<typename L > object_links<L> scribo::primitive::link::with_several_left_links (const component_set< L > & *comps*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.

7.15.2.11 template<typename L > object_links<L> scribo::primitive::link::with_several_right_closest_links (const component_set< L > & *comps,* unsigned *neighb_max_distance*) [inline]

Map each character bounding box to its right bounding box neighbor if possible.

If there are several right neighbor, the closest one is chosen.

Returns

an **mln::util::array**. Map a bounding box to its right neighbor.

7.15.2.12 template<typename L > **object_links**<L> scribo::primitive::link::with_several_right_closest_links (const component_set< L > & *comps*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.

7.15.2.13 template<typename L > **object_links**<L> scribo::primitive::link::with_several_right_links (const component_set< L > & *comps*, unsigned *neighb_max_distance*) [inline]

Map each character bounding box to its right bounding box neighbor if possible.

Iterate to the right but link boxes to the right.

Returns

Object links.

7.15.2.14 template<typename L > **object_links**<L> scribo::primitive::link::with_several_right_links (const component_set< L > & *comps*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.

7.15.2.15 template<typename L > **object_links**<L> scribo::primitive::link::with_several_right_links_overlap (const object_image(L)& *objects*, unsigned *neighb_max_distance*) [inline]

Map each character bounding box to its right bounding box neighbor if possible.

Iterate to the right but link boxes to the right.

Returns

an **mln::util::array**. Map a bounding box to its right neighbor.

7.15.2.16 template<typename L > **object_links**<L> scribo::primitive::link::with_several_right_links_overlap (const object_image(L)& *objects*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.

7.15.2.17 template<typename L > **object_links**<L> scribo::primitive::link::with_single_down_link (const component_set< L > & *comps*, unsigned *neighb_max_distance*, anchor::Type *anchor*) [inline]

Link objects with their down neighbor if exists.

Parameters

in	*comps*	A component set.
in	*neighb_max_-distance*	The maximum distance allowed to seach a neighbor object.
in	*anchor*	The neighborhod lookup start point.

Returns

Object links data.

7.15.2.18 template<typename L > object_links<L> scribo::primitive::link::with_single_down_link (const component_set< L > & *comps*, unsigned *neighb_max_distance*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.Anchor type is set to anchor::MassCenter.

7.15.2.19 template<typename L > object_links<L> scribo::primitive::link::with_single_down_link (const component_set< L > & *comps*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.Max distance is set to mln_max(unsigned).

7.15.2.20 template<typename L > object_links<L> scribo::primitive::link::with_single_left_link (const component_set< L > & *components*, unsigned *neighb_max_distance*) [inline]

Link components with their left neighbor if exists.

Parameters

in	*components*	A component set.
in	*neighb_max_-distance*	The maximum distance allowed to seach a neighbor object.

Returns

Object links data.

7.15.2.21 template<typename L > object_links<L> scribo::primitive::link::with_single_left_link (const component_set< L > & *components*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.Max distance is set to mln_max(unsigned).

7.15.2.22 template<typename L , typename F > object_links<L> scribo::primitive::link::with_single_left_link_dmax_ratio (const component_set< L > & *components*, const DMax_Functor< F > & *dmax_f*, anchor::Type *anchor*) [inline]

Link components with their left neighbor if exists.

Parameters

in	*components*	A component set.
in	*anchor*	Starting point for the neighbor lookup.
in	*dmax_f*	DMax functor defining the maximum lookup distance.

Returns

Object links data.

Look for a neighbor until a maximum distance is reached. The maximum distance is defined thanks to a functor `dmax_f`.

7.15.2.23 template<typename L , typename F > object_links<L> scribo::primitive::link::with_single_left_link_dmax_ratio (const component_set< L > & *components,* float *dmax_ratio,* anchor::Type *anchor*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts. The default dmax functor is used (internal::dmax_default).

7.15.2.24 template<typename L > object_links<L> scribo::primitive::link::with_single_left_link_dmax_ratio (const component_set< L > & *components,* float *dmax_ratio*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.anchor is set to anchor::MassCenter.

7.15.2.25 template<typename L , typename F > object_links<L> scribo::primitive::link::with_single_left_link_dmax_ratio (const component_set< L > & *components*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts. The default dmax functor is used (internal::dmax_default) with dmax_ratio set to 3.

anchor is set to anchor::MassCenter.

7.15.2.26 template<typename L , typename F > object_links<L> scribo::primitive::link::with_single_left_link_dmax_ratio_aligned (const component_set< L > & *components,* const DMax_Functor< F > & *dmax_f,* float *min_angle,* float *max_angle,* anchor::Type *anchor*) [inline]

Link objects with their left neighbor if exists.

Parameters

in	*components*	A component set.
in	*dmax_f*	A function defining the maximum lookup distance.
in	*min_angle*	Minimum difference allowed for alignement angle.
in	*max_angle*	Maximum difference allowed for alignement angle.
in	*anchor*	Starting point for the neighbor lookup.

Returns

Object links data.

Look for a neighbor until a maximum distance defined by :

dmax = w / 2 + dmax_ratio $*$ max(h, w)

where w is the bounding box width and h the bounding box height.

7.15.2.27 template<typename L > object_links<L> scribo::primitive::link::with_single_left_link_dmax_ratio_aligned (const component_set< L > & *components,* float *dmax_ratio,* float *min_angle,* float *max_angle*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.anchor is set to MassCenter.

dmax_f functor is set to internal::dmax_default.

7.15.2.28 template<typename L > object_links<L> scribo::primitive::link::with_single_left_link_dmax_ratio_aligned (const component_set< L > & *components*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.dmax_ratio is set to 3.

anchor is set to MassCenter.

7.15.2.29 template<typename L > object_links<L> scribo::primitive::link::with_single_right_link (const component_set< L > & *components*, unsigned *neighb_max_distance*, anchor::Type *anchor* = anchor::MassCenter) [inline]

Link components with their right neighbor if exists.

Lookup startup point is the object mass center.

Parameters

in	*components*	An object image.
in	*neighb_max_-distance*	The maximum distance allowed to seach a neighbor object.
in	*anchor*	Anchor from where the neighbor lookup is performed.

Returns

Object links data.

7.15.2.30 template<typename L > object_links<L> scribo::primitive::link::with_single_right_link (const component_set< L > & *components*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.Max distance is set to mln_max(unsigned).

7.15.2.31 template<typename L > object_links<L> scribo::primitive::link::with_single_right_link_bottom (const component_set< L > & *components*, unsigned *neighb_max_distance*) [inline]

Link components with their right neighbor if exists.

Lookup startup point is the object bottom center.

Parameters

in	*components*	A component set.
in	*neighb_max_-distance*	The maximum distance allowed to seach a neighbor object.

Returns

Object links data.

7.15.2.32 template<typename L > object_links<L> scribo::primitive::link::with_single_right_link_bottom (const component_set< L > & *components*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.Max distance is set to mln_max(unsigned).

7.15.2.33 template<typename L , typename F > object_links<L> scribo::primitive::link::with_single_right_link_dmax_ratio (const component_set< L > & *components,* const DMax_Functor< F > & *dmax_f,* anchor::Type *anchor*) [inline]

Link objects with their right neighbor if exists.

Parameters

in	*components*	A component set.
in	*anchor*	Starting point for the neighbor lookup.
in	*dmax_f*	DMax functor defining the maximum lookup distance.

Returns

Object links data.

Look for a neighbor until a maximum distance is reached. The maximum distance is defined thanks to a functor `dmax_f`.

7.15.2.34 template<typename L > object_links<L> scribo::primitive::link::with_single_right_link_dmax_ratio (const component_set< L > & *components,* float *dmax_ratio,* anchor::Type *anchor*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.`dmax_f` is set to internal::dmax_default.

7.15.2.35 template<typename L > object_links<L> scribo::primitive::link::with_single_right_link_dmax_ratio (const component_set< L > & *components,* float *dmax_ratio*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.anchor is set to MassCenter.

7.15.2.36 template<typename L > object_links<L> scribo::primitive::link::with_single_right_link_dmax_ratio (const component_set< L > & *components*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.dmax_ratio is set to 3.

anchor is set to MassCenter.

7.15.2.37 template<typename L > object_links<L> scribo::primitive::link::with_single_right_link_dmax_ratio_aligned (const component_set< L > & *components,* float *dmax_ratio,* float *min_angle,* float *max_angle,* anchor::Type *anchor*) [inline]

Link objects with their right neighbor if exists.

Parameters

in	*components*	A component set.
in	*dmax_ratio*	Size ratio defining the maximum lookup distance.
in	*min_angle*	Minimum difference allowed for alignement angle.
in	*max_angle*	Maximum difference allowed for alignement angle.
in	*anchor*	Starting point for the neighbor lookup.

Returns

Object links data.

Look for a neighbor until a maximum distance defined by :

dmax = w / 2 + dmax_ratio ∗ max(h, w)

where w is the bounding box width and h the bounding box height.

7.15.2.38 template<typename L > object_links<L> scribo::primitive::link::with_single_right_link_dmax_ratio_aligned (const component_set< L > & *components,* float *dmax_ratio,* float *min_angle,* float *max_angle*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.anchor is set to MassCenter.

7.15.2.39 template<typename L > object_links<L> scribo::primitive::link::with_single_right_link_dmax_ratio_aligned (const component_set< L > & *components*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.dmax_ratio is set to 3.

anchor is set to MassCenter.

7.15.2.40 template<typename L > object_links<L> scribo::primitive::link::with_single_right_link_top (const component_set< L > & *components,* unsigned *neighb_max_distance*) [inline]

Link components with their right neighbor if exists.

Lookup startup point is the object top center.

Parameters

in	*components*	A component set.
in	*neighb_max_-distance*	The maximum distance allowed to seach a neighbor object.

Returns

Object links data.

7.15.2.41 template<typename L > object_links<L> scribo::primitive::link::with_single_right_link_top (const component_set< L > & *components*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.Max distance is set to mln_max(unsigned).

7.15.2.42 template<typename L > object_links<L> scribo::primitive::link::with_single_up_link (const component_set< L > & *comps,* unsigned *neighb_max_distance,* anchor::Type *anchor*) [inline]

Link components with their up neighbor if exists.

Parameters

in	*comps*	A component set.
in	*neighb_max_-distance*	The maximum distance allowed to seach a neighbor object.
in	*anchor*	The neighborhod lookup start point.

Returns

Object links data.

7.15.2.43 template<typename L > object_links<L> scribo::primitive::link::with_single_up_link (const component_set< L > & *comps*, unsigned *neighb_max_distance*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.Anchor type is set to anchor::Center.

7.15.2.44 template<typename L > object_links<L> scribo::primitive::link::with_single_up_link (const component_set< L > & *comps*) [inline]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.Max distance is set to mln_max(unsigned).

7.16 scribo::table Namespace Reference

Namespaces

- namespace internal

Functions

- template<typename I , typename L >
  mln::util::array< **int** > align_lines_horizontaly (const **Image**< I > &input, const component_set< L > &lines, component_set< L > &aligned_lines, **unsigned** max_alignment_diff)
- template<typename I >
  mln::util::array< **int** > align_lines_verticaly (const **Image**< I > &input, **mln::util::array**< **box**< typename I::site > > &line_bboxes, **unsigned** max_alignment_diff)
- template<typename I >
  void connect_horizontal_lines (const **mln::util::array**< **int** > &aligned_cols, **mln::util::couple**< **mln::util::array**< **box**< typename I::site > >, **mln::util::array**< **box**< typename I::site > > > &tablebboxes, const **Image**< I > &input, **unsigned** max_distance)
- template<typename I >
  void connect_vertical_lines (const **mln::util::array**< **int** > &aligned_rows, **mln::util::couple**< **mln::util::array**< **box**< typename I::site > >, **mln::util::array**< **box**< typename I::site > > > &tablebboxes, const **Image**< I > &input, **unsigned** max_distance)
- template<typename I , typename L >
  mln::trait::concrete< I >::**ret** erase (const **Image**< I > &input, const component_set< L > &hlines, const component_set< L > &vlines)
- template<typename I , typename V >
  mln::util::couple< typename
  mln::trait::ch_value< I, V >
  ::**ret**, **mln::util::couple**
  < **mln::util::array**< **box**
  < typename I::site >
  >, **mln::util::array**< **box**
  < typename I::site > > > > extract (const **Image**< I > &input_, V &ncells)
- template<typename I , typename L >
  mln::util::couple< L,
  mln::util::couple
  < component_set< L >
  , component_set< L > > > rebuild (const **Image**< I > &input, const component_set< L > &vlines, const component_set< L > &hlines, **unsigned** max_dist_lines, typename **L::value** &ncells)

- template<typename I >
  void repair_horizontal_lines (const **Image**< I > &input, **mln::util::couple**< **mln::util::array**< **box**< typename I::site > >, **mln::util::array**< **box**< typename I::site > > > &tablebboxes, **unsigned** max_discontinuity)
- template<typename I >
  void repair_vertical_lines (const **Image**< I > &input, **mln::util::couple**< **mln::util::array**< **box**< typename I::site > >, **mln::util::array**< **box**< typename I::site > > > &tablebboxes, **unsigned** max_discontinuity)

7.16.1 Detailed Description

Namespace of routines working on tables.

7.16.2 Function Documentation

7.16.2.1 template<typename I , typename L > mln::util::array<int> scribo::table::align_lines_horizontaly (const Image< I > & *input,* const component_set< L > & *lines,* component_set< L > & *aligned_lines,* unsigned *max_alignment_diff*)

Align line bounding boxes horizontaly.

Parameters

in	*input*	Image from which the line bboxes are extracted from.
in	*lines*	Component set corresponding to table lines.
in,out	*aligned_lines*	Component set where aligned table lines are stored.
in	*max_alignment-_diff*	max space between two lines to consider they are potentialy on the same line.

Returns

A list of the resulting aligned rows. Each integer is actually a row number.

7.16.2.2 template<typename I > mln::util::array<int> scribo::table::align_lines_verticaly (const Image< I > & *input,* mln::util::array< box< typename I::site > > & *line_bboxes,* unsigned *max_alignment_diff*)

Align line bounding boxes verticaly.

Parameters

in	*input*	Image from which the line bboxes are extracted from.
in,out	*line_bboxes*	vertical lines bounding boxes.
in	*max_alignment-_diff*	max space between two lines to consider they are potentialy on the same line.

Returns

A list of the resulting aligned cols. Each integer is actually a col number.

7.16.2.3 template<typename I > void scribo::table::connect_horizontal_lines (const mln::util::array< int > & *aligned_cols,* mln::util::couple< mln::util::array< box< typename I::site > >, mln::util::array< box< typename I::site > > > & *tablebboxes,* const Image< I > & *input,* unsigned *max_distance*)

Connect horizontal lines with the new aligned columns.

Parameters

in	*aligned_cols*	a list of new aligned cols.
in,out	*tablebboxes*	the vertical and horizontal lines bounding boxes.
in	*input*	The image from where the lines are extracted.
in	*max_distance*	max distance allowed between a vertical and horizontal lines.

7.16.2.4 template< typename I > void scribo::table::connect_vertical_lines (const mln::util::array< int > & *aligned_rows,* mln::util::couple< mln::util::array< box< typename I::site > >, mln::util::array< box< typename I::site > > > & *tablebboxes,* const Image< I > & *input,* unsigned *max_distance*)

Connect vertical lines with the new aligned rows.

Parameters

in	*aligned_rows*	a list of new aligned rows.
in,out	*tablebboxes*	the vertical and horizontal lines bounding boxes.
in	*input*	The image from where the lines are extracted.
in	*max_distance*	max distance allowed between a vertical and horizontal lines.

7.16.2.5 template< typename I , typename L > mln::trait::concrete< I >::ret scribo::table::erase (const Image< I > & *input,* const component_set< L > & *hlines,* const component_set< L > & *vlines*)

Erase vertical and horizontal lines from an image.

Parameters

in	*input*	A binary image from which the table lines are extracted.
in	*hlines*	A component set with horizontal lines.
in	*vlines*	A component set with vertical lines.

Returns

A copy of `in` where the table lines are removed.

7.16.2.6 template< typename I , typename V > mln::util::couple< typename mln::trait::ch_value< I , V >::ret, mln::util::couple< mln::util::array< box< typename I ::site> >, mln::util::array< box< typename I ::site> > > > scribo::table::extract (const Image< I > & *input_,* V & *ncells*)

Extract tables from a binary image.

Use arbitrary criterions.

7.16.2.7 template< typename I , typename L > mln::util::couple< L, mln::util::couple< component_set< L>, component_set< L> > > scribo::table::rebuild (const Image< I > & *input,* const component_set< L > & *vlines,* const component_set< L > & *hlines,* unsigned *max_dist_lines,* typename L::value & *ncells*)

Rebuild a table from its line bounding boxes.

Parameters

in	*input*	A binary image.
in	*vlines*	Component set corresponding to vertical lines.
in	*hlines*	Component set corresponding to horizontal lines.
in	*max_dist_lines*	The maximum distance allowed between vertical and horizontal lines to connect them eachother.
out	*ncells*	Store the number of cells found in the rebuilt tables.

Returns

A couple. The first argument is a label image in which each table cell is labeled. The second argument are the aligned and connected table line bounding boxes.

7.16.2.8 template< typename I > void scribo::table::repair_horizontal_lines (const Image< I > & *input,* mln::util::couple< mln::util::array< box< typename I::site > >, mln::util::array< box< typename I::site > > > & *tablebboxes,* unsigned *max_discontinuity*)

Repair horizontal lines which have small discontinuities.

-— -— ==> ------—

Parameters

in	*input*	Image from which the table bounding boxes are extracted.
in,out	*tablebboxes*	Table line bounding boxes.
in	*max_-discontinuity*	Repair discontinuity which are smaller than this value.

7.16.2.9 template< typename I > void scribo::table::repair_vertical_lines (const Image< I > & *input,* mln::util::couple< mln::util::array< box< typename I::site > >, mln::util::array< box< typename I::site > > > & *tablebboxes,* unsigned *max_discontinuity*)

Repair vertical lines which have small discontinuities.

| |

| |

==> |

| |

| |

Parameters

in	*input*	Image from which the table bounding boxes are extracted.
in,out	*tablebboxes*	Table line bounding boxes.
in	*max_-discontinuity*	Repair discontinuity which are smaller than this value.

7.17 scribo::table::internal Namespace Reference

Functions

- template< typename L >
  mln::util::array< **int** > align_lines (**unsigned nsites**, **int** min_coord, **int** max_coord, const component_set< L > &lines, component_set< L > &aligned_lines, **unsigned** dim, **unsigned** max_alignment_diff)
- template< typename P >
  void connect_lines (const **mln::util::array**< **int** > &aligned_lines, **mln::util::array**< **box**< P > > &boxes, **unsigned** dim, **unsigned** dim_size, **unsigned** max_distance)
- template< unsigned axis, typename I >
  void repair_lines (const **Image**< I > &input, **mln::util::array**< **box**< typename I::site > > &tableboxes, **unsigned** max_discontinuity)

7.17.1 Detailed Description

Namespace of internal routines working on tables.

7.17.2 Function Documentation

7.17.2.1 template<typename L > mln::util::array<int> scribo::table::internal::align_lines (unsigned *nsites,* int *min_coord,* int *max_coord,* const component_set< L > & *lines,* component_set< L > & *aligned_lines,* unsigned *dim,* unsigned *max_alignment_diff*)

Align table lines bboxes according to a given dimension.

Parameters

`in`	*nsites*	Number of sites in the given dimension `dim`.
`in`	*min_coord*	The minimal coordinate in the dimension `dim`.
`in`	*max_coord*	The maximal coordinate in the dimension `dim`.
`in`	*lines*	The line components.
`out`	*aligned_lines*	The components of the aligned lines.
`in`	*dim*	The dimension according which the lines are aligned.
`in`	*max_alignment-_diff*	Maximum alignment difference.

Returns

A list of the resulting aligned cols. Each integer is actually a col number.

7.17.2.2 template<typename P > void scribo::table::internal::connect_lines (const mln::util::array< int > & *aligned_lines,* mln::util::array< box< P > > & *boxes,* unsigned *dim,* unsigned *dim_size,* unsigned *max_distance*)

Connect vertical and horizontal lines if they are close to each other.

—> | | | | |

FIXME: doc arguments.

7.17.2.3 template<unsigned axis, typename I > void scribo::table::internal::repair_lines (const Image< I > & *input,* mln::util::array< box< typename I::site > > & *tableboxes,* unsigned *max_discontinuity*)

Repair lines which have small discontinuities.

FIXME: buggy. Sometimes few lines move or shrink!

Parameters

`in`	*input*	A binary image.
`in,out`	*tableboxes*	Vertical or horizontal lines.
`in`	*max_-discontinuity*	The maximum discontinuity length which can be repaired.

7.18 scribo::text Namespace Reference

Functions

- template<typename L , typename I >

mln::trait::concrete< I >::**ret** clean (const line_info< L > &**line**, const **Image**< I > &input_)

- template<typename L , typename I >
  void clean_inplace (const line_info< L > &**line**, **Image**< I > &input)
- template<typename I , typename N >
  line_set< typename
  mln::trait::ch_value< I,
  scribo::def::lbl_type >::**ret** > extract_lines (const **Image**< I > &input, const **Neighborhood**< N > &nbh, const typename **mln::trait::ch_value**< I, **bool** >::**ret** &separators)
- template<typename I , typename N >
  line_set< typename
  mln::trait::ch_value< I,
  scribo::def::lbl_type >::**ret** > extract_lines (const **Image**< I > &input, const **Neighborhood**< N > &nbh)
- template<typename L >
  line_set< typename
  mln::trait::ch_value< L,
  scribo::def::lbl_type >::**ret** > extract_lines (const component_set< L > &input)
- template<typename I , typename J , typename N >
  line_set< typename
  mln::trait::ch_value< I,
  scribo::def::lbl_type >::**ret** > extract_lines_with_features (const **Image**< I > &input, const **Image**< J > &input_binary, const **Neighborhood**< N > &nbh, const typename **mln::trait::ch_value**< I, **bool** >::**ret** &separators)
- template<typename I , typename J , typename N >
  line_set< typename
  mln::trait::ch_value< I,
  scribo::def::lbl_type >::**ret** > extract_lines_with_features (const **Image**< I > &input, const **Image**< J > &input_binary, const **Neighborhood**< N > &nbh)
- template<typename I , typename N >
  line_set< typename
  mln::trait::ch_value< I,
  scribo::def::lbl_type >::**ret** > extract_lines_wo_merge (const **Image**< I > &input, const **Neighborhood**< N > &nbh, const typename **mln::trait::ch_value**< I, **bool** >::**ret** &separators)
- template<typename I , typename N >
  line_set< typename
  mln::trait::ch_value< I,
  scribo::def::lbl_type >::**ret** > extract_lines_wo_merge (const **Image**< I > &input, const **Neighborhood**< N > &nbh)
- template<typename L , typename N >
  line_set< typename
  mln::trait::ch_value< L,
  scribo::def::lbl_type >::**ret** > extract_lines_wo_merge (const document< L > &doc, const **Neighborhood**< N > &nbh)
- template<typename L , typename N >
  line_set< typename
  mln::trait::ch_value< L,
  scribo::def::lbl_type >::**ret** > extract_lines_wo_merge (const document< L > &doc, const **Neighborhood**< N > &nbh, const typename **mln::trait::ch_value**< L, **bool** >::**ret** &separators)
- template<typename L >
  paragraph_set< L > extract_paragraphs (line_set< L > &lines, const **image2d**< **bool** > &input)
- template<typename L >
  paragraph_set< L > extract_paragraphs_hdoc (line_set< L > &lines, const **image2d**< **bool** > &input)
- template<typename L >
  line_links< L > link_lines (const line_set< L > &lines)
- template<typename L >
  line_set< L > look_like_text_lines (const scribo::line_set< L > &l)
- template<typename L >
  void look_like_text_lines_inplace (scribo::line_set< L > &**line**)

- template<typename L >
  line_set< L > merging (const scribo::line_set< L > &lines)
- template<typename L >
  line_set< L > merging_hdoc (const scribo::line_set< L > &lines)
- template<typename L >
  mln::trait::concrete< L >**::ret paragraphs_closing** (const paragraph_set< L > &parset)
- template<typename L >
  void recognition (line_set< L > &lines, const char ∗language)
- template<typename I >
  void recognition (const **Image**< I > &**line**, const char ∗language, const std::string &output_file=std::string())

7.18.1 Detailed Description

Namespace of routines working on text components.

7.18.2 Function Documentation

7.18.2.1 template<typename L , typename I > mln::trait::concrete< I >::ret scribo::text::clean (const line_info< L > & *line,* const Image< I > & *input_*)

Improve quality of an image with text.

Parameters

in	*line*	Line info providing statistics about the text in the corresponding image `input`.
in	*input_*	A binary image. Object are set to 'false' and backgroud to 'true'.

Returns

An image. The text have better quality.

7.18.2.2 template<typename L , typename I > void scribo::text::clean_inplace (const line_info< L > & *line,* Image< I > & *input*)

Improve quality of an image with text.

Parameters

in	*line*	Line info providing statistics about the text in the corresponding image `input`.
in,out	*input*	A binary image. Object are set to 'false' and backgroud to 'true'.

7.18.2.3 template<typename L > line_set<L> scribo::text::look_like_text_lines (const scribo::line_set< L > & *l*)

Set line type to line::Text according to criterion.

7.18.2.4 template<typename L > void scribo::text::look_like_text_lines_inplace (scribo::line_set< L > & *line*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.Inplace version.

Chapter 8

Class Documentation

8.1 cluster_stats< T > Class Template Reference

Public Member Functions

- **cluster_stats** (const **unsigned** size)
- **T max** ()
- **T mean** ()
- **T median** ()
- **T min** ()
- **unsigned nelements** ()
- **T operator[ ]** (const **unsigned** index)
- void **reset** ()
- void **sort** ()
- **T standard_deviation** ()
- void **take** (const **T** &**value**)
- **T variance** ()

8.1.1 Detailed Description

template< typename T > class cluster_stats< T >

Definition at line 31 of file stats.hh.

8.2 compare_values< T > Struct Template Reference

Public Member Functions

- **bool operator()** (const **T** &lhs, const **T** &rhs)

8.2.1 Detailed Description

template< typename T > struct compare_values< T >

Definition at line 16 of file stats.hh.

8.3 mln::info Struct Reference

Public Member Functions

- **int height** () const
- void **init** (**unsigned p**, **int** row, **int** col)
- void **update** (info &r)
- **int width** () const

Public Attributes

- **unsigned card**
- **float col_sum**
- **point2d p_max**
- **point2d p_min**
- **unsigned parent**
- **float row_sum**

8.3.1 Detailed Description

Definition at line 71 of file components.hh.

8.4 scribo::binarization::internal::niblack_formula Struct Reference

Public Member Functions

- **double** operator() (const **double** m_x_y, const **double** s_x_y, const **double** K) const
- **double** operator() (const **double** m_x_y, const **double** s_x_y) const

8.4.1 Detailed Description

Definition at line 52 of file niblack_formula.hh.

8.4.2 Member Function Documentation

8.4.2.1 double scribo::binarization::internal::niblack_formula::operator() (const double *m_x_y,* const double *s_x_y,* const double *K*) const

compute a threshold using Niblack's formula.

Parameters

in	*m_x_y*	Mean value.
in	*s_x_y*	Standard deviation.
in	*K*	Control the threshold value in the local window. The higher, the lower the threshold form the local mean m(x, y).

Returns

A threshold.

8.4.2.2 double scribo::binarization::internal::niblack_formula::operator() (const double *m_x_y,* const double *s_x_y*) const

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.K is set to -0.2.

8.5 scribo::binarization::internal::niblack_functor< I > Struct Template Reference

Public Types

- enum { **step** = 1 }

Public Member Functions

- **niblack_functor** (const **Image**< I > &input, **double** K)
- void **end_of_row** (**int**)
- void **exec** (**double** mean, **double** stddev)
- void **finalize** ()
- void **init** ()

Public Attributes

- scribo::binarization::internal::niblack_formula **formula_**
- const I **input**
- **double K_**
- **unsigned next_line**
- **mln::trait::ch_value**< I, **bool** > ::**ret output**
- const **I::value** ∗ **pi**
- **bool** ∗ **po**

8.5.1 Detailed Description

template<typename I>struct scribo::binarization::internal::niblack_functor< I >

Definition at line 57 of file niblack_functor.hh.

8.6 scribo::binarization::internal::niblack_functor_fast< I > Struct Template Reference

Public Types

- enum { **step** = 3 }

Public Member Functions

- **niblack_functor_fast** (const **Image**< I > &input, **double** K)
- void **end_of_row** (**int**)
- void **exec** (**double** mean, **double** stddev)
- void **finalize** ()
- void **init** ()

Public Attributes

- scribo::binarization::internal::niblack_formula **formula_**
- const I **input**
- **double K_**
- **unsigned next_line3**
- **unsigned offset1**
- **unsigned offset2**
- **mln::trait::ch_value**< I, **bool** > ::**ret output**
- const **I::value** ∗ **pi**
- **bool** ∗ **po**

8.6.1 Detailed Description

template<typename I>struct scribo::binarization::internal::niblack_functor_fast< I >

Definition at line 58 of file niblack_functor_fast.hh.

8.7 scribo::binarization::internal::niblack_threshold_functor< I > Struct Template Reference

Public Types

- enum { **step** = 3 }
- typedef **mln::trait::concrete** < I >::**ret th_t**

Public Member Functions

- **niblack_threshold_functor** (const **Image**< I > &input, **double** K)
- void **end_of_row** (**int**)
- void **exec** (**double** mean, **double** stddev)
- void **finalize** ()
- void **init** ()

Public Attributes

- scribo::binarization::internal::niblack_formula **formula_**
- **mln::trait::concrete**< I >::**ret input**
- **double K_**
- **unsigned next_line3**
- **unsigned offset1**
- **unsigned offset2**
- th_t **output**
- **I::value** ∗ **po**

8.7.1 Detailed Description

template<typename I>struct scribo::binarization::internal::niblack_threshold_functor< I >

Definition at line 58 of file niblack_threshold_functor.hh.

8.8 scribo::binarization::internal::sauvola_formula Struct Reference

Public Member Functions

- **double** operator() (const **double** m_x_y, const **double** s_x_y, const **double** K, const **double** R) const
- **double** operator() (const **double** m_x_y, const **double** s_x_y) const

8.8.1 Detailed Description

Definition at line 63 of file sauvola_formula.hh.

8.8.2 Member Function Documentation

8.8.2.1 double scribo::binarization::internal::sauvola_formula::operator() (const double *m_x_y,* const double *s_x_y,* const double *K,* const double *R*) const

Compute a threshold using Sauvola's formula.

Parameters

in	*m_x_y*	Mean value.
in	*s_x_y*	Standard deviation.
in	*K*	Controls the threshold value in the local window. The higher, the lower the threshold form the local mean m(x, y).
in	*R*	Maximum value of the standard deviation (128 for grayscale documents).

Returns

A threshold.

8.8.2.2 double scribo::binarization::internal::sauvola_formula::operator() (const double *m_x_y,* const double *s_x_y*) const

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.K is set to 0.34 and R to 128.

8.9 scribo::binarization::internal::sauvola_functor< I > Struct Template Reference

Public Types

- enum { **step** = 3 }

Public Member Functions

- **sauvola_functor** (const **Image**< I > &input, **double** K, **double** R)
- void **end_of_row** (**int**)
- void **exec** (**double** mean, **double** stddev)
- void **finalize** ()
- void **init** ()

Public Attributes

- scribo::binarization::internal::sauvola_formula **formula_**
- const I **input**
- **double K_**
- **unsigned next_line3**
- **unsigned offset1**
- **unsigned offset2**
- **mln::trait::ch_value**< I, **bool** >
  ::**ret output**
- const **I::value** ∗ **pi**
- **bool** ∗ **po**
- **double R_**

8.9.1 Detailed Description

template<typename I>struct scribo::binarization::internal::sauvola_functor< I >

Definition at line 58 of file sauvola_functor.hh.

8.10 scribo::binarization::internal::sauvola_ms_functor< I > Struct Template Reference

Public Member Functions

- **sauvola_ms_functor** (const I &input, **double** R, const **image2d**< **value::int_u8** > &e_2, **unsigned** i, **unsigned** q)
- void **end_of_row** (**int** row)
- void **exec** (**double** mean, **double** stddev)
- void **finalize** ()
- void **init** ()
- **mln_fwd_pixter** (const I) pxl

Public Attributes

- **image2d**< **unsigned** > **card**
- **mln::util::array**< **int** > **dp**
- const **image2d**< **value::int_u8** > & **e_2**
- sauvola_formula **formula_**
- **unsigned i**
- **int i_**
- const I & **input**
- **double K_**
- **image2d**< **bool** > **msk**
- **unsigned n_nbhs**
- **image2d**< **unsigned** > **parent**
- **unsigned q**
- **double R_**
- **double res**
- **image2d**< **value::int_u8** > **t_sub**

8.10.1 Detailed Description

template<typename I>struct scribo::binarization::internal::sauvola_ms_functor< I >

Definition at line 66 of file sauvola_ms_functor.hh.

8.11 scribo::binarization::internal::sauvola_threshold_functor< I > Struct Template Reference

Public Types

- enum { **step** = 3 }
- typedef **mln::trait::concrete**
  < I >::**ret th_t**

Public Member Functions

- **sauvola_threshold_functor** (const **Image**< I > &input, **double** K, **double** R)
- void **end_of_row** (**int**)
- void **exec** (**double** mean, **double** stddev)
- void **finalize** ()
- void **init** ()

Public Attributes

- scribo::binarization::internal::sauvola_formula **formula_**
- **mln::trait::concrete**< I >::**ret input**
- **double K_**
- **unsigned next_line3**
- **unsigned offset1**
- **unsigned offset2**
- th_t **output**
- **I::value** ∗ **po**
- **double R_**

8.11.1 Detailed Description

template<typename I>struct scribo::binarization::internal::sauvola_threshold_functor< I >

Definition at line 58 of file sauvola_threshold_functor.hh.

8.12 scribo::binarization::internal::singh_formula< V > Struct Template Reference

Public Member Functions

- **double** operator() (const V &v, const **double** m_x_y, const **double** K) const
- **double** operator() (const V &v, const **double** m_x_y) const

8.12.1 Detailed Description

template<typename V>struct scribo::binarization::internal::singh_formula< V >

Definition at line 58 of file singh_formula.hh.

8.12.2 Member Function Documentation

8.12.2.1 template<typename V > double scribo::binarization::internal::singh_formula< V >::operator() (const V & *v,* const double *m_x_y,* const double *K*) const

Compute a threshold using Singh's formula.

Parameters

in	*v*	The current image value.
in	*m_x_y*	Mean value.
in	*K*	Control the threshold value in the local window. The higher, the lower the threshold form the local mean m(x, y).

Returns

A threshold.

8.12.2.2 template<typename V > double scribo::binarization::internal::singh_formula< V >::operator() (const V & *v,* const double *m_x_y*) const

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts. K = 0.34.

8.13 scribo::binarization::internal::singh_functor< I > Struct Template Reference

Public Types

- enum { **step** = 3 }
- typedef **I::value V**

Public Member Functions

- **singh_functor** (const **Image**< I > &input, **double** K)
- void **end_of_row** (**int**)
- void **exec** (**double** mean, **double** stddev)
- void **finalize** ()
- void **init** ()

Public Attributes

- scribo::binarization::internal::singh_formula
< V > **formula_**
- const I **input**
- **double K_**
- **unsigned next_line3**

- **unsigned offset1**
- **unsigned offset2**
- **mln::trait::ch_value**< I, **bool** >
  ::**ret output**
- const V ∗ **pi**
- **bool** ∗ **po**

8.13.1 Detailed Description

template<typename I>struct scribo::binarization::internal::singh_functor< I >

Definition at line 58 of file singh_functor.hh.

8.14 scribo::binarization::internal::wolf_formula< V > Struct Template Reference

Public Member Functions

- **double** operator() (const **double** m_x_y, const **double** s_x_y, const **double** K, const **double** global_max_stddev, const V &global_min) const

8.14.1 Detailed Description

template<typename V>struct scribo::binarization::internal::wolf_formula< V >

Definition at line 58 of file wolf_formula.hh.

8.14.2 Member Function Documentation

8.14.2.1 template<typename V > **double scribo::binarization::internal::wolf_formula**< V >::operator() (const **double** *m_x_y*, const **double** *s_x_y*, const **double** *K*, const **double** *global_max_stddev*, const V & *global_min*) const

Compute a threshold using Wolf's formula.

Returns

A threshold.

8.15 scribo::binarization::internal::wolf_functor< I > Struct Template Reference

Public Types

- enum { **step** = 1 }
- typedef **I::value V**

Public Member Functions

- **wolf_functor** (const **Image**< I > &input, **double** K, const typename **I::value** &global_min, **double** global_max_stddev)
- void **end_of_row** (**int**)
- void **exec** (**double** mean, **double** stddev)

- void **finalize** ()
- void **init** ()

Public Attributes

- scribo::binarization::internal::wolf_formula
  < V > **formula_**
- **double global_max_stddev_**
- V **global_min_**
- const I **input**
- **double K_**
- **unsigned next_line**
- **mln::trait::ch_value**< I, **bool** >
  ::**ret output**
- const **I::value** ∗ **pi**
- **bool** ∗ **po**

8.15.1 Detailed Description

template<typename I>struct scribo::binarization::internal::wolf_functor< I >

Definition at line 58 of file wolf_functor.hh.

8.16 scribo::binarization::internal::wolf_functor_fast< I > Struct Template Reference

Public Types

- enum { **step** = 3 }
- typedef **I::value V**

Public Member Functions

- **wolf_functor_fast** (const **Image**< I > &input, **double** K, const typename **I::value** &global_min, **double** global_max_stddev)
- void **end_of_row** (**int**)
- void **exec** (**double** mean, **double** stddev)
- void **finalize** ()
- void **init** ()

Public Attributes

- scribo::binarization::internal::wolf_formula
  < V > **formula_**
- **double global_max_stddev_**
- V **global_min_**
- const I **input**
- **double K_**
- **unsigned next_line3**
- **unsigned offset1**
- **unsigned offset2**
- **mln::trait::ch_value**< I, **bool** >
  ::**ret output**

- const **I::value** $*$ **pi**
- **bool** $*$ **po**

8.16.1 Detailed Description

template<typename I>struct scribo::binarization::internal::wolf_functor_fast< I >

Definition at line 58 of file wolf_functor_fast.hh.

8.17 scribo::component_features_data Struct Reference

Public Attributes

- **float boldness**
- **scribo::def::color_type color**
- **bool valid**

8.17.1 Detailed Description

Definition at line 39 of file component_features_data.hh.

8.18 scribo::component_info< L > Class Template Reference

```
#include <scribo/core/component_info.hh>
```

Public Types

- typedef **Object**< void > **category**
- typedef component_info< L > **exact_t**

Public Member Functions

- **component_info** (const **component_id_t** &id, const **mln::box2d** &bbox, const **mln::point2d** &mass_center, **unsigned** card, component::Type type=component::Undefined)
- void accept (const SerializeVisitor< E2 > &visitor) const
- const **mln::box2d** & **bbox** () const
- **unsigned card** () const
- const component_features_data & **features** () const
- **bool has_features** () const
- **component_id_t id** () const
- **bool is_valid** () const
- const **mln::point2d** & **mass_center** () const
- component::Tag **tag** () const
- component::Type **type** () const
- void **update_features** (const component_features_data &features)
- void **update_tag** (component::Tag tag)
- void **update_type** (component::Type type)

Protected Attributes

- **mln::box2d bbox_**
- **unsigned card_**
- component_features_data **features_**
- **component_id_t id_**
- **mln::point2d mass_center_**
- component::Tag **tag_**
- component::Type **type_**

8.18.1 Detailed Description

template<typename L>class scribo::component_info< L >

Component information data structure.

Definition at line 64 of file component_info.hh.

8.18.2 Member Function Documentation

8.18.2.1 void scribo::Serializable< component_info< L > >::accept (const SerializeVisitor< E2 > & *visitor*) const [inherited]

Allow this object to be serialized by `visitor`.

8.19 scribo::component_set< L > Class Template Reference

```
#include <scribo/core/component_set.hh>
```

Public Types

- typedef **Object**< void > **category**
- typedef component_set< L > **exact_t**

Public Member Functions

- component_set ()
- component_set (const **mln::util::tracked_ptr**< data_t > &**data**)
- component_set (const L &ima, const typename **L::value** &ncomps)
- component_set (const L &ima, const typename **L::value** &ncomps, const **mln::util::array**< **pair_accu_t** > &attribs, component::Type type=component::Undefined)
- component_set (const L &ima, const typename **L::value** &ncomps, const **mln::util::array**< pair_data_t > &attribs, component::Type type=component::Undefined)
- void accept (const SerializeVisitor< E2 > &visitor) const
- void add_separators (const typename **mln::trait::ch_value**< L, **bool** >::**ret** &ima)
- void clear_separators ()
- component_set< L > duplicate () const
- **bool** has_separators () const
- const component_info< L > & info (const typename **L::value** &id) const
- component_info< L > & info (const typename **L::value** &id)
- **bool** is_valid () const
- const L & labeled_image () const

- **L::value** nelements () const
- component_info< L > & operator() (const component_id_t &id)
- const component_info< L > & operator() (const component_id_t &id) const
- const **mln::trait::ch_value**< L,
  bool >::**ret** & separators () const
- template<typename F >
  void update_tags (const **mln::Function_v2b**< F > &f, component::Tag tag)

Protected Member Functions

- void init_ (const component_set< L > &model)

Protected Attributes

- **mln::util::tracked_ptr**< data_t > **data_**

Related Functions

(Note that these are not member functions.)

- template<typename L >
  std::ostream & operator<< (std::ostream &ostr, const component_set< L > &info)
- template<typename L >
  bool operator== (const component_set< L > &lhs, const component_set< L > &rhs)

8.19.1 Detailed Description

template<typename L>class scribo::component_set< L >

Represents all the components in a document image.

This structure is used to store rich information related to components in an image.

It can be constructed directly from a labeled image and the number of components, or using primitive::extract::components (easier and recommanded) with a binary image.

```
mln::image2d<bool> ima;
mln::io::pbm::load(ima, "document.pbm");

typedef scribo::def::lbl_type V;
typedef mln::image2d<V> L;
V ncomps;
component_set<L> comps = primitive::extract::components
       (ima, c4(), ncomps);
```

Each component is associated to an id, component_id_t. This id is equivalent to the component label in the label image (labeled_image()). They are labeled from 1 to nelements() included. Id 0 is reserved for the background which is not considered as an actual component.

A component_set is considered as valid if it has been initialized (i.e. not instantiated with the default constructor).

On construction, information is automatically computed for each components. To iterate over all the components and get that information, use operator()() or info():

```
for_all_comps(c,comps)
{
  const scribo::component_info<L>& comp_info = comps.
      info(c);

  // Some code here...
}
```

Among component information, a component::Tag tag is stored and used to mark specific components. This tag can be massively updated for all components using update_tags(). The function passed to this method must implement the following interface:

```
template <typename L>
struct my_function
: mln::Function_v2b< my_function<L> >
  {
    // Constructor
    my_function(const scribo::component_set<L>&
      components)
      : components_(components)
    {
    }

    // Core function
    bool operator()(const mln_value(L)& l) const
    {
      if (l == mln::literal::zero)
        return false;
      return true;// your test here
    }

    scribo::component_set<L> components_;
  };
```

Tags can be used to filter/make a selection of components. Routines performing this selection/filtering automatically are listed in Component Filtering section.

Components considered as separators in the input image (lines, blocks, ...) may be processed separately and are useful of the rest of the layout analysis. This structure allows you to store a binary image of separators which will be used in some routines to prevent wrong regrouping. Note that no component information will be computed for separators. The separator image can be managed with has_separators(), add_separators(), separators(), and clear_separators().

scribo::component_set is the first data structure in the hierarchical representation of a document as explained in Data structures.

See Also

component_info, component::Tag, component::Type, Data structures, Component Filtering.

Definition at line 217 of file component_set.hh.

8.19.2 Constructor & Destructor Documentation

8.19.2.1 template<typename L> scribo::component_set< L >::component_set ( )

Default.

8.19.2.2 template<typename L> scribo::component_set< L >::component_set (const mln::util::tracked_ptr< data_t > & *data*)

Constructor from internal data.

8.19.2.3 template<typename L> scribo::component_set< L >::component_set (const L & *ima*, const typename L::value & *ncomps*)

Constructor from an image `ima` and the number of labels `ncomps`.

8.19.2.4 template<typename L> scribo::component_set< L >::component_set (const L & *ima*, const typename L::value & *ncomps*, const mln::util::array< pair_accu_t > & *attribs*, component::Type *type* = component::Undefined)

Constructor from an image `ima`, the number of labels `ncomps` and attributes values (bounding box and mass center).

8.19.2.5 template<typename L> scribo::component_set< L >::component_set (const L & *ima*, const typename L::value & *ncomps*, const mln::util::array< pair_data_t > & *attribs*, component::Type *type* = component::Undefined)

Constructor from an image `ima`, the number of labels `ncomps` and attributes values (bounding box and mass center).

8.19.3 Member Function Documentation

8.19.3.1 void scribo::Serializable< component_set< L > >::accept (const SerializeVisitor< E2 > & *visitor*) const [inherited]

Allow this object to be serialized by `visitor`.

8.19.3.2 template<typename L> void scribo::component_set< L >::add_separators (const typename mln::trait::ch_value< L, bool >::ret & *ima*)

Add a new separator binary image or perform a logical OR with the existing one.

8.19.3.3 template<typename L> void scribo::component_set< L >::clear_separators ( )

Clear separator image.

8.19.3.4 template<typename L> component_set<L> scribo::component_set< L >::duplicate ( ) const

Create a copy of this component_set<L>

8.19.3.5 template<typename L> bool scribo::component_set< L >::has_separators ( ) const

Return true if an image of separator exists.

8.19.3.6 template<typename L> const component_info<L>& scribo::component_set< L >::info (const typename L::value & *id*) const

Return component information for a given component id `id`.

8.19.3.7 template<typename L> component_info<L>& scribo::component_set< L >::info (const typename L::value & *id*)

Return component information for a given component id `id`.

8.19.3.8 template<typename L> void scribo::component_set< L >::init_ (const component_set< L > & *model*) [protected]

INTERNAL_API Duplicate the underlying image and create a new component_set.

8.19.3.9 template<typename L> bool scribo::component_set< L >::is_valid ( ) const

A component_set is considered as valid if it has been initialized (i.e.

not instantiated with the default constructor).

8.19.3.10 template<typename L> const L& scribo::component_set< L >::labeled_image ( ) const

Return the underlying labeled image.

8.19.3.11 template<typename L> L ::value scribo::component_set< L >::nelements ( ) const

Return the number of components, background excluded.

8.19.3.12 template<typename L> component_info<L>& scribo::component_set< L >::operator() (const component_id_t & *id*)

Return component information for a given component id `id`.

8.19.3.13 template<typename L> const component_info<L>& scribo::component_set< L >::operator() (const component_id_t & *id*) const

Return component information for a given component id `id`.

8.19.3.14 template<typename L> const mln::trait::ch_value< L , bool >::ret& scribo::component_set< L >::separators ( ) const

Return the binary image of separators.

8.19.3.15 template<typename L> template<typename F > void scribo::component_set< L >::update_tags (const mln::Function_v2b< F > & *f*, component::Tag *tag*)

Update component tag for which `f` returns 'False', with `tag`.

8.19.4 Friends And Related Function Documentation

8.19.4.1 template<typename L > std::ostream & operator<< (std::ostream & *ostr*, const component_set< L > & *info*) [related]

8.19.4.2 template<typename L > bool operator== (const component_set< L > & *lhs*, const component_set< L > & *rhs*) [related]

8.20 scribo::debug::arg_data Struct Reference

Public Attributes

- const char ∗ **desc**

- const char ∗ **name**

8.20.1 Detailed Description

Definition at line 39 of file option_parser.hh.

8.21 scribo::debug::internal::logger_ Class Reference

Public Member Functions

- template<typename V >
  logger_ & operator<< (const V &v)
- logger_ & **operator**<< (std::ostream &(∗f)(std::ostream &))

- **bool** is_verbose () const
- **bool** is_at_verbose_mode (VerboseMode mode) const
- **bool** set_default_verbose_mode (VerboseMode mode)
- VerboseMode default_verbose_mode () const
- **bool** set_verbose_mode (VerboseMode mode)
- VerboseMode verbose_mode () const
- void set_verbose_prefix (const std::string &prefix)
- void log (VerboseMode mode, const std::string &text)

- **bool** is_enabled () const
- **bool** is_at_level (Level level) const
- void set_level (Level level)
- Level level () const
- void set_filename_prefix (const char ∗name)
- const char ∗ filename_prefix () const
- template<typename I >
  void log_image (Level dbg_level, const **Image**< I > &ima, const char ∗name)

- void start_time_logging ()
- void stop_time_logging (const std::string &time_title)

Static Public Member Functions

- static logger_ & **instance** ()

8.21.1 Detailed Description

Definition at line 99 of file logger.hh.

8.21.2 Member Function Documentation

8.21.2.1 VerboseMode scribo::debug::internal::logger_::default_verbose_mode ( ) const

Text Logging.

8.21.2.2 const char∗ scribo::debug::internal::logger_::filename_prefix ( ) const

Image Logging.

8.21.2.3 bool scribo::debug::internal::logger_::is_at_level (Level *level*) const

Image Logging.

8.21.2.4 bool scribo::debug::internal::logger_::is_at_verbose_mode (VerboseMode *mode*) const

Text Logging.

8.21.2.5 bool scribo::debug::internal::logger_::is_enabled ( ) const

Image Logging.

8.21.2.6 bool scribo::debug::internal::logger_::is_verbose ( ) const

Text Logging.

8.21.2.7 Level scribo::debug::internal::logger_::level ( ) const

Image Logging.

8.21.2.8 void scribo::debug::internal::logger_::log (VerboseMode *mode,* const std::string & *text*)

Text Logging.

8.21.2.9 template<typename I > void scribo::debug::internal::logger_::log_image (Level *dbg_level,* const Image< I > & *ima,* const char ∗ *name*)

Image Logging.

8.21.2.10 template<typename V > logger_& scribo::debug::internal::logger_::operator<< (const V & *v*)

Quickly logs text. **It** uses the default log mode to know if the text.

8.21.2.11 bool scribo::debug::internal::logger_::set_default_verbose_mode (VerboseMode *mode*)

The default verbose mode used while logging with operator<<.

8.21.2.12 void scribo::debug::internal::logger_::set_filename_prefix (const char ∗ *name*)

Image Logging.

8.21.2.13 void scribo::debug::internal::logger_::set_level (Level *level*)

Image Logging.

8.21.2.14 bool scribo::debug::internal::logger_::set_verbose_mode (VerboseMode *mode*)

Set the current verbose mode, filtering the debug output logged through this object.

8.21.2.15 void scribo::debug::internal::logger_::set_verbose_prefix (const std::string & *prefix*)

Text Logging.

8.21.2.16 void scribo::debug::internal::logger_::start_time_logging ( )

Time Logging.

This class provides timers in order to performs benchmarks inside a program.

Stopping time logging will output the computed time if Verbose mode is higher or equal to scribo::debug::Time.

8.21.2.17 void scribo::debug::internal::logger_::stop_time_logging (const std::string & *time_title*)

Time Logging.

This class provides timers in order to performs benchmarks inside a program.

Stopping time logging will output the computed time if Verbose mode is higher or equal to scribo::debug::Time.

8.21.2.18 VerboseMode scribo::debug::internal::logger_::verbose_mode ( ) const

Text Logging.

8.22 scribo::debug::opt_data Struct Reference

Public Attributes

- const char ∗ **by_default**
- const char ∗ **desc**
- **bool**(∗ **fcheck**)(const std::vector< const char ∗ > &)
- const char ∗ **format**
- **int n_args**
- const char ∗ **name**

8.22.1 Detailed Description

Definition at line 54 of file option_parser.hh.

8.23 scribo::debug::option_parser Class Reference

Public Member Functions

- **option_parser** (const arg_data arg_desc[ ], const toggle_data toggle_desc[ ], const opt_data opt_desc[ ])
- const char ∗ **arg** (const char ∗) const
- **bool is_enabled** (const char ∗toggle_name) const
- **bool is_set** (const char ∗opt_name) const
- std::string **opt_value** (const char ∗opt_name) const

- std::vector< const char ∗ > **opt_values** (const char ∗opt_name) const
- **bool parse** (**int** argc, char ∗argv[ ])
- void **print_help** () const
- void **set_verbose_enabled** (**bool** b)

8.23.1 Detailed Description

Definition at line 65 of file option_parser.hh.

8.24 scribo::debug::toggle_data Struct Reference

Public Attributes

- const char ∗ **desc**
- **bool enabled**
- const char ∗ **name**

8.24.1 Detailed Description

Definition at line 46 of file option_parser.hh.

8.25 scribo::DMax_Functor< E > Class Template Reference

```
#include <scribo/core/concept/dmax_functor.hh>
```

Public Types

- typedef **Object**< void > **category**
- typedef E **exact_t**

8.25.1 Detailed Description

template<typename E>class scribo::DMax_Functor< E >

Dmax functor concept.

Definition at line 40 of file dmax_functor.hh.

8.26 scribo::doc_serializer< E > Class Template Reference

```
#include <scribo/core/internal/doc_serializer.hh>
```

Public Types

- typedef **Object**< void > **category**
- typedef E **exact_t**

Public Member Functions

- template<typename L >
  void **visit** (const document< L > &doc) const
- template<typename L >
  void **visit** (const line_links< L > &llinks) const
- template<typename L >
  void **visit** (const object_groups< L > &groups) const
- template<typename L >
  void **visit** (const object_links< L > &links) const
- template<typename L >
  void **visit** (const component_set< L > &comp_set) const
- template<typename L >
  void **visit** (const component_info< L > &info) const
- template<typename L >
  void **visit** (const paragraph_set< L > &parset) const
- template<typename L >
  void **visit** (const line_info< L > &**line**) const

8.26.1 Detailed Description

template<typename E>class scribo::doc_serializer< E >

Link functor concept.

Definition at line 49 of file doc_serializer.hh.

8.27 scribo::document< L > Class Template Reference

```
#include <scribo/core/document.hh>
```

Public Types

- typedef **Object**< void > **category**
- typedef document< L > **exact_t**

Public Member Functions

- **document** (const char ∗filename)
- **document** (const char ∗filename, const **mln::image2d**< **mln::value::rgb8** > &input)
- void accept (const SerializeVisitor< E2 > &visitor) const
- const **mln::image2d**< **bool** > & **binary_image** () const
- const **mln::image2d**< **bool** > & **binary_image_wo_seps** () const
- const component_set< L > & **elements** () const
- const char ∗ **filename** () const
- **bool has_elements** () const
- **bool has_hline_seps** () const
- **bool** has_text () const
- **bool has_vline_seps** () const
- **bool has_whitespace_seps** () const
- **mln::def::coord height** () const
- const **mln::image2d**< **bool** > & **hline_seps** () const
- const component_set< L > & **hline_seps_comps** () const

- const **mln::image2d**< **value::rgb8** > & **image** () const
- **bool is_open** () const
- **bool is_valid** () const
- const line_set< L > & **lines** () const
- void **open** ()
- const paragraph_set< L > & **paragraphs** () const
- void **set_binary_image** (const **mln::image2d**< **bool** > &binary_image)
- void **set_binary_image_wo_seps** (const **mln::image2d**< **bool** > &binary_image_wo_seps)
- void **set_elements** (const component_set< L > &elements)
- void **set_filename** (const char ∗name)
- void **set_hline_separators** (const **image2d**< **bool** > &line_seps)
- void **set_hline_separators** (const **image2d**< **bool** > &line_seps, const component_set< L > &hline_seps_comps)
- void **set_image** (const **mln::image2d**< **value::rgb8** > &image)
- void **set_paragraphs** (const paragraph_set< L > &parset)
- void **set_vline_separators** (const **image2d**< **bool** > &vline_seps)
- void **set_vline_separators** (const **image2d**< **bool** > &vline_seps, const component_set< L > &vline_seps_comps)
- void **set_whitespace_separators** (const **image2d**< **bool** > &whitespace_seps, const component_set< L > &whitespace_seps_comps)
- const **mln::image2d**< **bool** > & **vline_seps** () const
- const component_set< L > & **vline_seps_comps** () const
- const **mln::image2d**< **bool** > & **whitespace_seps** () const
- const component_set< L > & **whitespace_seps_comps** () const
- **mln::def::coord width** () const

8.27.1 Detailed Description

template<typename L>class scribo::document< L >

Represent document data and structure.

Definition at line 90 of file document.hh.

8.27.2 Member Function Documentation

8.27.2.1 void scribo::Serializable< document< L > >::accept (const SerializeVisitor< E2 > & *visitor*) const `[inherited]`

Allow this object to be serialized by `visitor`.

8.27.2.2 template<typename L> bool scribo::document< L >::has_text ( ) const

Check whether this document contains text.

If it returns true, that document contains paragraphs, lines and text components.

8.28 scribo::fun::v2b::components_large_filter< L > Struct Template Reference

`#include <scribo/fun/v2b/objects_large_filter.hh>`

Public Types

- typedef **accu::math::count**
  < typename L::psite > **card_t**
- typedef **Function_v2b**< void > **category**
- typedef
  components_large_filter< L > **exact_t**
- typedef void **mutable_result**
- typedef **bool result**

Public Member Functions

- components_large_filter (const component_set< L > &components, **unsigned** max_size)
- **bool** operator() (const typename **L::value** &l) const

Public Attributes

- const component_set< L > components_
- **mln::util::array**< **bool** > marked_
- **unsigned** max_size_
- **L::value** nlabels_

8.28.1 Detailed Description

template<typename L>struct scribo::fun::v2b::components_large_filter< L >

Filter Functor.

Return false for all components which are too large.

Definition at line 62 of file objects_large_filter.hh.

8.28.2 Constructor & Destructor Documentation

8.28.2.1 template<typename L > scribo::fun::v2b::components_large_filter< L >::components_large_filter (const component_set< L > & *components,* unsigned *max_size*)

Constructor.

Parameters

in	*components*	Component bounding boxes.
in	*max_size*	Maximum component size.

8.28.3 Member Function Documentation

8.28.3.1 template<typename L > bool scribo::fun::v2b::components_large_filter< L >::operator() (const typename L::value & *l*) const

Check if the component is large enough.

Parameters

l	A label.

Returns

false if the component area is strictly inferion to `max_size_`.

8.28.4 Member Data Documentation

8.28.4.1 template<typename L > const component_set<L> scribo::fun::v2b::components_large_filter< L >::components_

The component set to filter.

Definition at line 89 of file objects_large_filter.hh.

8.28.4.2 template<typename L > mln::util::array<bool> scribo::fun::v2b::components_large_filter< L >::marked_ [mutable]

Has already been taken into account.

Definition at line 95 of file objects_large_filter.hh.

8.28.4.3 template<typename L > unsigned scribo::fun::v2b::components_large_filter< L >::max_size_

The minimum area.

Definition at line 86 of file objects_large_filter.hh.

8.28.4.4 template<typename L > L ::value scribo::fun::v2b::components_large_filter< L >::nlabels_ [mutable]

The number of labels remaining after filtering.

Definition at line 92 of file objects_large_filter.hh.

8.29 scribo::fun::v2b::components_on_border_filter< L > Struct Template Reference

`#include <scribo/fun/v2b/objects_on_border_filter.hh>`

Public Types

- typedef **accu::math::count** < typename L::psite > **card_t**
- typedef **Function_v2b**< void > **category**
- typedef components_on_border_filter< L > **exact_t**
- typedef void **mutable_result**
- typedef **bool result**

Public Member Functions

- components_on_border_filter (const component_set< L > &components)
- **bool** operator() (const typename **L::value** &l) const

Public Attributes

- const **box2d** & b_
- const component_set< L > components_

8.29.1 Detailed Description

template<typename L>struct scribo::fun::v2b::components_on_border_filter< L >

Filter Functor.

Return false for all components which are too large.

Definition at line 61 of file objects_on_border_filter.hh.

8.29.2 Constructor & Destructor Documentation

8.29.2.1 template<typename L > scribo::fun::v2b::components_on_border_filter< L >::components_on_border_filter (const component_set< L > & *components*)

Constructor.

Parameters

in	*components*	Component bounding boxes.

8.29.3 Member Function Documentation

8.29.3.1 template<typename L > bool scribo::fun::v2b::components_on_border_filter< L >::operator() (const typename L::value & *l*) const

Check if the component is large enough.

Parameters

in	*l*	A label.

Returns

false if the component area is strictly inferion to `max_size_`.

8.29.4 Member Data Documentation

8.29.4.1 template<typename L > const box2d& scribo::fun::v2b::components_on_border_filter< L >::b_

Labeled image bounding box.

Definition at line 86 of file objects_on_border_filter.hh.

8.29.4.2 template<typename L > const component_set<L> scribo::fun::v2b::components_on_border_filter< L >::components_

The component set to filter.

Definition at line 83 of file objects_on_border_filter.hh.

8.30 scribo::fun::v2b::components_small_filter< L > Struct Template Reference

```
#include <scribo/fun/v2b/objects_small_filter.hh>
```

Public Types

- typedef **accu::math::count**
  < typename L::psite > **card_t**
- typedef **Function_v2b**< void > **category**
- typedef
  components_small_filter< L > **exact_t**
- typedef void **mutable_result**
- typedef **bool result**

Public Member Functions

- components_small_filter (const component_set< L > &components, **unsigned** min_size)
- **bool** operator() (const typename **L::value** &l) const

Public Attributes

- const component_set< L > components_
- **mln::util::array**< **bool** > marked_
- **unsigned** min_size_
- **L::value** nlabels_

8.30.1 Detailed Description

template<typename L>struct scribo::fun::v2b::components_small_filter< L >

Filter Functor.

Return false for all components which are too small.

Definition at line 62 of file objects_small_filter.hh.

8.30.2 Constructor & Destructor Documentation

8.30.2.1 template<typename L > scribo::fun::v2b::components_small_filter< L >::components_small_filter (const component_set< L > & *components*, unsigned *min_size*)

Constructor.

Parameters

`in`	*components*	Component bounding boxes.
`in`	*min_size*	Minimum component size.

8.30.3 Member Function Documentation

8.30.3.1 template<typename L > bool scribo::fun::v2b::components_small_filter< L >::operator() (const typename L::value & *l*) const

Check if the component is large enough.

Parameters

l	A label.

Returns

false if the component area is strictly inferion to `min_size_`.

8.30.4 Member Data Documentation

8.30.4.1 template<typename L > const component_set<L> scribo::fun::v2b::components_small_filter< L >::components_

The component set to filter.

Definition at line 89 of file objects_small_filter.hh.

8.30.4.2 template<typename L > mln::util::array<bool> scribo::fun::v2b::components_small_filter< L >::marked_ [mutable]

Has already been taken into account.

Definition at line 95 of file objects_small_filter.hh.

8.30.4.3 template<typename L > unsigned scribo::fun::v2b::components_small_filter< L >::min_size_

The minimum area.

Definition at line 86 of file objects_small_filter.hh.

8.30.4.4 template<typename L > L ::value scribo::fun::v2b::components_small_filter< L >::nlabels_ [mutable]

The number of labels remaining after filtering.

Definition at line 92 of file objects_small_filter.hh.

8.31 scribo::fun::v2b::label_to_bool< L > Struct Template Reference

Public Types

- typedef **Function_v2b**< void > **category**
- typedef label_to_bool< L > **exact_t**
- typedef void **mutable_result**
- typedef **bool result**

Public Member Functions

- **label_to_bool** (const **mln::fun::i2v::array**< **bool** > &f)
- **bool operator()** (const L &v) const

Public Attributes

- **mln::fun::i2v::array**< **bool** > **f_**

8.31.1 Detailed Description

template<typename L>struct scribo::fun::v2b::label_to_bool< L >

Definition at line 43 of file label_to_bool.hh.

8.32 scribo::fun::v2v::highlight< R > Struct Template Reference

Public Types

- typedef **Function_v2v**< void > **category**
- typedef highlight< R > **exact_t**
- typedef void **mutable_result**
- typedef R **result**

Public Member Functions

- result **operator()** (const result &v) const

8.32.1 Detailed Description

template<typename R>struct scribo::fun::v2v::highlight< R >

Definition at line 43 of file highlight.hh.

8.33 scribo::group_info Class Reference

Public Member Functions

- **group_info** (**unsigned** id, const **mln::util::array**< component_id_t > &comps, **unsigned** pixel_area, const **box2d** &bbox)
- **group_info** (**unsigned** id, **unsigned** pixel_area, const **box2d** &bbox, **bool** valid=false)
- const **box2d** & **bbox** () const
- **unsigned card** () const
- const **mln::util::array**
  < component_id_t > & **component_ids** () const
- **mln::util::array**
  < component_id_t > & **component_ids_** ()
- **unsigned id** () const
- void **invalidate** ()
- **bool is_valid** () const
- void **merge** (group_info &rhs)
- **unsigned pixel_area** () const

8.33.1 Detailed Description

Definition at line 49 of file group_info.hh.

8.34 scribo::internal::component_set_data< L > Struct Template Reference

Public Types

- typedef **mln::accu::shape::bbox**
  < typename L::site > **bbox_accu_t**
- typedef **mln::accu::center**
  < typename L::site > **center_accu_t**
- typedef **center_accu_t::result center_t**
- typedef **mln::accu::pair**
  < **bbox_accu_t**, **center_accu_t** > **pair_accu_t**
- typedef std::pair< mln_box(L),
  std::pair< typename L::site,
  unsigned > > **pair_data_t**

Public Member Functions

- **component_set_data** (const L &ima, const typename **L::value** &ncomps)
- **component_set_data** (const L &ima, const typename **L::value** &ncomps, const **mln::util::array**< **pair_accu_t** > &attribs, component::Type type=component::Undefined)
- **component_set_data** (const L &ima, const typename **L::value** &ncomps, const **mln::util::array**< pair_data_t > &attribs, component::Type type=component::Undefined)
- **component_set_data** (const L &ima, const typename **L::value** &ncomps, const **mln::util::array**< scribo::component_info< L > > &infos)
- void **fill_infos** (const **mln::util::array**< **pair_accu_t** > &attribs, component::Type type=component::Undefined)
- void **fill_infos** (const **mln::util::array**< pair_data_t > &attribs, component::Type type=component::Undefined)
- void **soft_init** (const typename **L::value** ncomps)

Public Attributes

- L **ima_**
- **mln::util::array**
  < scribo::component_info< L > > **infos_**
- **L::value ncomps_**
- **mln::trait::ch_value**< L, **bool** >
  ::**ret separators_**

8.34.1 Detailed Description

template<typename L>struct scribo::internal::component_set_data< L >

Definition at line 82 of file component_set.hh.

8.35 scribo::internal::document_data< L > Struct Template Reference

```
#include <scribo/core/document.hh>
```

Public Member Functions

- **document_data** (const char ∗**filename**)
- **document_data** (const char ∗**filename**, const **mln::image2d**< **mln::value::rgb8** > &input)

Public Attributes

- **mln::image2d**< **bool** > **binary_image_**
- **mln::image2d**< **bool** > **binary_image_wo_seps_**
- component_set< L > **elements_**
- std::string **filename_**
- **mln::image2d**< **bool** > **hline_seps_**
- component_set< L > **hline_seps_comps_**
- **mln::image2d**< **mln::value::rgb8** > **image_**
- paragraph_set< L > **parset_**
- **mln::image2d**< **bool** > **vline_seps_**
- component_set< L > **vline_seps_comps_**
- **mln::image2d**< **bool** > **whitespace_seps_**
- component_set< L > **whitespace_seps_comps_**

8.35.1 Detailed Description

template<**typename L**>**struct scribo::internal::document_data**< **L** >

Data structure for `scribo::document<L>`.

Definition at line 57 of file document.hh.

8.36 scribo::internal::line_info_data< L > Struct Template Reference

```
#include <scribo/core/line_info.hh>
```

Public Member Functions

- **line_info_data** (const line_set< L > &holder, const group_info &group)
- **line_info_data** (const line_set< L > &holder, const **mln::util::array**< component_id_t > &component_ids)

Public Attributes

- **int a_height_**
- **int baseline_**
- stats< **float** > **baseline_clusters_**
- **mln::box2d bbox_**
- **float boldness_**
- **float boldness_reliability_**
- **unsigned char_space_**
- **unsigned char_width_**
- **mln::value::rgb8 color_**
- **float color_reliability_**
- **mln::util::array**< component_id_t > **component_ids_**
- component_set< L > **components_**
- **int d_height_**
- **mln::box2d ebbox_**
- **bool hidden_**
- std::string **html_text_**
- **bool indented_**
- object_links< L > **links_**

- **int meanline_**
- stats< **float** > **meanline_clusters_**
- **float orientation_**
- **unsigned pixel_area_**
- line::ReadingDirection **reading_direction_**
- **float reading_orientation_**
- **bool reverse_video_**
- line::Tag **tag_**
- std::string **text_**
- **float text_confidence_**
- line::Type **type_**
- **unsigned word_space_**
- **unsigned x_height_**

8.36.1 Detailed Description

template< typename L >struct scribo::internal::line_info_data< L >

Data structure for `scribo::line_info<I>`.

Definition at line 81 of file line_info.hh.

8.37 scribo::internal::line_links_data< L > Struct Template Reference

```
#include <scribo/core/line_links.hh>
```

Public Member Functions

- **line_links_data** (const line_set< L > &lines, **unsigned** size)
- **line_links_data** (const line_set< L > &lines, **unsigned** size, **line_id_t value**)

Public Attributes

- **mln::util::array< line_id_t > line_to_link_**
- line_set< L > **lines_**

8.37.1 Detailed Description

template< typename L >struct scribo::internal::line_links_data< L >

Data structure for `scribo::line_links<I>`.

Definition at line 54 of file line_links.hh.

8.38 scribo::internal::line_set_data< L > Struct Template Reference

```
#include <scribo/core/line_set.hh>
```

Public Member Functions

- **line_set_data** (const object_groups< L > &comp_set)
- **line_set_data** (const **mln::util::array**< scribo::line_info< L > > &infos, const object_groups< L > &comp_set)

Public Attributes

- component_set< L > **components_**
- object_groups< L > **groups_**
- **mln::util::array**
  < scribo::line_info< L > > **infos_**
- object_links< L > **links_**

8.38.1 Detailed Description

template<typename L>struct scribo::internal::line_set_data< L >

Data structure for `scribo::line_set<I>`.

Definition at line 69 of file line_set.hh.

8.39 scribo::internal::object_groups_data< L > Struct Template Reference

```
#include <scribo/core/object_groups.hh>
```

Public Member Functions

- **object_groups_data** (const object_links< L > &links)
- **object_groups_data** (const object_links< L > &links, const **mln::util::array**< group_info > &info)

Public Attributes

- **mln::util::array**< **unsigned** > **comp_to_group_**
- component_set< L > **components_**
- **mln::util::array**< group_info > **group_info_**
- object_links< L > **links_**

8.39.1 Detailed Description

template<typename L>struct scribo::internal::object_groups_data< L >

Data structure for `scribo::object_groups<I>`.

Definition at line 71 of file object_groups.hh.

8.40 scribo::internal::object_links_data< L > Struct Template Reference

```
#include <scribo/core/object_links.hh>
```

Public Member Functions

- **object_links_data** (const component_set< L > &components, **unsigned** size)
- **object_links_data** (const component_set< L > &components, **unsigned** size, **unsigned** default_link_id)

Public Attributes

- **mln::util::array< unsigned > comp_to_link_**
- component_set< L > **components_**

8.40.1 Detailed Description

template<typename L>struct scribo::internal::object_links_data< L >

Data structure for `scribo::object_links<I>`.

Definition at line 56 of file object_links.hh.

8.41 scribo::internal::paragraph_set_data< L > Struct Template Reference

```
#include <scribo/core/paragraph_set.hh>
```

Public Member Functions

- **paragraph_set_data** (const line_links< L > &llines, **unsigned** npars)

Public Attributes

- line_set< L > **lines_**
- line_links< L > **links_**
- **mln::util::array**
  < paragraph_info< L > > **pars_**

8.41.1 Detailed Description

template<typename L>struct scribo::internal::paragraph_set_data< L >

Data structure for `scribo::paragraph_set<I>`.

Definition at line 51 of file paragraph_set.hh.

8.42 scribo::internal::sort_comp_ids< L > Struct Template Reference

Public Member Functions

- **sort_comp_ids** (const component_set< L > &comp_set)
- **bool operator()** (const component_id_t &l, const component_id_t &r) const

Public Attributes

- component_set< L > **comps_**

8.42.1 Detailed Description

template<typename L>struct scribo::internal::sort_comp_ids< L >

Definition at line 44 of file sort_comp_ids.hh.

8.43 scribo::io::img::internal::debug_img_visitor< L > Class Template Reference

Public Types

- typedef **Object**< void > **category**
- typedef debug_img_visitor< L > **exact_t**

Public Member Functions

- **debug_img_visitor** (**mln::image2d**< **value::rgb8** > &out, **unsigned** output_ratio)
- void **visit** (const line_links< L > &llinks) const
- void **visit** (const object_groups< L > &groups) const
- void **visit** (const object_links< L > &links) const
- void **visit** (const component_set< L > &comp_set) const
- void **visit** (const document< L > &doc) const
- void **visit** (const component_info< L > &info) const
- void **visit** (const paragraph_set< L > &parset) const
- void **visit** (const line_info< L > &**line**) const

8.43.1 Detailed Description

template<typename L>class scribo::io::img::internal::debug_img_visitor< L >

Definition at line 66 of file debug_img_visitor.hh.

8.44 scribo::io::img::internal::full_img_visitor< L > Class Template Reference

Public Types

- typedef **Object**< void > **category**
- typedef full_img_visitor< L > **exact_t**

Public Member Functions

- **full_img_visitor** (**mln::image2d**< **value::rgb8** > &out)
- void **visit** (const line_links< L > &llinks) const
- void **visit** (const object_groups< L > &groups) const
- void **visit** (const object_links< L > &links) const
- void **visit** (const component_set< L > &comp_set) const
- void **visit** (const document< L > &doc) const
- void **visit** (const component_info< L > &info) const
- void **visit** (const paragraph_set< L > &parset) const
- void **visit** (const line_info< L > &**line**) const

8.44.1 Detailed Description

template<typename L>class scribo::io::img::internal::full_img_visitor< L >

Definition at line 63 of file full_img_visitor.hh.

8.45 scribo::io::img::internal::non_text_img_visitor Class Reference

```
#include <scribo/io/img/internal/non_text_img_visitor.hh>
```

Public Types

- typedef **Object**< void > **category**
- typedef non_text_img_visitor **exact_t**

Public Member Functions

- **non_text_img_visitor** (**mln::image2d**< **value::rgb8** > &out)
- void **visit** (const line_links< L > &llinks) const
- void **visit** (const object_groups< L > &groups) const
- void **visit** (const object_links< L > &links) const
- template<typename L >
  void **visit** (const document< L > &doc) const
- template<typename L >
  void **visit** (const component_set< L > &comp_set) const
- template<typename L >
  void **visit** (const component_info< L > &info) const
- void **visit** (const paragraph_set< L > &parset) const
- void **visit** (const line_info< L > &**line**) const

8.45.1 Detailed Description

Save non-text information as an image.

Definition at line 57 of file non_text_img_visitor.hh.

8.46 scribo::io::img::internal::text_img_visitor Class Reference

Public Types

- typedef **Object**< void > **category**
- typedef text_img_visitor **exact_t**

Public Member Functions

- **text_img_visitor** (**mln::image2d**< **value::rgb8** > &)
- void **visit** (const line_links< L > &llinks) const
- void **visit** (const object_groups< L > &groups) const
- void **visit** (const object_links< L > &links) const
- template<typename L >
  void **visit** (const document< L > &doc) const

- void **visit** (const component_info< L > &info) const
- template<typename L >
  void **visit** (const component_set< L > &comp_set) const
- template<typename L >
  void **visit** (const paragraph_set< L > &parset) const
- template<typename L >
  void **visit** (const line_info< L > &**line**) const

8.46.1 Detailed Description

Definition at line 59 of file text_img_visitor.hh.

8.47 scribo::io::xml::internal::color_t Struct Reference

Public Attributes

- const char ∗ **name**
- **float res**
- const **algebra::vec**
  < 3, **value::int_u8** > **v**

8.47.1 Detailed Description

Definition at line 57 of file compute_text_colour.hh.

8.48 scribo::io::xml::internal::extended_page_xml_visitor< L > Class Template Reference

Public Types

- typedef **Object**< void > **category**
- typedef
  extended_page_xml_visitor< L > **exact_t**

Public Member Functions

- **extended_page_xml_visitor** (std::ofstream &out)
- void **visit** (const line_links< L > &llinks) const
- void **visit** (const object_groups< L > &groups) const
- void **visit** (const object_links< L > &links) const
- void **visit** (const document< L > &doc) const
- void **visit** (const component_set< L > &comp_set) const
- void **visit** (const component_info< L > &info) const
- void **visit** (const paragraph_set< L > &parset) const
- void **visit** (const line_info< L > &**line**) const

8.48.1 Detailed Description

template<typename L>class scribo::io::xml::internal::extended_page_xml_visitor< L >

Definition at line 75 of file extended_page_xml_visitor.hh.

8.49 scribo::io::xml::internal::full_xml_visitor Class Reference

Public Types

- typedef **Object**< void > **category**
- typedef full_xml_visitor **exact_t**

Public Member Functions

- **full_xml_visitor** (std::ofstream &out)
- template<typename L >
  void **visit** (const document< L > &doc) const
- template<typename L >
  void **visit** (const line_links< L > &llinks) const
- template<typename L >
  void **visit** (const object_groups< L > &groups) const
- template<typename L >
  void **visit** (const object_links< L > &links) const
- template<typename L >
  void **visit** (const component_set< L > &comp_set) const
- template<typename L >
  void **visit** (const component_info< L > &info) const
- template<typename L >
  void **visit** (const paragraph_set< L > &parset) const
- template<typename L >
  void **visit** (const line_info< L > &**line**) const

8.49.1 Detailed Description

Definition at line 67 of file full_xml_visitor.hh.

8.50 scribo::io::xml::internal::page_xml_visitor< L > Class Template Reference

```
#include <scribo/io/xml/internal/page_xml_visitor.hh>
```

Public Types

- typedef **Object**< void > **category**
- typedef page_xml_visitor< L > **exact_t**

Public Member Functions

- **page_xml_visitor** (std::ofstream &out)
- void **visit** (const line_links< L > &llinks) const
- void **visit** (const object_groups< L > &groups) const
- void **visit** (const object_links< L > &links) const
- void **visit** (const line_info< L > &**line**) const
- void **visit** (const document< L > &doc) const
- void **visit** (const component_set< L > &comp_set) const
- void **visit** (const component_info< L > &info) const
- void **visit** (const paragraph_set< L > &parset) const

8.50.1 Detailed Description

template<typename L>class scribo::io::xml::internal::page_xml_visitor< L >

Save document information as XML.

We use a XML Schema part of the PAGE (Page Analysis and Ground truth Elements) image representation framework.

This schema was used in the Historical Document Layout Analysis COMPetition (HDLAC) for ICDAR 2011.

Its XSD file is located here: http://schema.primaresearch.org/PAGE/gts/pagecontent/2010-03-19/pagecc xsd

Definition at line 72 of file page_xml_visitor.hh.

8.51 scribo::layout::internal::hist_info Struct Reference

Public Attributes

- **unsigned horizontal**
- **unsigned vertical**

8.51.1 Detailed Description

Definition at line 44 of file hist_info.hh.

8.52 scribo::layout::internal::node< B > Class Template Reference

Public Member Functions

- **node** (const B &**bbox**)
- const B & **get_bbox** () const
- node< B > ∗ **get_ls** ()
- const node< B > ∗ **get_ls** () const
- node< B > ∗ **get_rs** ()
- const node< B > ∗ **get_rs** () const
- **bool is_leaf** () const
- void **set_ls** (node< B > ∗n)
- void **set_rs** (node< B > ∗n)

8.52.1 Detailed Description

template<typename B>class scribo::layout::internal::node< B >

Definition at line 47 of file node.hh.

8.53 scribo::line_info< L > Class Template Reference

Public Types

- typedef **Object**< void > **category**
- typedef line_info< L > **exact_t**

Public Member Functions

- **int a_height** () const
- void accept (const SerializeVisitor< E2 > &visitor) const
- **int ascent** () const
- **int baseline** () const
- const **mln::box2d** & **bbox** () const
- **float boldness** () const
- **float boldness_reliability** () const
- **unsigned card** () const
- **unsigned char_space** () const
- **unsigned char_width** () const
- **bool chars_same_width** () const
- const **mln::value::rgb8** & **color** () const
- **float color_reliability** () const
- const **mln::util::array**
  < **component_id_t** > & **component_ids** () const
- **int d_height** () const
- **int** delta_of_line () const
- **int descent** () const
- const **mln::box2d** & ebbox () const
- void force_stats_update ()
- **unsigned get_first_char_height** () const
- **bool has_text** () const
- const std::string & **html_text** () const
- **line_id_t** id () const
- **bool indented** () const
- **bool is_textline** () const
- **bool is_valid** () const
- **int meanline** () const
- line_info< L > & operator= (const line_info< L > &other)
- **float orientation** () const
- **unsigned pixel_area** () const
- line::ReadingDirection **reading_direction** () const
- **float reading_orientation** () const
- **bool reverse_video** () const
- line::Tag **tag** () const
- const std::string & **text** () const
- **float text_confidence** () const
- line::Type **type** () const
- void update_ebbox ()
- void **update_tag** (line::Tag tag)
- void **update_text** (const std::string &str, **float** confidence=100.0f)
- void **update_type** (line::Type type)
- **unsigned word_space** () const
- **unsigned x_height** () const

- line_info ()
- line_info (const **line_id_t** &id, data_t ∗**data**)
- line_info (const line_set< L > &holder, const **line_id_t** &id, const group_info &group)
- line_info (const line_info< L > &other)

- **bool** is_hidden () const
- void set_hidden (**bool** b)

- void fast_merge (line_info< L > &other, **bool** hide=true)
- void precise_merge (line_info< L > &other, **bool** hide=true)

8.53.1 Detailed Description

template<typename L>class scribo::line_info< L >

Definition at line 160 of file line_info.hh.

8.53.2 Constructor & Destructor Documentation

8.53.2.1 template<typename L> scribo::line_info< L >::line_info ( )

Constructors.

8.53.2.2 template<typename L> scribo::line_info< L >::line_info (const line_id_t & *id*, data_t ∗ *data*)

Constructors.

8.53.2.3 template<typename L> scribo::line_info< L >::line_info (const line_set< L > & *holder*, const line_id_t & *id*, const group_info & *group*)

Constructors.

8.53.2.4 template<typename L> scribo::line_info< L >::line_info (const line_info< L > & *other*)

The line id of the target instance is preserved if it is valid.

8.53.3 Member Function Documentation

8.53.3.1 void scribo::Serializable< line_info< L > >::accept (const SerializeVisitor< E2 > & *visitor*) const `[inherited]`

Allow this object to be serialized by `visitor`.

8.53.3.2 template<typename L> int scribo::line_info< L >::delta_of_line ( ) const

Returns the delta used to compute the extended bbox.

8.53.3.3 template<typename L> const mln::box2d& scribo::line_info< L >::ebbox ( ) const

Extended bounding box.

The width is extended with char_width() + char_space() on each side. The height is adjusted to max(a_height, -d_height) on each side.

8.53.3.4 template<typename L> void scribo::line_info< L >::fast_merge (line_info< L > & *other*, bool *hide* = `true`)

Merge related routines.

This merge only updates the component list and the bounding box.

After this merge, the line is tagged with line::Needs_Precise_Stats_Update.

The `other` line is tagged with line::Merged and if `hide` is set to 'True', it is set as hidden as well.

8.53.3.5 template<typename L> void scribo::line_info< L >::force_stats_update ( )

Force a new computation of statistics.

8.53.3.6 template<typename L> line_id_t scribo::line_info< L >::id ( ) const

If the line info is valid, the line id never changes for a given instance.

8.53.3.7 template<typename L> bool scribo::line_info< L >::is_hidden ( ) const

Hidden status.

When a line is hidden, it should not be used in routines computing data over lines.

8.53.3.8 template<typename L> line_info<L>& scribo::line_info< L >::operator= (const line_info< L > & *other*)

The line id of the target instance is preserved if it is valid.

8.53.3.9 template<typename L> void scribo::line_info< L >::precise_merge (line_info< L > & *other*, bool *hide* = true)

This merge updates the component list and recompute from scratch statistics, bounding box and other line attributes.

After this merge, the line is tagged with line::None.

8.53.3.10 template<typename L> void scribo::line_info< L >::set_hidden (bool *b*)

Hidden status.

When a line is hidden, it should not be used in routines computing data over lines.

8.53.3.11 template<typename L> void scribo::line_info< L >::update_ebbox ( )

Update the extended bbox.

8.54 scribo::line_links< L > Class Template Reference

```
#include <scribo/core/line_links.hh>
```

Public Types

- typedef **Object**< void > **category**
- typedef line_links< L > **exact_t**

Public Member Functions

- **line_links** (const line_set< L > &lines)
- **line_links** (const line_set< L > &lines, **line_id_t value**)
- void accept (const SerializeVisitor< E2 > &visitor) const
- line_links< L > **duplicate** () const
- void **init** ()

- **bool is_valid** () const
- const **mln::util::array**
  < **line_id_t** > & **line_to_link** () const
- const line_set< L > & **lines** () const
- **unsigned nelements** () const
- **line_id_t** & **operator()** (**line_id_t** comp_id)
- const **line_id_t** & **operator()** (**line_id_t** comp_id) const

8.54.1 Detailed Description

template<typename L>class scribo::line_links< L >

Line links representation.

Definition at line 75 of file line_links.hh.

8.54.2 Member Function Documentation

8.54.2.1 void scribo::Serializable< line_links< L > >::accept (const SerializeVisitor< E2 > & *visitor*) const [inherited]

Allow this object to be serialized by `visitor`.

8.55 scribo::line_set< L > Class Template Reference

```
#include <scribo/core/line_set.hh>
```

Public Member Functions

- const component_set< L > & components () const
- component_set< L > & components_ ()
- void compute_lines (const object_groups< L > &groups)
- line_set< L > duplicate () const
- const object_groups< L > & groups () const
- const line_info< L > & info (const typename **L::value** &id) const
- line_info< L > & info (const typename **L::value** &id)
- const **mln::util::array**
  < line_info< L > > & infos () const
- **bool** is_valid () const
- const object_links< L > & links () const
- **L::value** nelements () const
- line_info< L > & operator() (const **line_id_t** &id)
- const line_info< L > & operator() (const **line_id_t** &id) const
- void **update_line_data_** (const **mln::util::array**< line_info< L > > &line_data)
- template<typename F >
  void update_tags (const **mln::Function_v2b**< F > &f, line::Tag tag)
- template<typename F >
  void update_types (const **mln::Function_v2b**< F > &f, line::Type type)

- line_set ()
- line_set (const object_groups< L > &groups)
- line_set (const object_groups< L > &groups, const **mln::util::array**< line_info< L > > &line_data)

- void force_stats_update ()
- const **mln::util::array**
  < scribo::line_info< L > > & infos_ () const

8.55.1 Detailed Description

template<typename L>class scribo::line_set< L >

Lines container.

Line ids start from 1.

Definition at line 94 of file line_set.hh.

8.55.2 Constructor & Destructor Documentation

8.55.2.1 template<typename L> scribo::line_set< L >::line_set ( )

Constructors

Constructor without argument.

8.55.2.2 template<typename L> scribo::line_set< L >::line_set (const object_groups< L > & *groups*)

Constructor from object groups.

8.55.2.3 template<typename L> scribo::line_set< L >::line_set (const object_groups< L > & *groups*, const mln::util::array< line_info< L > > & *line_data*)

Constructor useful for delayed construction (loading from file).

8.55.3 Member Function Documentation

8.55.3.1 template<typename L> const component_set<L>& scribo::line_set< L >::components ( ) const

Return the underlying component set.

8.55.3.2 template<typename L> component_set<L>& scribo::line_set< L >::components_ ( )

Return the underlying component set (non-const version).

8.55.3.3 template<typename L> void scribo::line_set< L >::compute_lines (const object_groups< L > & *groups*)

Compute line stats and fill the underlying information.

8.55.3.4 template<typename L> line_set<L> scribo::line_set< L >::duplicate ( ) const

Create a copy of this line_set<L>

8.55.3.5 template<typename L> void scribo::line_set< L >::force_stats_update ( )

Massive line computation.

8.55.3.6 template<typename L> const object_groups<L>& scribo::line_set< L >::groups ( ) const

Return the underlying component group.

8.55.3.7 template<typename L> const line_info<L>& scribo::line_set< L >::info (const typename L::value & *id*) const

Return line information for a given line id `id`.

8.55.3.8 template<typename L> line_info<L>& scribo::line_set< L >::info (const typename L::value & *id*)

Return line information for a given line id `id`.

8.55.3.9 template<typename L> const mln::util::array<line_info<L> >& scribo::line_set< L >::infos ( ) const

Return all the line information.

8.55.3.10 template<typename L> const mln::util::array<scribo::line_info<L> >& scribo::line_set< L >::infos_ ( ) const

Internal methods.

Return all the line infos.

8.55.3.11 template<typename L> bool scribo::line_set< L >::is_valid ( ) const

Return false if it is not initialized (built with the default constructor).

8.55.3.12 template<typename L> const object_links<L>& scribo::line_set< L >::links ( ) const

Return the underlying links.

8.55.3.13 template<typename L> L ::value scribo::line_set< L >::nelements ( ) const

Return the line count.

8.55.3.14 template<typename L> line_info<L>& scribo::line_set< L >::operator() (const line_id_t & *id*)

Return line information for a given line id `id`.

8.55.3.15 template<typename L> const line_info<L>& scribo::line_set< L >::operator() (const line_id_t & *id*) const

Return line information for a given line id `id`.

8.55.3.16 template<typename L> template<typename F > void scribo::line_set< L >::update_tags (const mln::Function_v2b< F > & *f*, line::Tag *tag*)

Update tag of lines set to 'false' in `f` with `tag`.

8.55.3.17 template<typename L> template<typename F > void scribo::line_set< L >::update_types (const mln::Function_v2b< F > & *f*, line::Type *type*)

Update Type of lines set to 'false' in `f` with `tag`.

8.56 scribo::Link_Functor< E > Class Template Reference

```
#include <scribo/core/concept/link_functor.hh>
```

Public Types

- typedef **Object**< void > **category**
- typedef E **exact_t**

8.56.1 Detailed Description

template<typename E>class scribo::Link_Functor< E >

Link functor concept.

Definition at line 43 of file link_functor.hh.

8.57 scribo::object_groups< L > Class Template Reference

```
#include <scribo/core/object_groups.hh>
```

Public Types

- typedef **Object**< void > **category**
- typedef object_groups< L > **exact_t**

Public Member Functions

- **object_groups** (const object_links< L > &links)
- **object_groups** (const object_links< L > &links, const **mln::util::array**< group_info > &info)
- void accept (const SerializeVisitor< E2 > &visitor) const
- const **mln::util::array**
  < **unsigned** > & **comp_to_group** () const
- const component_set< L > & **components** () const
- object_groups< L > **duplicate** () const
- const group_info & group_of (**unsigned** comp_id) const
- group_info & **group_of** (**unsigned** comp_id)
- **bool is_valid** () const
- const object_links< L > & **links** () const
- void **merge** (**unsigned** group_id_from, **unsigned** group_id_to)
- **unsigned nelements** () const
- const group_info & operator() (**unsigned** group_id) const
- group_info & **operator()** (**unsigned** group_id)

8.57.1 Detailed Description

template<typename L>class scribo::object_groups< L >

Object group representation.

Definition at line 95 of file object_groups.hh.

8.57.2 Member Function Documentation

8.57.2.1 void **scribo::Serializable< object_groups< L > >**::accept (const **SerializeVisitor**< E2 > & *visitor*) const [inherited]

Allow this object to be serialized by `visitor`.

8.57.2.2 template<typename L> const **group_info& scribo::object_groups**< L >::group_of (**unsigned** *comp_id*) const

Return the group id of the component `comp_id`.

8.57.2.3 template<typename L> const **group_info& scribo::object_groups**< L >::operator() (**unsigned** *group_id*) const

Return group info data for group with id `group_id`.

Valid id starts from 1.

8.58 scribo::object_links< L > Class Template Reference

`#include <scribo/core/object_links.hh>`

Public Types

- typedef **Object**< void > **category**
- typedef object_links< L > **exact_t**

Public Member Functions

- object_links ()
- object_links (const component_set< L > &components)
- object_links (const component_set< L > &components, **unsigned** default_link_id)
- void accept (const SerializeVisitor< E2 > &visitor) const
- const **mln::util::array**
  < **unsigned** > & comp_to_link () const
- const component_set< L > & components () const
- object_links< L > duplicate () const
- **bool** has_linking_enabled (**unsigned** comp_id) const
- void init ()
- **bool** is_linked (**unsigned** comp_id) const
- **bool** is_valid () const
- **unsigned** nelements () const

- void update (**unsigned** from_id, **unsigned** to_id)
- void clear (**unsigned** id)
- void disable_linking (**unsigned** id)
- const **unsigned** & operator() (**unsigned** comp_id) const

Related Functions

(Note that these are not member functions.)

- template<typename L >
  std::ostream & operator<< (std::ostream &ostr, const object_links< L > &links)

8.58.1 Detailed Description

template<typename L>class scribo::object_links< L >

Object links representation.

This structure is meant to store link information between components. Linking components can be considered as a first step towards component grouping.

It requires a component_set to be constructed. Each component existing in the component_set may have link in an object_link structure. If no component_set is used for construction, this object is invalid (

See Also

is_valid()).

Definition at line 88 of file object_links.hh.

8.58.2 Constructor & Destructor Documentation

8.58.2.1 template<typename L> scribo::object_links< L >::object_links ( )

Default constructor. **It** produces an invalid structure.

8.58.2.2 template<typename L> scribo::object_links< L >::object_links (const component_set< L > & *components*)

Construct a valid object_links.

Links is enabled for each valid component but no link is set. Invalid components links are disabled.

8.58.2.3 template<typename L> scribo::object_links< L >::object_links (const component_set< L > & *components*, unsigned *default_link_id*)

Construct a valid object_links.

Links is enabled for each valid component and set by default towards component with id `default_link_id`. Invalid components links are disabled.

8.58.3 Member Function Documentation

8.58.3.1 void scribo::Serializable< object_links< L > >::accept (const SerializeVisitor< E2 > & *visitor*) const [inherited]

Allow this object to be serialized by `visitor`.

8.58.3.2 template<typename L> void scribo::object_links< L >::clear (unsigned *id*)

Reset link for component with id `id`.

This component can be linked later.

8.58.3.3 template<typename L> const mln::util::array<unsigned>& scribo::object_links< L >::comp_to_link ( ) const

Returns the underlying array encoding the component links.

Indexes in array correspond to component ids and the corresponding value is the component id involved in the link.

8.58.3.4 template<typename L> const component_set<L>& scribo::object_links< L >::components ( ) const

Return the underlying component_set.

8.58.3.5 template<typename L> void scribo::object_links< L >::disable_linking (unsigned *id*)

Do not allow component with id `id` to be linked to another component.

8.58.3.6 template<typename L> object_links<L> scribo::object_links< L >::duplicate ( ) const

Make a deep copy of this structure.

8.58.3.7 template<typename L> bool scribo::object_links< L >::has_linking_enabled (unsigned *comp_id*) const

Return True if component `comp_id` can be linked to another component.

8.58.3.8 template<typename L> void scribo::object_links< L >::init ( )

Initialize links.

Each component is linked to itself (i.e. has no link). Invalid components have linking disabled.

8.58.3.9 template<typename L> bool scribo::object_links< L >::is_linked (unsigned *comp_id*) const

Return True if component `comp_id` has a link starting from itself to another component.

8.58.3.10 template<typename L> bool scribo::object_links< L >::is_valid ( ) const

Return True if this object_links structure is correctly constructed.

8.58.3.11 template<typename L> unsigned scribo::object_links< L >::nelements ( ) const

Return the number of links.

This is equivalent to the number of components + the background.

8.58.3.12 template<typename L> const unsigned& scribo::object_links< L >::operator() (unsigned *comp_id*) const

Get link id for component `comp_id`.

8.58.3.13 template<typename L> void scribo::object_links< L >::update (unsigned *from_id,* unsigned *to_id*)

Link related methods.

Set link between component `from_id` and `to_id`.

8.58.4 Friends And Related Function Documentation

8.58.4.1 template<typename L > std::ostream & operator<< (std::ostream & *ostr,* const object_links< L > & *links*) `[related]`

8.59 scribo::paragraph_info< L > Class Template Reference

`#include <scribo/core/paragraph_info.hh>`

Public Member Functions

- **paragraph_info** (const line_links< L > &llinks)
- void add_line (const line_info< L > &line)
- const **mln::box2d** & **bbox** () const
- const **mln::value::rgb8** & **color** () const
- **float color_reliability** () const
- **int delta_baseline** () const
- void **fast_merge** (paragraph_info< L > &info)
- void **force_stats_update** ()
- void **invalidate** ()
- **bool is_valid** () const
- const line_info< L > & **line** (**line_id_t** id) const
- const **mln::util::array**
  < **line_id_t** > & **line_ids** () const
- const line_links< L > & **llinks** () const
- **bool needs_stats_update** () const
- **unsigned nlines** () const
- void **set_color_** (const **mln::value::rgb8** &v)
- void **set_color_reliability_** (**float** v)
- void **set_delta_baseline** (const **int** delta_baseline)
- paragraph::Tag **tag** () const
- void **update_tag** (paragraph::Tag tag)

8.59.1 Detailed Description

template<typename L>class scribo::paragraph_info< L >

Paragraph structure information.

Definition at line 43 of file paragraph_info.hh.

8.59.2 Member Function Documentation

8.59.2.1 template<typename L> void scribo::paragraph_info< L >::add_line (const line_info< L > & *line*)

Add a new line to this paragraph.

This method is provided for an incremental construction.

Once this method has been called, needs_stats_update() will return true until force_stats_update() is called.

8.60 scribo::paragraph_set< L > Class Template Reference

```
#include <scribo/core/paragraph_set.hh>
```

Public Types

- typedef **Object**< void > **category**
- typedef paragraph_set< L > **exact_t**

Public Member Functions

- **paragraph_set** (internal::paragraph_set_data< L > ∗**data**)
- **paragraph_set** (const line_links< L > &llinks, **unsigned** npars)
- void accept (const SerializeVisitor< E2 > &visitor) const
- paragraph_set< L > **duplicate** () const
- template<typename F >
  void **invalidate** (const **Function_v2b**< F > &f)
- **bool is_valid** () const
- const line_set< L > & **lines** () const
- const line_links< L > & **links** () const
- **unsigned nelements** () const
- paragraph_info< L > & **operator()** (const **paragraph_id_t** &i)
- const paragraph_info< L > & **operator()** (const **paragraph_id_t** &i) const

8.60.1 Detailed Description

template<typename L>class scribo::paragraph_set< L >

Paragraph container.

Paragraph ids start from 1.

Definition at line 71 of file paragraph_set.hh.

8.60.2 Member Function Documentation

8.60.2.1 void scribo::Serializable< paragraph_set< L > >::accept (const SerializeVisitor< E2 > & *visitor*) const `[inherited]`

Allow this object to be serialized by `visitor`.

8.61 scribo::preprocessing::internal::Hough Class Reference

Public Member Functions

- **Hough** (**int** width, **int** height)
- **image2d**< **unsigned** > & **acc** ()
- **double get_cos** (**int** index) const
- **double get_sin** (**int** index) const
- **int height** () const
- void **look_up_table** ()
- **double mrho** () const
- **int mrhoi** () const
- **double mtheta** () const
- **int mthetai** () const
- **int width** () const

8.61.1 Detailed Description

Definition at line 54 of file deskew.hh.

8.62 scribo::preprocessing::internal::QCompare Struct Reference

Public Member Functions

- **bool operator()** (const s_angle &s1, const s_angle &s2)

8.62.1 Detailed Description

Definition at line 102 of file deskew.hh.

8.63 scribo::preprocessing::internal::s_angle Struct Reference

Public Attributes

- **unsigned max**
- **int pos**

8.63.1 Detailed Description

Definition at line 95 of file deskew.hh.

8.64 scribo::primitive::link::internal::dmax_default Class Reference

```
#include <scribo/primitive/link/internal/dmax_default.hh>
```

Public Types

- typedef **Object**< void > **category**
- typedef dmax_default **exact_t**

Public Member Functions

- **dmax_default** (**float** dmax_factor)
- **float compute_** (const **box2d** &b) const
- **float operator()** (const **box2d** &b) const

Protected Attributes

- **float dmax_factor_**

8.64.1 Detailed Description

Base class for dmax functors.

Definition at line 50 of file dmax_default.hh.

8.65 scribo::primitive::link::internal::dmax_functor_base< E > Class Template Reference

```
#include <scribo/primitive/link/internal/dmax_functor_base.hh>
```

Public Types

- typedef **Object**< void > **category**
- typedef E **exact_t**

Public Member Functions

- **dmax_functor_base** (**float** dmax_factor)
- **float operator()** (const **box2d** &b) const

Protected Attributes

- **float dmax_factor_**

8.65.1 Detailed Description

template<typename E>class scribo::primitive::link::internal::dmax_functor_base< E >

Base class for dmax functors.

Definition at line 57 of file dmax_functor_base.hh.

8.66 scribo::primitive::link::internal::dmax_hrules Class Reference

```
#include <scribo/primitive/link/internal/dmax_hrules.hh>
```

Public Types

- typedef **Object**< void > **category**
- typedef dmax_hrules **exact_t**

Public Member Functions

- **dmax_hrules** (**float** dmax_factor, **unsigned** fixed_dmax)
- **float compute_** (const **box2d** &b) const
- **float operator()** (const **box2d** &b) const

Protected Attributes

- **float dmax_factor_**
- **unsigned fixed_dmax_**

8.66.1 Detailed Description

Base class for dmax functors.

Definition at line 50 of file dmax_hrules.hh.

8.67 scribo::primitive::link::internal::dmax_width_and_height Class Reference

```
#include <scribo/primitive/link/internal/dmax_width_and_height.hh>
```

Public Types

- typedef **Object**< void > **category**
- typedef dmax_width_and_height **exact_t**

Public Member Functions

- **dmax_width_and_height** (**float** dmax_factor)
- **float compute_** (const **box2d** &) const
- **float operator()** (const **box2d** &b) const

Protected Attributes

- **float dmax_factor_**

8.67.1 Detailed Description

Base class for dmax functors.

Definition at line 53 of file dmax_width_and_height.hh.

8.68 scribo::primitive::link::internal::dmax_width_only Class Reference

```
#include <scribo/primitive/link/internal/dmax_width_only.hh>
```

Public Types

- typedef **Object**< void > **category**
- typedef dmax_width_only **exact_t**

Public Member Functions

- **float compute_** (const **box2d** &) const
- **float operator()** (const **box2d** &b) const

Protected Attributes

- **float dmax_factor_**

8.68.1 Detailed Description

Base class for dmax functors.

Definition at line 53 of file dmax_width_only.hh.

8.69 scribo::primitive::link::internal::link_functor_base< L, E > Class Template Reference

```
#include <scribo/primitive/link/internal/link_functor_base.hh>
```

Public Types

- typedef **Object**< void > **category**
- typedef component_set< L > **component_set_t**
- typedef **mln::util::couple**
  < anchor::Type, P > **couple_t**
- typedef E **exact_t**
- typedef L::site **P**
- typedef L **support**

Public Member Functions

- link_functor_base (const component_set< L > &components)
- const component_set< L > & **components** () const
- void **compute_next_site** (P &p)
- void **compute_next_site_** (P &p)
- **couple_t finalize_link** (**unsigned** current_object)
- **couple_t finalize_link_** (**unsigned** current_object)
- void **initialize_link** (**unsigned** current_object)
- void **initialize_link_** (**unsigned** current_object)
- void **invalidate_link** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- void invalidate_link (**unsigned** current_object, const P &start_point, const P &p)
- void **invalidate_link_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- **bool is_potential_link** (**unsigned** current_object, const P &start_point, const P &p) const
- **bool is_potential_link_** (**unsigned** current_object, const P &start_point, const P &p) const
- const L & **labeled_image** () const

- **unsigned link** (**unsigned** object) const
- const object_links< L > & **links** () const
- L::site **start_point** (**unsigned** current_object, anchor::Type anchor)
- L::site start_point (**unsigned** current_object)
- L::site **start_point_** (**unsigned** current_object, anchor::Type anchor)
- void **start_processing_object** (**unsigned** current_object)
- void **start_processing_object_** (**unsigned** current_object)
- **bool valid_link** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- **bool valid_link_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- void **validate_link** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- void validate_link (**unsigned** current_object, const P &start_point, const P &p)
- void **validate_link_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- **bool verify_link_criterion** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor) const
- **bool verify_link_criterion_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor) const

Protected Attributes

- const component_set< L > **components_**
- const L & **labeled_image_**
- object_links< L > **links_**

8.69.1 Detailed Description

template<typename L, typename E>class scribo::primitive::link::internal::link_functor_base< L, E >

Base class for link functors.

Definition at line 70 of file link_functor_base.hh.

8.69.2 Constructor & Destructor Documentation

8.69.2.1 template<typename L, typename E> scribo::primitive::link::internal::link_functor_base< L, E >::link_functor_base (const component_set< L > & *components*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.

8.69.3 Member Function Documentation

8.69.3.1 template<typename L, typename E> void scribo::primitive::link::internal::link_functor_base< L, E >::invalidate_link (unsigned *current_object,* const P & *start_point,* const P & *p*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.`anchor` is set to anchor::MassCenter.

8.69.3.2 template<typename L, typename E> L ::site scribo::primitive::link::internal::link_functor_base< L, E >::start_point (unsigned *current_object*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.`anchor` is set to anchor::MassCenter.

8.69.3.3 template<typename L, typename E> void scribo::primitive::link::internal::link_functor_base< L, E >::validate_link (unsigned *current_object,* const P & *start_point,* const P & *p*)

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.`anchor` is set to anchor::MassCenter.

8.70 scribo::primitive::link::internal::link_several_dmax_base< L, E > Class Template Reference

`#include <scribo/primitive/link/internal/link_several_dmax_base.hh>`

Public Types

- typedef **Object**< void > **category**
- typedef component_set< L > **component_set_t**
- typedef E **exact_t**
- typedef L **support**

Public Member Functions

- **link_several_dmax_base** (const component_set< L > &comps, **unsigned** neighb_max_distance)
- const **util::array**< anchor::Type > & **anchors** () const
- const component_set< L > & **components** () const
- void **compute_next_site** (P &p)
- void **compute_next_site_** (P &p)
- **couple_t finalize_link** (**unsigned** current_object)
- **couple_t finalize_link_** (**unsigned** current_object)
- void **initialize_link** (**unsigned** current_object)
- void **initialize_link_** (**unsigned** current_object)
- void **invalidate_link** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- void invalidate_link (**unsigned** current_object, const P &start_point, const P &p)
- void **invalidate_link_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- **bool is_potential_link** (**unsigned** current_object, const P &start_point, const P &p) const
- **bool is_potential_link_** (**unsigned** current_object, const P &start_point, const P &p) const
- const L & **labeled_image** () const
- **unsigned link** (**unsigned** object) const
- const object_links< L > & **links** () const
- **unsigned nanchors** () const
- L::site **start_point** (**unsigned** current_object, anchor::Type anchor)
- L::site start_point (**unsigned** current_object)
- L::site **start_point_** (**unsigned** current_object, anchor::Type anchor)
- void **start_processing_object** (**unsigned** current_object)
- void **start_processing_object_** (**unsigned** current_object)
- **bool valid_link** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- **bool valid_link_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- void **validate_link** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- void validate_link (**unsigned** current_object, const P &start_point, const P &p)
- void **validate_link_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- **bool verify_link_criterion** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor) const
- **bool verify_link_criterion_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor) const

Protected Attributes

- **mln::util::array**< anchor::Type > **anchors_**
- const component_set< L > **components_**
- anchor::Direction **direction_**
- **float dmax_**
- const L & **labeled_image_**
- object_links< L > **links_**
- **float neighb_max_distance_**
- **mln::util::array**< **couple_t** > **potential_links_**

8.70.1 Detailed Description

template<typename L, typename E>class scribo::primitive::link::internal::link_several_dmax_base< L, E >

Base class for link functors using several anchors and a maximum lookup distance.

Definition at line 70 of file link_several_dmax_base.hh.

8.70.2 Member Function Documentation

8.70.2.1 template<typename L, typename E> void scribo::primitive::link::internal::link_functor_base< L, E >::invalidate_link (unsigned *current_object,* const P & *start_point,* const P & *p*) `[inherited]`

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.`anchor` is set to anchor::MassCenter.

8.70.2.2 template<typename L, typename E> L ::site scribo::primitive::link::internal::link_functor_base< L, E >::start_point (unsigned *current_object*) `[inherited]`

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.`anchor` is set to anchor::MassCenter.

8.70.2.3 template<typename L, typename E> void scribo::primitive::link::internal::link_functor_base< L, E >::validate_link (unsigned *current_object,* const P & *start_point,* const P & *p*) `[inherited]`

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.`anchor` is set to anchor::MassCenter.

8.71 scribo::primitive::link::internal::link_single_dmax_base< L, E > Class Template Reference

```
#include <scribo/primitive/link/internal/link_single_dmax_base.hh>
```

Public Types

- typedef **Object**< void > **category**
- typedef component_set< L > **component_set_t**
- typedef **mln::util::couple**
  < anchor::Type, P > **couple_t**
- typedef E **exact_t**
- typedef L::site **P**
- typedef L **support**

Public Member Functions

- **link_single_dmax_base** (const component_set< L > &components, **unsigned** neighb_max_distance, anchor::Direction direction)
- const component_set< L > & **components** () const
- void **compute_next_site** (P &p)
- void **compute_next_site_** (P &p)
- **couple_t finalize_link** (**unsigned** current_object)
- **couple_t finalize_link_** (**unsigned** current_object)
- void **initialize_link** (**unsigned** current_object)
- void **initialize_link_** (**unsigned** current_object)
- void **invalidate_link** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- void invalidate_link (**unsigned** current_object, const P &start_point, const P &p)
- void **invalidate_link_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- **bool is_potential_link** (**unsigned** current_object, const P &start_point, const P &p) const
- **bool is_potential_link_** (**unsigned** current_object, const P &start_point, const P &p) const
- const L & **labeled_image** () const
- **unsigned link** (**unsigned** object) const
- const object_links< L > & **links** () const
- L::site **start_point** (**unsigned** current_object, anchor::Type anchor)
- L::site start_point (**unsigned** current_object)
- L::site **start_point_** (**unsigned** current_object, anchor::Type anchor)
- void **start_processing_object** (**unsigned** current_object)
- void **start_processing_object_** (**unsigned** current_object)
- **bool valid_link** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- **bool valid_link_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- void **validate_link** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- void validate_link (**unsigned** current_object, const P &start_point, const P &p)
- void **validate_link_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- **bool verify_link_criterion** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor) const
- **bool verify_link_criterion_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor) const

Protected Attributes

- const component_set< L > **components_**
- const L & **labeled_image_**
- object_links< L > **links_**

8.71.1 Detailed Description

template<typename L, typename E>class scribo::primitive::link::internal::link_single_dmax_base< L, E >

Base class for link functors using mass centers and a given max distance.

Definition at line 69 of file link_single_dmax_base.hh.

8.71.2 Member Function Documentation

8.71.2.1 template<typename L, typename E> void scribo::primitive::link::internal::link_functor_base< L, E >::invalidate_link (unsigned *current_object*, const P & *start_point*, const P & *p*) [inherited]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.`anchor` is set to anchor::MassCenter.

8.71.2.2 template<typename L, typename E> L ::site scribo::primitive::link::internal::link_functor_base< L, E >::start_point (unsigned *current_object*) [inherited]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.`anchor` is set to anchor::MassCenter.

8.71.2.3 template<typename L, typename E> void scribo::primitive::link::internal::link_functor_base< L, E >::validate_link (unsigned *current_object,* const P & *start_point,* const P & *p*) [inherited]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.`anchor` is set to anchor::MassCenter.

8.72 scribo::primitive::link::internal::link_single_dmax_ratio_aligned_base< L, F, E > Class Template Reference

Public Types

- typedef **Object**< void > **category**
- typedef component_set< L > **component_set_t**
- typedef **mln::util::couple**
  < anchor::Type, P > **couple_t**
- typedef E **exact_t**
- typedef L::site **P**
- typedef L **support**

Public Member Functions

- **link_single_dmax_ratio_aligned_base** (const component_set< L > &components, const DMax_Functor< F > &dmax_f, **float** min_angle, **float** max_angle, anchor::Type anchor)
- const component_set< L > & **components** () const
- void **compute_next_site** (P &p)
- void **compute_next_site_** (P &p)
- **couple_t finalize_link** (**unsigned** current_object)
- **couple_t finalize_link_** (**unsigned** current_object)
- void **initialize_link** (**unsigned** current_object)
- void **initialize_link_** (**unsigned** current_object)
- void **invalidate_link** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- void invalidate_link (**unsigned** current_object, const P &start_point, const P &p)
- void **invalidate_link_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- **bool is_potential_link** (**unsigned** current_object, const P &start_point, const P &p) const
- **bool is_potential_link_** (**unsigned** current_object, const P &start_point, const P &p) const
- const L & **labeled_image** () const
- **unsigned link** (**unsigned** object) const
- const object_links< L > & **links** () const
- L::site **start_point** (**unsigned** current_object, anchor::Type anchor)
- L::site start_point (**unsigned** current_object)
- L::site **start_point_** (**unsigned** current_object, anchor::Type anchor)
- void **start_processing_object** (**unsigned** current_object)
- void **start_processing_object_** (**unsigned** current_object)
- **bool valid_link** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- **bool valid_link_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- void **validate_link** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)

- void validate_link (**unsigned** current_object, const P &start_point, const P &p)
- void **validate_link_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- **bool verify_link_criterion** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor) const
- **bool verify_link_criterion_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor) const

Public Attributes

- **mln::trait::ch_value**< L, **value::rgb8** >::**ret debug_**
- anchor::Type **debug_anchor_**
- **mln::trait::ch_value**< L, **value::rgb8** >::**ret debug_angle_**
- **float max_alpha_rad**
- **float min_alpha_rad**

Protected Attributes

- const component_set< L > **components_**
- anchor::Direction **direction_**
- **float dmax_**
- const F **dmax_f_**
- const L & **labeled_image_**
- object_links< L > **links_**

8.72.1 Detailed Description

template<typename L, typename F, typename E>class scribo::primitive::link::internal::link_single_dmax_ratio_aligned_base< L, F, E >

Definition at line 72 of file link_single_dmax_ratio_aligned_base.hh.

8.72.2 Member Function Documentation

8.72.2.1 template<typename L, typename E> void scribo::primitive::link::internal::link_functor_base< L, E >::invalidate_link (unsigned *current_object,* const P & *start_point,* const P & *p*) [inherited]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.`anchor` is set to anchor::MassCenter.

8.72.2.2 template<typename L, typename E> L ::site scribo::primitive::link::internal::link_functor_base< L, E >::start_point (unsigned *current_object*) [inherited]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.`anchor` is set to anchor::MassCenter.

8.72.2.3 template<typename L, typename E> void scribo::primitive::link::internal::link_functor_base< L, E >::validate_link (unsigned *current_object,* const P & *start_point,* const P & *p*) [inherited]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.`anchor` is set to anchor::MassCenter.

8.73 scribo::primitive::link::internal::link_single_dmax_ratio_aligned_delta_base< L, F, E > Class Template Reference

Public Types

- typedef **Object**< void > **category**
- typedef component_set< L > **component_set_t**
- typedef **mln::util::couple**
  < anchor::Type, P > **couple_t**
- typedef E **exact_t**
- typedef L::site **P**
- typedef L **support**

Public Member Functions

- **link_single_dmax_ratio_aligned_delta_base** (const component_set< L > &components, const DMax_Functor< F > &dmax_f, **int** delta_pixel, anchor::Direction delta_direction)
- const component_set< L > & **components** () const
- void **compute_next_site** (P &p)
- void **compute_next_site_** (P &p)
- **couple_t finalize_link** (**unsigned** current_object)
- **couple_t finalize_link_** (**unsigned** current_object)
- void **initialize_link** (**unsigned** current_object)
- void **initialize_link_** (**unsigned** current_object)
- void **invalidate_link** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- void invalidate_link (**unsigned** current_object, const P &start_point, const P &p)
- void **invalidate_link_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- **bool is_potential_link** (**unsigned** current_object, const P &start_point, const P &p) const
- **bool is_potential_link_** (**unsigned** current_object, const P &start_point, const P &p) const
- const L & **labeled_image** () const
- **unsigned link** (**unsigned** object) const
- const object_links< L > & **links** () const
- L::site **start_point** (**unsigned** current_object, anchor::Type anchor)
- L::site start_point (**unsigned** current_object)
- L::site **start_point_** (**unsigned** current_object, anchor::Type anchor)
- void **start_processing_object** (**unsigned** current_object)
- void **start_processing_object_** (**unsigned** current_object)
- **bool valid_link** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- **bool valid_link_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- void **validate_link** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- void validate_link (**unsigned** current_object, const P &start_point, const P &p)
- void **validate_link_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- **bool verify_link_criterion** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor) const
- **bool verify_link_criterion_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor) const

Public Attributes

- anchor::Direction **delta_direction_**
- **int delta_pixel_**

Protected Attributes

- const component_set< L > **components_**
- anchor::Direction **direction_**
- **float dmax_**
- const F **dmax_f_**
- const L & **labeled_image_**
- object_links< L > **links_**

8.73.1 Detailed Description

template<typename L, typename F, typename E>class scribo::primitive::link::internal::link_single_dmax_ratio_aligned_delta_base< L, F, E >

Definition at line 72 of file link_single_dmax_ratio_aligned_delta_base.hh.

8.73.2 Member Function Documentation

8.73.2.1 template<typename L, typename E> void scribo::primitive::link::internal::link_functor_base< L, E >::invalidate_link (unsigned *current_object,* const P & *start_point,* const P & *p*) [inherited]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.`anchor` is set to anchor::MassCenter.

8.73.2.2 template<typename L, typename E> L ::site scribo::primitive::link::internal::link_functor_base< L, E >::start_point (unsigned *current_object*) [inherited]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.`anchor` is set to anchor::MassCenter.

8.73.2.3 template<typename L, typename E> void scribo::primitive::link::internal::link_functor_base< L, E >::validate_link (unsigned *current_object,* const P & *start_point,* const P & *p*) [inherited]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.`anchor` is set to anchor::MassCenter.

8.74 scribo::primitive::link::internal::link_single_dmax_ratio_base< L, F, E > Class Template Reference

```
#include <scribo/primitive/link/internal/link_single_dmax_ratio_base.hh>
```

Public Types

- typedef **Object**< void > **category**
- typedef component_set< L > **component_set_t**
- typedef **mln::util::couple**
  < anchor::Type, P > **couple_t**
- typedef E **exact_t**
- typedef L::site **P**
- typedef L **support**

Public Member Functions

- **link_single_dmax_ratio_base** (const component_set< L > &components, anchor::Direction direction, const DMax_Functor< F > &dmax_f)
- const component_set< L > & **components** () const
- void **compute_next_site** (P &p)
- void **compute_next_site_** (P &p)
- **couple_t finalize_link** (**unsigned** current_object)
- **couple_t finalize_link_** (**unsigned** current_object)
- void **initialize_link** (**unsigned** current_object)
- void **initialize_link_** (**unsigned** current_object)
- void **invalidate_link** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- void invalidate_link (**unsigned** current_object, const P &start_point, const P &p)
- void **invalidate_link_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- **bool is_potential_link** (**unsigned** current_object, const P &start_point, const P &p) const
- **bool is_potential_link_** (**unsigned** current_object, const P &start_point, const P &p) const
- const L & **labeled_image** () const
- **unsigned link** (**unsigned** object) const
- const object_links< L > & **links** () const
- L::site **start_point** (**unsigned** current_object, anchor::Type anchor)
- L::site start_point (**unsigned** current_object)
- L::site **start_point_** (**unsigned** current_object, anchor::Type anchor)
- void **start_processing_object** (**unsigned** current_object)
- void **start_processing_object_** (**unsigned** current_object)
- **bool valid_link** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- **bool valid_link_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- void **validate_link** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- void validate_link (**unsigned** current_object, const P &start_point, const P &p)
- void **validate_link_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor)
- **bool verify_link_criterion** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor) const
- **bool verify_link_criterion_** (**unsigned** current_object, const P &start_point, const P &p, anchor::Type anchor) const

Protected Attributes

- const component_set< L > **components_**
- anchor::Direction **direction_**
- **float dmax_**
- const F **dmax_f_**
- const L & **labeled_image_**
- object_links< L > **links_**

8.74.1 Detailed Description

template<typename L, typename F, typename E>class scribo::primitive::link::internal::link_single_dmax_ratio_base< L, F, E >

Base class for link functors using bounding box center and a proportional max distance.

Definition at line 72 of file link_single_dmax_ratio_base.hh.

8.74.2 Member Function Documentation

8.74.2.1 template<typename L, typename E> void scribo::primitive::link::internal::link_functor_base< L, E >::invalidate_link (unsigned *current_object*, const P & *start_point*, const P & *p*) [inherited]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.`anchor` is set to anchor::MassCenter.

8.74.2.2 template<typename L, typename E> L ::site scribo::primitive::link::internal::link_functor_base< L, E >::start_point (unsigned *current_object*) [inherited]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.`anchor` is set to anchor::MassCenter.

8.74.2.3 template<typename L, typename E> void scribo::primitive::link::internal::link_functor_base< L, E >::validate_link (unsigned *current_object*, const P & *start_point*, const P & *p*) [inherited]

This is an overloaded member function, provided for convenience. **It** differs from the above function only in what argument(s) it accepts.`anchor` is set to anchor::MassCenter.

8.75 scribo::Serializable< E > Class Template Reference

`#include <scribo/core/concept/serializable.hh>`

Public Types

- typedef **Object**< void > **category**
- typedef E **exact_t**

Public Member Functions

- template<typename E2 >
  void accept (const SerializeVisitor< E2 > &visitor) const

8.75.1 Detailed Description

template<typename E>class scribo::Serializable< E >

Concept for objects that can be serialized.

Definition at line 42 of file serializable.hh.

8.75.2 Member Function Documentation

8.75.2.1 template<typename E> template<typename E2 > void scribo::Serializable< E >::accept (const SerializeVisitor< E2 > & *visitor*) const

Allow this object to be serialized by `visitor`.

8.76 scribo::SerializeVisitor< E > Class Template Reference

```
#include <scribo/core/concept/serialize_visitor.hh>
```

Public Types

- typedef **Object**< void > **category**
- typedef E **exact_t**

8.76.1 Detailed Description

template<typename E>class scribo::SerializeVisitor< E >

Link functor concept.

Definition at line 40 of file serialize_visitor.hh.

8.77 scribo::toolchain::internal::content_in_doc_functor< I > Struct Template Reference

```
#include <scribo/toolchain/internal/content_in_doc_functor.hh>
```

Public Types

- typedef **mln::trait::ch_value**
  < I, **V** >::**ret L**
- typedef **scribo::def::lbl_type V**

Public Member Functions

- **content_in_doc_functor** (const char ∗doc_filename)
- virtual **int nsteps** () const
- virtual void **on_end** ()
- virtual void **on_new_progress_label** (const char ∗label)
- virtual void **on_progress** ()
- virtual void **on_start** ()
- virtual void **on_xml_saved** ()
- template<typename J >
  scribo::document< L > **operator()** (const **Image**< J > &original_image, const **Image**< I > &processed_image)

Public Attributes

- document< L > **doc**
- **bool enable_denoising**
- **bool enable_line_seps**
- **bool enable_ocr**
- **bool enable_whitespace_seps**
- **mln::util::timer gt**
- std::string **ocr_language**
- std::string **output_file**
- **bool save_doc_as_xml**

- **mln::util::timer t**
- **bool verbose**
- scribo::io::xml::Format **xml_format**

8.77.1 Detailed Description

template<typename I>struct scribo::toolchain::internal::content_in_doc_functor< I >

Functor analysing and extracting document image content.

Definition at line 103 of file content_in_doc_functor.hh.

8.78 scribo::toolchain::internal::content_in_hdoc_functor< I > Struct Template Reference

```
#include <scribo/toolchain/internal/content_in_hdoc_functor.hh>
```

Public Types

- typedef **mln::trait::ch_value**
  < I, **V** >::**ret L**
- typedef **scribo::def::lbl_type V**

Public Member Functions

- **content_in_hdoc_functor** (const char ∗doc_filename)
- virtual **int nsteps** () const
- virtual void **on_end** ()
- virtual void **on_new_progress_label** (const char ∗label)
- virtual void **on_progress** ()
- virtual void **on_start** ()
- virtual void **on_xml_saved** ()
- template<typename J >
  scribo::document< L > **operator()** (const **Image**< J > &original_image, const **Image**< I > &processed_image)

Public Attributes

- document< L > **doc**
- **bool enable_denoising**
- **bool enable_line_seps**
- **bool enable_ocr**
- **bool enable_whitespace_seps**
- **mln::util::timer gt**
- std::string **ocr_language**
- std::string **output_file**
- **bool save_doc_as_xml**
- **mln::util::timer t**
- **bool verbose**
- scribo::io::xml::Format **xml_format**

8.78.1 Detailed Description

template<typename I>struct scribo::toolchain::internal::content_in_hdoc_functor< I >

Functor analysing and extracting content in degraded/historical documents.

Definition at line 116 of file content_in_hdoc_functor.hh.

8.79 scribo::toolchain::internal::text_in_doc_functor< I > Struct Template Reference

```
#include <scribo/toolchain/internal/text_in_doc_functor.hh>
```

Public Types

- typedef **mln::trait::ch_value**
  < I, **V** >::**ret L**
- typedef **scribo::def::lbl_type V**

Public Member Functions

- virtual **int nsteps** () const
- virtual void **on_end** ()
- virtual void **on_new_progress_label** (const char ∗label)
- virtual void **on_progress** ()
- virtual void **on_start** ()
- line_set< L > **operator()** (const **Image**< I > &input_)

Public Attributes

- **bool enable_denoising**
- **bool enable_line_seps**
- **bool enable_whitespace_seps**
- **mln::util::timer gt**
- std::string **ocr_language**
- line_set< L > **output**
- **mln::util::timer t**
- **bool verbose**

8.79.1 Detailed Description

template<typename I>struct scribo::toolchain::internal::text_in_doc_functor< I >

Functor extracting text lines from a document image.

Definition at line 87 of file text_in_doc_functor.hh.

8.80 scribo::toolchain::internal::text_in_doc_preprocess_functor< I > Struct Template Reference

```
#include <scribo/toolchain/internal/text_in_doc_preprocess_functor.hh>
```

Public Member Functions

- virtual **int nsteps** () const
- virtual void **on_end** ()
- virtual void **on_new_progress_label** (const char ∗label)
- virtual void **on_progress** ()
- virtual void **on_start** ()
- **mln::trait::ch_value**< I, **bool** >
  ::**ret operator()** (const **Image**< I > &input_)

Public Attributes

- **mln::trait::concrete**< I >::**ret bg**
- Binarization_Algo **binarization_algo**
- **bool enable_denoising**
- **bool enable_deskew**
- **bool enable_fg_extraction**
- **bool enable_subsample**
- **mln::trait::concrete**< I >::**ret fg**
- **mln::util::timer gt**
- **unsigned lambda**
- **image2d**< **bool** > **output**
- **double sauvola_k2**
- **double sauvola_k3**
- **double sauvola_k4**
- **unsigned sauvola_win**
- **mln::util::timer t**
- **bool verbose**

8.80.1 Detailed Description

template<typename I>struct scribo::toolchain::internal::text_in_doc_preprocess_functor< I >

Functor performing custom preprocessing algorithms on documents.

Whatever the options selected, this functor converts an input image into a graylevel image and try to binarize the latter.

Optional algorithms can be performed too:

- Subsampling (enable_subsample), disabled by default.
- Extract Background/Foreground (enable_fg_extraction), split background and foreground objects, disabled by default.
- Deskew (enable_deskew), disabled by default.
- Denoise (enable_denoise), enabled by default.

The binarization algorithm can be chosen through binarization_algo option. **It** can be one of the algorithms detailed in enum Binarization_Algo.

Few parameters can be set for algorithms:

- Sauvola Multi-scale: sauvola_win, the window size used in Sauvola based algorithms (default 101). sauvola_-K2, a user parameter for Sauvola's threshold formula at scale 2. (default 0.34). sauvola_K3, a user parameter for Sauvola's threshold formula at scale 3. (default 0.34). sauvola_K4, a user parameter for Sauvola's threshold formula at scale 4. (default 0.34).

- Extract background/foreground lambda, the maximum area of the possible foreground objects (default 0, auto).

Definition at line 115 of file text_in_doc_preprocess_functor.hh.

8.81 scribo::toolchain::internal::text_in_picture_functor< I > Struct Template Reference

```
#include <scribo/toolchain/internal/text_in_picture_functor.hh>
```

Public Types

- typedef **mln::trait::ch_value < I, V >::ret L**
- typedef **scribo::def::lbl_type V**

Public Member Functions

- virtual **int nsteps** () const
- virtual void **on_end** ()
- virtual void **on_new_progress_label** (const char ∗label)
- virtual void **on_progress** ()
- virtual void **on_start** ()
- component_set< L > **operator()** (const **Image**< I > &)

Public Attributes

- **float bbox_h_ratio**
- **float bbox_overlap**
- **bool enable_bg_removal**
- **bool enable_multi_scale_bin**
- **unsigned group_min_holes**
- object_groups< L > **groups**
- **mln::util::timer gt**
- **unsigned lambda**
- **unsigned max_dim_size**
- **unsigned mean_width**
- component_set< L > **output**
- **unsigned regroup_dmax**
- **unsigned sauvola_min_w**
- **unsigned sauvola_s**
- **unsigned small_groups**
- **mln::util::timer t**
- **bool verbose**

8.81.1 Detailed Description

template<typename I>struct scribo::toolchain::internal::text_in_picture_functor< I >

Localize text in a picture.

Definition at line 112 of file text_in_picture_functor.hh.

8.82 scribo::toolchain::internal::Toolchain_Functor Class Reference

```
#include <scribo/toolchain/internal/toolchain_functor.hh>
```

Public Member Functions

- virtual **int nsteps** () const =0
- virtual void **on_end** ()
- virtual void **on_new_progress_label** (const char ∗label)
- virtual void **on_progress** ()
- virtual void **on_start** ()

Public Attributes

- **bool verbose**

8.82.1 Detailed Description

Base class for toolchain functors.

Definition at line 47 of file toolchain_functor.hh.

8.83 scribo::util::integral_sub_sum_sum2_functor< I, S > Class Template Reference

Public Types

- typedef **mln::util::couple**< S, S > **result**

Public Member Functions

- **integral_sub_sum_sum2_functor** (const I &ima, **unsigned** scale)
- **integral_sub_sum_sum2_functor** (**unsigned** scale, const mln_box(I)&output_domain, **unsigned** border)
- void **begin_of_col** ()
- void **begin_of_first_row** ()
- void **begin_of_row** ()
- void **end_of_col** ()
- void **end_of_row** ()
- void **take** (const V &v)
- **result to_result** (const **result** &up_result) const
- **result to_result_first_row** () const

Public Attributes

- J **sub**

8.83.1 Detailed Description

template<typename I, typename S = typename mln::value::props< typename I ::value >::sum>class scribo::util::integral_sub_sum_sum2_functor< I, S >

Definition at line 48 of file integral_sub_sum_sum2_functor.hh.

8.84 scribo::util::integral_sum_sum2_functor< V, S > Class Template Reference

Public Types

- typedef **mln::util::couple**< S, S > **result**

Public Member Functions

- void **begin_of_col** ()
- void **begin_of_first_row** ()
- void **begin_of_row** ()
- void **end_of_col** ()
- void **end_of_row** ()
- void **take** (const V &v)
- **result to_result** (const **result** &up_result) const
- **result to_result_first_row** () const

8.84.1 Detailed Description

template< typename V, typename S = typename mln::value::props< V >::sum>class scribo::util::integral_sum_sum2_functor< V, S >

Definition at line 44 of file integral_sum_sum2_functor.hh.

8.85 scribo::util::integral_sum_sum2_global_min_functor< V, S > Class Template Reference

Public Types

- typedef **mln::util::couple**< S, S > **result**

Public Member Functions

- void **begin_of_col** ()
- void **begin_of_first_row** ()
- void **begin_of_row** ()
- void **end_of_col** ()
- void **end_of_row** ()
- const V & **global_min** () const
- void **take** (const V &v)
- **result to_result** (const **result** &up_result) const
- **result to_result_first_row** () const

8.85.1 Detailed Description

template< typename V, typename S = typename mln::value::props< V >::sum>class scribo::util::integral_sum_sum2_global_min_functor< V, S >

Definition at line 44 of file integral_sum_sum2_global_min_functor.hh.

8.86 stats< T > Class Template Reference

Public Member Functions

- **stats** (const **int** size)
- util::array< cluster_stats< **T** > > & **clusters** ()
- **T max** ()
- **T mean** ()
- **T median** ()
- **T min** ()
- **unsigned nelements** ()
- void **reset** ()
- **T standard_deviation** ()
- void **take** (const **T** &**value**)
- **T variance** ()

8.86.1 Detailed Description

template<typename T>class stats< T >

Definition at line 195 of file stats.hh.

Bibliography

[1] E. Badekas and N. Papamarkos. Automatic evaluation of document binarization results. pages 1005–1014, 2005.

[2] Boontee Kruatrachue, Narongchai Moongfangklang, and Kritawan Siriboon. Fast document segmentation using contour and XY-Cut technique. In *Proceedings of World Academy of Science, Engineering and Technology*, volume 5, pages 27–29, 2005.

[3] J.-L. Meunier. Optimized XY-Cut for determining a page reading order. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 347–351. IEEE, 2005.

[4] George Nagy, Sharad Seth, and Mahesh Viswanathan. A prototype document image analysis system for technical journals. *Computer*, 25(7):10–22, 1992.

[5] J. Sauvola and M. Pietikäinen. Adaptive document image binarization. *PATTERN RECOGNITION*, 33:225–236, 2000.

[6] Faisal Shafait, Daniel Keysers, and Thomas M. Breuel. Efficient implementation of local adaptive thresholding techniques using integral images. *Document Recognition and Retrieval XV*, Jan 2008.

Index