# Documentation: Segmentation and Graph Construction of HMRF

Martin Drauschke

martin.drauschke@uni-bonn.de, martin.drauschke@unibw.de

TR-IGG-P-2009-03

19th October 2009


Technical Report Nr. 3, 2009

Department of Photogrammetry
Institute of Geodesy and Geoinformation
University of Bonn

Available at
http://www.ipb.uni-bonn.de/martindrauschke/

# Documentation: Segmentation and Graph Construction of HMRF

Martin Drauschke

martin.drauschke@uni-bonn.de

**Abstract**

This is a technical report for presenting a documentation on the segmentation and the graph construction of a Hierarchical Markov random field (HMRF). The segmentation is based on multiscale analysis and watershed regions as presented in [Drauschke *et al.*, 2006]. The region's development is tracked over the scales, which defines a region hierarchy graph. This graph is used to improve the segmentation by reforming the regions geometrically more precisely. This work is taken from [Drauschke, 2009]. Furthermore, we determine a region adjacency graph from each image partition of all scales. The detected image regions, their adjacent regions and their hierarchical neighbors are saved into an xml-file for a convenient output.

## Contents

# 1 Introduction

This is the first version of the Segmentation and Graph Construction IPM for building a hierarchical Markov random field. Most functions of the IPM are taken from the Stable Regions IPM, which has been developed within the project *eTraining for Interpreting Images of Man-Made Scenes (eTRIMS)*. In the end of this project, we integrated the multiscale segmentation into the Markov random field framework, and so this IPM was needed.

The Segmentation and Graph Construction IPM analyzes an input image in its scale-space. It determines image regions using the watershed algorithm on the gradient image at several scales. Since the watershed algorithm returns an image partition where each pixel belongs to exactly one region, it is simple to determine the according region adjacency graph for each scale. The hierarchical Markov random field is determined by the region hierarchy graph, which is taken from the Stable Regions IPM. We summarize our segmentation approach and present our algorithms for constructing the graphs in section 2.

We save the segmented image regions and their graph structures into an xml file. The structure of this output is described in section 3. Furthermore, we include a rather technical section into this report, section 4, where we present the functionality of the IPM. The choice of system parameters is presented in sec. 5. Before concluding, we also show some small results and list all possible error statements that can be thrown by the IPM, cf. sec. 6 and sec. refsec:err.

# 2 Algorithms for Segmentation and Graph Construction

In this section, we document the theoretical considerations, which underlie the Segmentation and Graph Construction IPM. First, we summarize the multiscale segmentation using the watershed algorithm. Then we present the construction of the region hierarchy graph (RHG). It is used to define an irregular pyramid which leads to geometrically more precise regions. Conse-

quently the region hierarchy graph has to be updated again. Finally, we are able to determine the region adjacency graph (RAG).

## 2.1 Multiscale Segmentation

We analyze the image at different scales, therefore we smooth the image in the Gaussian scale-space by adequate kernels. So, we form a discrete scale-space, and in each of its layers, we use the gradient's magnitude as input for the watershed algorithm, cf. [Drauschke *et al.*, 2006]. The result of the watershed algorithm is a labeled image with regions, which were surrounded by edge-pixels which are characterized by the label 0. The image labeling is performed using a 4-neighborhood between region pixels. Afterwards, each edge-pixel is removed and put to the region with the smallest area around it. Fig. 1 shows a smoothed image and the original border pixels between the watershed regions are visualized in yellow.

For scene interpretation, we usually consider manually rectified images. The rectification is a transformation namely a homography. Thus, the new image contains image pixels which are outside the recorded scene, and so their image values are set to a certain gray value. When segmenting the smoothed images, we ignore these pixels and replace their region label by 0.

## 2.2 Region Hierarchy Graph

The region hierarchy graph (RHG) reflects the development of regions through the scale-space. Therefore, the graph contains directed edges in upwards direction. We define the hierarchy relation such that each region of a given scale is mapped to exactly one region of the next scale. Thus, the undirected RHG has the structure of a forest with trees having one node, their root, at the highest scale. The leaves of such a tree is either a region at the smallest scale or a region that is never a successor of a region of the previous scale. Fig. 2 shows a RHG with regions at three scales, the gray node represents a region, which has no predecessor.

We define the directed relations of the RHG using the maximum intersection. If a region at scale $s$ is given having $P$ pixels $x_1, \cdots, x_P$ then we construct a histogram over the labels of these pixels at the next scale, $s+1$. The region with the label that occurs most often is taken for being the successor of the previous region.

Figure 1: Smoothed image with result of watershed algorithm in yellow.

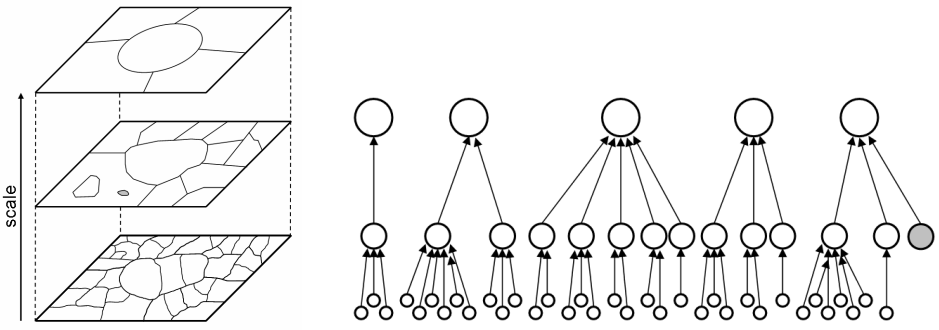


Figure 2: Regions in scale-space with three scales and according region hierarchy graph.

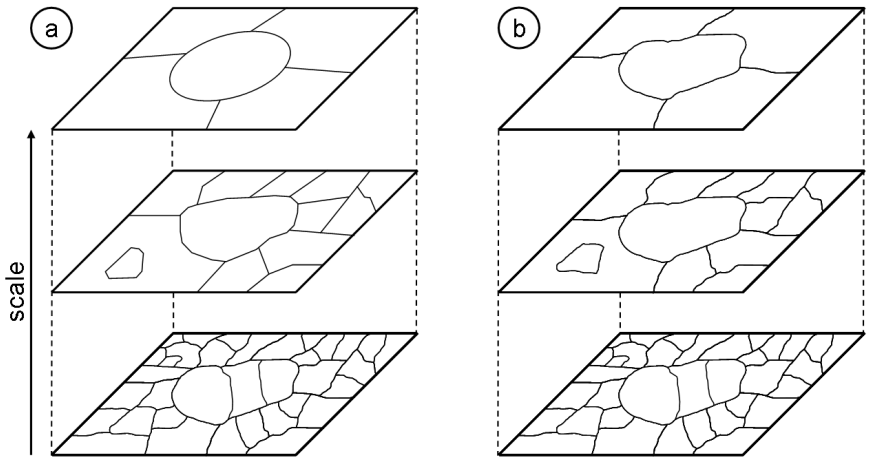


Figure 3: Left: watershed regions in scale-space. Right: regions from irregular pyramid.

## 2.3 Irregular Pyramid

In the next step, we combine the multi-scale segmentation and the region hierarchy graph to obtain geometrically more precise regions at the upper scales. There, the regions have round corners due to the smoothing of the image. Now, we replace a region in a higher scale by the union of all regions from the first scale, which develop towards that one particular region at the higher scale. This forms an irregular pyramid of regions. Fig. 3 shows the regions in scale-space (left) and the derived segmentations of the pyramid (right).

Our pyramid framework is based on the multi-scale segmentation approach by [Gauch, 1999]. There, the Gaussian scale-space is also used for deriving several image layers. The regions of the smallest scale are also extracted using the watershed algorithm with the gradient's magnitude as its input. Then, the region hierarchy is defined by observing the shifts of the seed point of each region. So, this approach is only applicable for watershed regions or other segmentation approaches which are based on region growing starting at seed points. So, our approach with comparing the pixel labels is more general.

This irregular pyramid framework leads to a loss of regions, and therefore to a change in topology. Since we construct each region of a higher scale only by regions from the first scale, region neighborhoods may change and all those regions disappear which have no predecessor region at the first scale. Fig. 4 shows two segmentations in comparison, the left side shows the original

Figure 4: Left: segmented watershed regions. Right: image regions from the irregular pyramid).

watershed segmentation, and the right side shows the geometrically more precise regions. Obviously, the boundaries of the more precise regions are not so generalized as in the original segmentation. The changes in topology are here clearly visible.

## 2.4 Region Adjacency Graph

Since the topology changed when improving the segmentation by applying an irregular pyramid, the construction of the region adjacency graph (RAG) at each scale is performed afterwards. At each scale, the image regions form a partition of the image, i. e. each pixel belongs one region. So, we are able to apply the algorithm for constructing a RAG as described in [Pan, 1994]. In this first version of the IPM, we determine the RAG at each scale independently, but the RAGs of higher scales could also get constructed by updating the RAG of the first scale using the RHG.

## 2.5 Hierarchical Markov Random Field

Finally, we combine the segmented regions, the RAGs and the RHG to one big network, a hierarchical Markov random field. Therefore, we rename the regions such that each label does not occur at every scale anymore, but exist only once. Fig. 5 shows the pyramid segmentation at three scales and the according RHG. When combining these regions with the according RAGs and RHG, we obtain a graphical model which is shown in fig. 6. This graphical model is describable an hierarchical Markov random field, when interpreting all edges in the graph as undirected.

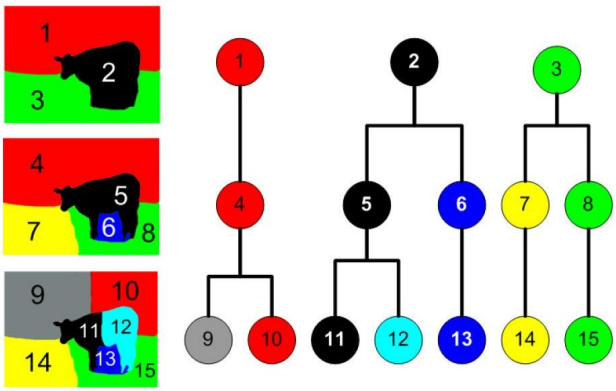


Figure 5: Image regions at three scales and according region hierarchy graph. Graphic by Michael Ying Yang.

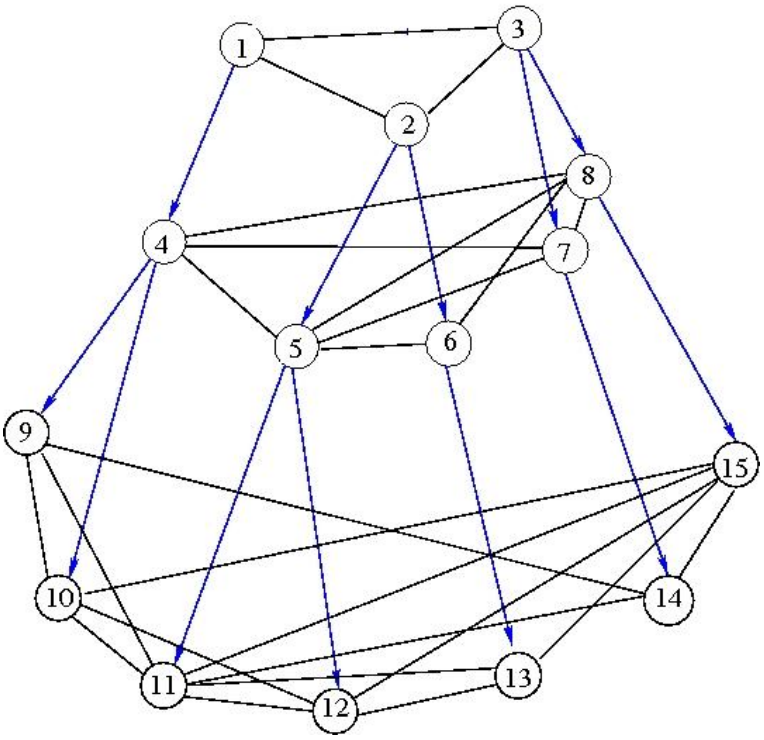


Figure 6: Graphical Model for combining regions with RAGS and RHG. Graphic by Michael Ying Yang.

# 3 In- and output of the IPM

In this section, we document the ingoing and outgoing data of the Segmentation and Graph Construction IPM.

## 3.1 Input

Of course, the IPM is able to segment any given color-image and to construct RAGs and a RHG from these image regions. But the IPM is originally designed for integrating into a scene interpretation system for man-made scenes namely buildings. Therefore, the IPM is also able to work on rectified images, if they have been rectified using the annotation tool, cf. [Korč & Schneider, 2007]. Then, the transformation is saved according to the specification of the annotation tool, and our IPM is able to reconstruct the frame of the original image.

## 3.2 Output

Here, we specify the output of the Segmentation and Graph Construction IPM: the xml file. The top tag's name is *annotation* with one child tag *image*. At that level, two tags are defined: *filename* for storing the filename of the processed image and *entity*, which is a list of all detected image regions (entities). One such entity consists of the following items:

- *type* for saving the type of a region. Here, we only have one type: `ws_region`.
- *identity* contains the number of the region after relabeling all regions to obtain unique region labels over all scales. Thus, we use this number as identifier of the region. Furthermore, this number is identical to the index in the list of entities.
- *scale* stores the scale of the region, i. e. the layer of the scale-space, where the region was detected, respectively the level of the irregular pyramid, the region belongs to.
- *label* stores the original label of the region that was given to it in the pyramid.
- *neighbors* contains an array with *identity*-numbers of all adjacent regions (taken from the specified RAG). If a region has no neighbors, this item still exists, but the array is empty.

- *tree* inludes two further items: *children* and *parent*. In *children* we store the list of child regions from the RHG analogously to the neighbors. Since the RHG is a forest with tree-structured components, a region typically has exactly one parent region. Its *identity*-number is stored there. Regions at the highest scale and regions at the lowest scale do not have a parent or child regions, respectively. Then the tags contain an empty array.

- *polygon* saves the boundary of the region. Therefore, each corner point consisting of the elements $x$ and $y$ of the boundary is saved in the list tag *pt*.

- *boundingbox* saves the four points of the bounding box analogously to the list in *polygon*.

# 4 Functionality of Interface Functions

We distinguish between the interface functions of the Segmentation and Graph Construction IPM and the private functions. The private functions are stored in the directory `src` should not be called individually. Thus, we do not list their functionality in this report. A short description of these routines and the explanation of the input and output variables are listed in the beginning of code of each method.

The interface functions are prepared for usage by e. g. a scene interpretation system. These functions set up the environment of the IPM and there, we call the private functions. All interface functions are stored in the main directory of the IPM, which should be used as *current directory* when using matlab.

- `init`. This functions creates directories for saving temporary data, adds a path to the directory with the private functions to the Matlab search path, and loads a configuration file. The input of this function is the name of the configuration file and, voluntary, the file extension. The accepted file formats for a configuration file are `mat` and `xml`. In the directory `test`, there is a method `create_option_file` for making such a configuration file. The routine returns a structure where the system parameters are stored in.

- `set_options`. This functions resets the system configuration by reading a new configuration file and returning an updated structure with the system parameters. Thus, input and output are identical to the function `init`.

- `set_optionvalue` and `get_optionvalue`. Both functions are implemented for a controlled reading from and writing into the structure with the system parameters, respectively. Thus, such a structure is an input of both functions and the name of the selected parameter. The writing method additionally needs the new value as input and has no further output, and the reading method returns that parameter's value. If the parameter's name is not one of the list (cf. next subsection), the functions stops the process and returns an error message.

- `derive_further_parameters`. Most of the system's parameters are set from the configuration file, but some parameters depend on the value of others. The validation of the read parameters and the derivation of further dependent parameters in done in this routine, e. g. the base $\sigma$ for the smoothing of the images depends on the number of layers and the range of scale which can be set by the user. Thus, we determine its value directly before starting the detection of the stable regions. There two other input variables (both boolean), where the user may direct, if the system shall write images into temporary directories for subtask inspection and if the output stream shall be verbose.

- `find_all_stable_regions`. Here, the detection of the stable region takes place. This routine is made as an all-ine-one-function, where no interaction can be done. But the routine also returns important information about the scale-space structure that has been saved, and which can be used for further activities.

- `cleanup`. If this function is called, all temporary directories and files are removed again.

# 5 System Parameters

The Segmentation and Graph Construction IPM is based on the Stable Regions IPM of the eTRIMS project. We did not adjust the list of parameters, so the configuration file containing the system parameters includes values of no interest. We list and comment those parameters, an example is given in table 1.

- `avoid_oversegmentation`. This parameter is used in segmentation process. It describes a factor which will be multiplied to the median of the average gradient. Then, all gradients below this product are set to 0. For additional information, cf. [Brügelmann & Förstner, 1992, Drauschke *et al.*, 2006].

- `maximum_scale`. This parameter is used when constructing the scale-space. In Gaussian scale-space, the different levels have the following meaning: 0 means the layer with the original image, 1 is the pyramid's level where original image is set on half size (or smoothed by Gaussian kernel with sigma=1), 2,4,8 are next pyramid's levels. Further levels do not seem to be appropriate, because the smoothing of the image with a Gaussian kernel will cause to many deformations of the regions.

- `nb_big_scales`. This parameter is also used when constructing the scale space. Big scales are all scales of pyramid's levels ¿= 1. At the Stable Regions IPM we preferred to work on 10 scales from one pyramid's level to the next one. Then, we need 31 scale for modeling the scale-space starting at scale 1 and ending at scale 8. At the Segmentation and Graph Construction IPM we are only interested in the five major scale corresponding to sigma = 1,2,4,8 and 16.

- `nb_small_scales`. This parameter is also used when constructing the scale-space. Small scales are all scales below 1. We have set this parameter to 0, because we already very small regions at the scale with sigma = 1.

- `start_tree_layer`. The layers in the scale space have an index, 1 is the lowest scale, the original image, then the small scales come, finally the big ones. This parameter is used when building the region's hierarchy. Its value refers to the index of a scale where we start building the hierarchy and later, we begin there to derive the tree structure.

- `end_tree_layer`. The layers in the scale space have an index, 1 is the lowest scale, the original image, then the small scales come, finally the big ones. This parameter is used when building the region's hierarchy. Its value refers to the index of a scale where we stop building the hierarchy and later, we stop there to derive the tree structure. The value 0 for this parameter means that the work shall be done until the last possible layer has been reached.

- `minimum_size_of_a_region`. This parameter is not used anymore.

- `stability_threshold`. This parameter is not used anymore.

- `stability_range`. This parameter is not used anymore.

- `pruning_size`. This parameter is not used, yet. Originally, small regions are suppressed to avoid feature extraction for thousands of very small regions, but it is not applied in this version of the IPM.

- `pruning_overlap`. This parameter is not used, yet. Originally, small regions are suppressed to avoid feature extraction for thousands of very small regions, but it is not applied in this version of the IPM.

Table 1: List of System Parameters as set in a Configuration File.

```
  avoid_oversegmentation: 1
           maximum_scale: 16
           nb_big_scales: 5
         nb_small_scales: 0
minimum_size_of_a_region: 0
        start_tree_layer: 1
          end_tree_layer: 0
     stability_threshold: 0.7000
         stability_range: 10
            pruning_size: 30
         pruning_overlap: 0.7
```

Other system parameters depend directly on parameters from the configuration file. Furthermore, the user may select two options for the output: the systems behavior concerning the saving of temporary images and the printing into the command line is managed by the parameters `save_images` and `verbose`. The two new derived system parameters are:

- `sigma0`. This parameter is needed to determine the correct smoothing parameter. The scale space layers are logarithmically ordered, so this value is used as a base for the determination of the smoothing values. For additional information, cf. [Drauschke *et al.*, 2006].

- `nb_all_scales`. The scale space shall consist of the original image and all available small and bigger scales.

The following table shows the additional parameters of the options structure.

# 6 System Properties and Complexity

Version 1.0 of the Segmentation and Graph Construction IPM has been tested using Matlab 7.9.0 (R2009b). The Image Processing Toolbox must be available when running the IPM since we use various implemented functions of

Table 2: List of derived System Parameters.

```
save_images: 1
    verbose: 0
     sigma0: 2
nb_all_scales: 5
```

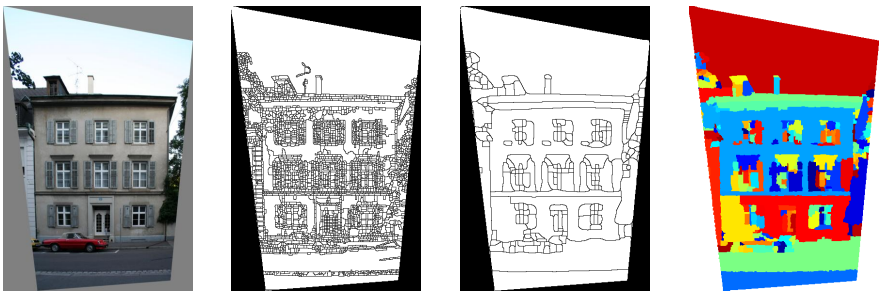

Figure 7: Image `basel_000003` with extracted watershed regions at scales 1 and 3 and the merged regions of scale 3 (f. l. t. r.).

this toolbox. So far, we only have performed the IPM on Windows, but we don't see any problems for using Linux as operating system.

We have tested the IPM intensively, our last tests were based on 194 images from Basel (Switzerland) with 53 images, Hamburg (Germany) with 85 images, Heidelberg (Germany) with 26 images and the other 29 images from the German cities Berlin, Bonn, Karlsruhe and Munich and 1 image from Great Britain. We document the complexity of our IPM on 7 selected images. Therefore, we document the number of extracted regions in 5 scales and the CPU-time of the complete calculations (beginning when reading the image and ending after saving the xml file). All 7 images are rectified and have an approximate size of $600 \times 400$ pixels, if its format is portrait or vice versa if its format is landscape. We present the images and some segmentation results in the figs. 7-13. The number of extracted regions and the needed cpu time is shown in tab. 3.

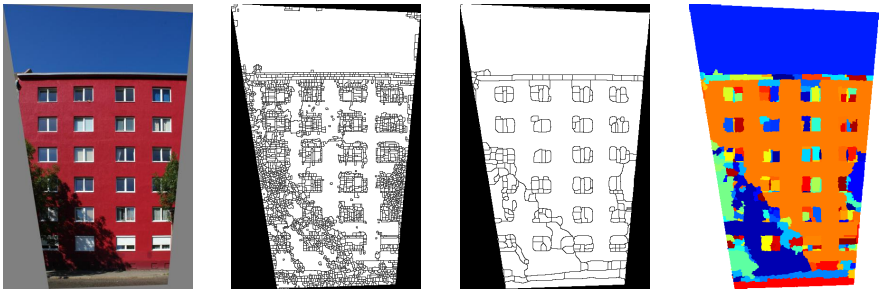

Figure 8: Image `berlin_000003` with extracted watershed regions at scales 1 and 3 and the merged regions of scale 3 (f. l. t. r.).

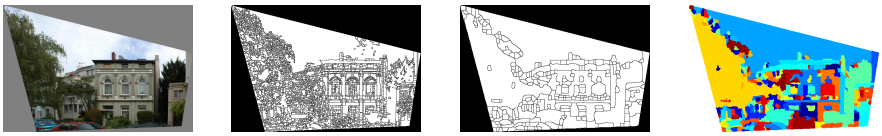

Figure 9: Image `bonn_000027` with extracted watershed regions at scales 1 and 3 and the merged regions of scale 3 (f. l. t. r.).

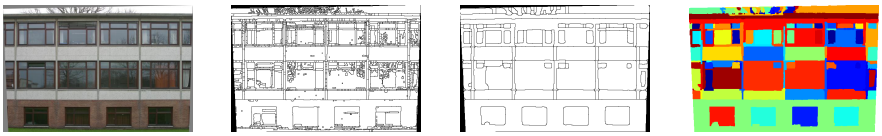

Figure 10: Image `hamburg_000006` with extracted watershed regions at scales 1 and 3 and the merged regions of scale 3 (f. l. t. r.).

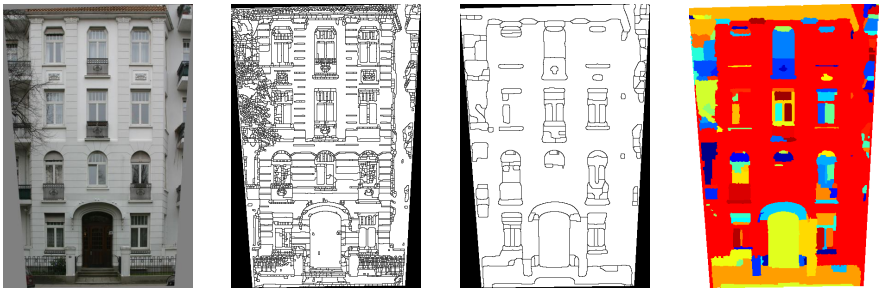

Figure 11: Image `hamburg_000041` with extracted watershed regions at scales 1 and 3 and the merged regions of scale 3 (f. l. t. r.).

Figure 12: Image `heidelberg_000001` with extracted watershed regions at scales 1 and 3 and the merged regions of scale 3 (f. l. t. r.).

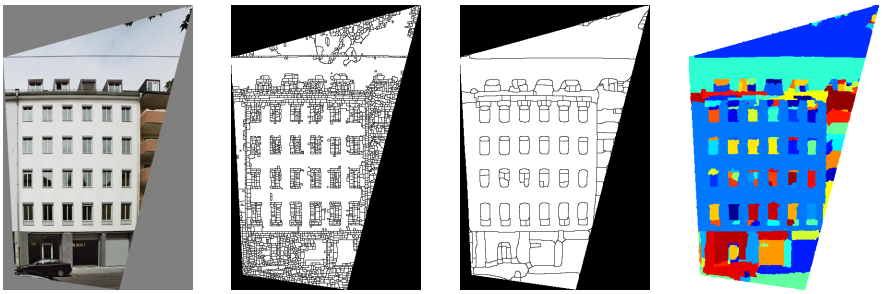

Figure 13: Image `munich_000005` with extracted watershed regions at scales 1 and 3 and the merged regions of scale 3 (f. l. t. r.).

Table 3: Complexity of the IPM.

| Image and scale | regs 1 | regs 2 | regs 3 | regs 4 | regs 5 | $\sum$ | CPU-time in [sec.] |
|---|---|---|---|---|---|---|---|
| basel_000003 | 3448 | 1053 | 208 | 26 | 7 | 4742 | 88.1406 |
| berlin_000003 | 3074 | 948 | 206 | 37 | 17 | 4282 | 82.5625 |
| bonn_000027 | 4457 | 1165 | 230 | 22 | 4 | 5878 | 111.1406 |
| hamburg_000006 | 1919 | 471 | 115 | 19 | 6 | 2530 | 60.8444 |
| hamburg_000041 | 3402 | 784 | 134 | 36 | 8 | 4364 | 91.4531 |
| heidelberg_000001 | 3084 | 975 | 143 | 39 | 4 | 4245 | 82.0000 |
| munich_000005 | 2942 | 878 | 157 | 53 | 9 | 4039 | 75.5938 |

# 7 Error Statements

The Segmentation and Graph Construction IPM returns the following error statements, if the IPM is not used properly or other problems occur. Table 4 lists all error statements, sorted by their number, and also contains the throwing function and the statement itself.

Table 4: Error statements of Segmentation and Graph Construction IPM.

| Nr. | Throwing Functions | Statement |
|---|---|---|
| 1 | stab_init, stab_set_options | No file name for configuration file. |
| 2 | stab_init, stab_set_options | Configuration file must have valid file extension (mat or xml). |
| 3 | stab_init, stab_set_options | Configuration file not found. |
| 4 | stab_init, stab_set_options | Wrong file extension for configuration file. |
| 5 | stab_set_optionvalue, stab_get_optionvalue, stab_derive_further_parameters | Three / Two parameters are needed for setting / deriving new option value. |
| 6 | stab_set_optionvalue, stab_get_optionvalue | Wrong parameters name - could not set / get option value. |
| 7 | stab_get_frame | Could not find file with annotations. |
| 8 | stab_get_frame | Problem with reading annotation file and input image. |
| 9 | disperse_filename | Problem with filename of input image. |
| 10 | stab_derive_further_parameters, detect_regions_for_tree, merge_regions, calculate_tree_of_regions, calculate_RAGs_of_regions, construct_xml_structure | Option structure is not valid. |
| 11 | detect_regions_for_tree | Problem with reading the input image. |

Table 4: Error statements of Segmentation and Graph Construction IPM.

| Nr. | Throwing Functions | Statement |
|---|---|---|
| 12 | detect_regions_for_tree | Invalid input image must have 3 channels (RGB). |
| 13 | get_scale_space_layer, detect_regions_for_tree, merge_regions | Problem with saving temporary data. |
| 14 | calculate_tree_of_regions, merge_regions, calculate_RAGs_of_regions, construct_xml_structure | Problem with loading temporary data. |
| 15 | construct_partition | Label image contains only one label: 0. |
| 16 | construct_partition | Error when removing the zeros. |
| 17 | calculate_tree_of_regions | Wrong parameter configuration. |
| 18 | douglas_peucker_algorithm | Error when executing peckerpan_c-routine in libary. |
| 19 | write_xml_structure | Problem with saving regions into xml file. |

# 8 Further Work

In the next steps we include the feature extraction and target validation to the IPM. This shall be used for classifying each region.

# References

[Brügelmann & Förstner, 1992] Brügelmann, R., & Förstner, W. 1992. Noise Estimation for Color Edge Extraction. *Pages 90–107 of: Robust Computer Vision*. Wichmann, Karlsruhe.

[Drauschke, 2009] Drauschke, M. 2009. An Irregular Pyramid for Multi-Scale Analysis of Objects and their Parts. *Pages 293–303 of: GbR'09*. LNCS 5534.

[Drauschke *et al.*, 2006] Drauschke, M., Schuster, H.-F., & Förstner, W. 2006. Detectability of Buildings in Aerial Images over Scale Space. *Pages 7–12 of: PCV'06*. IAPRS 36 (3).

[Gauch, 1999] Gauch, J. M. 1999. Image Segmentation and Analysis via Multiscale Gradient Watershed Hierarchies. *Image Processing*, **8**(1), 69–79.

[Korč & Schneider, 2007] Korč, F., & Schneider, D. 2007 (June). *Annotation Tool*. Tech. rept. TR-IGG-P-2007-01. IGG University of Bonn.

[Pan, 1994] Pan, H.-P. 1994. Two-level Global Optimization for Image Segmentation. *P& RS*, **49**(2), 21–32.