Description of Stable Regions IPM

Martin Drauschke

martin.drauschke@uni-bonn.de

TR-IGG-P-2008-03

18th March 2008

Technical Report Nr. 3, 2008

Department of Photogrammetry
Institute of Geodesy and Geoinformation
University of Bonn

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

Desription of Stable Regions IPM

Martin Drauschke

martin.drauschke@uni-bonn.de

Abstract

The Stable Regions Image Processing Module is a low-level region detector. It delivers image parts of interest without any further interpretation. These image parts are all regions of an image which do not change much over a certain range in scale space of the image. The output of this IPM is a list of polygons of any shape and their rectangular bounding boxes, which both are saved into an xml-file.

Contents

1 Introduction

The Stable Regions IPM is written by Martin Drauschke from Department of Photogrammetry, Institute of Geodesy and Geoinformation, University Bonn. It is based on first experiments on image regions in scale space, cf. [Drauschke *et al.*, 2006]. The work for the first version of the implementation was done within the project "Ontological Scales for automated detection, efficient processing and fast visualization of landscape models" which is founded by the German Research Council. Due to the complexity of the calculations, some programme routines had to optimized and re-structured. The work for the second version was done within the project "e-Training for Interpreting Images of Man-Made Scenes" which is founded by the European Union.

As a low-level detector, it can be integrated into the framework of SCENIC. Therefore, we implemented routines for setting up the IPM and for choosing the systems parameters. The output of the IPM is a list of image regions which are represented by their boundary and described by polygons. These regions may have any shape, especially they may have holes. For easier integration into SCENIC we also return the rectangular bounding box of each polygon. So far, we only detect stable regions. There is no semantic interpretation of these regions yet. Thus, the user should take the output of the IPM as a list of "interesting image parts" only.

This report contains four chapters. In chapter 2, we describe mathematically the detection procedure of the stable regions. The chapter 3 is more technically, there we present the functionality of the IPM and introduce all implemented functions. Finally, in chapter 4 we document the work by presenting some examples.

2 Detection of Stable Regions

This section is taken from the project review "D2.1: Compositional hierarchies in Bayesian Networks", submitted on May 21st 2007 to Commission Services of the eTRIMS project.

2.1 Segmentation and Scale-Space Hierarchy.

Scale space analysis has been intensively studied, cf. e. g. [Alvarez *et al.*, 1993, Lindeberg, 1994, Florack & Kuijper, 2000, Kuiper *et al.*, 2003]. We build the Gaussian scale-space of the original image $\mathbf{I}(x, y)$ with $N = 41$ logarithmically ordered scales $\sigma_i = 2^i$ and $i \in [-0.5, 3]$. Thus, we obtain 10 small scales below 1, and the other scales between 1 and 8 in steps of a tenth octave. The gradients are used as an input for a watershed algorithm that returns a partitioning with regions $R^{\nu,\sigma}, \nu = 1, ...$ for each layer of the image's scale-space with scale σ, cf. [Drauschke *et al.*, 2006].

The adjacencies of regions within one scale-space layer are already specified by the partitioning. Additionally, we need a well defined neighbourhood for regions of different scale-space layers. We define the across-scale-neighbourhood by a region mapping. Let σ_i and σ_{i+1} two adjacent scale-space layers, then region $R^{\nu_{j_1},\sigma_i}$ is adjacent to a region $R^{\nu_{j_2},\sigma_{i+1}}$, if there is no other region in layer σ_{i+1} that shares more pixel with $R^{\nu_{j_1},\sigma_i}$ than $R^{\nu_{j_2},\sigma_{i+1}}$ does. More precisely,

$$R^{\nu_{j_1},\sigma_i} \mapsto R^{\nu_{j_2},\sigma_{i+1}} \Leftrightarrow R^{\nu_{j_1},\sigma_i} \cap R^{\nu_{j_2},\sigma_{i+1}} > R^{\nu_{j_1},\sigma_i} \cap R^{\nu_k,\sigma_{i+1}} \forall k \neq j_2. \quad (1)$$

Note that this asymmetric relation between two regions of different scale-space layers does not depend on a threshold. However it not transitive, thus we are not allowed to skip one scale layer, thus need to sample the scales densely enough in contrast to our previous approach in [Drauschke *et al.*, 2006].

We therefore obtain an image intrinsic graph structure which is a forest of trees and reflects the partonomy. If we consider region adjacencies defined as above, but in both directions (scale up- and downwards) then we may also handle scale-space events, where regions split. However, regions may be split when increasing scale, which however does not occur frequently and can be neglected, i e. resolved with some heuristics in a first approximation. Due to the huge number of very small regions, especially in the lower scales, we build the tree of regions only over regions with a minimum size, see Fig. 1. In our experiments we chose 30 pixels for the minimum size of a region.

2.2 Stability of Regions.

We are of course interested in scale-space events *annihilation* and *merging*. However, it appears decisive whether a region only changes a little bit over a wide range in scale-space.

In our first approach, cf. [**?**], we chose the area (size) of a region as criterion for measuring a region's stability. The area is a good measure since it changes dramatically, if e. g. two regions melt together, and it is nearly constant, if a region is similarly segmented over several adjacent scale-space layers. As this approach, in its first realization was a manual one, we replaced it by an automated process. While tracing the regions through the scale-space layers, we focused on those regions which do not vary too much over a certain range of scales. In fact, we looked for approximate prisms or cylinders in the scale-space image which are built by regions from adjacent layers. Thus, we observed the region R^{ν_i,σ_i} and its mappings into adjacent scales R^{ν_s,σ_s} with $s \in [\sigma_0, \sigma_1]$ and $\sigma_0 \leq \sigma_i \leq \sigma_1$. Finally, we term all regions R^{ν_i,σ_i} stable, if it obeys the condition

$$\frac{\bigcap_{s=\sigma_0}^{\sigma_2} R^{\nu_s,s}}{\bigcup_{s=\sigma_0}^{\sigma_2} R^{\nu_s,s}} > t \quad (2)$$

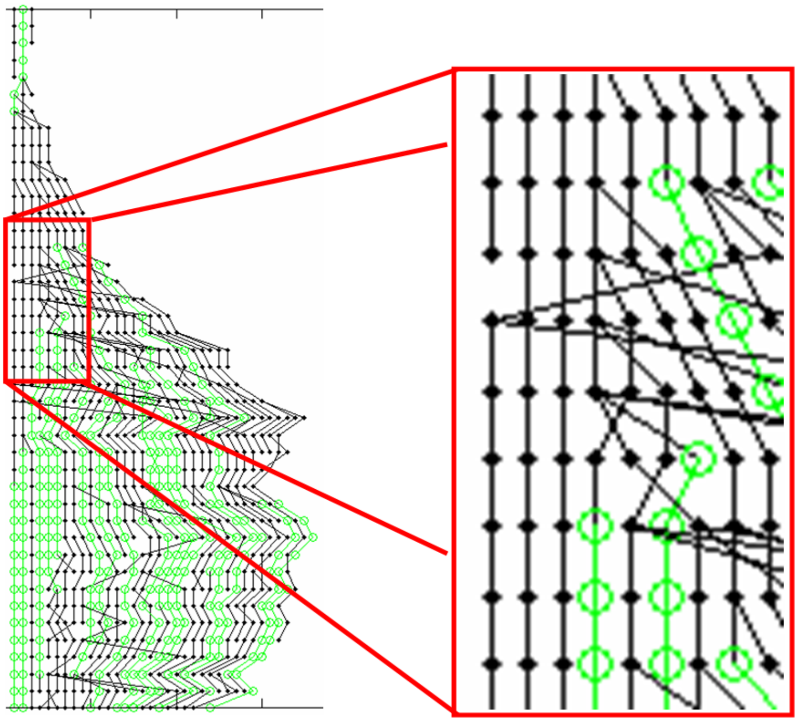

Figure 1: Example graph for regions neighbourhoods across scale-space layers. Nodes of the same height are regions in one layer. Edges between regions of adjacent layers mark the mappings between those regions. Green nodes symbolize stable regions, cf. equ. 2.

where the threshold t can be used to weaken or strengthen the measurement of stability. We used $t = 0.75$ for the determinations that we present in fig. 2 and 3.

As important regions may not be stable over a large scale range, larger than t, we discarded this early decision. Therefore, we do not preselect stable regions, but treat the region's behavior in scale space as the region's feature.

More precisely, we use the stability measure ς_s for a region R^{ν_s,σ_s} to the adjacent region in the next scale-space level $s+1$ by

$$\varsigma_s = \frac{R^{\nu_s,\sigma_s} \cap R^{\nu_{s+1},\sigma_{s+1}}}{R^{\nu_s,\sigma_s} \cup R^{\nu_{s+1},\sigma_{s+1}}} \tag{3}$$

and the stability measure $\overline{\varsigma}$ of scale range with d scale-space levels by

$$\overline{\varsigma_s} = \max_{i=0..d} \left\{ \min_{j=s-d+i..s+i} \varsigma_j \right\}. \tag{4}$$

The regions where $\overline{\varsigma}$ is over a threshold, e. g. $t = 0.75$, are exactly the stable

Figure 2: Aerial image and its segmentations at scale-space layers with $\sigma = 1$, $\sigma = 2$ and $\sigma = 4$ (row-wise f.l.t.r.) where we highlighted the stable regions.

region of the previous approach. However, we now have as a feature of a region the scale range within that region exists.

3 Functionality of Stable Regions IPM

3.1 Image Description

The Stable Regions IPM may work on all image formats that can be read by the Matlab Image Processing Toolbox (`imread`). Its supported file types are jpeg, tiff, gif, bmp, png, hdf, pcx, xwd, ico, cur, ras, pgm and ppm. Each input image must have 3 colour channels (RGB), otherwise the IPM will stop and return an error message. So far, we used images in jpg and tiff format in our tests. We have not tested the IPM on other material than 8-bit coded images. There is no further request on the input image, it may show any scene (objects may be rectified as well as perspectively deformed).

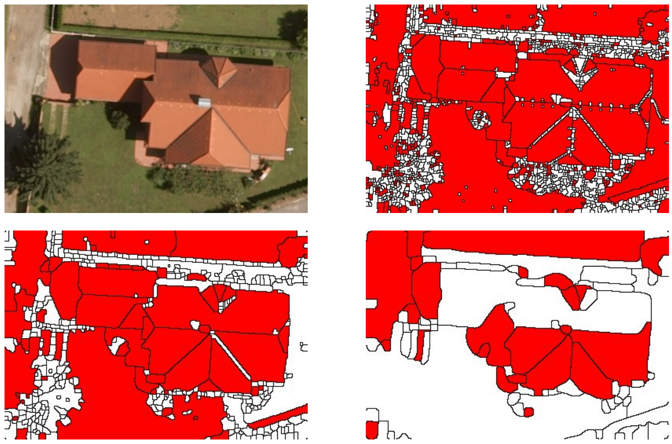

Figure 3: Aerial image and its segmentations at scale-space layers with $\sigma = 1$, $\sigma = 2$ and $\sigma = 4$ (row-wise f.l.t.r.) where we highlighted the stable regions.

3.2 Principle of the Stable Regions Detection

The Stable Regions IPM recognizes image regions which stay stable for at least a predefined range d in scale space. Therefore, we construct the scale space function of an image f, using a Gaussian kernel for the smoothing operation:

$$f(x, y, \sigma) = f(x, y) * G_\sigma(x, y).$$

There is no resizing of the image done, thus we analyze an image cuboid, not an image pyramid. The scale space is implemented discretely with l layers, and we call a family of regions

$$S = \{s_1, s_2, s_3, \ldots, s_n\} \quad , d \leq n \leq l$$

stable, if for all regions s_i, $i = 1..n-1$:

1. If s_i is a region of layer i, then is s_{i+1} a region of layer $i + 1$.
2. The regions do not change much from one layer to the next one:

$$s_i \approx s_{i+1}$$

.

Many stable regions can be found, where image parts have a clearly visible border. Man-made objects often have these borders, additionally shadow edges also often cause stable regions along them. Usually, single vegetation objects only lead to stable regions in upper scales, but lawn might also lead to stable regions at lower scales. Due to the image smoothing with a Gaussian kernel, the regions get rounder shape when the scale increases. These regions often do not represent a single object anymore. Therefore, the maximum scale should not set too high.

3.3 Functionality of Interface Functions

We distinhguish between the interface functions of the Stable Regions 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 each method.

The interface functions are prepared for usage by SCENIC, these functions set up the environment of the Stable Regions IPM and there, we call the private functions. All interface functions are stored in the main directory of the Stable Regions 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 contolled 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 σ 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.

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

- `show_stable_regions`. The purpose of this function is the visualization of the detection. So far, we only detect stable regions without having any interpretation done. So we draw the boundary of all regions into the image. Therefore, we need three input strings, the filename of the image, the filename of xml-file, where the stable regions have been saved, and the filename of the output image.

- `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. The process five major private functions which are described in section 3.5.

- `find_stable_regions_in_selected_area`. It could be reasonable, to search for stable regions in certain image parts. Then, not the whole tree of regions has to be determined, but only a subtree of it. Thus, all regions which do not intersect a rectangular image part are deleted before constructing the regions hierarchy. This selection process is realized in a private routine, the other difference to the previous function is that the construction of the scale-space structure can be skipped, if it already has been done.

3.4 System Parameters

The detection of stable regions depends on various parameters. Most of these parameters are set in a configuration file. They are listed below, 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 additonal information, cf. [Brügelmann & Förstner, 1992, Drauschke *et al.*, 2006].

- `maximum_scale`. This parameter is used when constructing the scale space. In 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 pyrmaid'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. We prefer the work on 10 scales from one pyramid's level to the next one. Then, we need 31 scale for modelling the scale space starting at scale 1 and ending at scale 8.

- `nb_small_scales`. This parameter is also used when constructing the scale space. Small scales are all scales below 1. We work with 10 scales what is like observing the pyramid's levels between 0.5 and 1. Further small scales are not recommendable.

- `minimum_size_of_a_region`. This parameter is used when analyzing the scale space: The region's hierarchy is only performed on regions with at least this size (number of pixels). Thus, this criterion is kind of a preselection of regions.

- `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.

- `stability_threshold`. This parameter is used when analyzing the regions in scale space. There, we focus on those regions which do not change a lot over a certain scale space range. This parameter describes the maximum allowed difference between two adjacent regions (in scale space), 0.7 says 70% of the regions area (list of pixels) may not change.

- `stability_range`. This parameter is used when analyzing the regions in scale space, we focus on those regions which do not change a lot over a certain scale space range. This parameter describes the minimum number of scale space layers where regions may not change much.

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

```
   avoid_oversegmentation: 1
            maximum_scale: 8
             nb_big_scales: 31
          nb_small_scales: 10
minimum_size_of_a_region: 30
         start_tree_layer: 1
           end_tree_layer: 0
      stability_threshold: 0.7000
          stability_range: 10
```

Other system parameters depend directly on parameters from the configuration file. Furthermore, the user may select two options for the output: the systems behaviour 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 additonal 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.

Table 2: List of derived System Parameters.

```
  save_images: 1
      verbose: 0
       sigma0: 1.1
nb_all_scales: 42
```

3.5 System Properties and Complexity

The Stable Regions IPM is implemented using Matlab 7.0.0.19920 (R14). The Image Processing Toolbox 4.2 has been used and must also be available, when running the IPM. We have tried the IPM only on Windows, so far, but we don't see any problems for using Linux as operating system (if Matlab is installed there, too).

We have tested the IPM intensively, we also recorded the computational CPU-time for inspecting the following 18 images:

- *cups.jpg* It is a very small image (134 x 71), where the calculations are manageable in about 15 seconds.
- *Graz16.jpg* and *Graz22.jpg* Both images are extracts of aerial images, but their size (each is 1409 x 1500) is still too big for fast testing. The region detection for each scale needs about half an hour, the calculations on the hierarchy and stability measures needs a couple of days! Thus, we used 5 and 9 smaller extracts of these aerial images, respectively.
- *London01.jpg*, *London02.jpg* and *London03.jpg* They are taken from the Imperial-group, all three images have an similar size of about 500 x 350 pixels.

The following table shows the computational costs of the five major routines that have to be called by each interface operator that wants to detect the stable regions:

- m_1 **detect_regions_for_tree.** Here, we construct the scale space of the input image, determine the image partition using the watershed transform, and finally, we select the appropriate regions for the region hierarchy graph without restricting the calculations on a pre-selected image part.

- m_2 **calculate_tree_of_regions.** In this method, we determine the hierarchy graph of all detected regions, and we obtain the local stability measure.

- m_3 **determine_range_stability.** In this routine, we derive a range stability measure for each region using the tree structure and the local stability measures.

- m_4 **reduce_tree_of_regions.** Here, we select the stable regions and appoint the reference regions of each stable regions family and the merging regions in the region hierarchy graph.

- m_5 **write_stable_regions.** This function is used for saving the stable regions (vector-representation and bounding box) into an xml-file.

Table 3: CPU-time needed for each major function.

Image	m_1 [sec]	m_2 [sec]	m_3 [sec]	m_4 [sec]	m_5 [sec]
cups	5.0	3.8	3.0	1.4	1.0
Graz16_1	196.0	179.2	1 216.8 ≈ 20 [min]	58.3	15.7
Graz16_2	402.3	321.0	3 265.1 ≈ 54 [min]	114.5	24.9
Graz16_3	713.0	525.2	6 996.0 ≈ 2 [h]	156.1	27.1
Graz16_4	118.4	87.9	174.8	22.6	8.9
Graz16_5	101.6	101.5	241.6	25.8	10.8
Graz22_1	75.1	79.0	75.4	12.5	10.4
Graz22_2	105.4	86.6	105.8	18.4	9.3
Graz22_3	84.5	72.3	83.6	11.3	9.1
Graz22_4	296.3	233.2	1 602.5 ≈ 27 [min]	74.1	18.3
Graz22_5	65.8	50.9	54.0	9.2	5.1
Graz22_6	233.7	224.9	1 511.0 ≈ 25 [min]	80.5	20.1
Graz22_7	138.7	111.8	373.5	28.0	9.8
Graz22_8	75.6	73.1	177.7	21.4	10.6
Graz22_9	164.3	141.7	672.4	42.0	9.6
London01	160.0	184.7	829.7	58.3	20.6

Table 3: CPU-time needed for each major function.

Image	m_1 [sec]	m_2 [sec]	m_3 [sec]	m_4 [sec]	m_5 [sec]
London02	174.1	204.5	1 792.3 $\approx$ 30 [min]	97.1	13.0
London03	209.2	257.9	1 752.7 $\approx$ 29 [min]	69.4	12.5
sum of above	3 319.0 $\approx$ 55 [min]	2 939.2 $\approx$ 49 [min]	20 928.0 $\approx$ 5.8 [h]	900.9 $\approx$ 15 [min]	236.9 $\approx$ 4 [min]
Graz16	29 250 $\approx$ 8 [h]	26 440 $\approx$ 7.5 [h]	96 270 $\approx$ 27 [h]	46 400 $\approx$ 13 [h]	690 11.5 [min]
Graz22	31 790 $\approx$ 9 [h]	26 180 $\approx$ 7 [h]	133 900 $\approx$ 37 [h]	23 430 $\approx$ 6.5 [h]	680 $\approx$ 11 [min]

3.6 Error Statements

The Stable Regions 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 Stable Regions 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.

Table 4: Error statements of Stable Regions IPM.

Nr.	Throwing Functions	Statement
6	stab_set_optionvalue, stab_get_optionvalue	Wrong parameters name - could not set / get option value.
7	stab_derive_further_parameters, detect_regions_for_tree, calculate_tree_of_regions, determine_range_stability, get_paths_from_region, reduce_tree_of_regions, visualize_stable_regions, write_stable_regions, select_regions_from_area	Option structure is not valid.
8	detect_regions_for_tree	Invalid input image must have 3 channels (RGB).
9	detect_regions_for_tree	Problem with reading the input image.
10	detect_regions_for_tree	Problem with filename of input image.
11	get_scale_space_layer	Problem with smoothing the image.
12	get_scale_space_layer	Problem with determining the watershed regions.
13	get_scale_space_layer, detect_regions_for_tree, visualize_stable_regions, select_regions_from_area	Problem with saving temporary data.
14	calculate_tree_of_regions, visualize_stable_regions, write_stable_regions, select_regions_from_area	Problem with loading temporary data.
15	calculate_tree_of_regions	Calculation of histogram.
16	determine_range_stability	Inversion of tree of region failed.
17	get_paths_from_region	Problem with tree analysis upwards / downwards.
18	determine_range_stability	Problem by determination of range stabilities.

Table 4: Error statements of Stable Regions IPM.

Nr.	Throwing Functions	Statement
19	reduce_tree_of_regions	Marking of regions failed.
20	reduce_tree_of_regions	Gathering marked regions into paths failed.
21	reduce_tree_of_regions	Determination of reduced tree nodes failed.
22	reduce_tree_of_regions	Arrangement of reduced stable regions tree failed.
23	reduce_tree_of_regions	Saving the Reduced Stable Regions Tree failed.
24	write_stable_regions	Problem with converting a region-bitmap into vector-representation.
25	write_stable_regions	Problem with saving stable regions into xml file.

4 Examples

In this chapter, we present some examples of the Stable Regions IPM. In table 5, we list the numbers detected stable regions for each of our 20 test images. The table also includes the sizes of each image. All calculations have been done with the same set of system parameters, whose values are listed in table 1.

Table 5: Size and number of detected stable regions per test image.

Image	Size	Regions		Image	Size	Regions
cups	134×71	33		Graz22_1	298×316	147
London01	458×334	324		Graz22_2	386×266	178
London02	458×334	357		Graz22_3	312×326	141
London03	530×352	292		Graz22_4	450×424	334
Graz16_1	438×368	303		Graz22_5	286×292	118
Graz16_2	618×368	426		Graz22_6	600×310	361
Graz16_3	756×410	462		Graz22_7	322×402	213
Graz16_4	295×356	196		Graz22_8	376×250	212
Graz16_5	370×304	200		Graz22_9	398×348	248

Table 5: Size and number of detected stable regions per test image.

Image	Size	Regions	Image	Size	Regions
Graz16	1409×1500	3 634	Graz22	1409×1500	4 328

Furthermore, we have visualized our results regarding the three images from London. You can see all detected regions in figs. 4, 6 and 8. Since the number of detected regions is relatively high, and many regions overlap others, we do also show 10 images showing only 10% of the detected stable regions, cf. figs. 5, 7 and 9.

Figure 4: All detected stable regions of London01.jpg

References

[Alvarez *et al.*, 1993] Alvarez, Luis, Guichard, Frederic, Lions, Pierre-Louis, & Morel, Jean-Michel. 1993. Axioms and fundamental equations in image processing. *Archive for Rational Mechanics*, **123**(3), 199–257.

[Brügelmann & Förstner, 1992] Brügelmann, Regine, & Förstner, Wolfgang. 1992. Noise Estimation for Color Edge Extraction. *Pages 90–107 of:* Förstner, W., & Ruwiedel, S. (eds), *Robust Computer Vision*. Wichmann, Karlsruhe.

[Drauschke *et al.*, 2006] Drauschke, Martin, Schuster, Hanns-Florian, & Förstner, Wolfgang. 2006. Detectability of Buildings in Aerial Images over Scale Space. *Pages 7–12 of:* Förstner, Wolfgang, & Steffen, Richard (eds), *Symposium of ISPRS Commission III: Photogrammetric Computer Vision*, vol. XXXVI. Bonn: ISPRS Commission III, for ISPRS.

[Florack & Kuijper, 2000] Florack, Luc, & Kuijper, Arjan. 2000. The Topological Structure of Scale-Space Images. *Journal of Mathematical Imaging and Vision*, **12**(1), 65–79.

[Kuiper *et al.*, 2003] Kuiper, Arjan, Florack, Luc, & Viergever, Max. 2003. Scale Space Hierarchy. *Journal of Mathematical Imaging and Vision*, **18**, 169–189.

[Lindeberg, 1994] Lindeberg, Tony. 1994. *Scale space theory in computer vision*. Kluwer Academic.

Figure 5: Results of London01.jpg

Figure 6: All detected stable regions of London01.jpg

Figure 7: Results of London02.jpg

Figure 8: All detected stable regions of London01.jpg

Figure 9: Results of London03.jpg