Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

# Interactive Hierarchical Image Segmentation on Irregular Pyramids

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Medizinische Informatik**

eingereicht von

**Michael Gerstmayer**
Matrikelnummer 0204613

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Univ.Ass. Dipl.-Ing. Dr.techn. Yll Haxhimusa

Wien, 14.03.2013

(Unterschrift Verfasser) (Unterschrift Betreuung)

# Erklärung zur Verfassung der Arbeit

Michael Gerstmayer
Rennweg 17, 4810 Gmunden

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst, die verwendeten Quellen und Hilfsmittel vollständig angegeben und die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen – die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

| | |
|---|---|
| (Ort, Datum) | (Unterschrift Verfasser) |

# Acknowledgements

I owe my gratitude to many people for various kinds of help with this work. To begin with I would like to thank my supervisor Yll Haxhimusa, who had a sympathetic ear and patience at all times. Also, I would like to thank Walter G. Kropatsch, Head of the Pattern Recognition and Image Processing Group, and all other colleagues at PRIP for their helpful questions and comments during my internal presentations. I highly appreciate the help of my friends and all companions on my way to climb the peak. Finally, thanks to my family, especially my parents for their continuous love and support.

# Abstract

Image segmentation, in general, is the process of dividing a digital image into segments having a strong correlation with objects in it. Various techniques exist to locate objects of interest formed by visual cues. However, general purpose segmentation methods cannot produce a perfect final segmentation by using low-level cues only. A way round the problem is rather to create a stack of segmentations with different resolution levels. Higher level knowledge shall then be used to confirm or select regions for further processing. In automatic region-based segmentation, usually such a stack of segmentations is built in a bottom-up manner, guided by low-level image feature data and the defined homogeneity criteria. We should take into account as well that the accuracy of an image segmentation is measurable, but its quality and usability are highly subjective and depend also on the scope of the application.

This thesis deals with modifications of such an irregular image segmentation pyramid and embedding additional knowledge about the problem domain such that the results of the image segmentation best suit the user. Based on an existing automatic segmentation framework – where the minimum spanning tree based method tries to capture perceptually important groupings – we bring the user into the loop and define interactive operations guiding the segmentation process. Semi-automatic approaches show multiple benefits (like flexibility and acceptance), but may sometimes be required also from juridical point of view. The interactive operations of merging and inhibition from merging require a representation that encodes the edge and the parent-child relationship of the merging tree. In this work each level of the irregular pyramid is represented by a combinatorial map, encoding both region and boundary information in a single combinatorial map structure. Using the connecting paths between the different levels of the pyramid, it is possible to set focus on regions from different granularities. In contrast to related approaches, this work is not limited to a single working level and pure sequential processing. Moreover, regions having different resolutions – down to pixel level – may be selected in parallel. This requires dedicated (pre-)processing and conflict resolution methods which guarantee consistency of applying the operations throughout the hierarchy. The output is a stack of segmentations with a final result that best suits the users' applications, in the topmost level of the hierarchy.

We try to find out answers related to usability questions of the interactive segmentation tool developed and empirical values on the operations defined. As it turned out, the candidates (beginners) were able to produce results satisfying their expectations. The data recorded during the segmentation-sessions reveals different strategies and gives evidence on the usage of the interactive operations. This work can be used for problems where accuracy in image segmentation, annotating images or creating ground truth among others is needed.

# Kurzfassung

Unter Bildsegmentierung versteht man im Allgemeinen den Prozess des Aufteilens eines digitalen Bildes in Segmente, wobei diese eine starke Korrelation mit den im Bild enthaltenen Objekten aufweisen sollen. Um ein vorab definiertes Zielobjekt – das aus unterschiedlichsten visuellen Reizen bestehen kann (z.B. Farbe, Symmetrie oder bekannte Formen) – zu extrahieren, gibt es mehrere Methoden. Jedoch lässt sich ein perfektes Endergebnis nicht mit Ansätzen, die für eine breites Anwendungsgebiet gedacht sind und nur mit einfachen Bildinformationen arbeiten, erzielen. Ein Weg, um dieses Problem zu lösen, besteht darin, nicht ein perfektes Ergebnis, sondern eine Hierarchie von Segmentierungen mit unterschiedlichen Auflösungen zu erstellen. Spezielles Wissen über den Problembereich kann dann dazu verwendet werden, einzelne Segmente zu bestätigen oder sie detaillierter zu bearbeiten. Automatische regionenorientierte Verfahren bauen eine solche Hierarchie von unten nach oben auf. Die verschiedenen Segmentierungstufen werden dabei durch die im Bild enthaltenen Informationen sowie ein definiertes Homogenitätskriterium bestimmt. Ebenfalls muss berücksichtigt werden, dass sich die Genauigkeit einer Bildsegmentierung zwar messen lässt, die Qualität und Verwendbarkeit des Ergebnisses jedoch stark subjektiv ist und von der weiteren Anwendung abhängt.

Diese Arbeit nimmt diesen Ansatz auf. Einzelne Segmente einer irregulären Bildpyramide werden dabei – unter Einbindung von zusätzlichem Wissen über das Segmentierungsziel – bestätigt oder bearbeitet. Basierend auf einer automatischen Methode, die Borůvkas minimalen aufspannenden Baum (MST) dazu verwendet, wichtige Gruppierungen – ähnlich dem Vorgang der menschlichen Wahrnehmung – zu segmentieren, binden wir zusätzlich den Benutzer in den Prozess ein. Speziell dafür definierte interaktive Operationen sollen so den Segmentierungsprozess beeinflussen. Solche halbautomatisierten Ansätze haben in Bezug auf Flexibilität und Akzeptanz viele Vorteile, können aber manchmal auch aus juristischen Gründen erforderlich sein.

Die Operationen 'merging' (dt.: Verschmelzen) von zwei durch den Benutzer ausgewählten Regionen und 'inhibition' (dt.: Verhindern) derselbigen basieren auf dem Konzept der Kanten eines Graphen und der Eltern-Kind-Information von Segmenten der Hierarchie. In dieser Arbeit wird jede Segmentierung durch einen kombinatorischen Graph auf struktureller Ebene repräsentiert. Dieser duale Graph beinhaltet sowohl die Information über Regionen als auch deren Kanten gleichzeitig. Mithilfe der Verbindungen (Eltern-Kind) zwischen den verschiedenen Ebenen der irregulären Pyramide ist es möglich, den Fokus auf mehrere Segmente gleichzeitig in verschiedenen Ebenen der Hierarchie zu setzen. Im Gegensatz zu anderen Arbeiten ist diese nicht auf eine einzige Ebene und eine reine sequenzielle Verarbeitung

beschränkt. Darüber hinaus können die zu bearbeitenden Segmente mit verschiedenen Auflösungen - bis hin zur Pixelebene - parallel ausgewählt werden. Dies erfordert eine spezielle (Vor-) Verarbeitung und Methoden, um die Konsistenz der irregulären Pyramide und des Ergebnisses zu gewährleisten. Das Endergebnis ist ein Stapel von Segmentierungen, in dem sich das Endergebnis im obersten Level befindet und vollständig den Erwartungen des Benutzers entspricht.

Im Rahmen der Evaluierung versuchten wir, Antworten einerseits zu Fragen der Benutzerfreundlichkeit der interaktiven Segmentierungmethode zu finden und andererseits empirische Werte zu den interaktiven Operationen selbst zu bekommen. Wie sich herausstellte, konnten die Kandidaten, die alle Anfänger waren, Ergebnisse produzieren, die ihre Erwartungen erfüllten. Die Daten, die während den einzelnen Segmentierung-Einheiten aufgenommen wurden, zeigen verschiedene Strategien und ermöglichen Erkenntnisse über die Nutzung der interaktiven Operationen. Offene Probleme und zukünftige Entwicklungen werden zusätzlich in dieser Arbeit diskutiert.

Dieses Verfahren kann zur Problemlösung verwendet werden, wenn Genauigkeit bei der Bildsegmentierung, der Annotation von Bildern oder dem Erstellen von Ground Truth benötigt wird. Weiters können mit dieser Methode auch eigene Abstrahierungsebenen eingefügt werden. Dieses Konzept ist allgemein und offen für Erweiterungen oder Optimierungen.

# Contents


**1 Introduction** **2**
- 1.1 Problem Statement . . . 3
- 1.2 Aim of the Work . . . 4
- 1.3 Contribution of the Work . . . 5
- 1.4 Structure of the Thesis . . . 5

**2 Basics of Hierarchical Image Segmentations** **6**
- 2.1 Image Segmentation . . . 6
- 2.2 Topological Data Structures . . . 11
- 2.3 (Irregular) Pyramid Representation . . . 16
- 2.4 Knowledge and Image Segmentation Algorithms . . . 21
- 2.5 State of the Art . . . 23

**3 Automatic Hierarchical Image Segmentation** **30**
- 3.1 Minimum Spanning Tree based Method . . . 30
- 3.2 Graph based Segmentation Algorithm Alternatives . . . 37

**4 Interactive Hierarchical Image Segmentation** **40**
- 4.1 Interface to Automatic Approach . . . 40
- 4.2 Definition of Operations . . . 42
- 4.3 Processing . . . 49

**5 Evaluation** **64**
- 5.1 Setup . . . 64
- 5.2 Results . . . 67

**6 Conclusion** **74**

**Appendices** **75**

**A Interactive Segmentation Tool** **76**
- A.1 Installation Guide . . . 76
- A.2 Tutorial . . . 80

**B Evaluation Documents** **86**
B.1 Questionare Score . . . . . 87
B.2 Declaration of Consent . . . . . 99

**Bibliography** **100**

CHAPTER 1

# Introduction

Referring to [Sonka et al., 2008, p. 175], “Image Segmentation is one of the most important steps leading to the analysis of processed image data - its main goal is to divide an image into parts that have a strong correlation with objects... contained in the image”. There are several algorithms available for the purpose of creating image segmentations, each dedicated to a certain domain of problems. Their quality and usability of the result for further processing depends on various factors. Among others, the type of cues and knowledge used to create partitions is one of the essential. This all indicates to [Shi and Malik, 2000] “that image segmentation based on low-level cues cannot and should not aim to produce a complete final ’correct’ segmentation”. Instead, the objective should be to use the low-level coherence in order to create hierarchical partitions. Finally they conclude, mid- and high-level knowledge may than be used to either confirm these groups or select some for further attention.

[Shi and Malik, 2000] summarize a general problem “since there are many possible partitions of the domain of an image into subsets, how do we pick the ’right’ one?”. Regarding the current problems in image processing, the authors state that in general the following two aspects need to be considered:

- There are several possible interpretations (i.e. partitions of an image into subsets) in the context of prior world knowledge.
- The partitioning is inherently hierarchical (relations of parts to any given whole)

Regarding the first point, as there may not be a single correct answer on the question which partition to choose, [Shi and Malik, 2000] conclude that a “Bayesian view is appropriate”. For the second aspect, they suggest to “think of returning a tree structure corresponding to a hierarchical partition instead of a single ’flat’ partition”.

Usually the results delivered by automatic segmentation methods – building a stack of segmentations by using low-level cues – do not meet neither the requirements nor the users’ expectations. This diploma thesis is about the interactive selection and modification of partitions

in an hierarchical irregular pyramidal data structures. Categorizing this work within the field of image processing, there are the following keywords to describe the overall context:

- Image Segmentation
- User Interaction
- Graph Theory
- Topology
- Hierarchical Data Representation
- Perceptual Grouping
- Annotation

## 1.1 Problem Statement

In this work we attempt to satisfy both annotation and segmentation requirements, so far both processes share various similarities. The annotation process itself is hierarchical. Moreover hierarchies are plausible by nature [Culhane and Tsotsos, 1990] - e.g. a car consists of wheels, a car body and its interior. Using LabelMe approach [Martin et al., 2001], each level of granularity has to be created individually. Furthermore it lacks the explicit and formal encoding of the hierarchical annotation relations. The segmentation process itself is hierarchical too as smaller parts are iteratively grouped together [Zahn, 1971]. Usually it all starts on pixel level, where each pixel corresponds to a region. However, working on pixel level may not be feasible for every application and working on higher level of abstraction/granularity is desired. In order to create different levels of abstraction, the regions are iteratively merged together according to the defined homogeneity criteria. In the approach presented, such a stack of segmentations is automatically built using the minimum spanning tree based method (see Chapter 3 for details) proposed by [Haxhimusa and Kropatsch, 2004, Haxhimusa et al., 2005].

In pure interactive approaches, working close to pixel precision can be time consuming and requires considerable amount of human interaction. This problem has been described and considered to be a drawback in recent image annotation tools like LabelMe [Torralba et al., 2010], where on pixel-level segmented objects are annotated with vocabulary. One possibility to automate the process is again to delegate generic segmentation tasks to an algorithm and retain the user for initialization or optimization steps. Not only for the purpose of interactively controlling the merging of regions, this proceeding allows introducing additional knowledge about the problem domain into the segmentation process to improve the results. Moreover, the acceptance of a partition is highly subjective and embedding the user can be required also from juridical point of view. In conclusion, using a semi-automatic segmentation method has several benefits.

Depending on the composition of automatic/interactive methods and the different types of knowledge, a framework can either be restricted to a certain domain of images or generically applicable on a variety of image classes. Section 2.4 in the basics' chapter covers various types of knowledge, its use in image segmentation algorithms and the associated tradeoff. We decided to use a general purpose and semi-automatic approach. After a study on the state-of-the-art in Section 2.5 and having also the task of image annotation in mind, we sketched the cornerstones of the contribution:

- Integration of user-interaction into an existing automatic hierarchical image-segmentation framework
- Definition of *merging* and *inhibition of merging* operations on regions to guide the segmentation process
- Working at different granularity levels within the hierarchy
- Using of pixel-level precision versus super-pixel approach
- Consistency of operations within the hierarchy
- Ability to produce any segmentation result

## 1.2 Aim of the Work

[Meine et al., 2004] point out the benefits of reusing components of one segmentation method in another one and the possibility of switching to the most appropriate method for a specific segmentation task. The authors state that the combination is difficult because different algorithms "use incompatible data representations for almost all levels beyond the pixel matrix". They invent a unified representation called GeoMap which covers the requirements of a large number of algorithms. It is based on an extended topological representation and the user will be integrated into the loop as well. In contrast to regular pyramids, which have a rigid structure and known drawbacks [Bister et al., 1990], irregular pyramids dynamically adapt to the image structure. They are widely used for such kind of problems in image analysis [Simon et al., 2006] and are able to represent image partitions at different resolution levels. In conclusion, the data structure plays a key role.

However, the initialization at the base and sequent decimation steps in the first levels are computationally intense and very crucial to the overall segmentation result. Consequently these steps are often automatized. As most approaches rely on an initial over-segmentation, it is impossible to access lowest levels of detail for refinement. Furthermore boundaries not detected in the beginning also cannot be corrected later on. In this work this is not the case and pixel-level precision is reached.

While the GeoMap [Meine et al., 2004] framework is rather specific concerning the representation chosen, this diploma thesis is about interactive modifications of irregular image segmentation pyramids [Kropatsch, 1995] in general. In order to exert influence on the segmentation process, we use the topological relations defined within the levels and connecting paths [Glantz and Kropatsch, 1994] that relate the elements between the different levels of resolution in irregular combinatorial pyramids. Another important aspect of this work is the duality concept of the combinatorial map structures that allows to switch between the relevant representations in terms of regions and boundaries for processing.

Doing it this way it is possible to guide the process of creating the stack of segmentations. It shall be possible to access all levels within the hierarchy and select regions in multiple levels of granularity in one step. The output is a stack of segmentations with a final result that best suits the users' applications on top.

## 1.3 Contribution of the Work

The definition of interactive operations on the irregular graph pyramid and its embedding into the automatic framework – in order to guide the segmentation process – is an essential part of the work presented. Based on the information encoded in the combinatorial map structure and the merging tree, dedicated processing steps within the hierarchy are introduced to guarantee consistency of the operations and propagate the changes. In addition we provide methods to select regions from different levels of resolution (until pixel-level precision) for further attention and which support the user in creating image segmentation results that meet his expectations.

An evaluation of the framework developed against the state-of-the-art (see Section 2.5) is not intended at this time. Among others, the problems making it difficult to compare them are based on technical differences and framework parametrization. Furthermore segmentation accuracy is not relevant in our case as it is possible to produce any result using the approach developed. Instead we evaluate questions regarding the usability/functionality implemented and discuss the findings based on the empirical values gathered in the segmentation sessions.

## 1.4 Structure of the Thesis

After a first overview it is necessary to describe in Chapter 2 the basic theory and state-of-the-art related to this work. We discuss the segmentation framework at the base in Chapter 3, followed by the definition of the interactive operations on top of it in Chapter 4. The introduced interaction methods on irregular pyramids, possible segmentation strategies and related user experience are subject of research and will be evaluated in Chapter 5. A conclusion of this work and a list of improvements to be added in future is given in the final Chapter 6. The two Appendices A and B contain a documentation/tutorial on the tools developed and the detailed answers of the questionnaire in the evaluation part.

CHAPTER 2

# Basics of Hierarchical Image Segmentations

In this chapter we cover the basic concepts needed in the rest of the thesis. Starting with image segmentation as governing theme, the graph-based foundations required in the subsequent chapters are introduced. We will discuss hierarchical representations based on topological data structures representing the image-segmentations results and the different types of knowledge to improve the same. This finally leads to a comparison of the state-of-the-art, having focus on semi-automatic and interactive segmentation frameworks.

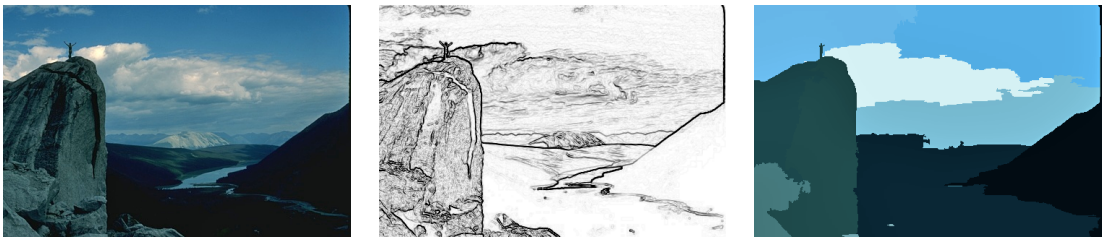
## 2.1 Image Segmentation

Any object contained in an image is formed by different cues. According to [Martin et al., 2001], cues are ranked with respect to the following levels of complexity:

- **Low:** Coherence of brightness, texture, continuity of a contour.
- **Intermediate:** Symmetry and convexity
- **High:** Recognition of familiar objects

They are fundamental in the process of creating image segmentations, both for computer programs and humans. Various techniques exist to segment an input image into the objects of interest (corresponding to the foreground) and its background. A general categorization is presented in [Sonka et al., 2008, pp. 175 – 256]:

- **Thresholding:** By choosing a brightness constant as threshold, an image is separated into foreground and background.



(a) Input image (b) Edge image (c) Region image

Figure 2.1: An input image (a) and its example segmentations created using (b) edge-based or (c) region-based methods.

- **Edge-based segmentation:** Edges mark discontinuities – like a significant change of brightness – in an image. They are detected using local filtering operators (convolution kernels) returning e.g. first (gradient) or second (Laplace) derivative in the spatial domain. Chained significant edges lead to borders. Among other methods also border tracing or graph search based algorithms exist. In the end its output is used to construct regions.

- **Region-based segmentation:** The methods following this paradigm create regions directly. Based on a homogeneity criterion, the region-growing process iteratively merges – in a bottom-up approach – smaller/adjacent regions together. Alternatively there may be dividing (top-down) and hybrid approaches.

In general the segmentations derived by latter two paradigms are dual, i.e. regions can be constructed from their borders and vice versa. These results can be combined in a single structure like graphs (see Section 2.2 for details). However, the output of edge- and region based methods are usually not the same. Figure 2.1 shows some examples, the image in (a) is taken from Berkeley image segmentation database [Martin et al., 2001]. The segmentation in subfigure (b) shows edges extracted by the Sobel operator. This edge-based method calculates a gradient image using the first derivative of pixel-intensity values. In (c) a region image having coarse granularity in a step of the region-merging process is shown.

Please note that the usage of the technical terms varies in literature as there exist several synonyms. By definition from [Sonka et al., 2008, p. 332], a region is identified by a label assigned "with a unique (integer) number; such identification is called labeling or coloring (also connected component labeling), and the largest integer label usually gives the number of regions in the image". Hence all pixels belonging to the same region in Figure 2.1c share a common label [Braquelaire and Brun, 1998] and are visualized using e.g. the mean-color. Depending on the context, the terms "division, area, section, partition, face and segment" are used equivalently for a region[1]. Due to [Meine and Köthe, 2005], "there are several definitions of regions and its boundaries in discrete images":

[1] See definitions and thesaurus in Oxford Dictionaries of English.

- Crack-edges based interpretation of region images
- 8-connected boundaries between 4-connected regions or vice versa.

Limit, border and boundary are also often used equivalently. The difference between *boundaries* and *edges* in general will become more clear when having a look at the definition from [Sonka et al., 2008, pp. 21 – 22]. "An edge is a... property of a pixel and its immediate neighborhood" represented by a vector and "tells us how fast the image intensity varies". The (inner) border of a region $R$ is "the set of pixels within the region that have one or more neighbors outside $R$." Therefore a border is a global concept and an edge represents related local properties.

Concluding this first section, for the problem of extracting objects from complex images, several methods lead to a result. The decision which one is chosen depends on several factors and also on the future scope of an application. Referring to [Braquelaire and Brun, 1998], "it quickly appeared that this problem cannot be solved without an priori knowledge of the objects to be recognized". Therefore another classification is performed for such algorithms:

- **Domain-dependent:** which attempt to recognize specific objects in a scene;
- **Domain-independent:** which produces a partition of the original image into a set of regions according to a homogeneity criterion.

Additional knowledge about the problem domain takes in a key role in optimizing the results for further processing. In addition a specific pre-processing phase (like application of a median filter in order to remove disturbing noise), combinations of different approaches could also be used for improvement. Such combinations are usually restricted by technical means, e.g. incompatibilities with respect to the data structure [Braquelaire and Brun, 1998, Meine et al., 2004] used. More on external knowledge, like the integration of user-interaction, is presented in Section 2.4 within this chapter as it is a key feature of this work too.

As a premise for this work, the input images need not to be pre-processed, furthermore the approach falls into the second category of domain-independent applications. It is based on region-based segmentations, so thresholding and edge-based segmentation will not be covered in detail.

### Region-based Segmentation

But how to derive these regions in the region-based segmentation paradigm? [Braquelaire and Brun, 1998] list the three approaches that are possible:

1. **The bottom-up approach:** Small regions are aggregated into larger ones by using merging algorithms.
2. **The top-down approach:** An image is iteratively split into smaller and smaller regions by using splitting algorithms.
3. **The split-and-merge approach:** Implements a combination of both.

**Procedure 1** Region growing

1: Define some starting method to segment the image into many small regions satisfying condition (2.2);
2: Define a criterion for merging two adjacent regions;
3: Merge all adjacent regions satisfying the merging criterion. If no two regions can be merged maintaining condition (2.1), stop;

In general, the first procedure, as defined in Procedure 1, is known as region growing[2]. In more detail, *first* a starting method is chosen which segments the input-image into many small regions. The following two may be used in order to get a starting level:

- **Identity:** Take the original input image where each pixel represents a single region ($1:1$ correspondence).
- **Super-pixels:** [Ren and Malik, 2003] introduce a preceding stage which groups pixels into *super-pixels*. The method returns a segmentation of coarser granularity than pixel level. This grouping can be characterized best as over-segmentation (relating the output to a ground truth segmentation). As this initial phase is crucial for the overall result, this task is often delegated. Resulting segmentations based on super-pixels are often used as input/basis for other algorithms, however regions not identified in the beginning will not be available for further processing.

Anticipating the following two formulas[3], the *second* step of Procedure 1 gets more clear.

$$H(R_i \cup R_j) = \text{FALSE}, \quad i \neq j \quad R_i \text{ adjacent to } R_j \tag{2.1}$$

$$H(R_i) = \text{TRUE}, \quad i = 1, 2, \ldots, S \tag{2.2}$$

For $S$ being the total number of regions, the resulting must be both homogenous (due to an evaluation function $H$) satisfying Equation 2.2 and maximal with respect to Equation 2.1. The regions in the beginning most certainly do not satisfy Equation 2.1 [Sonka et al., 2008, p. 224].

In the *third* and last step we define a criterion for merging two adjacent regions. As an example one can use distance-measures based on color-means (low level cues). Later in Chapter 3, related concepts like perceptual grouping [Felzenszwalb and Huttenlocher, 2004] are introduced. If the resulting difference between two adjacent regions exceeds a given threshold (merging criterion), then the regions are merged together. This iteratively happens for all regions as long as the homogeneity criterion, in Equation 2.2 remains satisfied, then it stops. During this iterative reduction one can track the merging/segmentation tree as shown in Figure 2.2. In this picture a circle represents a region in the level specified on the left (indicated by the dotted horizontal bar). Already in the next step, e.g. from level $0$ to $1$, adjacent regions are merged together. The parent-child relation of successor and predecessor regions defines a reduction window. This is the abstract procedure and basis for lots of similar approaches

[2] Procedure 6.17 taken from [Sonka et al., 2008, p. 225].
[3] Equations 6.30 and 6.31 taken from [Sonka et al., 2008, p. 224].

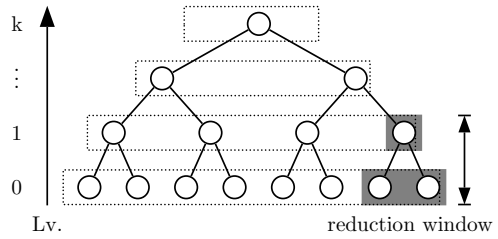



Figure 2.2: Merging Tree, modified according to [Haxhimusa, 2007, p. 31].

creating region-based image segmentations. The questions which remains to be answered, how to evaluate the resulting output? Parametrized on a threshold $T$ that specifies the amount of mutual overlap between a segmented region and the corresponding ground truth region, the resulting segmentation of a region-of-interest is classified by [Sonka and Fitzpatrick, 2001] into the following five conditions:

- *A correct segmentation of a region:* overlap between a region in the segmented image and a region in the ground truth.
- *An over-segmentation of a region:* occurs when a collection of regions in the segmented image corresponds to one region in the ground truth.
- *An under-segmentation of a region:* occurs when a collection of regions in the ground truth corresponds to one region in the segmented image.
- *A missed region:* has no good correspondence in the segmented image.
- *A noise region:* has no good correspondence in the ground truth.

After a rough definition on the main topic of this work and covering related theoretical aspects, there are some questions which remain to be answered:

- In order to merge adjacent regions, how to encode topological relations?
- Regions get iteratively merged together, what is the benefit of using a hierarchical merging tree instead of a flat representation?
- Cluttered background, shadows or noise make the segmentation task even more difficult. How to improve the results?
- Moreover, the concept of a good partition may depend on the purpose of its use and may be highly subjective [Klava et al., 2009]. So why not integrating the user?

The first two questions will be answered in detail in Section 2.2, the remaining two within Section 2.4 in this chapter.

## 2.2 Topological Data Structures

### Definition of Topology

As important as the choice for the appropriate segmentation method is the decision for the image representations at its base. Topological properties like spatial relationships between regions are necessary for computation. They are usually described in terms of adjacency, proximity, connectivity, membership and orientation. For the purpose of defining modification-operations on adjacent regions and moreover within the hierarchical structure, the graph-based representations (presented later in this chapter) need to be able to encode this type of information.

In contrast to distance measurement in Euclidean space, these properties are invariant to continuos transformations. Translated from the introducing book on topology, [Ossa, 2009, pp. 2-3] describes the term as follows:

> Topology is also referred as 'Continuous' or 'Rubber-Sheet Geometry'. This geometry is defined on the fundamental term of Homeomorphic Relations.
>
> $$\overline{f} : M \to N, \widetilde{f} = (x, f(x))$$
>
> Transformations $f$ between the sets $M$ and $N$ such as as bending and denting have no influence on the topological properties (imagine inclusions such as holes). The sets stay topologically equivalent, whereas tearing is not allowed since the space is no longer continuous.

[Egenhofer, 1989] formalizes the relations above algebraically using a representation model based on minimal objects called *simplices*. As per his definition, a simplex exists for each dimension (0...n) of a spatial object for example:

- *0-simplex* represents a node,
- *1-simplex* an edge,
- *2-simplex* a triangle

and even a n-simplex can be defined. In order to depict the relations, he gives the following example: "Any $n$-simplex is composed of $n + 1$ geometrically independent simplices of dimension $n - 1$. For example a triangle (2-simplex) is bounded by three 1-simplices."

Furthermore [Egenhofer, 1989] combines those elements to a *simplicial complex*, which is a (finite) collection of such simplices. Based on empty and non-empty intersection of its simplicial elements the spatial relationships mentioned in the beginning are defined. Hence objects are considered as e.g. neighbors if they share a common edge (1-simplex). Finally also clockwise or counterclockwise orientation of 1-simplices (edges) is introduced, a concept which we will need again in Section 2.2 of this chapter.

## Graphs

The terminology of nodes and edges joining them is also used in graph theory. Hence the same are often used to represent topological relations [Sonka et al., 2008, p. 104] like adjacency which is needed for processing of split and merge operations. [Lienhardt, 1991] says that "a solid is defined by a subdivision of a surface (informally, a partition of this surface into vertices, edges and faces, that is into cells of dimension 0,1, and 2)". Following this definition, two types of topological models can be distinguished [Lienhardt, 1991]:

- **Incidence Graphs:** Vertices can have regions/cells of modeled subdivision assigned. The edges represent the adjacency and incidence relations between them.
- **Ordered Models:** A single type of basic element on which element functions act, are used to describe the relations.

Both of these graphs have a different representation. An incidence graph is defined by $G$ containing vertices $V$ and arcs/edges $E$ connecting pairs of them[4].

$$G = (V, E)$$
$$V = x_0, x_1, \ldots, x_k$$
$$E = x_0x_1, x_1x_2, \ldots, x_{k-1}x_k$$

This neighborhood graph is also called primal graph $G$. Figure 2.3 shows an undirected graph, where the black dots form the Vertices $x \in V$ and black lines the edges $e \in E$ of $G$. Its topologically dual $\overline{G}$ or $G^*$ can be deduced by the following scheme:

> "Every connected plane multigraph has a plane dual. Indeed... we start by picking from each face $f$ of $G$ a point $v^*(f)$ as vertex for $G^*$. We can then link these vertices up by independent arcs..." [Diestel, 2010, p. 108]

[4] Graph notions used from [Diestel, 2010].

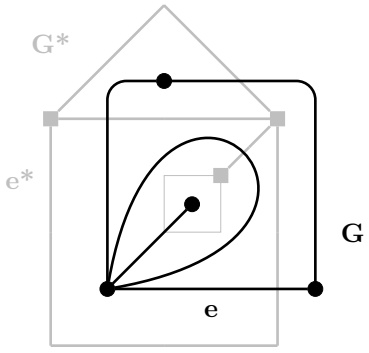
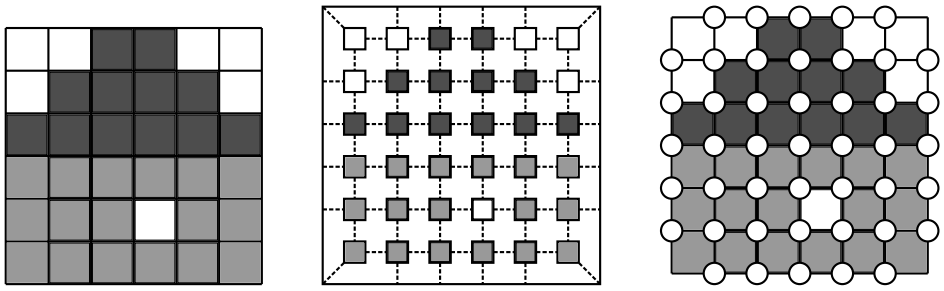



Figure 2.3: A planar graph $G$ and its dual $G^*$.



(a) Image (b) Adjacency Graph (c) Boundary Graph

Figure 2.4: Embedding graphs of a 2D image. (a) Regions on pixel-level. (b) Adjacency relations between the former, visualized using a RAG. (c) Boundary Graph containing nodes and interpixel-boundaries.

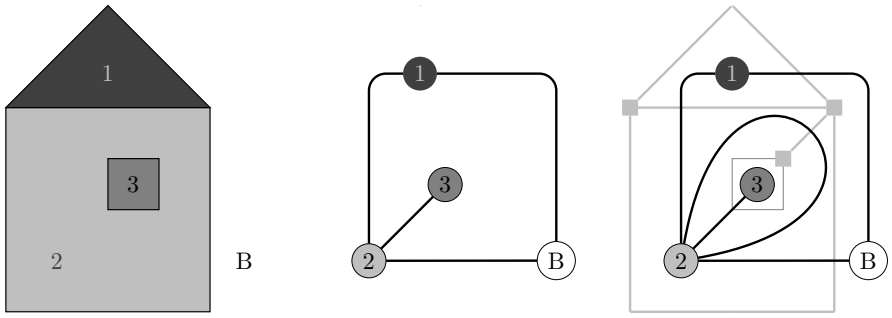
On interpreting the graphs in terms of region-based segmentation (see Figure 2.4a), the primal encodes the adjacency relations between the regions (see Figure 2.4b), i.e. vertices have regions assigned and edges indicate relations. Whereas in this case the dual represents the boundaries of the regions (see Figure 2.4c), i.e. the interpixel vertices and edges [Braquelaire and Brun, 1998]. Interpixel boundaries, in contradiction to 'natural' edges derived from edge-based segmentation, have no 'visible' correspondence in the image.

The second type, ordered models, is explained in Section 2.2 covering combinatorial maps. In conclusion, when using 2D images, the assignment of graphs' basic-elements to match the image structure yield these two complementary types of graphs $G$ and $\overline{G}$. They are used to describe the structure (spatial relationships) within the image. Furthermore, a weight value can be stored on the edges for computation and indicating e.g. a likelihood of being a true boundary [Shi and Malik, 2000, Arbelaez et al., 2009].

In the following list of topological graphs, the assignment of which graph to be primal and dual varies in literature. To avoid misconceptions we define the primal to be the one capturing the main aspect of a representation, this will be indicated in the text accordingly. Starting from typically used region adjacency graphs, we proceed to more advanced representations like combinatorial maps. The choice of the underlying data structure "considerably effects ... an algorithm and is therefore a fundamental question" [Sonka et al., 2008, p. 175]. Using graphs at the base has various benefits. Energy or cost minimization may be realised using graph-optimization algorithms. Well known frameworks using graph-cuts approach [Boykov and Jolly, 2001], as well as this work make use of them (see Section 3.1). The benefits and drawbacks of each graph type will be discussed.

### RAG

An example of a 2D region adjacency graph is shown in Figure 2.5b. Nodes correspond to regions shown in Figure 2.5a and an arc/edge between two regions indicates their adjacency. To specify a relationship (e.g. to be left/right/inside), descriptions can be put on an edge [Sonka




(a) Regions (b) RAG (c) Dual Graph

Figure 2.5: Image regions and corresponding graphs. (a) Shows the region image. (b) Representation of adjacency relations using the RAG. (c) Precise topological correct representation using dual graph.

et al., 2008, p. 104, p. 385]. Simple- or multi-adjacency and inclusion are represented in the same way [Simon et al., 2006].

Due to this 'informal' representation of topology, structural modifications and resolving conflicts/ambiguities are a difficult task. Nevertheless a RAG is a simple and popular model. Computation over topological properties requires – especially in higher dimensions (e.g. loss of information in $n$D-images) – an advanced representation formalism.

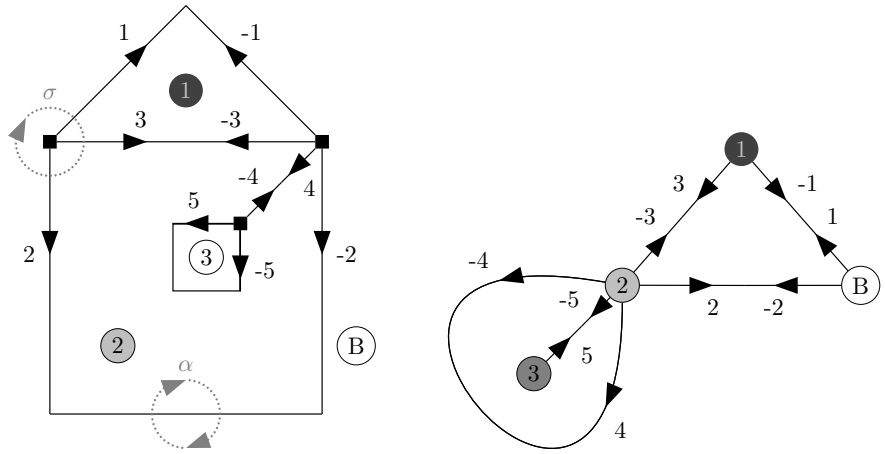
### Dual Graphs

A dual image graph ($G, \overline{G}$, see Figure 2.5c) consists both of a 2D region adjacency (primal) and boundary (dual) representation. It is created using the method shown in [Diestel, 2010]. Multi-adjacency and inclusion relations are now represented correctly, the latter clearly by a self loop in the region graph. But still not all topological information is available, i.e. the ordering of regions around a vertex [Simon et al., 2006] to distinguish them. Furthermore, two separate graphs need to be stored.

### 2-Dimensional Combinatorial Maps

2-Dimensional combinatorial maps (see Figure 2.6), as defined by [Lienhardt, 1994], belong to the class of ordered models and can be used to represent the topology of subdivisions of oriented topological surfaces. Inclusions were already handled correctly using dual graphs, but they are still missing ordering information. Combinatorial maps overcome this restriction by adding the concept of orientation [Egenhofer, 1989].

**Definition 2.2.1.** *A combinatorial map $G$ is the triplet $G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha)$, where $\boldsymbol{\mathcal{D}}$ is the set of darts and $\sigma, \alpha$ are two permutations defined on $\boldsymbol{\mathcal{D}}$ such that $\alpha$ is an involution:* $(\forall d \in \boldsymbol{\mathcal{D}} \alpha^2(d) = d)$.




(a) 2D Combinatorial Map (b) 2D Combinatorial Map - Dual

Figure 2.6: Combinatorial Map of Figure 2.5a.

Therefore each edge is split into two half-edges, which is the basic element of this representation called dart. Since each edge connects two vertices, a dart uniquely belongs to one vertex. Hence two darts on the same edge have opposite directions. The darts $\in \boldsymbol{\mathcal{D}}$ in combination with the permutation $\sigma$ (ordering along a vertex) and involution $\alpha$ fully characterize the ordered graph in 2-D space [Brun and Kropatsch, 1999]:

$$G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha) \tag{2.3}$$

By definition of a fixed orientation (clockwise, counterclockwise) it is possible to enumerate all darts in the specified order. This ordering/permutation can be traversed with the $\sigma$ permutation (see Figure 2.6a, clockwise). All darts (indicated by arrows) of a vertex (indicated by a black square) are enumerated by consecutive application of this operation, this cycle/orbit can be abbreviated with e.g. $\sigma^*(1) = (1, 3, 2)$. To retrieve the opposite dart on the same edge, the $\alpha$ involution (see Figure 2.6a) is defined, e.g. applying $\alpha(-2)$ yields 2 and vice versa. Mapping to the definition of $G$ in terms of vertices and edges, the vertex set corresponds to $V = \sigma^*(\boldsymbol{\mathcal{D}})$ and the edge set to $E = \alpha^*(\boldsymbol{\mathcal{D}})$ [Haxhimusa et al., 2005] accordingly.

In contrast to dual graphs before, there is no need to store its dual separately. Through definition of a new operation $\varphi = \sigma \circ \alpha$ (by consecutively applying the defined operations before), it is possible to implicitly derive the adjacency representation:

$$\overline{G} = (\boldsymbol{\mathcal{D}}, \varphi, \alpha) \tag{2.4}$$

The darts $\varphi^*(-1) = (-1, 3)$ encode both the boundary of region 1 in the primal graph and its adjacency relations (see Figure 2.6b) in the dual. By now applying the operation $\varphi$ on the opposite dart ($\alpha$), one gets the darts defining the adjacent region $B$ (representing the background) as they are touching the same edge: $\varphi^*(1) = (1, -2)$. This is shown in the dual correspondent in Figure 2.6b. The new vertices in the dual representation are equivalent to the region indices, the

$$
\begin{aligned}
&\boldsymbol{\mathcal{D}}\ (1,-1,2,-2,3,-3,4,-4,5,-5)\\
&\alpha\ = (1,-1)(2,-2)(3,-3)(4,-4)(5,-5)\\
&\sigma\ = (1,3,2)(2,4,-3,-1)(-4,-5,5)\\
&\varphi\ = (-1,3)(-3,2,4,-5,-4)\ \underbrace{(5)}_{\text{inclusion}}\ \underbrace{(1,-2)}_{f_{\varphi}^{\infty}}
\end{aligned}
$$

Figure 2.7: Dartset $\boldsymbol{\mathcal{D}}$ and its operations encoding the example-graphs in Figure 2.6.

darts leaving each vertex encode all the information necessary for deriving adjacency/ordering relations. $f_{\varphi}^{\infty}$ denotes the infinite face or the background $B$. $\boldsymbol{\mathcal{D}}$ together with the operations $\alpha, \sigma, \varphi$ implicitly encodes all we need just in one representation, see Figure 2.7 for an example. This topological representation is able to handle ordering and inclusion information correctly. A formal description for n-dimensional combinatorial maps can be found in [Lienhardt, 1994].

### n-Dimensional Generalized Combinatorial Maps

[Lienhardt, 1991, Lienhardt, 1994] describes the n-dimensional generalized maps as "combinatorial model for representing the topology of subdivisions of orientable or non-orientable quasi manifolds[5] with or without boundaries". Unlike before, where either topological information was lost in higher dimensions or the formalism was restricted to 2D images only, n-G-maps overcome these problems. They are homogeneously defined for any dimension - this applies also to removal operations. An example graph is defined as follows:

$$G = (\boldsymbol{\mathcal{D}}, \alpha_0, \alpha_1, \ldots, \alpha_n) \tag{2.5}$$

where $\boldsymbol{\mathcal{D}}$ is a non-empty set of darts and $\alpha_0, \alpha_1, \ldots, \alpha_n$ are involutions on $\boldsymbol{\mathcal{D}}$ . In contrast to before, all operations are involutions defined equivalently:

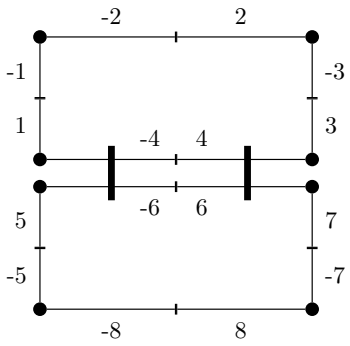
$$\forall i, 0 \leq i \leq n, \forall d \in \boldsymbol{\mathcal{D}}, \alpha_i^2(d) = d \tag{2.6}$$

An example of tiled frame – represented as a 2-Dimensional Generalized Combinatorial Map – is shown in Figure 2.8. All involutions $\alpha_i$ represent adjacencies, where $\alpha_0$ corresponds to $\alpha$ permutation defined before and $\alpha_1$ to $\sigma$ similarly. The black bar in Figure 2.8 indicates the new operation $\alpha_2$ encoding the adjacency relation.

## 2.3 (Irregular) Pyramid Representation

The usage of hierarchical representations in opposite to flat data structures in image analysis is motivated by several findings. [Culhane and Tsotsos, 1990] claim that hierarchies are "plausible

[5]For an explanation see [Brickell and Clark, 1970].




$$\alpha_0 = \{(1,-1),(2,-2),(3,-3),(4,-4),(5,-5),$$
$$(6,-6),(7,-7),(8,-8)\}$$
$$\alpha_1 = \{(1,-4),(-1,-2),(2,-3),(3,4),(5,-6),$$
$$(6,7),(-7,8),(-8,-5)\}$$
$$\alpha_2 = \{(-4,-6),(4,6)\}$$

Figure 2.8: 2D Generalized Combinatorial Map and the cycles for involutions $\alpha_{0...n}$.

by biology". When working with multi-resolution pyramids [Kropatsch, 1995], one may work at the level of granularity only of those parts of the image which are essential [Sonka et al., 2008, p. 106]. Often computer vision has to deal with high amounts of data, so reducing the load to the minimum (resolution) required may help to save resources. From algorithmic point of view this representation fits to region-based segmentation frameworks. There, in the first place, each step may be represented as a level in the pyramid. In the second place the merging of regions from level to level builds up a hierarchical merging tree. [Sonka et al., 2008, p. 98, 109] describe this concept in terms of regular matrix $m$ and tree $t$ pyramids explained in the following. Based on the concept of topological (dual) graphs presented in Section 2.2, [Kropatsch, 1995] defines irregular pyramids.

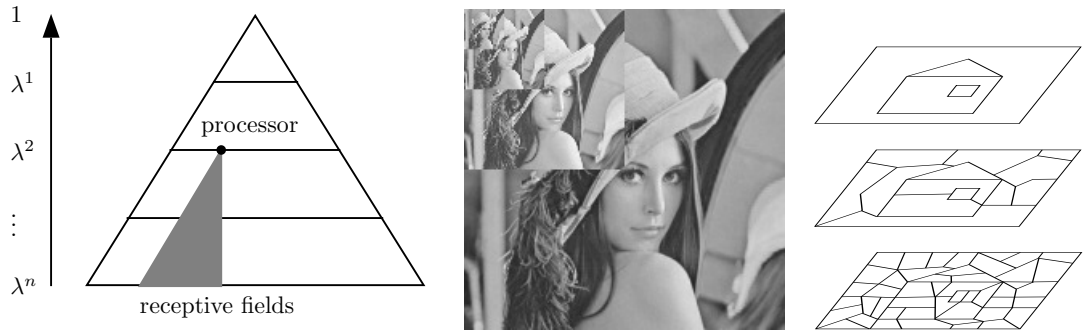
### Regular Pyramids

[Sonka et al., 2008, pp. 99-100] state that "traditional image data structures such as matrices, chains, graphs, ... are important not only for the direct representation of image information". Thus matrices may also be used as a basis for more complex hierarchical methods and data structures.

- *Binary images:* a matrix containing only zeros or ones;
- *Multispectral images:* each matrix contains corresponds to a representation in one spectral band;
- *Multi-resolution pyramids:* matrices containing of different levels of resolutions (pyramidal structure);

Such a matrix pyramid, is defined as follows [Sonka et al., 2008, p. 106]:

**Definition 2.3.1.** *A matrix pyramid is a sequence $\{M_0, \ldots, M_i, M_{i+1}, \ldots, M_k\}$ of images, where $M_0$ has the same dimensions and elements as the original image. $M_{i+1}$ is derived from $M_i$ by reducing the resolution by one half.*




(a) Pyramid Reduction Factor (b) Gaussian Image Pyramid (c) Stack of segmentations

Figure 2.9: Pyramidal Structures

This reduction defines a fixed *reduction window, e.g.* $2 \times 2$ *pixels* (see dotted lines in Figure 2.10) where for every cell $c$ (single element, e.g. region) of $M_{i+1}$, $w(c)$ is its set of children in $M_i$. The *reduction factor* $\lambda$ defines at which rate the image decreases between the levels:

$$\lambda \leq \frac{|M_i|}{|M_{i+1}|}, i = 0, 1, \ldots, k-1 \tag{2.7}$$

If the images are organized in regular manner, i.e. all cells have the same number of neighbors (square grid) and the same number of children, the pyramid is called regular. In a regular pyramid the number of pixels in level $i-1$ is $\lambda$ times higher than in the next level $i$ (Figure 2.9a). This reduction window of a cell $c$ at level $i$ can be propagated down to higher resolution than level $i-1$. $w^l(c_i)$ is the equivalent window [Sonka et al., 2008, p. 109] that covers all cells at level $i-l-1$:

$$w^{l-1}(c_i) = \bigcup_{q \in w(c_i)} w^l(q) \tag{2.8}$$

This defines also the *merging tree* and the *receptive field* (see Figure 2.9a, modified according to [Haxhimusa, 2007, p. 31]). Figure 2.9b shows a self-created Gaussian Pyramid with reduction factors $\lambda = 1, 2, 3$, resulting in a resolution being $1/2, 1/4, 1/16$ of the size of the input image. Regular pyramids are fully characterized by the term (*reduction window*, e.g. $2 \times 2$)/(*reduction factor*).

Regular pyramids have a rigid structure where the intra-level relationships are fixed (only inter-level relationships could adapt itself to the image layout) and the reduction factors are constant [Marfil et al., 2006]. This 'inflexibility' has the advantage that size and layout of the structure are always fixed and well known already in the beginning. In consequence, this data structure has logarithmic tapering and efficient algorithms can be implemented in $O(\log n)$ runtime.

In order to create segmentation either bottom-up and/or top-down approaches can be chosen. What needs to be considered is that even if the same homogeneity criteria are used, merging

is not a dual operation to splitting [Sonka et al., 2008, p. 227] and inevitably leads to different results! In the latter case of splitting, one could start on top of the pyramid (image is represented as one region) and apply splitting operations to produce segments. Approaches following this paradigm use T-Pyramids (tree-pyramids or modifications called quadtrees) as data structure [Sonka et al., 2008, pp. 106-110]. Tree-pyramids correspond to the tree version of M-pyramids.

**Definition 2.3.2.** *Nodes of a T-pyramid correspond for a given $k$ with image points of an M-Pyramid; elements of the set of nodes $P = \{(k, i, j)\}$ correspond with individual matrices in the M-Pyramid (k is called the level of the pyramid). $F$ is the so called parent mapping, which is defined for all nodes $P_k$ of the T-pyramid except its root $(0, 0, 0)$ [Sonka et al., 2008, p. 107].*

Quadtrees are modifications of T-pyramids, where it is not necessary to keep nodes at all levels – which is less expensive. [Braquelaire and Brun, 1998] claim, that above mentioned "hierarchical structures are devoted to top-down region based algorithms and do not provide efficient implementations of merge algorithms". Thus they break these structures and use region adjacency graphs (as presented in Subsection 2.2) which are more adapted to perform merge operations. See Figure 2.10 for an example showing the split-and-merge operations in an hierarchical data structure. [Bister et al., 1990] show, that these regular data structures suffer

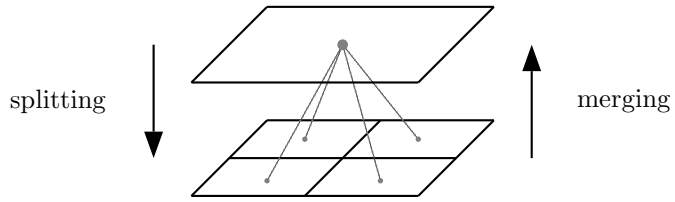



Figure 2.10: Split-and-merge in a hierarchical data structure.

from global shift-, scale- and rotation variance. E.g. moving the grid structure by one pixel leads to a complete different segmentation. Therefore they conclude that regular pyramids have to be rejected as general purpose image segmentation algorithms.

### Irregular Pyramids

Irregular pyramids were introduced to overcome the limitations of global shift-, scale- and rotation variance. Here – analogous to regular M-pyramids – a stack of graphs/combinatorial maps is used for representation. Similarly, [Kropatsch, 1995] distinguishes ordered levels of decreasing sizes (higher abstraction) in an irregular pyramid. This way image partitions at different resolution levels [Simon et al., 2006] may be represented.

Adjacent levels are related by the fact that the vertex set (representing regions) of the reduced level is a subset of the vertices in the level below. This way 'connecting paths' (structures) are created that relate the edges of the reduced graph with paths between surviving vertices in the level below. This variable data-structure and decimation process dynamically adapts to the image layout [Marfil et al., 2006], in consequence they are also called *adaptive pyramids*. In general,

the authors say, there is no fixed reduction factor between the levels nor is the size of each level and the height of the pyramid known.

From any of the boundary representation data structures [Lienhardt, 1991] (topological graphs) presented before in Section 2.2 an irregular pyramid can be built. When representing 2D-images, the simplest one is a RAG-pyramid. Reminding Procedure 1 for region growing, this data structure could be used to easily apply the merge-operations. [Glantz and Kropatsch, 1994] define dual graph contraction to create dual graph pyramids. In further consequence combinatorial pyramids [Brun and Kropatsch, 1999] are used to capture the whole topological information. In general an irregular pyramid is a stack of graphs $(G_0, \ldots, G_k)$. As described in [Kropatsch, 1995], "the methods for building differ in the way they select the survivors and in the way they derive the neighborhood relations of the reduced level". In Procedure 2, [Simon et al., 2006] presents a general approach:

**Procedure 2** Build a new pyramid level

1: mark $(n-1)$ cells between regions to merge homogenous regions ($n$-cells);
2: duplicate last level;
3: apply decimation;

The first step of 'selecting the regions to merge' and some more details on this procedure are given in Section 3. E.g. level 0 corresponds to the initial image/segmentation and each graph $k+1$ is built from the one below ($k$). Especially the third step is more involved than before and will be discussed here in detail. The preceding section on regular pyramids already contained - in general terms - the concept of a reduction window. In order to obtain different resolution levels in irregular pyramids, this concept needs to be adapted to the new structure.

When working with topology and dual representations, the decimation phase is crucial since contraction and removal operations need to be done in a structure preserving way [Glantz and Kropatsch, 1994]. Either one can apply:

- primal edge contraction and removal of its dual *or*
- dual edge contraction and removal of its primal.

[Brun and Kropatsch, 1999] say that "in order to preserve the number of connected components of the initial combinatorial map, bridges and self-loops must be respectively excluded from removal and contraction operations". Hence contraction kernels [Brun and Kropatsch, 2003a] are defined to solve this constraint and to be able to perform several contraction operations simultaneously. The definitions 2.3.3 and 2.3.4 are taken from [Haxhimusa et al., 2005].

**Definition 2.3.3** (Contraction Kernel)**.** *Given a connected combinatorial map $G = (\mathcal{D}, \sigma, \alpha)$, the set $K \subset \mathcal{D}$ will be called a contraction kernel iff it is a forest*[6] *of $G$ not including all darts of $G$ : $SD = D - K \neq \emptyset$. The set $SD$ is called the set of surviving darts.*

[6]Definition from [Diestel, 2010, p. 12]: An *acyclic* graph, one not containing any cycles, is called a *forest*. A connected forest is called a *tree*. Thus a forest is a graph whose components are trees.

**Definition 2.3.4** (Equivalent Contraction Kernel)**.** *Given a combinatorial map $G_0 = (\mathcal{D}, \sigma, \alpha)$, a contraction kernel $K_1$ of $G_0$, the contracted combinatorial map $G_1 = G_0/K_1$, and $K_2$ a contraction kernel of $G_1$, the contraction kernel $K_3$ of $G_0$ for which $G_0/K_3 = (G_0/K_1)/K_2$, is called the equivalent contraction kernel of $K_1$ and $K_2$.*

A successive reduction/decimation of vertices by the set of operations leads to a stack of combinatorial maps, defining a combinatorial pyramid $(G_0, ..., G_k)$. Several decimation schemes may be applied to control the height of the pyramid. This applies also to the process of selecting the edges to be removed. An overview and detailed definition of both aspects is given in Chapter 3. Formally each $0 \dots k$ represents a level $G$ within the pyramid. Each level (map) in $k + 1$ is build from the one below $k$, by selecting a set of contraction kernels $K_{k,k+1}$ and applying it to a given combinatorial map $G_k$ to get the reduced $G_{k+1} = \mathcal{C}[G_k, K_{k,k+1}] = G_k \setminus K_{k,k+1}$ [Haxhimusa and Kropatsch, 2004]. More on removal of the redundant edges can be found in [Brun and Kropatsch, 2003c] and will be discussed in Section 4.2. In contrast to n-dimensional combinatorial maps where removing and contracting operations need to be defined explicitly for each dimension, a n-dimensional generalized map pyramid is based on a general operation [Simon et al., 2006].

## 2.4 Knowledge and Image Segmentation Algorithms

The performance of segmentation algorithms can be increased when additional knowledge about the problem domain is added. As this is a key part of the work presented (see Chapter 4), this section gives a more detailed assignment. Figure 2.11 shows the tradeoff between general purpose and highly specialized methods.

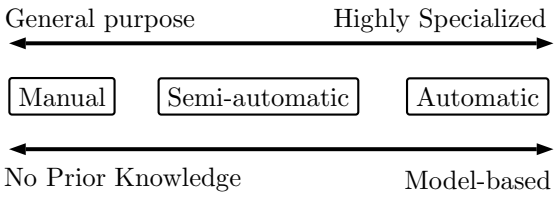



Figure 2.11: Classification of Segmentation Methods, modified according to in [Hug, 2000].

Beginning on the left top end on the scale, the more generic an application is designed, the less prior knowledge is necessary. But domain-independent algorithms therefore require additional sources of knowledge during runtime to produce acceptable results, thus requiring a flexible interface. If we do manual segmentation, probably pixel by pixel, the result for sure is quite accurate and of course lives up to the users' expectations. It is time-consuming and hence not acceptable for repeated usage, but very flexible. One possibility to automate the process is to delegate generic segmentation tasks to an algorithm and retain the user for initialization or optimization steps. In the center of Figure 2.11, parts of the process are calculated automatically. The user is freed from providing input all the time and parts are delegated. If there is noise badly

influencing the automatic segmentation process, additional information is needed to produce acceptable results. For both aspects, flexibility and prior-knowledge, bringing a user or an expert into the loop has advantages. It is not as time-consuming as the fully manual approach and the user still has the possibility to exert influence on the result. In the fully automatic approach the user is no longer in the loop of the segmentation task itself. Prior knowledge in shape of e.g. a training-set is used to create a model of the object in question. It is also possible to extract rules guiding the algorithm. The algorithm produces a result mostly without user input needed, but it is fitted to a specific domain and probably no longer applicable for generic image classes.

As already stated, the acceptance and usage of a partition depends on the application and may be highly subjective. Therefore semi-automatic segmentation yields good acceptance. A similar argument is also given by [Meine et al., 2004], for the use of medical applications: "Having a user in the loop is necessary from both a clinical and a legal point of view". What can be observed, is that the notion of 'semi-automatic' and 'interactive' is ambiguously used. In some publications a simple mouse click to trigger an operation is already considered as interaction as well as manually preprocessing of an image. Restricting or parametrizing the segmentation process also cannot be counted to this class. [Hug, 2000] tries to categorize them and shows various ways of user interaction on (semi-)automatic segmentation algorithms (see Table 2.1).

| Initialization | Optimization | Postprocessing |
|---|---|---|
| manually initialized | manually guided | manually corrected |
| manually restricted | manually influenced | manually modified |

Table 2.1: Interaction Types.

Some of the methods in recent literature are initialized (e.g. statistic shape models, rule sets, training sets), others use seed points or strokes for initializing and restricting a segmentation process [Heimann et al., 2009]. Besides, existing methods can also be categorized either as optimizing (manually guided, influenced) or post-processing methods (manually corrected, modified) [Hug, 2000].

The computer sees the image through a keyhole (local neighborhood) which makes it very difficult to understand more global context [Sonka et al., 2008, p. 5]. In consequence the amount and quality of additional data used has a major impact on both sides, i.e. the design of the segmentation tool and its resulting output. User-interaction gives an input of knowledge about the domain that can exert influence on the algorithm to produce more acceptable segmentation results. The work presented can be put into both categories of the schema presented in Table 2.1. The user is integrated to modify an existing automatic segmentation and also to guide the process of creating segments. This will be discussed in more detail in Chapter 4 but first existing algorithms are reviewed to point out the differences.

## 2.5 State of the Art

Up to now a classification of this work with respect to others would not have been adequate as the theoretical background was missing. Therefore this section now describes alternate approaches while comparing them to the one developed. They do not contradict necessarily, but partially inspire and complement each other.

### Automatic Image Segmentation Algorithms

In general, three algorithms in this category appear to be the most widely used according to [Arbelaez et al., 2009] as sources of automatically created image segments in recent applications:

- Graph based region merging [Felzenszwalb and Huttenlocher, 2004]
- Normalized Cuts [Shi and Malik, 2000]
- Mean shift [Comaniciu and Meer, 2002]

The approach of [Meine et al., 2004] presented in Subsection 2.5 fits into the category of graph-based region merging algorithms as well as the framework used here [Haxhimusa and Kropatsch, 2004]. The others will be covered in Chapter 3. Among others, also watershed [Sonka et al., 2008, p. 233] approaches are used widely. In this case catchment basins of the topographic surface (gray-levels are interpreted as altitude) are created by e.g. synchronous flooding. In the beginning water is placed into each regional minimum (source, lowest gray-level). On increasing the gray-level (flooding), different sources melt together, this is where the boundaries are located. As a result a hierarchy is constructed by this greedy graph-based region merging algorithm. The over-segmentation created this way (super-pixels) is used as base or beginning of other algorithms. The approach presented in [Felzenszwalb and Huttenlocher, 2004] is discussed in detail in Chapter 3 as it is an important part of the framework introduced in this thesis.

### Interactive Image Segmentation Algorithms

In contradiction to the ones before, the approaches in the following are pure interactive:

- Snakes [Kass et al., 1988] or the active contour model are based on a energy-minimizing spline. Its energy depends on the shape and location within the image.
- Live Wire [Mortensen and Barrett, 1998], also called intelligent scissors, is based on dynamic programming and graph search for the path with the lowest cost.
- Active paintbrush [Van Leemput et al., 1998] a method used to paint over region boundaries to initiate region merging.
- Interactive Graph Cuts [Boykov and Jolly, 2001], based on the *minimum* cut principle.

Implementations of the above are available as plugins for recent image manipulation and graphics editing software.

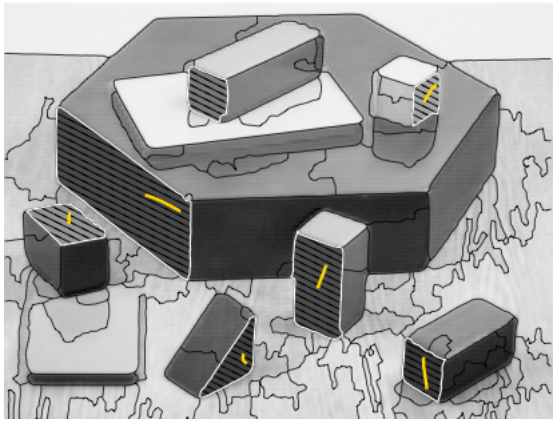


Figure 2.12: Processing critical regions with a few strokes. Image taken from [Meine, 2009a].

### Interactive Image Segmentation in the GeoMap Framework

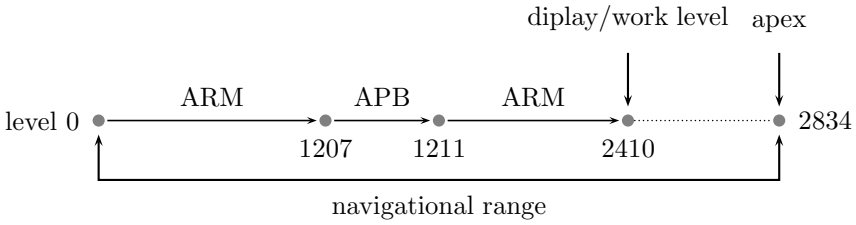
[Meine et al., 2004] states that "no single method achieves the optimal balance for all classes of images". Therefore he presents an interactive image segmentation framework combining several methods, both edge- and region based. In order to do so he was challenged to create a unified data representation fitting the purpose of all approaches and capturing both the geometrical and topological aspect of a segmentation.

The framework developed by [Meine et al., 2004] is called GeoMap and being able to represent also the results of sub-pixel segmentation algorithms. Previous approaches were limited to pixel grid, but – due to the authors – most algorithms already deliver edges with sub-pixel accuracy. This is achieved by inserting crack-edges, in the first place to directly represent boundaries and in the second to derive them in a topologically correct way from label images. Image dimensions are doubled and explicit edges inserted. In order to overcome the staircase affect (a well known problem of crack-edge boundaries) some optimizations are embedded (smoothing) in the visualization part.

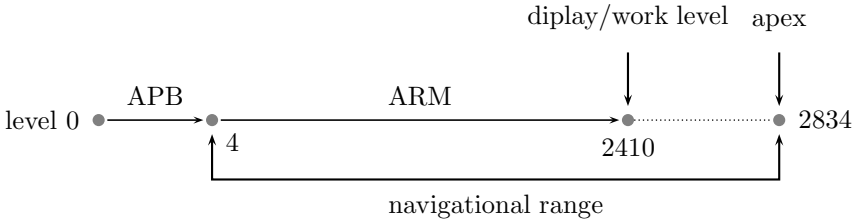
Another motivation behind GeoMaps is the unified representation of topological relations and geometrical features. While the first is required for encoding of adjacency and boundary enumeration, the second is used to derive statistical properties of a region. An internal mechanism called 'hooks' keeps track of changes, updates the components accordingly and guarantees consistency. This sound combination is new, as most frameworks have a separated implementation, i.e. a topological data structure side-by-side with a label image.

Besides automatic region merging, the previously mentioned interactive segmentation algorithms Snakes [Kass et al., 1988], Live Wire [Mortensen and Barrett, 1998] and Active paintbrush [Van Leemput et al., 1998] are implemented on top of the GeoMap framework in order to control the merging process (see Figure 2.12).

Processing of these modifications in the implementation of [Meine et al., 2004] is described in the following example. In the beginning usually an initial over-segmentation (super-pixels) defines the entry point. In Figure 2.13a the steps are aligned sequentially. Starting at level 0, the input regions are first processed by the automatic region merging (ARM) method until level




(a) Naive representation of generated pyramid




(b) Pyramid after reordering to protect manual changes from disappearing

Figure 2.13: Alternating application of automatic and interactive methods, figures modified according to [Meine, 2009b].

1207. Using the active paintbrush (APB) method, the user then performed manual corrections (in 4 levels) on the current image segmentation stack. From level 1211 to 2410 again the automatic region merging process was used for decimation. In this example, the latter level is also the current working level of the generated stack. There any new operation defined may be started. The apex is the uppermost point of the hierarchy, representing the future final segmentation result. Following the order of processing shown in Figure 2.13a, we run into the problem that any manual operations/refinements is lost once the process is started over at lower level. Therefore – as shown in Figure 2.13b – the manual changes induced by the user (APB) are moved to the beginning of the process. Doing it this way these particular levels do no longer appear within the navigational range and remain protected throughout the process.

Concluding, this approach relates to a specific implementation on top of the irregular pyramid paradigm, whereas our work is based on the general concept of irregular pyramids. Moreover, the process starts on pixel-level instead. Nevertheless the PhD-thesis of [Meine, 2009b] mainly inspired the developments presented in this work, we will refer to these concepts later in Chapter 2.4. Finally categorizing it into the schema presented in Table 2.1, class *optimization* suites this approach best.

### Annotated Contraction Kernels for Interactive Image Segmentation

In [Meine, 2009a] the authors present a modified framework (based on [Meine et al., 2004] introduced in the preceeding section) optimized with respect to performance and interactive

operations. In the structural representation, the merging tree is now annotated and pruned. First a label showing the level where the merge happened is added to the merged node. This way the partitioning in a particular level $l$ can be retrieved (due to the authors) easily by pruning all branches below nodes with a $level \leq l$.

Furthermore [Meine, 2009a] state that "in practice, it is unneeded to represent all levels at the same time". Using only the bottom layer (GeoMap) and the concept of contraction kernels [Brun and Kropatsch, 2003a], they can recreate every level and thus reduce memory consumption. Finally also 'face inhibition' is introduced in order to protect resulting segmentation throughout the hierarchy. The authors also argue that logarithmic tapering with a fixed reduction factor is irrelevant in their context. This way a 'continuous pyramid' is created in which only one region per step is merged. No new boundaries are introduced. The process relies on an initial over-segmentation where important boundaries already need to be present [Meine, 2009a]. The 'face inhibition' operation appears to be similar to our approach, however, the overall concept is different.

### Interactive image segmentation with integrated use of the markers and the hierarchical watershed approaches

In general, watersheds and markers belong to the category of morphological particle methods [Sonka et al., 2008, p. 687]. First markers (used to indicate objects, not boundaries) are located, in the second step the objects can be grown from the markers using e.g. watersheds. This concept is well known in topography where watershed lines divide individual catchment basins as already described in the beginning.

Several optimizations and extensions exist to the first efficient watershed approach of [Vincent and Soille, 1991]. [Klava et al., 2009] combine in their work hierarchical watershed and watershed from markers. In their implementation it is possible to switch back and forth between the two taking advantage of the individual strengths. This approach is also considered as interactive, markers can be set (using a welding brush) and two local operations (merge and refine) can be used on the hierarchy. At the base they use a region adjacency graph for

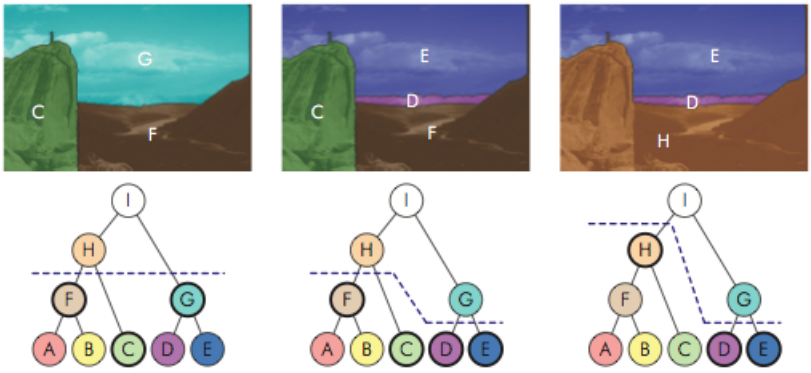



Figure 2.14: Obtaining partitions, subfigures taken from [Klava et al., 2009].

representing the topological relationships. Figure 2.14 shows different segmentations created by operations on the merging tree.

On testing the system developed by [Klava et al., 2009], it shows that this approach strongly benefits from the two approaches supplementing each other. In the beginning e.g. an initial over-segmentation can be performed using the watershed from markers approach. Afterwards a region can be selected for hierarchical segmentation. Within this region one can apply split and merge operations following certain 'hierarchical criterion'. Only one depth, area or volume level up/down in the hierarchy can be addressed by each modification-operation.

Afterwards, for further refinement, again the markers' approach may be used. As it is relevant for the discussion on the approach developed, only within the markers' approach pixel accuracy can be used. This is not possible within the hierarchical watershed method, but one can merge regions efficiently using a welding brush. The idea of 'grabbing' a certain region within the hierarchy (see Figure 2.14) was adopted also in this work. Following the schema in Table 2.1, this approach is located in class *initialization* as well as *optimization*.

### Automatic Image Segmentation by Positioning a Seed

The work of [Mičušić and Hanbury, 2006] is about optimization of a seed-based segmentation technique, i.e. the interactive graph cut method [Boykov and Jolly, 2001] which takes color and texture into account. "The novelty is in introducing the positioning of the seed, and collecting and merging similar segments yielding the segmentation pyramid..." say the authors. In this approach the interactive part usually performed by a user is automatized. However, the amount of seeds (input) required does not seem to be economic and impossible to achieve for human operators. Categorizing it into the schema presented in Table 2.1, class *initialization* fits the application best.

### Semi-automatic image segmentation and annotation

The work of [Gelasca et al., 2007] is based on the concept of perceptual grouping introduced by [Felzenszwalb and Huttenlocher, 2004]. It is possible to refine regions within the segmentation hierarchy or simply create custom segments (on pixel level) by drawing a polygon-line around the object. Regions may be selected for further attention and to initiate merging. Resulting segmentation borders are visualized but may be smoothened afterwards (using median-filter) to get a non blurred result. It is possible to traverse the segmentation hierarchy up and down (for the entire image or a specific region). Finally one can put a label (defined in a custom catalog) on the region specified. The original approach and the segmentation criteria introduced by [Felzenszwalb and Huttenlocher, 2004] are discussed in more detail in Chapter 3.

### Labelme

LabelMe [Torralba et al., 2010] is a web-based image annotation tool[7] relying on the collaborative effort. Researchers are able to segment images and contained objects by clicking

[7] Contribute and visit `http://labelme.csail.mit.edu/`



Figure 2.15: Freehand segmentation and annotation of objects. Image taken from a LabelMe public directory.

along their boundary (drawing a polygon-line, see an example in Figure 2.15). Finally the object can be annotated with vocabulary (free text).

The developers set up simple labeling guidelines but apart from that there are no limitations. Aiming to create a database containing annotated images spanning the visual world, the output and statistics collected may be useful for training and evaluation of computer vision systems among others. Therefore they provide also a Matlab interface as a basis for custom developments.

This approach is entirely interactive without automated assistance and located – with respect to the schema presented in Table 2.1 – in class 'initialization'. From segmentation point of view this framework requires high amount of manual input. Furthermore it is not possible to grasp relationships between the objects (which are hierarchical by nature). Each level of abstraction has to be created individually. However, [Torralba et al., 2010] mainly set focus on the annotation part therefore improvements in the segmentation part (using a hierarchical data structure or optimized input methods) are not considered.

### Annotation tool

The annotation tool presented by [Korč and Schneider, 2007] is based on the Labelme [Torralba et al., 2010] approach. In comparison to before, the authors try to improve usability and add missing functionality in the user interface lacking in the original implementation. The segmentation method remains the same as in [Torralba et al., 2010] where the user indicates each boundary individually by placing corner points forming a polygon line. Zooming functionality is added to increase accuracy. Based on the Matlab framework provided by [Torralba et al., 2010] it is now also possible to limit the input to a certain set of vocabulary for object classes. On annotating objects the user now can specify its uncertainty, which is an estimate how precise the labeling is and values for occlusion and representativeness. Although [Korč and Schneider, 2007] use an hierarchical data-structure XML to store the results, there is no tree structure implemented which allows to define hierarchical relations between objects.

CHAPTER 3

# Automatic Hierarchical Image Segmentation

The basic theory of image segmentation and irregular pyramids was covered in the preceding chapter, while this chapter covers the combination of the different concepts. As already stated in the very beginning, the diploma thesis is based on the hierarchical image segmentation framework proposed by [Haxhimusa and Kropatsch, 2004, Haxhimusa et al., 2005, Haxhimusa, 2007]. The authors use extended RAG, dual graphs and combinatorial maps to encode topological relations properly – respectively RAG, dual graph and combinatorial map pyramids are used to represent the stack of segmentations. The process of merging of regions is based on the concept of perceptual grouping [Felzenszwalb and Huttenlocher, 2004] and the minimum spanning tree (MST), which in combination captures many perceptually important aspects of complex imagery.

## 3.1 Minimum Spanning Tree based Method

[Haxhimusa, 2007, p. 101] builds a minimum weight spanning tree of an image "in order to find regions borders quickly in a bottom-up 'stimulus driven' way". The selection of candidate edges is based on local differences in the feature data. In general, *perceptual grouping* is "about extracting the global impressions of a scene", therefore [Shi and Malik, 2000] state that most segmentation criteria based on local properties of the graph are not appropriate. Greedy decisions (like smallest weight edges) in turn produce segmentations that satisfy global properties [Felzenszwalb and Huttenlocher, 2004]. [Zahn, 1971] states that a minimum spanning tree reveals the hierarchical structure of clustering. This strong cut property is also supported by [Zahn, 1971].

The stack of segmentations represented by irregular graph pyramids is created by the BorůSeg framework (see outline in Procedure 3). The merging processes take low-level cues, e.g. images only having RGB or gray color values. In each decimation level in the pyramid,

a minimum spanning forest is built by Borůvka's method [Nesetril et al., 2001] (translation of original papers). In this approach the edges to be contracted are selected by a merging criteria based on perceptual grouping (capturing feature similarity). In the decimation phase contraction kernels assert the topologically correct creation of the new level. A general outline is shown in Procedure 3, the individual concepts are summarized afterwards. The lowest level $k = 0$ corresponds to the input image where each pixel maps to a region and each region is represented by a node in the (primal) region graph. Graph edges encode the adjacency relations which are built up in a preceding initialization step. Afterwards they are processed as follows:

**Procedure 3** Construct Hierarchy of Partitions (Boruseg)

1: k = 0;
2: **repeat**
3: **for all** vertices $v_i \in G$ **do**
4: $E_{\min} = argmin\{attr_e | e = (v_i, v_j) \in E_k \vee e = (v_j, v_i) \in E_k\}$
5: $E_{\min} = E_{\min} \wedge E_{\min}(v_i)$
6: **end for**
7: **for all** edges $e = (v_{k,i}, v_{k,j}) \in E_{\min}$ with $Dif(C_1, C_2) < MInt(C_1, C_2)$ **do**
8: include edge $e$ in contraction edges $N_{k,k+1}$
9: **end for**
10: contract graph $G_k$ with contraction kernels, $N_{k,k+1} : G_{k+1} \leftarrow C[G_k, N_{k,k+1}]$
11: **for all** $e_{k+1} \in G_{k+1}$ **do**
12: set edge attributes $attr_e(e_{k+1} = \min\{attr_e(e_k) | e_{k+1} = C[e_k, N_{k,k+1}]\}$
13: **end for**
14: k = k + 1;
15: **until** $G_k = G_{k+1}$

Procedure 3 describes the key steps of the framework proposed[1]. At the bottom of the pyramid $G_0$, the regions correspond to image pixels (compare to Figure 2.4c). The following list captures the essential parts of the algorithm:

1. **Initialization:** Lines 3–6 build up the MST.
2. **Selection:** In lines 7–9 we select the edges to be contracted.
3. **Decimation:** Line 10–13 the graph is decimated and post-processing steps are initiated.

The end is reached when there are no more edges available for contraction, indicated by condition $G_k = G_{k+1}$. We explain these three tasks in the following subsections. On segmenting an image into regions using graph based representations, a "predicate for measuring the evidence for a boundary between two regions" is necessary [Felzenszwalb and Huttenlocher, 2004]. Its calculation is based on attributes assigned to the elements of the graph. Following [Haxhimusa et al., 2005], a corresponding representation in terms of a combinatorial map representation is defined as graph:

$$G = (\boldsymbol{\mathcal{D}}, \sigma, \alpha, \text{attr}_v, \text{attr}_e)$$

[1] Modified according to Procedure 13 in [Haxhimusa, 2007, p. 117].

In order to store image feature data on the graph elements, attributes are defined on vertices $u \in V$ and edges $e \in E$ as follows:

$$\text{attr}_v : V \to \mathbb{R}^+$$
$$\text{attr}_e : V \to \mathbb{R}^+$$

Vertices have feature values (like pixel values in RGB color space) assigned, the edge attributes store differences between end point vertices. A feature vector $F(u_i)$ representing feature values of vertices $u_i \in V$ encodes the information [Shi and Malik, 2000]:

$F(u_i) = I(u_i)$: intensity value, for segmenting brightness images
$F(u_i) = [v, v * s * \sin h, v * s * \cos h](u_i)$: HSV values, for color segmentation

Three dimensional RGB values are implemented in $\mathbb{R}^+$ using a programming trick. The data type Integer, which has length $32\,\text{bit} = 4 * 8\,\text{bit}$ or $4 * 1\,\text{B}$, is used to encapsulate the three color channels having $1\,\text{B}$ each. The same trick is used to extract the RGB color information again. Similar approach can be found in HTML where a RGB color is encoded using hexadecimal digits. In order to calculate edge attributes $\text{attr}_e$ (representing differences), [Haxhimusa et al., 2005] use the Euclidean distance between RGB pixel values:

$$\text{attr}_e(u_i, u_j) = \sqrt{\text{R}(u_i) * \text{R}(u_j) + \text{G}(u_i) * \text{G}(u_j) + \text{B}(u_i) * \text{B}(u_j)}.$$

This weight $\text{attr}_e$ is stored on the edge between two regions. It is some kind of dissimilarity (contrast) measure between two pixels connected by that edge. In this implementation undirected graph edges are used. In case of combinatorial maps, the attribute is not stored on the dart itself but on the mapped edge (involution $\alpha$, see Section 2.2). In case of directed edges, which may be necessary because of special application requirements, they must not necessarily have the same weights.

## Perceptual Grouping

As already indicated in the beginning of this chapter, a pairwise region comparison predicate calculus [Felzenszwalb and Huttenlocher, 2004] based on such feature data is needed, indicating the presence of a boundary in the segmentation process. [Wertheimer, 1923] asks, "do such arrangements and divisions follow definite principles?". He claims that a number of stimuli are not experienced as a number of things, but instead "larger wholes separated from and related to one another are given". In his paper he examines the problem and lists the factors of proximity, similarity and continuation which form the appearing stimuli to the experienced arrangement following "Gestalt Theory". [Ren and Malik, 2003] summarize the principle of similarity whilst describing the concept of perceptual grouping as follows:

1. *Intra-region similarity:* the elements in a region are similar.
2. *Inter-region (dis)similarity:* the elements in different regions are dissimilar.

These assumptions were formalized by [Felzenszwalb and Huttenlocher, 2004], leading to the predicate calculus they present in the framework developed. The internal difference of a component $C \subseteq V$ is expressed by Equation 3.1.

$$Int(C) = \max_{e \in MST(C,E)} w(e) \tag{3.1}$$

The *internal difference* of a component $C$ is, by definition of [Felzenszwalb and Huttenlocher, 2004], calculated by using the maximum weight edge of its minimum spanning tree. According to [Diestel, 2010, p. 141], a topological spanning tree of a graph $G$ is an arc-connected standard subspace $T$ of $G$ that contains every vertex but contains no cycle. To explain the motivation behind MST, we need to go back to earliest attempts in the field of cluster detection. [Zahn, 1971] states, that the main advantage of the MST is its "close conformity to the 'proximity' principle of perceptual organization ... by Wertheimer". Furthermore [Zahn, 1971] draws an analogy between organization effected by MST (minimum principle) and perceptual mechanisms, but it also reveals the hierarchical structure of clustering.

To build the MST, one could implement the method proposed by Borůvka, Kruskal or Prim. A review of its specific properties is presented in [Haxhimusa, 2007, pp. 106 – 108]. In [Felzenszwalb and Huttenlocher, 2004], Kruskal's algorithm is used. Whereas in this work, allowing parallel implementation and overall algorithmic complexity of $O(|E| \log |V|)$, Borůvka's work has been chosen.

The *difference between* the components is determined by the definition given in equation 3.2

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j \in E)} w((v_i, v_j)) \tag{3.2}$$

The decision whether two adjacent regions (i.e. vertices connected by edges) are merged is guided by comparison of region similarity [Felzenszwalb and Huttenlocher, 2004]. To determine the evidence of a boundary the following predicate $D$ is defined:

$$D(C_1, C_2) = \begin{cases} true & \text{if } Dif(C_1, C_2) > MInt(C_1, C_2) \\ false & \text{otherwise} \end{cases}$$

where the minimum internal difference $MInt$ is:

$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2)) \tag{3.3}$$

$$\tau(C) = k/|C| \tag{3.4}$$

The threshold function, $\tau$ in Equation 3.3 for calculating the minimal internal differences, contains a scale factor. It is based on size of the component [Felzenszwalb and Huttenlocher, 2004]. This way the relative difference in comparison to the differences between the components is controlled. Following the author's statement, "for small components we require stronger evidence for a boundary". In consequence a larger $k$ causes a preference for larger components and vice versa. The general observation is that edges between two vertices in the same component should have relatively low weights and edges between vertices in different components should have higher weights [Felzenszwalb and Huttenlocher, 2004].

## Decimation

As already described in the basic's chapter, an irregular pyramid is a adaptive data structure which dynamically adapts to the image layout. Of course it would be desirable to retain the advantages of its regular counterpart also for them. In contrast to regular pyramids, there is neither a fixed reduction factor between the levels nor a fixed relationships. Therefore the size of each level and the height of the structure are unknown. [Marfil et al., 2006] summarize the problem:

> "Although original irregular pyramids overcome the drawbacks of regular ones, their main drawback is that they only grow to a reasonable height as long as the base level is small."

[Meer, 1989] presents a pure stochastic scheme that allowed no two neighbored vertices to be both survivors and required at least one survivor and in the neighborhood of each non survivor. These two constraints define a maximum independent set (MIS).

In order to retain logarithmic tapering for dual graph pyramids too, [Haxhimusa et al., 2002] implements a MIS of edges called MIES (maximal independent edge set). If the direction of contraction matters, a MIS of directed edges can be applied. This approach is called MIDES (maximal independent directed edge set) [Haxhimusa et al., 2003] and is of interest as the direction of edges determines the roots of the contraction kernels. All the vertices which are sources of directed edges are marked there as non survivors [Haxhimusa, 2007, p. 75]. Using the new decimation concepts, "the construction of irregular pyramids bounds logarithmically the height of the pyramid". Alternate decimation approaches (according to [Marfil et al., 2006]) not discussed at this time are:

- Data driven decimation scheme [Jolion, 2003]
- Specified rate and prioritized sampling approaches [Ip and Lam, 1996]
- Dual-graph contraction [Glantz and Kropatsch, 1994] and connectivity preserving relinking approach [Nacken, 1995]
- Union-find techniques [Tarjan, 1975]

## Implementation

The status output of the framework (see Listing 3.1) shows detailed information about the merging process, each level is identified by an increasing number and performance data. After initializing the edge weights at level 0, each iteration of Borůvka's MST algorithm (called *Borůvka's* phase, indicated with an asterisk '*') followed by relevance filtering of candidate edges and its contraction creates the next level in the stack of segmentations. In between [Haxhimusa and Kropatsch, 2004] add additional levels to have a simpler contraction kernel. In this case the edges to be removed (selected in *) are iteratively decimated using MIES decimation method. This way the irregular graph pyramid grows higher until no longer edges are selected for removal (see Figure 4.1 for reference). Finally the segmentation (*.ppm) and label images (*.bsi) are written to file.

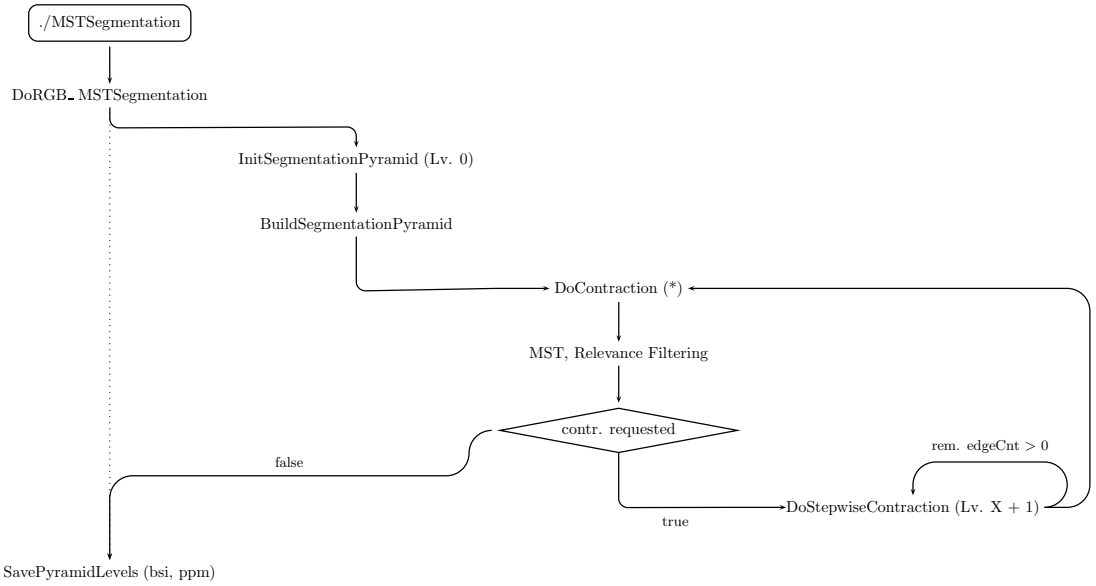



Figure 3.1: Flow chart of the automatic MST segmentation framework showing the most important processing steps. From the input image a stack of segmentations is generated.

```
Input image: 118035.ppm, 481 x 321
K: 300
*
 Level 1:  MIES 0.26s, Kernel 31.09s, Total 31.7s , Faces left 72244
 Level 2:  MIES 0.12s, Kernel 6.05s, Total 6.5s , Faces left 47793
 Level 3:  MIES 0.07s, Kernel 1.17s, Total 1.53s , Faces left 43976
 Level 4:  MIES 0.05s, Kernel 0.4s, Total 0.73s , Faces left 43671
 Level 5:  MIES 0.06s, Kernel 0.32s, Total 0.64s , Faces left 43652
 Level 6:  MIES 0.05s, Kernel 0.31s, Total 0.62s , Faces left 43650
*
 Level 7:  MIES 0.11s, Kernel 3.86s, Total 4.23s , Faces left 19390
 ...
*
 Level 13:  MIES 0.02s, Kernel 0.24s, Total 0.39s , Faces left 4355
 ...
*
 Level 31:  MIES 0s, Kernel 0s, Total 0.14s , Faces left 47
 Level 32:  MIES 0s, Kernel 0s, Total 0.14s , Faces left 35
 Level 33:  MIES 0s, Kernel 0s, Total 0.14s , Faces left 33
*
 Level 34:  MIES 0s, Kernel 0s, Total 0.14s , Faces left 23
 Level 35:  MIES 0s, Kernel 0s, Total 0.14s , Faces left 16
 Level 36:  MIES 0s, Kernel 0s, Total 0.13s , Faces left 14
*
 Level 37:  MIES 0s, Kernel 0s, Total 0.15s , Faces left 12
 Level 38:  MIES 0s, Kernel 0s, Total 0.14s , Faces left 11
*
Total time:  52.32s
```

Listing 3.1: Shortened runtime information of the automatic segmentation tool presented in Chapter 3. Console output shows the decimation parameters and contraction statistics.

### Other Merging Criteria

The image features to be processed take in a key role as the resulting segmentations are "affected by the order in which the regions are merged" [Zankl, 2010]. This order is influenced by the similarity of the regions, the higher the similarity, the sooner edges are selected for contraction. Based on the framework proposed by [Haxhimusa et al., 2005], [Zankl, 2010] implements higher level features in order to improve the process of evaluating the similarity of adjacent regions. He discerns among pixel-, border- and region features and considers also structural patterns, difference in intensity/color/motion or some other local attribute. The geometrical data needed is derived by [Zankl, 2010] implicitly using the hierarchical information:

- area (size of the receptive field, i.e. the number of nodes in level 0 which merge to one node in level n)
- perimeter (number of edges in level 0 corresponding to one of the edges of one node in level n)
- length of the border between 2 regions (number of edges in level 0 corresponding to one edge in level n)

In addition to the contrast measure defined before, also the results of the new feature set are encoded in the edge weight.

Another aim of the work presented by [Zankl, 2010] is to reduce the memory consumption of the pyramid. This is achieved on inventing the *Flat Graph Pyramid*, i.e. an aggregation of multiple sequent levels. Arrays of index changes (in merging tree) are combined together based on the concept of the equivalent contraction kernel. In the example shown in Listing 3.2, all levels from 1 to 10 (range determined individually) are compressed into a single one. What can be seen as well is that most regions are merged within these levels. The merging process then continues with level 2, having only 8129 regions instead of 64616. This procedure saves resources especially in lower levels of the pyramid, however it causes all merging information in between to be lost. Only the overall contraction result is passed on. The implementation in this diploma thesis contains both improvements of [Zankl, 2010] as they all share the same code-base.

```
Level 1: Flat Pyramid
-----------------------------------------------------------
*
 Flat Level 1:  MIES 0.16s, Kernel 1.16s, Total 1.33s , Faces left 79376
 Flat Level 2:  MIES 0.03s, Kernel 0.43s, Total 0.47s , Faces left 64616
 Flat Level 3:  MIES 0s, Kernel 0.29s, Total 0.3s , Faces left 63698
 Flat Level 4:  MIES 0s, Kernel 0.29s, Total 0.3s , Faces left 63673
*
 Flat Level 5:  MIES 0.08s, Kernel 0.36s, Total 0.44s , Faces left 28524
 Flat Level 6:  MIES 0.02s, Kernel 0.11s, Total 0.14s , Faces left 19533
 Flat Level 7:  MIES 0s, Kernel 0.06s, Total 0.06s , Faces left 18249
 Flat Level 8:  MIES 0s, Kernel 0.04s, Total 0.04s , Faces left 18129
 Flat Level 9:  MIES 0s, Kernel 0.04s, Total 0.04s , Faces left 18117
 Flat Level 10:  MIES 0s, Kernel 0.04s, Total 0.04s , Faces left 18115
-----------------------------------------------------------
*
Level 2:  MIES 0.02s, Kernel 0.07s, Total 0.17s , Faces left 8129
```

Listing 3.2: Flat Pyramid Concept, the merging information of the first 10 levels is condensed to a single level. The process then continues with level 2.

### Example segmentations

The segmentation images in Figures 3.2 on the next page are intermediate levels chosen from a stack of segmentations. The input images (a), (b) and (c) representing level 0 are taken from [Martin et al., 2001]. As the pyramid grows higher, the regions iteratively merge together. In the Figures shown, the label "level (# number of segments)" indicates the current height and the remaining amount of regions. The segments in Subfigures (d), (e) and (f) are fine-grained, the term over-segmentation describes the result best. The Subfigures (g), (h) and (i) show – subjectively – good segmentations, capturing perceptually important regions. Although some smaller details already disappear due to their thin structure and a lack of contrast causes regions to be merged early with the background, the automatic framework delivers promising results. Subfigures (j), (k) and(l) show higher levels of the pyramid, important regions are under-segmented as they merge with the background. On producing the stack of segmentations, no special parametrization was chosen. The default parameters chosen are $k = 300$, *MIES decimation method* and the *internal/external contrast measure*. On testing several parameter combinations and edge selection criteria it is possible to optimize results of the automatic framework presented.

## 3.2 Graph based Segmentation Algorithm Alternatives

So far, the following concepts for the purpose of creating segmentation graph hierarchies [Haxhimusa et al., 2005] were presented:

- *segmentation criteria* based on the MST concept [Zahn, 1971] like perceptual grouping [Felzenszwalb and Huttenlocher, 2004]
- *decimation methods:* dual graph contraction [Haxhimusa and Kropatsch, 2004] and the MIES/MIDES approach [Haxhimusa et al., 2003]

Among others, [Marfil et al., 2006] list alternate approaches based on irregular pyramid representation:

- Segmentation with a hierarchy of region adjacency graphs (RAG) and the adaptive pyramid [Montanvert et al., 1991]
- Image segmentation by connectivity preserving relinking [Burt et al., 1981]
- Segmentation based on combinatorial pyramids and union-find algorithm [Brun and Kropatsch, 2003b]

What can be observed is that each of the latter approaches is related to a specific decimation scheme. There are various hierarchical segmentation frameworks which use an e.g. RAG at its base. In Chapter 2 we already referred to watershed-based segmentation methods. Further – well known – graph-based segmentation frameworks like the minimum cut [Boykov and Jolly, 2001] or normalized cut [Shi and Malik, 2000] approach are not considered due to their different concept. Referring to [Sonka et al., 2008], "the pyramid approach is quite general and leads itself to many developments…" [Bister et al., 1990, Meer, 1989].



(a) Input image ID 41004 (b) Input image ID 118035 (c) Input image ID 12003



(d) 23 (#244) (e) 27 (#179) (f) 26 (#579)



(g) 35 (#15) (h) 33 (#33) (i) 32 (#73)



(j) 36 (#10) (k) 34 (#23) (l) 33 (#50)

Figure 3.2: Example segmentations (a)(b)(c) images without preprocessing (d)(e)(f) over-segmentation, high amount of details, fine granularity (g)(h)(i) subjectively good segmentations, smaller details already disappear (j)(k)(l) under-segmentation, regions merge with background.

CHAPTER 4

# Interactive Hierarchical Image Segmentation

This chapter first establishes a connection to the automatic hierarchical segmentation approach already presented and points out the design decisions made for the interactive extension. Then we define the interactive modification operations on the stack of segmentations. In order to create a custom segmentation meeting the users' expectations, additional processing steps – embedded in the hierarchical structure – are required. Besides the theoretical aspects covered, a part puts focus on the implementation details as well. In the end, examples showing the effect of the operations defined are provided.

## 4.1 Interface to Automatic Approach

As already mentioned in the introduction, the interactive image segmentation approach developed consists of two parts.

1. Building the irregular pyramid based on the MST [Haxhimusa and Kropatsch, 2004, Haxhimusa et al., 2005] which is already described in Chapter 3, and the

2. Interactive extension implemented on top of the framework where the user places its modification operations.

After a study on the state-of-the-art in Section 2.5 and having also the task of image annotation in mind, we sketched the cornerstones of the contribution:

- Integration of user-interaction into an existing automatic hierarchical image-segmentation framework

- Definition of *merging* and *inhibition of merging* operations on regions to guide the segmentation process

- Working at different granularity levels within the hierarchy
- Using of pixel-level precision versus super-pixel approach
- Consistency of operations within the hierarchy
- Ability to produce any segmentation result

In general, merging and inhibition from merging operations (defined later in Section 4.2) placed by a user exert influence on the segmentation process and modify the merging tree. The basic way of doing this is traversing through all the segmentation levels of the pyramid in strictly sequential and ascending order. This was considered to be a drawback. Therefore the hierarchical paradigm takes in a special role as the user is enabled to select regions from different granularity levels for the final segmentation. It is possible to set focus on arbitrary detail(s) within the stack of segmentations, not limited to a certain resolution level. This contributes also to the statement of [Sonka et al., 2008, p. 107] that "often it is advantageous to use several resolutions simultaneously rather than choose just one image". In comparison to the approaches already presented in Section 2.5 the user is now no longer forced to define a certain working level or follow a strictly sequential process. This method chosen is supported by the fact that also in the process of creating annotations one may choose many parts of interest in one step – regardless of the level of detail. Not limited to this, also the hierarchy is kept (merging tree) to allow a decomposition of object into its subparts or restart the process for further refinement.

There are several points where the approaches presented so far differ. In contrast to the approach presented by [Meine et al., 2004] this framework delivers as well sub-pixel precision but lacking the explicit edge representation using crack-edges. Furthermore there is no combined model of structural and geometrical data. As already covered in Section 3.1 in the features extension created by [Zankl, 2010] this information is derived implicitly and contributes to the edge weight. In consequence we only have the region information and the information contained in the input image. Hence an effective implementation of interactive tools like 'Intelligent Scissors' is not feasible, although one could think of related optimized and more efficient input methods than pixel or region-wise selection.

In the second place, the methods presented in Section 2.5 immediately propagate the modifications done in the graphical user interface to the structural representation. For the reasons already described in Chapter 1 and the conditions induced by the underlying framework, a different approach has been chosen. We collect several modification operations and finally pass them as argument to the segmentation framework. There the irregular pyramid is recalculated and the user-defined operations guide the image segmentation process. As the overall process of the interactive segmentation tool is crucial for understanding, it is visualized in Figure 4.1.

The interactive extension uses the output provided by the modified automatic segmentation framework, i.e. the merging tree and region-based segmentation images with their labeling information. This information is needed for visualization in the GUI and to record the modification operations. The region-based operations are again provided as input to the segmentation framework. In order to handle the input from the human, a 'translation' from regions to edges needs to be performed in a separate preprocessing phase. The operations placed in the user interface implicitly describe the desired effect in terms of affected regions,

a representation in terms of combinatorial maps explicitly addresses the inter-pixel boundaries between the regions which have to be modified.

Consequently we need to define processing methods in the segmentation framework dedicated to the interactive operations which is described in Section 4.3. On trying to categorize this approach we refer again to Table 2.1. It is located between optimization and post processing, the reason for this fuzziness is determined by the different capabilities:

1. Post processing: the framework presented uses the operations i.e. user input strictly, so the result of the segmentation-process may be directly corrected or modified.

2. Optimization: the collected external knowledge possibly affects the algorithms' decisions. It is not a live approach, nevertheless the framework is manually guided and influenced.

## 4.2 Definition of Operations

First of all two basic operations – merging and the inhibition of merging – of adjacent regions are defined. As already discussed in the sections on graph theory and perceptual grouping, the feature data may be stored directly as an attribute/weight on the graph edge. In the automatic procedure this information is used to determine the edges to be contracted. What remains to clarify, how do we encode the decisions taken by the interactive method? Is it possible to combine both?

On reviewing the possible implementations for the related approaches, the merging decisions may either be *soft* or *hard*. In the first option the edge weight is modified. There the interactive operation has a real-valued representative, i.e. the edge gets a normalized/cumulative weight in addition to the feature data is assigned. But this approach leaves some questions unanswered. How to weight the additional knowledge in relation to the feature data? Shall the interactive decision have a fixed value or set within a variable probability range? In case of edge removal, setting the edge weight to a value close to zero will cause a higher probability so that the incident

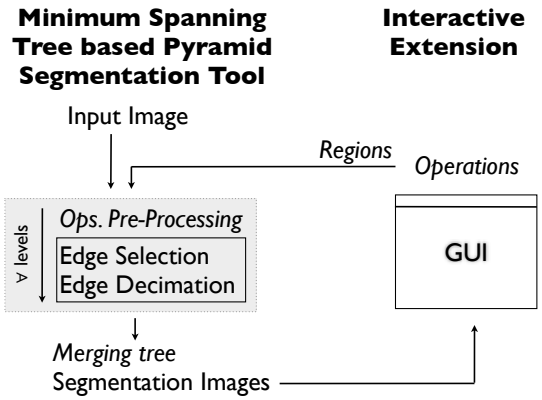
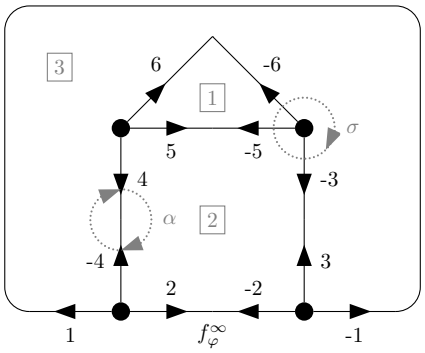



Figure 4.1: Data flow between the program components. The interactive extension provides the input required for the decimation steps performed in the MST-based pyramid segmentation tool.

regions are merged. But how about the inverse operation 'inhibition'? Shall the probability of the operation change as the segmentation pyramid grows higher? However the segmentation result would be unpredictable and the merging criteria computed by a black box. There is no guarantee that a contraction will happen in certain level (if ever). [Meine et al., 2004] had similar problems, but they handle this complex task by introducing weighting decisions and priorities for operations to the frame work. Especially overlapping operations required different protection flags and importance rankings. Due to the problems described and as the scope of this work is also creating image annotation (hierarchies), this implementation is based on hard merging or inhibition operations decisions. If an edge between two regions shall be removed as per the input of the user, then it will be removed. The same applies for the dual operation of inhibition of regions from merging, strictly following the users input. Nevertheless the interactive operations are not restricting the flexibility of the algorithm in merging (other) non selected (or not in the focus) regions automatically. In the previous chapters we explained the foundations, now the approach will be discussed from technical point of view.

## Basic Operations

In order to identify the edges for processing, first all darts belonging to and identifying a region are collected. Within the segmentation framework, more correctly in the hierarchical/topological data structure, the correspondence is encoded and can be retrieved easily. Figure 4.2 shows the affected combinatorial map elements of the regions in level $i$.




$\boldsymbol{\mathcal{D}} = \{1, -1, 2, -2, 3, -3, 4, -4, 5, -5\}$
$\alpha = \{(1, -1), (2, -2), (3, -3), (4, -4), (5, -5), (6, -6)\}$
$\sigma = \{(1, -4, 2), (4, 6, 5), (-3, -5, -6), (-1, -2, 3)\}$
Darts for $R_{i,1} = \varphi(5, -6)$
Darts for $R_{i,2} = \varphi(3, -5, 4, 2)$
Darts for $R_{i,3} = \varphi(-1, -4, 6, -3)$
Darts for $f_\varphi^\infty = \varphi(-2, 1)$

Figure 4.2: Combinatorial map at Level $R_i$ and its functions $\alpha$, $\sigma$ and $\varphi$ on $\boldsymbol{\mathcal{D}}$.

In Chapter 2 we already explained the fundamental iterations $\alpha$ and $\sigma$ which are also indicated in Figure 4.2. But how do we get the correspondence between e.g region $\boxed{1}$ and its boundary graph? Technically – at the beginning of this process – a dart that forms the vertex in the regions' graph is returned, this dart can then be iterated using $\sigma$. For easy access of this functionality during programming, the framework contains a method $GetDartForFace(faceIdx)$ which returns a dart belonging to the face index given (e.g. $R_{i,1}$: 5), similarly $GetFaceForDart(dartIdx)$ is defined which returns the index of a face for a given dart. This mapping is essential for explanation of the concepts presented in the following.

In order to modify the relations between two adjacent regions by only giving their corresponding indices, we need to get the inter-pixel boundary in the primal or the edge representing the adjacency in its dual. Using permutation $\varphi$ (corresponds to $\sigma \circ \alpha$) all darts belonging to a region can be traversed. Concluding, the orbit $\varphi^*$ describes the inter-pixel boundary of a region [Brun et al., 2004]. This schema is explained in more detail in Procedure 4.

**Procedure 4** Get all darts belonging to a specific region

*Input:* dart, *Output:* vector

```
1: dart = GetDartForFace(faceIdx);
2: for (var tempdart = φ(dart), tempdart != dart, tempdart = φ(tempdart)) do
3:     if GetFaceForDart(tempdart) != GetFaceForDart(dart) then
4:         vector.add(dart);
5:     end if
6: end for
```

Using the $dart$ retrieved by method $GetDartForFace(faceIdx)$, all darts belonging to a region are traversed until the input dart is reached again. Bridges and self-loops must be excluded from removal and contraction operations. This is asserted by the check in line 2, i.e. does this dart belong to an edge separating the same face. Due to [Brun and Kropatsch, 1999] the reason is to preserve the number of connected components of the initial combinatorial map, furthermore the contraction operation is not defined for self-loop in the primer graph.

The list of darts belonging to the roof (region 1) and the wall (region 2) is shown in Figure 4.2. The fundamental operations that can be applied on levels and regions $x_1$ and $x_2$ are:

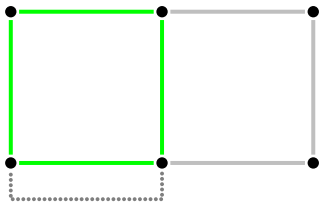
| | **Operation** | **Description** |
|---|---|---|
| **1** | *goto*$(i)$ | go to level $i$ in pyramid ($G_0 \leq G_i \leq G_k$, from pixel-level to top) |
| **2** | *imrg*$_1(R_{(i,x_1)})$ | inhibition of merging $x_1$ with other regions in higher levels $l > i$ |
| **3** | *mrg*$(R_{(i,x_1)}, R_{(i,x_2)})$ | merging regions $x_1$ and $x_2$ |

Table 4.1: Basic Operations.

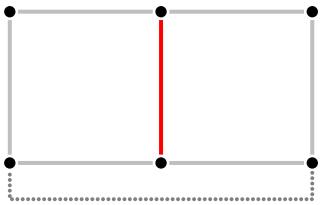
The **first** one, *goto*$(i)$, is no operation on segments but on the stack of segmentations. In the user interface of the interactive extension, one may choose arbitrary level ($0 \leq i \leq k$) within the hierarchy. Within a particular segmentation the user is then able to select a region for further interest. In the following such a region may be described using the two patterns, this depends on the context:

- *Generic*, a region $x_1$ in level $i$: $R_{(i,x_1)}$;
- *Specific*, a region e.g. $x_1 = 1$ in the context of level $i = 10$: $R_1$;

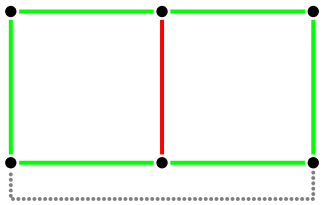
The **second** operation *imrg*$_1(R_{(i,x_1)})$ is inhibition of a region $x_1$ from merging with others (see Figure 4.3a). It takes only one region (therefore the subscript '$_1$') index of a certain level $i$ as input argument. All edges from the selected segment are excluded from the following merging



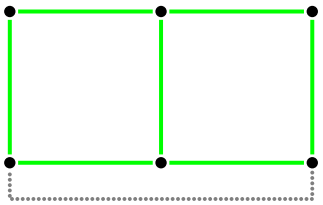

(a) Inhibition of a region, affected edges indicated in green.




(b) Merging of two regions, affected edge indicated in red.




(c) Merging of two regions, inhibition of their outer border.




(d) Inhibition of two regions.

Figure 4.3: Example operations in arbitrary level $i$ on specific regions.

process. The method for deriving those edges is already outlined in Procedure 4. Combining those two operations, $goto(i)$ and ${}_1(R_{(i,x_1)})$, shows one of the advantages of this approach. Figure 4.4 sketches this behavior roughly. It shows a stack of segmentations, i.e. image partitions at different resolution/granularity levels. The nodes representing a partition may be selected from any level within the merging tree (ranging from graph $G_0$ to $G_k$). This way it is possible to combine segments (indicated in green) from arbitrary level of detail to a final result containing all of them. The propagation of operations to higher levels within the merging tree is explained in Section 4.3.

The **third** operation, $mrg(R_{(i,x_1)}, R_{(i,x_2)})$, merges the adjacent regions $x_1$ and $x_2$ given and removes their common edge (see Figure 4.3b) in the structural representation. The calculation of the joining edge out of the region indices is again based on the cycles of their $\varphi^*$ permutations. The darts for each region are compared until both belong to the same cycle of $\alpha$ and consequently to the same edge. This schema is outlined in Procedure 5. Like before, all edges belonging to a region are calculated separately using the frameworks' method presented. Talking in terms of numbers which helps to understand the principle behind a little better, these edges are then compared until the common edge between the adjacent regions is found. This particular edge exists twice and in the cumulated set of edges, whereas all others appear only once.

Basically this test compares *dart1* with $\alpha(dart2)$. If the resulting dart of the involution matches the input dart we found the common edge to be removed. This is the technical

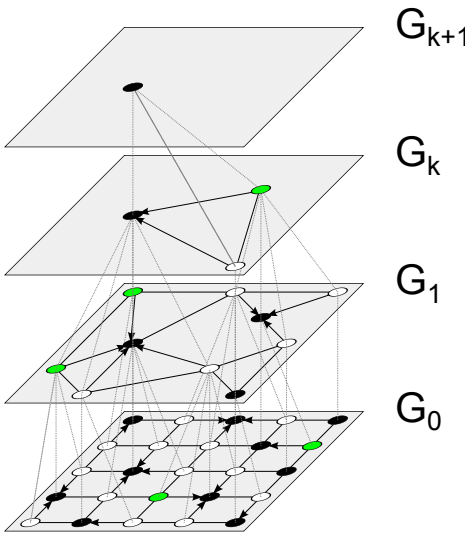



Figure 4.4: Simplified graph pyramid which contains vertices (regions) marked for inhibition of merging operation (green) placed in different levels. The arrow of an edge indicates the direction of a contraction.

background, now we can start to select the regions belonging to these operations.

There are three different ways implemented in the interactive extension of how to select a particular region for processing. In the *selection mode* each segment is selected individually by simply clicking on it, like in Figure 4.5a where the two regions to be merged are chosen individually. The colors assigned are customizable, in this example a selected region is indicated in yellow and the current region at the location of the mouse-pointer using its inverse RGB value. In general each operation has a different color which makes it easier to distinguish them. For visualization purpose we add a separate layer on top the segmentation image, there each input region is colored accordingly. The *brush mode* shown in Figure 4.5b can be used to select regions more efficiently by simply drawing over the regions affected. For the merging operation this is visualized using a red stroke, however as it must not have more than two input arguments the last input region of the recent operation is automatically passed as first argument to the new operation. This operation is comparable to the paintbrush method [Van Leemput et al., 1998]. As there is no explicit visualization of inter-pixel edges – like crack-edge visualization in [Meine et al., 2004] – it is not possible to directly select a particular edge, but only the regions

**Procedure 5** Finding the common edge

*Input:* vector1, vector2 *Output:* commonEdge

```
1: for all dart d1 in vector1 do
2:     for all dart d2 in vector2 do
3:         if getEdgeForDart(d1) == getEdgeForDart(d2) then
4:             commonEdge.add(getEdgeForDart(d1));
5:             return commonEdge;
6:         end if
7:     end for
8: end for
```




(a) Selection mode (b) Brush mode (c) Explore mode

Figure 4.5: Different selection modes for user interaction.

separated by the same. The *explore mode* shown in Figure 4.5c will be explained together with the corresponding operations later in this section.

## Nested Operations

But merging solely does not implicate that the resulting region will be inhibited automatically since the automatic algorithm can decide to merge it with other regions in higher levels of the pyramid. Through 'nesting' other combinations are possible which attempt to solve this problem, see Table 4.2 for reference. Furthermore the intention is to provide 'useful' combinations, which supports efficient interactive image-segmentation.

| | Operation | Description |
|---|---|---|
| **4** | $imrg_1(mrg(R_{(i,x_1)}, R_{(i,x_2)}))$ | merge $R_1$ and $R_2$ in level $i$ and inhibit the result from merging with others in higher levels |
| **5** | $imrg_2(R_{(i,x_1)}, R_{(i,x_2)})$ | inhibit $R_1$ and $R_2$ in level $i$ from merging with others in higher levels |

Table 4.2: Nested operations.

Similar to the basic operations, the nested operations are defined. The **fourth** operation is $imrg_1(mrg(R_{(i,x_1)}, R_{(i,x_2)}))$, see Figure 4.3c for an example. There the common edge between the adjacent regions $R_{(i,x_1)}$ and $R_{(i,x_2)}$ selected is removed while the others are inhibited from merging. The methods to calculate the common 'inner' (indicated using red color) and limiting 'outer' (green colored) edges are identical as before. In the combined set of edges all edges which appear twice will be removed, the others are going to be protected.

The **fifth** operation is $imrg_2(R_{(i,x_1)}, R_{(i,x_2)})$. In contrast to before, this inhibition operation now takes two regions as input (therefore the subscript '$_2$'). All edges belonging to the two adjacent regions $R_{(i,x_1)}$ and $R_{(i,x_2)}$ - i.e. their common and outer boundary - are inhibited from merging from the next level onwards. This is shown in Figure 4.3d, all corresponding edges are not put into the list of edges to be contracted. Our intention behind this operation is not to simply protect two adjacent regions in a specific level (which could also be done using the

corresponding basic-operation), but – formulated differently – rather splitting-up two regions and preventing them from merging in higher levels.

Finally there exist combined operations which are a combination out of both basic and nested operations. They relate to the explore-mode concept, as seen in Figure 4.5c, already mentioned before. With this method it is possible to divide a region into its components within its receptive field. This is preserving a selected region (surrounded red) in a level $j$ with lower granularity and restore a particular detail (indicated in yellow, like the birds beak) from a level $i$ having higher granularity (level $i \geq j$) in one step. Any detail of the object of interest which already disappeared in a higher level may be restored this way without going back to a lower level. Without this operation, the user is forced to leave the current 'working level', now we only change the level within a particular region. However, instead of creating a complex operation, in this case two of the previously presented operations are generated in different levels:

| | **Operation** | **Description** |
|---|---|---|
| **6** | $\mathit{imrg}_1(R_{(i,x_1)}) + \mathit{mrg}(\mathit{rem})$ | inhibit $R_1$ and in level $i$ and merge all others (placeholder $rem$) to get a clean foreground-segmentation |
| **7** | $\mathit{imrg}_1(R_{(i,x_1)}) + \mathit{imrg}_2(R_{(j,y_1)}, R_{(j,y_2)})$ | in $R_1$ of level $i$, split selected regions $R_1$ and $R_2$ contained in level $j$ |
| **8** | $\mathit{imrg}_1(R_{(i,x_1)}) + \mathit{imrg}_1(R_{(j,y_1)})$ | in $R_1$ of level $i$, restore selected region $R_1$ contained in level $j$ |

Table 4.3: Receptive Field Operations.

The **sixth** and last operation implemented $\mathit{imrg}_1(R_{(i,x_1)}) + \mathit{mrg}(\mathit{rem})$ is designed to get a clean foreground/background-segmentation. If the segmentation-image on top of the pyramid is very noisy, the list of operations containing regions to be merged will be very long (remember, each corresponding operation can take only two arguments) and confusing to the human operator. This way we introduce the placeholder *rem* representing all remaining regions. The wildcard will be expanded in the segmentation framework using the structural representation. However, we skip the preprocessing phase (see Section 4.3 for details) and choose a straight-forward approach. All remaining edges – except the ones belonging to region $R_{(i,x_1)}$ and the background face $f_{\varphi}^{\infty}$ – are directly selected for removal.

Other operations would be defined easily, like in opposite of removing a single edge keep it protected throughout a lifecycle. It turns out it is not possible to control such situation in the current setup (where only the region index changes are tracked). If the regions separated by this edge do (not) merge with others, this one would be left in a dangling state and causing a corrupted structural and visual representation. The selected edge somehow needs to evolve, i.e. develop gradually following the merging tree (using tracked edge index changes). However this approach leaves some open questions to be investigated in detail. Please note that the list of operations presented so far is not complete and may be extended (if needed).

Examples showing the effect of each operation will be presented in the next section. They all may be combined individually, in the end it is an entire set of operations placed in various levels within the stack of segmentations which needs to be processed. Therefore it is crucial

to implement special mechanisms which keep track of the consistency all the time and enforce correct processing. The mechanisms implemented check the overlap of operations' receptive fields, more on this is explained in Section 4.3.

What remains to be discussed, why is it necessary to use darts when the final-processing is based on edges? The concept of the segmentation framework at the base is general and can be implemented also on other topological representations like region adjacency graphs and dual graphs [Haxhimusa and Kropatsch, 2004]. Thus this work is not limited to the image representation with combinatorial maps even though it strongly benefits from them because of the implicit dual representation. Therefore we only need the concept of regions' boundary in order to define the user operations.

## 4.3 Processing

### Preprocessing of operations

The interactive extension creates a list of operations which is finally passed on to the modified automatic segmentation framework. Each operation entry in the list is of the form $\{i, \text{op.}, R_{\text{i},1}[, R_{\text{i},2}]\}$. An integer value $i$ denotes the level within the pyramid, the string op the operation identifier and $R_{\text{i},1}[, R_{\text{i},2}]$ the regions affected. The second region is optional as unary and binary operations are defined. This list and the basic operations are essential for the segmentation process. The translation of the region arguments $R_{\text{i},1}[, R_{\text{i},2}]$ into adjacency relations was already covered in the previous section, but how exactly are the different operations mapped/processed in the segmentation framework? The set of user operations is not immediately applied upwards in the merging tree. Since we allow operations to be selected on different levels we need to find a common starting level.

**Definition 4.3.1.** *A common starting level is the lowest level in which at least one operation is defined. From this level onwards the operations given as input are processed.*

However, the ones initially placed in higher levels need to be down-projected. In this part of the preprocessing phase the parent-child information contained in the merging tree is very important. The procedure for down-projection presented in the following, traverses the merging tree backwards. Starting from the root node of each operation (probably lying in different levels), all child nodes (regions) in the common starting level are determined. Internally and for computational reasons, it is not exactly the lowest level among the operations recorded.

```
 Level 5:  MIES 0.06s, Kernel 0.32s, Total 0.64s , Faces left 43652
 Level 6:  MIES 0.05s, Kernel 0.31s, Total 0.62s , Faces left 43650 <-- common starting level
*
 Level 7:  MIES 0.11s, Kernel 3.86s, Total 4.23s , Faces left 19390
 Level 8:  MIES 0.04s, Kernel 0.72s, Total 0.97s , Faces left 11785 <-- Operation 1 = lowest level
 Level 9:  MIES 0.02s, Kernel 0.18s, Total 0.36s , Faces left 10054
 Level 10:  MIES 0.01s, Kernel 0.08s, Total 0.25s , Faces left 9748 <-- Operation 2
 ...
*
 Level 13:  MIES 0.02s, Kernel 0.24s, Total 0.39s , Faces left 4355
 Level 14:  MIES 0s, Kernel 0.05s, Total 0.18s , Faces left 2583 <-- Operation 3
 Level 15:  MIES 0s, Kernel 0.01s, Total 0.13s , Faces left 2073
```

Listing 4.1: Operations' down-processing schema.

An example is shown in Listing 4.1. There, three operations were placed in different levels ($8, 10$ and $14$). The lowest level among these operations is level $8$, but the common starting level will be $6$. Level $6$ is the closest level prior to the latest Borůvka step indicated by an asterisk '*'. At this point it is necessary to describe the process of the evolving pyramid again. In the base combinatorial map at level $0$ (not shown in the output) each pixel corresponds to a region. This level is copied to the new level $1$. In the beginning of level $1$ not only the Borůvka step (building up the MST) is executed, also the final set of edges to be removed is determined using relevance filtering. This set is decimated until all edges selected for removal are processed, so at the end of level $6$ only $43650$ regions (or faces) are left in the segmentation image. Then the corresponding graph of level $6$ is copied and the process described before starts over again. This is why the final segmentation of level $6$ will be our common starting level and the highest one kept in the new stack. All levels above this one will be deleted. In the edge selection phase of newly added level 7, the interactive operations will then be considered as well.

What is important to mention as well is that the region indexes are reorganized in the post-processing steps at the end of each level. This causes the region indexes to be equivalent always at the end of level $i$ and the beginning of $i+1$. Comparing this to the approach of [Meine et al., 2004] presented in Section 2.5, the process here is slightly different (see Figure 4.6 for a summary). From the different levels where operations are set, the lowest one among them is chosen to be the starting level for applying the operations.

In this common starting level the candidate-edges affected from operations (contraction or inhibition) are determined, allowing rebuilding the pyramid from this level upwards (see Figure 4.7). More precisely, deleting all older levels above the common starting level and

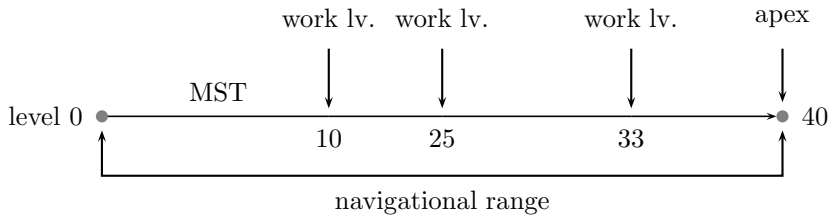



Figure 4.6: Alternating application of automatic and interactive methods.

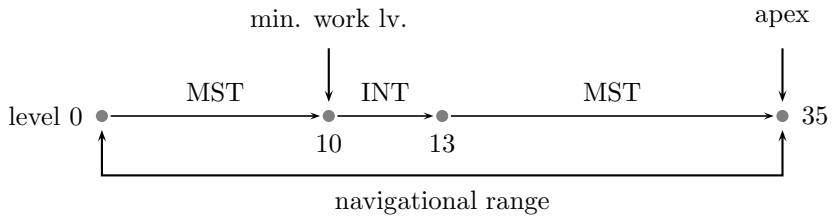


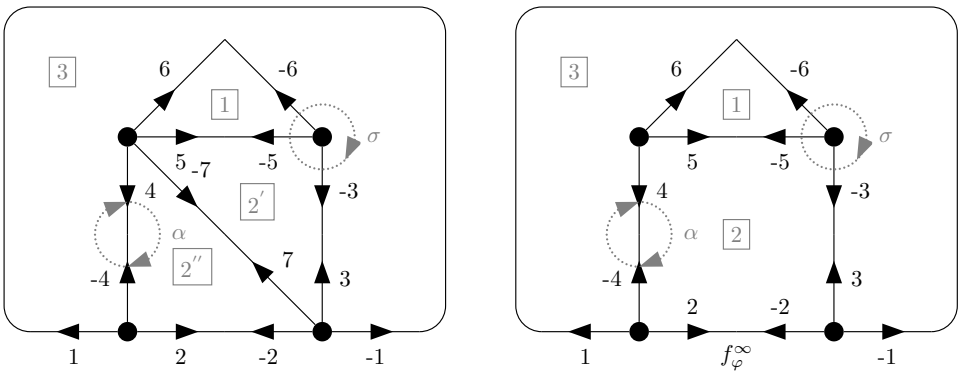

Figure 4.7: Processing overview after restart/recalculation.

recalculating the new levels. Consequently only the really affected parts of the merging tree are recalculated from the common starting level onwards instead of the entire pyramid. First – in a separate step – the interactive (INT) merging operations will be processed. Then the process continues with the inhibition of regions and automatic merging of the remaining regions. As we do not move the interactive operations to the base level like in the approach of [Meine, 2009b], the different (intermediate) resolution levels are retained for later usage.

This recalculation is achieved using the hierarchical information encoded in the merging tree, through permutations on the $\mathcal{D}$ with $\varphi$ in the combinatorial map and set operations as shown in the following example. Figure 4.8 shows two consecutive levels[1] $i$ and $i+1$ in the pyramid. Let Figure 4.8a be a common starting level and Figure 4.8b be a higher level where operation $mrg(R_{(i+1,x_1)}, R_{(i+1,x_2)})$ shall be applied. We intend to merge region 1 (roof) with 2 (wall) and inhibit the resulting region from merging. We can see that this operation leads to a different and non corresponding result $imrg_1(mrg(R_{(i,1)}, R_{(i,2')}))$ when applied on the combinatorial map in the level $i$ below. The wall of the house (region 2) in level $i$ is divided ($\xrightarrow{\text{div}}$) into two subregions $2'$ and $2''$, the boundary the house's roof (region 1) remains the same in both of the levels. Therefore we need to reconstruct the original operation of level $i+1$ to its correspondent in level $i$. Using the relation between the regions in the common starting level $i$ and the level above $i+1$ and its $\varphi$ permutation, the corresponding darts (or edges) can be reconstructed as follows:

$$
\begin{aligned}
&R_{i+1,1} \xrightarrow{\text{div}} R_{i,1} \\
&R_{i+1,2} \xrightarrow{\text{div}} R_{i,2} = R_{i,2'} \wedge R_{i,2''} \\
&\text{Darts for } R_{i,1} = \varphi(5, -6) \\
&\text{Darts for } R_{i,2} = \varphi(3, -5, -7) \wedge \varphi(2, 7, 4)
\end{aligned}
$$

[1] Figure 4.8(a) and (b) are based on the examples from [Brun and Kropatsch, 1999].




(a) Level $R_i$ (b) Level $R_{i+1}$

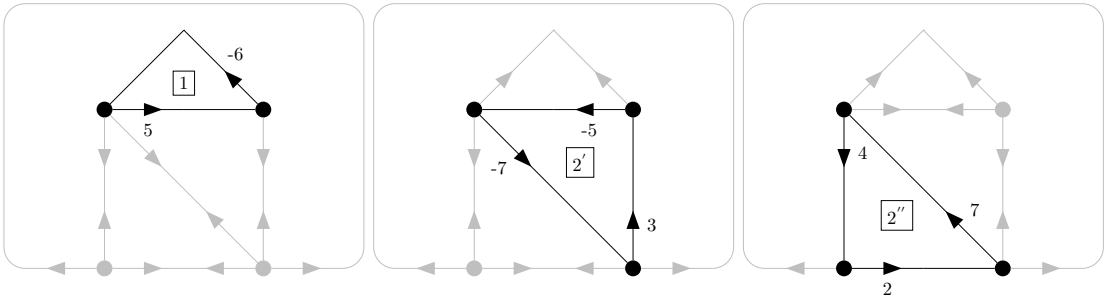
Figure 4.8: Combinatorial maps from different segmentation levels.

There are two different categories and collections of edges (inhibition and removal) necessary to process the operation correctly:

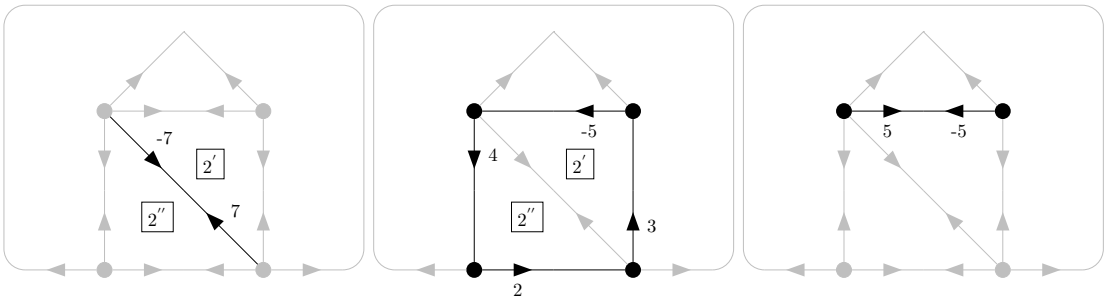
1. **Former Contractions:** in each set of $R_{i,1}$ and $R_{i,2}$ some darts belong to the same cycle of $\alpha$. This edge indicates the adjacency/subdivision from region $R_{i+1,2}$ in level $i$ and is marked for **removal**, i.e. edge $(7, -7)$ in Figure 4.9d.

2. **User Operations:**

   - all edges of $R_{i,1}$ (Figure 4.9a) and $R_{i,2}$ (Figure 4.9b and 4.9c) without the marked edges for removal from former contractions (Figure 4.9d), represent the original (outer) borders of regions $R_{i+1,1}$ and $R_{i+1,2}$. All these edges in level $i$ are marked for **inhibition**, e.g. darts $(3, -3)(5, -5)$, $(2, -2)(4, -4)$ of $R_{i,1}$ and $(6, -6)$, $(5, -5)$ of $R_{i,2}$.
   - the darts from the outer borders of $R_{i,1}$ (Figure 4.9a) and $R_{i,2}$ (Figure 4.9e) of each region are compared to find the common edge. This edge is marked for **removal**, see edge $(5, -5)$ in Figure 4.9f.




(a) Darts of region 1 (b) Darts of region $2^{'}$ (c) Darts of region $2^{''}$




(d) Shared edge of regions $2^{'}$ and $2^{''}$ (e) Outer border of regions $2^{'}$ and $2^{''}$, or region 2 in level $i+1$ (f) Shared edge of regions 1 and 2 belonging to level $i+1$

Figure 4.9: Edge reconstruction in level $i$.

This process is repeated for all operations. In the example shown, the selection of edge $(5, -5)$ leads to a conflict because it appears in both inhibition and removal edge sets of the **users' operations**. Because user intention is to merge, a rule defined decides that this edge has to be removed. Such an invalid overlap-situation has to be avoided in any case, for this purpose a dedicated mechanism is introduced and explained later in this section.

### Implementation of the processing phase

As it is important for understanding sequent steps, we present below the exact implementation of operations' processing performed in the automatic framework. The down-processing of operations w.r.t. the common starting level and identification of edges representing adjacency relations rely in the first place on the combinatorial map or in general graph operations.

We did not yet explicitly discuss these phases, but due to the conflict processing and interactive operation propagation phase it was necessary to store the operations list (and meta-data) for the purpose of computation in a slightly different way than $\{i, \text{op.}, R_{\text{i},1}[, R_{\text{i},2}]\}$. Irrespective of this the overall processing concept defined in the previous section remains the same as presented before. The propagation phase is needed in case that inhibition from merging operations are defined. Its corresponding edges will automatically be protected until the end of the segmentation process (top of the pyramid). Otherwise the user would have to protect the same region repeatedly in each level.

The list is now stored in a different format $\{i, \text{op.}, \text{items}, \text{reason}\}$, where items stands for a collection of edges using the following structure:

- *edge* affected
- *metadata*
  - *count*, how often does this edge appear for specified operation
  - *parent*, which are the regions belonging to the edge

The most significant change affects the representation in terms of affected regions (before) to joining edges. This separate data-structure may look redundant as this information can be retrieved easily from the combinatorial map as well. First of all it is required also by the automatic framework as we need to pass on a set of edges to be contracted. Second, due to fitting the new functionality into the existing automatic framework, this step was needed for computational reasons. Therefore an edge belonging to a region is now stored in the new format for interactive processing only. In general, the overall procedure remains the same as explained previously.

Each region given in the operation entry is processed separately. Because of the common-starting level, a region may split up within its receptive field in lower levels. Therefore we need to determine the edges belonging to former contractions in order to reconstruct the initial operation again. As mentioned before, this all is based on Procedure 4 where the edges for each child-region are determined and collected in the structure presented above. The edges belonging to former contractions can easily be identified as they share a count in the metadata of value two (i.e. exist twice). They are copied into a separate list (called 'wasteMap') having

| edge | metadata | |
|---|---|---|
| | *count* | *parent faces* |
| $(7,-7)$ | 2 | $2^{'}, 2^{''}$ |

(a) wasteMap

| edge | metadata | |
|---|---|---|
| | *count* | *parent faces* |
| $(3,-3)$ | 1 | $2^{'}, 3$ |
| $(5,-5)$ | 2 | $1, 2^{'}$ |
| $(2,-2)$ | 1 | $2^{''}$, BG |
| $(4,-4)$ | 1 | $2^{''}, 3$ |
| $(6,-6)$ | 1 | $1, 3$ |

(b) reducedMap

Figure 4.10: Edge maps (a) shows the edges of former contractions (b) the original ones.

| **Op.** | **Type** | **Description** | **Edge #** | **Set** |
|---|---|---|---|---|
| 0 | MF | $mrg(R_{(i,x_1)}, R_{(i,x_2)})$ | 2 | edgeRemovalSet |
| 1 | IF | $imrg_1(R_{(i,x_1)})$ | 1 | edgeInhibitionSet |
| 2 | IE | $imrg_2(R_{(i,x_1)}, R_{(i,x_2)})$ | 2 | edgeInhibitionSet |
| | | | 1 | edgeInhibitionSet |
| 3 | MI | $imrg_1(mrg(R_{(i,x_1)}, R_{(i,x_2)}))$ | 2 | edgeRemovalSet |
| | | | 1 | edgeInhibitionSet |
| 4 | WS | waste | 2 | edgeRemovalSet |
| 6 | MA | $imrg_1(R_{(i,x_1)}) + mrg(rem)$ | 1 | edgeInhibitionSet |

Table 4.4: Mapping of operations between the segmentation framework and its interactive extension in the first three columns. Plus assignment of edges to the correct removal/inhibition set depending on the operation (first column). The fourth column shows the count of edges retrieved from the metadata, the corresponding containers *edgeRemovalSet* and *edgeInhibitionSet* in the fifth column hold the index of the affected edge.

the same structure and holding all similar edges. Our edges belonging to the original operation (and appearing only once, see corresponding value for count in the metadata) are stored in a list called 'reducedMap'. Figure 4.10 shows the same example as presented before in Figure 4.8, but in a different format.

The mapping of interactive operations in the GUI and the automatic framework is shown in Table 4.4. Of course, for each basic and nested operation there is a correspondent defined, but again for the purpose of computation there are some slight differences. Column *operation* contains a real valued representative of each operation, the abbreviation in column *type* matches its name in the automatic framework. You will notice that some operations are missing here, but the others may also be expressed in terms of the basic operations. There is no explicit type defined for:

- $imrg_1(R_{(i,x_1)}) + imrg_2(R_{(j,y_1)}, R_{(j,y_2)})$
- $imrg_1(R_{(i,x_1)}) + imrg_1(R_{(j,y_1)})$

The new dedicated type '4' is required in order to distinguish the different types of edges to be removed (former vs. original). Finally a new list of operations is allocated, each operation adds two entries containing the:

- **Original edges**: $\{level, operation, \text{reducedMap}, reason\}$.
- **Former edges**: $\{level, 4, \text{wasteMap}, 4\}$

The usage of the *reason* argument will be explained in the subsection on conflict processing, its value comes as well as the one for the operation identifier from Table 4.4 and is intended to save the original purpose of the operation. Usually they are equivalent, the following example reveals its usage. In case of operation $5$ – in order to get a clean foreground segmentation – we remember instead of selecting all regions in the GUI this step is performed in the automatic part. As this type of merging is equivalent to operation $0$ we add the following entry $\{\text{level}, 0, \text{reducedMap}, 5\}$ to the list of operations. In the last column we save the 'original' operation, this is needed in order to resolve conflicts later and to distinguish between other merging operations. Finally the operations are ready to be processed in the Borůvka '*' level.

**Procedure 6** DoContraction - interactive

```
1:  I: edgeInhibitionSet, edgeRemovalSet = ProduceActionSets(operationsList);
2:  if edgeRemovalSet != empty then
3:      interactiveRemovalRequested = true;
4:  end if
5:  if interactiveRemovalRequested == false then
6:      A: mark all edges that could be safely contracted
7:      for all edges ∈ edgeInhibitionSet do
8:          edgeMarkedForRemoval[e] = false;
9:      end for
10:     A: first random selection
11:     A: build the MST using Boruvka's method
12:     A: second random selection
13:     A: filter candidates through adaptive thresholding (internal/external contrast)
14: else
15:     for all edges ∈ edgeRemovalSet do
16:         edgeMarkedForRemoval[e] = true;
17:     end for
18: end if
19: A: Do stepwise contraction
20: A: Update edge weights
21: I: Update operations
```

Procedure 6 shows an overview of the steps performed in this level. All lines with prefix '*A:*' belong to the functionality of the automatic framework, the ones with prefix '*I:*' to the interactive part. Besides the edges affected by the automatic process, we also need to determine the edges

affected by interactive selection as well. Therefore we allocate two separate lists which will hold the edges to be inhibited from merging in the *edgeInhibitionSet* and of course a list called *edgeRemovalSet* for the edges to be contracted. See Table 4.4, containing the mapping for each operation. But why do we need both sets? This separation is very important as further processing requires a separation, the reasons are shown in the following. Reminding the automatic edge selection approach in Chapter 3, in the first step all edges are selected for removal. Afterwards the candidate edges list is decimated using relevance filtering methods. In order to embed the interactively selected edges in the automatic process, consequently:

- the candidate-edges to be inhibited from merging need to be excluded already in the very beginning, i.e. deselected before the Borůvka phase and relevance filtering begins (lines 6 to 13).
- the candidate-edges for contraction will be processed in a custom contraction phase (lines 15 to 21). This way we omit conflicts between existing functionality and force low coupling/high cohesion. Furthermore the segmentation method may be replaced easily without damaging the interactive extension.

Listing 4.2 shows a sample logging-output of the segmentation framework. First the candidate-edges to be removed are processed. Using the flat-pyramid approach we isolate the corresponding contraction steps to a single (flat pyramid) level. On advancing to this new level, no other (automatic) contractions will be processed.

```
-- interactive edge selection removal begin--
*
 Level 16 Interactive Removal: Flat Pyramid
-------------------------------------------------------------
 Flat Level 1:  MIES 0s, Kernel 0s, Total 0s , Faces left 520
 Flat Level 2:  MIES 0s, Kernel 0s, Total 0s , Faces left 517
-------------------------------------------------------------
>> processing operations
*
 Level 17:  MIES 0s, Kernel 0.01s, Total 0.15s , Faces left 303
 Level 18:  MIES 0s, Kernel 0s, Total 0.16s , Faces left 238
 Level 19:  MIES 0s, Kernel 0s, Total 0.15s , Faces left 219
 Level 20:  MIES 0s, Kernel 0s, Total 0.13s , Faces left 212
 Level 21:  MIES 0s, Kernel 0s, Total 0.13s , Faces left 211
>> processing operations
*
...
```

Listing 4.2: Edge processing of interactive operations.

This way the interactive edge-removal phase is clearly separated from others. As of level 17 in this example, also the edges to be inhibited are propagated as shown in Procedure 6. Furthermore this separation allows the user to track the changes to the merging tree induced by the interactive method easily.

### Conflicting operations

As already mentioned, the instruction set – i.e. the collection of operations – has to be conflict free for certain combinations in order to produce predictable results. Predictable in this context means, the user will get exactly the segmentation result he wants to have. This is different to

other approaches like interactive graph cuts based frameworks [Boykov and Jolly, 2001] where the effect of a stroke is unpredictable for the user to a certain extent. An invalid overlap of the regions and their receptive fields within the hierarchy with respect to the operations recorded needs to be avoided in any case. Obviously some operations are not allowed to conflict. If there is a invalid overlap within the receptive field of a operations region, an ambiguous instruction set would be created. In order to exclude such inconsistencies internal rulesets are defined to validate the operations at runtime. Two cases may be distinguished and need to be tracked:

1. Invalid combinations

2. Multiple application of operations (compare brush-mode)

Regarding the **first step**, this evaluation is done on region basis. Therefore each region is tagged during runtime in the interactive GUI with a dedicated boolean flag in the merging tree, if a certain operation is set. After recording an operation (its arguments are omitted in the example below), each affected region within the merging tree (and receptive field) is iterated and checked if the truth value of the following terms is $\top$ (true):

$$
\begin{aligned}
&\forall \text{ affected regions } x \text{ within the receptive field between level } 1, i \\
&\text{boolean rule1} \;=\; \neg(\mathit{imrg}_1 \wedge \mathit{mrg}) \\
&\text{boolean rule2} \;=\; \neg(\mathit{imrg}_2 \wedge \mathit{mrg}) \\
&\text{boolean rule3} \;=\; \neg(\mathit{imrg}_1(\mathit{mrg}) \wedge \mathit{mrg}) \\
&\text{boolean rule4} \;=\; \neg(\mathit{imrg}_1(\mathit{mrg}) \wedge \mathit{imrg}_2) \\
&\text{boolean rule5} \;=\; \neg((\mathit{imrg}_1(x) + \mathit{mrg}(\mathit{rem})) \wedge \mathit{mrg})) \\
&\text{boolean rule6} \;=\; \neg((\mathit{imrg}_1(x) + \mathit{mrg}(\mathit{rem})) \wedge \mathit{imrg}_1(\mathit{mrg})) \\
&\text{return validOperation} \;=\; \text{rule1} \;\wedge\; \text{rule2} \;\wedge\; \text{rule3} \;\wedge\; \text{rule4} \;\wedge\; \text{rule5};
\end{aligned}
$$

In the first case, if this region has the boolean flags set both for $\mathit{imrg}_1(R_{(i,x_1)})$ and $\mathit{mrg}(R_{(i,x_1)}, R_{(i,x_2)})$, the current operation and the others already recorded are conflicting. The single term evaluates to false $\perp$, so the overall return value 'validOperation' combines all return sub-clauses. For several combinations of operations there exists a truth equation (see Table 4.5 for the full list and detailed explanations).

Most of the conflicts (invalid combination of operations) are already caught here and the user is prompted during runtime. The main reason behind the two phase conflict resolution is that for the situations presented in Table 4.5, there is no possibility to resolve them in the automatic part. These decisions on ambiguous instruction sets require user interaction.

A **second conflict processing step** is executed in the automatic segmentation framework itself. There is one situation which cannot be covered by the first step, i.e. conflicting edges when certain operations are applied one after another. Imagine the following overlap situation shown in Figure 4.11: Operation 1 is $\mathit{imrg}_1(\mathit{mrg}(x, y))$ and operation 2 and $\mathit{imrg}_1(\mathit{mrg}(y, z))$. The intention here is to merge regions $x, y, z$ and protect their resulting outer border.

If we look on each operation separately, the outer boundary created through merging region $x$ and $y$ shall be protected. This is true also for regions $y$ and $z$ of the second operation. As

| **Operation 1** | **Operation 2** |
|---|---|
| $imrg_1(x)$ | $mrg(x, y)$ |
| Inhibition against merging region $x$ with others (operation 1) and merging region $x$ with $y$ (operation 2) is not allowed. There are two contradicting operations defined on region $x$, leading to an ambiguous instruction set. | |
| $imrg_2(x, y)$ | $mrg(x, z), (y, z)$ or $mrg(x, y)$ |
| Inhibition against merging regions $x$ and $y$ together or with others (operation 1) and merging e.g. one of the regions of operation 1 with another one is not allowed. Again this leads to contradictions. If mrg$(x, y)$ is recorded as operation 2, the inhibition overrules the deletion of the common boundary. For protecting the outer boundary and removing the inner one, a particular operation is defined. | |
| $imrg_1(mrg(x, y))$ | $mrg(x, z), (y, z)$ |
| In operation 1 region $x$ and $y$ are first merged together while the outer boundary is protected. Hence merging (operation 2) one of the previous regions with a different one $z$ is not allowed. Region $x$ and $y$ may be merged in step 2 because the instruction set is identical to the inner part of operation 1. | |
| $imrg_1(mrg(x, y))$ | $imrg_2(x, y), (x, z), (y, z)$ |
| Merging regions $x$ and $y$ whilst inhibition of the outer boundary in operation 1 while preventing one of the affected regions from merging with others and of the common boundary is not allowed. Operation 2 with $x$ and $y$ is the inverse operation, $x$ and $z$ or $y$ and $z$ would lead to a conflicting instruction set on the common boundary of $x$ and $y$. | |
| $imrg_1(x) + mrg(rem)$ | $mrg(x, y \in rem)$ |
| Operation 1 is a combined one, nevertheless the rule (similar to the first one) applies also to this combination. Inhibition of region $x$ in operation 1 and merging all others while merging $x$ with region $y$ in operation 2 is not allowed. | |
| $imrg_1(x) + mrg(rem)$ | $imrg_1(mrg(x, y \in rem))$ |
| Operation 1 is a combined one, nevertheless the rule (similar to the second one) applies also to this combination. Inhibition of region $x$ in operation 1 and merging all others while merging $x$ with region $y$ and inhibition of the result in operation 2 is not allowed. | |

Table 4.5: Conflicting operations, evaluation based on regions.

we can see in Figure 4.11, the edges to be removed and protected intersect. In this case the users' intention is to merge, so the inhibition operations of the edges separating $x$ from $y$ and $y$ from $z$ is overruled. Finally $x, y$ and $z$ will be merged while protecting their outer border. Such complex decisions require processing with edge-based conflict resolution step in the automatic framework.

As we cannot ask the user for a decision here, we require a global rule for handling such situations. If the edge to be removed is also contained in the inhibition set at the same time, we have a look at the 'reason' column stored in the metadata. There we can see which operation

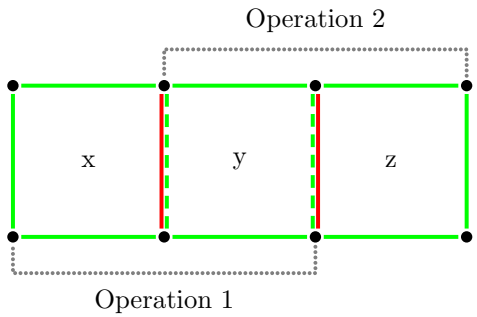



Figure 4.11: Conflicting operations, invalid overlap of merging (red line) and inhibition from merging (solid green line) operations. The dashed green line indicates the outer borders of operation 1 and 2 which finally will be removed.

marked a particular edge for removal:

- $imrg_1(mrg(R_{(i,x_1)}, R_{(i,x_2)}))$: this relates to the example shown in Figure 4.11 where multiple regions shall be merged while protecting their outer border. In this case the particular edge is removed from the inhibition set and kept in the removal set.
- *Other operations:* for all conflicting situations not handled explicitly, by definition inhibition overrules removal.

Using the two-phased conflict resolution approach, a large set of conflicting combinations is covered. This way we ensure a proper instruction set and preserve the consistency of the structural representation.

### Post-processing

At the end of each contraction phase there are some important steps to do. Both Procedure 3 and 6 list some post-processing actions which need to be performed. Because of the merging regions, in addition to edge-weight update also a structural reorganization (of e.g. the region indexes) is required and performed by the automatic framework at each level. As the structure changes, obviously also the operations need to be adapted. Consequently a new step is added which will take care of propagating the inhibition of merging operations. By definition, the edges to be inhibited from merging shall survive until the end of the segmentation process. All others are omitted as the merging operations (with identifier $0$ and $4$) were already processed by the latest Borůvka and contraction phase.

Due to limitations of the automatic framework – index changes of edges are not tracked by definition – we will use again the region numbering and the hierarchical merging tree. At this point the dual representation of combinatorial maps again is required and of great benefit. The procedure is comparable to the one presented in the top-down-processing step but of course in the opposite direction. The old operation list will be overwritten, the detailed steps for each remaining entry are:

1. Get all regions belonging to an operation from the edge metadata and sort them in ascending order to eliminate duplicates.

2. Get the level-to-level region index change from the latest Borůvka level to the topmost (current) map of the stack of segmentations.

3. For each region, determine the graph-edges affected and its metadata (using the specific edge-format presented).

4. Add a new operation following the pattern $\{toplevel, operation, processedMap, reason\}$

Let us take again the example from Figure 4.8. Region $R_{i+1,1}$ does not change with respect to level $i$, for region $R_{i+1,2}$ this is different because its child-regions $R_{i,2'}$ and $R_{i,2''}$ are merged ($\xrightarrow{mrg}$) together. The $processedMap$ list containing only the edges to be inhibited looks like:

$R_{i+1,1} \xrightarrow{mrg} R_{i,1}$

$R_{i+1,2} \xrightarrow{mrg} R_{i,2} = R_{i,2'} \wedge R_{i,2''}$

Darts for $R_{k,1} = \varphi(5, -6)$

Darts for $R_{k,2} = \varphi(3, -5, -7) \wedge \varphi(2, 7, 4)$

| **edge** | **metadata** | |
|---|---|---|
| | *count* | *parent faces* |
| $(2, -2)$ | 1 | 2, BG |
| $(3, -3)$ | 1 | 2, 3 |
| $(4, -4)$ | 1 | 2, 3 |
| $(6, -6)$ | 1 | 1, 3 |

### Examples for interactive operations

This subsection shows some examples and the effect of the operations presented. Please note that the colors indicating the affected regions in the segmentations are artificial and optimized for visualization.

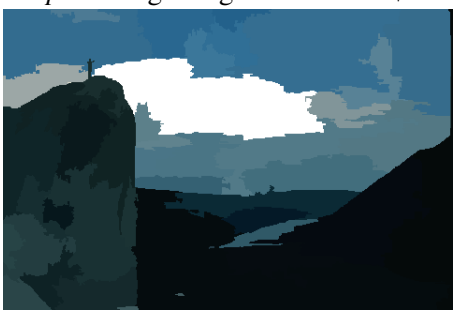
1. *goto*$(i)$ - Goto a higher/lower level

2. *mrg*$(R_{(i,x_1)}, R_{(i,x_2)})$ - Merge two adjacent regions

*Input:* two adjacent regions in level $i$



*Output:* merged region in level $i + 1$



*Description:* The regions (parts of a cloud, indicated in yellow/green) selected in level $i$ for merging, are already merged together (white cloud) in the next level.

3. $imrg_1(R_{(i,x_1)})$ - Inhibit a region from merging

*Input:* a specific region in level $x$



*Output:* region survives until the top



The operations are placed on 4 separate regions (colored green, indicating windows of a church). These regions will be excluded from any future merging decisions until the top-most level of the stack.

4. $imrg_2(R_{(i,x_1)}, R_{(i,x_2)})$ - Inhibit two regions from merging

*Input:* two adjacent regions



*Output:* regions survive until the top



The operation was placed on two adjacent regions (indicated in yellow/green). Both were excluded from any merging decisions and survive until top.

5. $imrg_1(mrg(R_{(i,x_1)}, R_{(i,x_2)}))$ - Merge two regions, and inhibit the resulting region

*Input:* two adjacent regions



*Output:* merged region survives until top



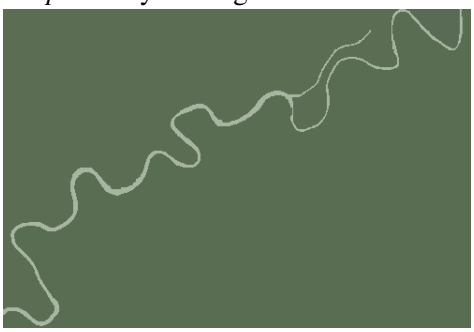
The operation was placed on two adjacent regions (indicated in yellow/red, parts of a cloud). Both merge already in the next level and the resulting region will survive.

6. $imrg_1(R_{(i,x_1)}) + mrg(rem)$ - Inhibit selected region from merging but merge all others

*Input:* one region



*Output:* only this region will survive



This operation is intended to get a clean foreground/background-segmentation. All regions except the one selected (indicated in yellow) will be merged in any case.

7. $imrg_1(R_{(i,x_1)}) + imrg_1(R_{(j,y_1)})$ - Restore a selected region

*Input:* one detail




*Output:* parent region and detail



Preserving a selected region (surrounded green) in a level j with lower granularity and restore a particular detail (indicated in yellow, the cross on top of the church) from a level i, j $\leq$ i.

### Level locking and other GUI operations

This framework could also be used easily for creating a nested image annotation tree. By locking all levels below the top of the pyramid it is possible to create a new stack of segmentations with custom graduation of decreasing granularity. Recording operations below the topmost level is not allowed, this way the remaining regions may iteratively be merged to build a new hierarchy, see for example Figure 4.4 level $k + 1$ on top of the pyramid.

An explicit life-time for operations was not considered in this approach. The main reason behind is that the initial design-decision aims to have the final segmentation on top of the pyramid. A living time for protections is implicitly encoded since it is possible to restart at a desired level and simply not considering a region for protection in future runs. Another approach to indirectly achieve the desired effect is to restart the tool on the topmost segmentation level without interactive operations.

An explicit undo operation has not been realized in this approach. However a history of operations is kept, this way it is sufficient to re-apply the set of operations collected (in appropriate order) on the initial segmentation in order to retrieve the final segmentation result again. Implicitly the user can therefore restore a particular segmentation up to a certain rebuild.

The parent-child relationship defined by the reduction window may be extended by transitivity down to the base level [Marfil et al., 2006]. The initial grid on level $0$ has a well-defined structure, therefore the region-indices remain the same even after starting a new segmentation. If all operations are down-propagated to the base level it is possible to collect, re-use or remove (undo) operations from the instruction set. This attempt is similar to [Meine et al., 2004] where the operations are moved to the very beginning of the process as already shown in Figure 2.13. Because of the design decisions described in the beginning of this chapter, a different approach has been chosen.

### Refine existing segmentation

Using the operations/extension developed it is theoretically also possible to correct an existing segmentation performed by a different method. The only input needed, besides the input image, is the correspondency between the base-layer and the segmentation mask. This can be done on mapping all the pixels of a region (by the use of its labeling) to its correspondents at the base. In a reverse manner an 'artificial' set of operations is built representing the merging process (receptive field), like using operation $imrg_2(R_{(i,x_1)}, R_{(i,x_2)})$. The resulting list of operations is later passed on to the framework which automatically computes the stack of segmentations containing the given 'input' partitioning. This partitioning may then be interactively modified. This feature remains to be implemented and was postponed for future tasks.

CHAPTER 5

# Evaluation

The aim of the evaluation part is to find answers to usability questions, mainly related to the process of creating hierarchical segmentations using the interactive tool developed. In the beginning we present the setup. The evaluation part in general is made up out of two parts which will be discussed in detail, the phase of interactively creating segmentations by a set of users followed by a questionnaire.

## 5.1 Setup

The conditions for the image segmentation task were assigned following the definition of [Martin et al., 2001]: "Divide each image into pieces, where each piece represents a distinguished thing in the image. It is important that all of the pieces have approximately equal importance...". This goal was adapted, finally matching the purpose of the tool developed. This is creating an image segmentation which in the first place is acceptable and in addition that meets the requirements of the user. To be more precise, the user was free to choose the object he wanted to segment. Hence the condition of 'equal importance' was not considered here because of the different scope of the application. It is not about comparing segmentations and measuring the quality compared to ground truth, it is about setting focus on a particular detail in the image – imagine the task of segmenting a computed tomography (CT scan) image of the human brain by a medical practitioner. In certain situations he may not be interested in a clear separation of the bone or the gray/white matter of a brain, but rather in limiting the tumor. In the final survey we ask the candidate to answer the following question, "How satisfied are you with the segmentation result?". This question about acceptance can only be answered if the segmentation goal has been chosen individually and not defined a priori.

For the first phase of creating image segmentations 8 volunteers agreed to participate voluntarily. They signed a declaration of consent, summarizing the evaluation process and data to be recorded during the session (see the form attached to Section B.2). Additionally it is important to mention that these persons neither worked before with the tools developed nor

they have experience in this field of image analysis. Therefore I started with a short technical introduction and demonstration of the concepts.

There is no transcript of this introduction available as this part was based on examples and individual discussions, however a summary of the key topics discussed is provided in the following:

> We started to give a definition of image segmentation by providing a concrete example and application. This is for example locating objects in an image and annotating them with vocabulary. Then we had a closer look on the output of the automatic segmentation process. Using the interactive tool, we explained the concept of the different (resolution) levels in the stack of segmentations: "there, starting on pixel level, the regions iteratively merge together following certain merging criteria". Doing it this way we began to build up basic knowledge in this domain. At this point, however, no detailed information was given about the structural representation (like the concept of irregular pyramids). The user agreed that the automatic method delivers good results, but these are located in different resolution levels. This way we defined the goal of interactive extension developed, which is the combination of these segments to a final segmentation on top of the pyramid which matches the users' expectations. Starting with the two basic operations – inhibition of a region from merging and merging two adjacent regions – a first example of the new functionality was provided. After a short introduction of the GUI, we explained the remaining operations and the brush mode.

This first part and an explanation of the evaluation questionnaire took about $25$ minutes in total. Afterwards the candidates had the chance to test out the interactive segmentation tool on their own and create segmentations as long as they wanted. In this phase any remaining open questions were answered. After two training images, all of the users felt ready to start the exercises.

The five images in Figure 5.1 which are presented to the users were taken randomly (using a helper tool based on class`java.util.Random`) out of the $300$ training and test images from [Martin et al., 2001]. We assumed that each candidate will segment on average $3$ images (estimated response rate). Therefore we specified two of them to be the *control images*, but only one of them will be presented to the user. The others are again chosen randomly. Concluding, the set of images assigned per session consists out of one control image and up to three random images. If a candidate wants to do all five segmentations, the set of images consists out of the two control and the three random images. Doing it this way, the result is expected to be more representative than a complete random presentation of the $5$ or all $300$ images.

The questionnaire at the end was implemented using the open source tool LimeSurvey. This way the users were able to fill in the evaluation form online and anonymously. We asked questions grouped into different categories. The participants should provide feedback on the process of creating *Image Segmentations*, the functionality and usability of the *Segmentation Tool* itself and questions regarding the *Skills* required. The full set of questions can be found in Section B.1 of the appendix.



(a) 113016 (481 × 321 px)



(b) 135037



(c) 108070



(d) 175032



(e) 374020

Figure 5.1: Images for evaluation randomly chosen from [Martin et al., 2001], the captions show the image ID, (a) and (b) are the control images. The others are presented in random order.

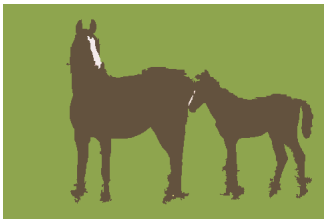
Together with the feedback received when creating the segmentations, we try to answer questions related to possible segmentation strategies among others:

- How often and when (at which level of granularity) was a particular operation used?
- Is there any preferred type of input method (selection/brush)?
- How much time was needed to create a segmentation?
- Does operating the interactive segmentation tool require prior/expert knowledge or can it be used effectively by beginners as well?
- Are there any parts in the process of creating image segmentations that require improvement?

## 5.2 Results

The detailed results of the segmentation phase can be found on the DVD media at `<drive:>/Evaluation`. Images and raw segmentation data is located there as well as the segmentation statistics. Figure 5.2 shows some examples.

The score of the questionnaire is attached to Section B.1 of the appendix. A summary will be provided in the following. Each segmentation has a random identification tag assigned (5 hexadecimal-digits) which helps to establish a correspondence with the related survey questions.



(a) 2A4A6 (7 regions); Focus set on the mare, foal and their pallor.



(b) 3ED15 (3 regions); Focus set on the eagle and fine details.



(c) 337D5 (4 regions); Rough segmentation of the different areas in the scenery.



(d) 2CDC2 (5 regions); Focus set on the foal and fine details.



(e) 25662 (3 regions); Focus set on the eagle, no details selected.



(f) AD4D6 (3 regions); Failed to set focus on tiger and river only.



(g) 12857 (7 regions); Focus set on the branch.



(h) 5FAD6 (45 regions) Focus set on the snake, the big and the fir branch.

Figure 5.2: Images segmentation results, the captions show the test identifier ID and the region count. The candidates set focus on different objects and level of detail.

Overall 21 segmentations have been created by 8 persons. Each image (see Figure 5.1) – randomly chosen out of [Martin et al., 2001] – can be identified by a file name having 6-digits. The session ID which identifies each segmentation is emphasized in the following using the typewriter font:

- Image 113016 × 5: `T2A4A6`, `2CDC2`, `3ABE4`, `4C584`, `78B48`
- Image 135037 × 5: `3ED15`, `5DFB6`, `25662`, `BD498`, `FE38F`
- Image 374020 × 4: `337D5`, `494B3`, `B64DB`, `C3D2D`
- Image 108070 × 3: `54DFB`, `452B2`, `AD4D6`
- Image 175032 × 2: `5FAD6`, `12857`

Two segmentations (`6C6DA`, `5814C`) of one candidate are not listed here as they are not taken into consideration due to the fact that the process has been aborted twice because of runtime errors. There one candidate extensively used the brush-mode on a very low level with high resolution. The implementation of the MouseListener (Java) waiting for the double-click to activate this mode and recording the regions, did not react as expected. In conclusion, the participant was not able to produce a satisfying segmentation according to his strategy and expectations (see comments in questions `G2_Q0002` and `G2_Q0004` in the questionnaire score). However, these problems did not occur again during the remaining sessions.

Summarizing the list of images and sessions, each user segmented on average 2.7 images per session. One session took about 1.5 to 2 hours including the introduction, getting familiar with the tool, performing the segmentations and filling in the questionnaire. The chart in Figure 5.3

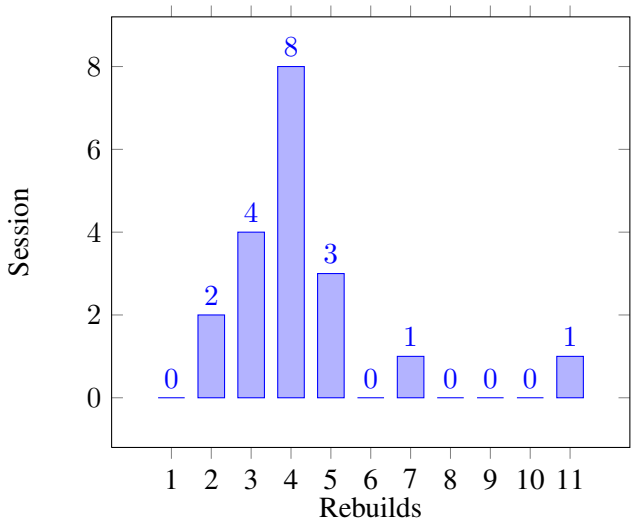



Figure 5.3: Number of rebuilds needed (x-axis) in order to produce a final image segmentation result. The y-axis shows the number of sessions matching the amount of rebuilds).

| U. | Session | Image | Inter. | Total | Ratio |
|---|---|---|---|---|---|
| 1 | 3ABE4 | 113016 | 01:37 | 08:22 | 0, 194 |
| 1 | 25662 | 135037 | 01:02 | 05:59 | 0, 175 |
| 1 | 452B2 | 108070 | 00:58 | 05:20 | 0, 183 |
| 2 | C3D2D | 374020 | 03:17 | 13:33 | 0, 243 |
| 2 | 78B48 | 113016 | 01:58 | 07:35 | 0, 259 |
| 4 | BD498 | 135037 | 04:59 | 13:45 | 0, 362 |
| 4 | 337D5 | 374020 | 03:50 | 15:06 | 0, 251 |
| 5 | 4C584 | 113016 | 01:34 | 07:39 | 0, 204 |
| 5 | 5DFB6 | 135037 | 04:50 | 12:29 | 0, 387 |
| 6 | 3ED15 | 135037 | 03:03 | 14:50 | 0, 205 |

| U. | Session | Image | Inter. | Total | Ratio |
|---|---|---|---|---|---|
| 6 | 12857 | 175032 | 01:51 | 20:00 | 0, 093 |
| 6 | B64DB | 374020 | 02:17 | 07:16 | 0, 314 |
| 8 | 2CDC2 | 113016 | 08:08 | 21:19 | 0, 381 |
| 8 | 54DFB | 108070 | 05:05 | 19:19 | 0, 263 |
| 9 | 2A4A6 | 113016 | 05:17 | 18:07 | 0, 292 |
| 9 | 5FAD6 | 175032 | 10:55 | 25:38 | 0, 426 |
| 9 | 494B3 | 374020 | 01:38 | 10:34 | 0, 155 |
| 9 | AD4D6 | 108070 | 05:39 | 08:49 | 0, 641 |
| 9 | FE38F | 135037 | 02:31 | 07:19 | 0, 345 |

Table 5.1: Interaction times (columns 'U.' = user, 'Inter.' = Interaction Time, minutes:seconds) recorded in each session, compared agains the total time required.

shows the amount of user-triggered rebuilds needed in order to produce a final segmentation in each session. This is $3, 4$ or $5$ partial recalculations from the common starting level of the irregular pyramid that were used in approximately $79\%$ of all cases.

The required time for creating the segmentations is visualized in Table 5.1, on average each session took 12:47 minutes. There the interaction time (selection of regions) is compared against the total time consumed. The total time includes the waiting time for the segmentation result which is generated by the external MST segmentation tool and navigating through the hierarchy, finding regions or thinking about the strategy. Time recording starts as soon as the initial stack of segmentations is loaded successfully and ends when the user is satisfied with the result. This ratio 'Interaction Time/Total Time' is on average $29.01\%$ and shown in Figure 5.4 (visualized by the red line). The

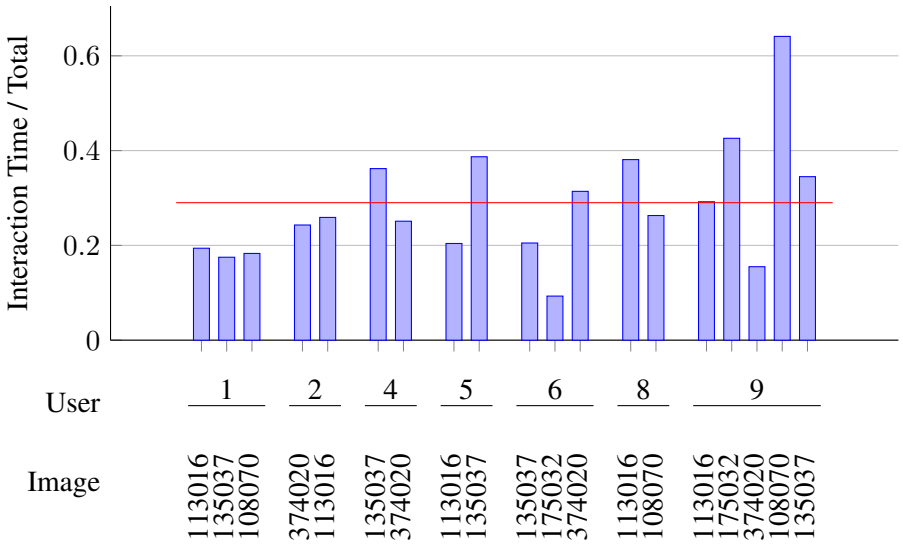



Figure 5.4: Normalized Interaction Time, grouped by user. Red line shows the average ratio.

| Operations/Height in % | $0-9$ | $10-19$ | $20-29$ | $30-39$ | $40-49$ | $50-59$ |
|---|---|---|---|---|---|---|
| $\text{imrg}_1(R_{(i,x_1)})$ | 4 | 4 | 5 | 5 | 9 | 6 |
| $\text{imrg}_1(\text{mrg}(R_{(i,x_1)}, R_{(i,x_2)}))$ | 663 | 168 | 984 | 1579 | 406 | 178 |
| $\text{imrg}_1(R_{(i,x_1)}) + \text{mrg}(\text{rem})$ | 0 | 0 | 0 | 0 | 0 | 2 |

| Operations/Height in % | $60-69$ | $70-70$ | $80-89$ | $90-99$ | 100 |
|---|---|---|---|---|---|
| $\text{imrg}_1(R_{(i,x_1)})$ | 3 | 1 | 8 | 0 | 13 |
| $\text{imrg}_1(\text{mrg}(R_{(i,x_1)}, R_{(i,x_2)}))$ | 49 | 9 | 2 | 5 | 177 |
| $\text{imrg}_1(R_{(i,x_1)}) + \text{mrg}(\text{rem})$ | 0 | 0 | 12 | 1 | 64 |

Table 5.2: Amount of operations in relation to the height of the pyramid. The columns show the corresponding intervals in % for the operations used (rows).

In total 4357 interactive operations have been placed in the GUI with the following distribution:

- $96,87\%$ (4220) belong to $\textit{imrg}_1(\textit{mrg}(R_{(i,x_1)}, R_{(i,x_2)}))$
- $1.81\%$ (79) to $\textit{imrg}_1(R_{(i,x_1)}) + \textit{mrg}(\textit{rem})$ and
- $1.32\%$ (58) to $\textit{imrg}_1(R_{(i,x_1)})$

Concluding, operation $\textit{imrg}_1(\textit{mrg}(R_{(i,x_1)}, R_{(i,x_2)}))$ is the one used most. There also the *brush mode* has been used for placing $3313\times$ this operation in a more efficient way, the remaining 1044 operations (and their corresponding regions) up to the total amount of 4357 were selected individually. What needs to be considered here is the fact that this operation – merge regions and inhibit resulting region from merging – takes two input regions, this is $77\%$ of all regions recorded which were selected using the brush. The other types of operations only require one argument and have a different effect in the merging tree. Nevertheless they are very important as well.

This is supported by the results summarized in the Table 5.2 showing the amount of operations used (rows) in relation to the height of the pyramid (columns, increasing from left to right). The $100\%$ column indicates that an operation was placed in the highest level to add a new level on top of the stack with the changes requested. Operation $\textit{imrg}_1(R_{(i,x_1)})$ appears to be used in all level of the stack to inhibit regions from merging in higher levels. Operation $\textit{imrg}_1(\textit{mrg}(R_{(i,x_1)}, R_{(i,x_2)}))$ has been applied extensively in lower levels (of high resolution) to merge over-segmented regions correctly together. The purpose of $\textit{imrg}_1(R_{(i,x_1)}) + \textit{mrg}(\textit{rem})$ is used to produce a clean foreground-background segmentation if there are a lot of (noisy) regions remaining in the topmost level. During the process of creating segmentations, we were able to observe different strategies. This enumeration claims not to be complete:

1. Use one specific level – where most of the details are available – as long as possible. Even if the region resolution is high (strong over-segmentation) and more operations are needed in the end.

2. Select details from arbitrary level and rebuild the stack of segmentations in order to preview the changes. This has been repeated again and again until the final segmentation result was acceptable to the user.

3. Use different levels and pick as much details (having different granularities) as possible. Then, in a second step, refine the result with regions from levels of higher resolution.

4. No explicit selection of the object of interest, but implicitly limit its boundary. Let the segmentation framework merge the regions, where no operations are defined, automatically.

However, the data recorded in the logs confirms these observations (see Figure 5.5 for reference). In the *first strategy* (session `2CDC2`) only regions from the same level $8$ (of high granularity) were chosen initially to fully outline the object of interest using $259$ operations. Even if there were bigger or good partitions in higher levels, they were not taken into consideration. Afterwards the remaining parts were merged together to the final segmentation. In the first run of `25662`, the user starts in the intermediate level $26$ (of $39$). Only in this level interactive operations were applied in the first run, then the pyramid is rebuilt to review the result. In an iterative manner details from other levels are chosen and combined to the final result. This matches the *second strategy*. The segmentation process of `3ED15` follows strategy three where a lot of regions of different granularity levels ($1, 6, 7, 8, 9, 12, 17, 28, 29$) are already selected in the first rebuild. However there is no preferred strategy or strategy enforced by the interactive segmentation tool. This question has also been asked by the participants, in general the users are free to follow whatever strategy they have in mind. If the interactive segmentation tool does not support this particular way, this should be addressed in the questionnaire.

| Rb. | Level | Operation | Amount |
|---|---|---|---|
| 1 | 8 | IMRG_MRG | 259 |
| 2 | 32 | IMRG_MRG | 27 |
| 3 | 34 | IMRG_MRG | 1 |
| 4 | 35 | IMRGMRG | 1 |

(a) Strategy in 2CDC2

| Rb. | Level | Operation | Amount |
|---|---|---|---|
| 1 | 26 | IMRG_MRG | 32 |
| 2 | 40 | IMRG_MRG | 1 |
| 3 | 13 | IMRG_MRG | 11 |
| 3 | 43 | IMRG | 1 |
| 4 | 48 | IMRG_MRG | 2 |
| 5 | 49 | IMRGMRG | 1 |

(b) Strategy in 25662

| Rb. | Level | Operation | Amount |
|---|---|---|---|
| 1 | 1 | IMRG_MRG | 3 |
| 1 | 6 | IMRG_MRG | 1 |
| 1 | 7 | IMRG | 4 |
| 1 | 7 | IMRG_MRG | 13 |
| 1 | 8 | IMRG | 3 |
| 1 | 8 | IMRG_MRG | 8 |
| 1 | 9 | IMRG_MRG | 1 |
| 1 | 12 | IMRG_MRG | 14 |
| 1 | 17 | IMRG_MRG | 2 |
| 1 | 28 | IMRG_MRG | 7 |
| 1 | 29 | IMRG_MRG | 1 |
| 2 | 45 | IMRGMRG | 12 |
| 3 | 46 | IMRG_MRG | 11 |
| 4 | 47 | IMRG_MRG | 1 |

(c) Strategy in 3ED15

Figure 5.5: Different segmentation strategies.

The interpretation of the questionnaire (Section B.1) shows that the users rate the usability of the tool on average as good (school grade), this goes together with the grade of satisfaction with the final segmentation result. The candidates – having no image-segmentation skills prior to this evaluation - provide also motivating feedback and agree that the tool may be operated by beginners with the amount of preliminary information provided. However, 7 out of 8 users feel that their skills in using the interactive segmentation tool and creating segmentations improved during the sessions. On a scale from 1 (easy) to 5 (hard), the difficulty of this process has been rated on average with 2.13. The working experience on coarse grained levels has been rated better than on fine-grained levels. From the feedback provided, this turns out to be a point where still improvement is needed. The *brush mode* helps to save clicks and supports the user in selecting regions more efficiently, whereas on levels of high resolution the human operator needs to be precise. This clashes to the character self-assessment of some participants of being impatient and 'clumsy' when it comes to operating on pixel-level. Here we run into a limitation of the tool which is caused by external constraints. Due to the embedding of interactive functionality into the external segmentation tool – collecting modification operations instead of applying them immediately – the changes are here only visible after rebuilding the pyramid. As it turned out the users expect the interactive operations on regions to be transitive. They intend to merge the regions number one with two and its result with region three (which is probably on the other end), this will cause an error. In this implementation we required the regions to be adjacent and selected in pairs of two. These regions are only marked for merging but not yet merged, therefore finding the adjacent region on pixel level can be very hard, especially when the colors of the neighboring segments are very similar to each other.

In a different approach like [Meine et al., 2004] this is not a problem as the region borders may be addressed and modified directly due to the explicit crack-edge representation of inter-pixel boundaries. In the GUI (used for region selection) of the interactive extension we only have the region information available as it is used for region/operation selection only. In the external segmentation tool this translation to edges to be modified is finally performed. The zoom functionality provided and the possibility to replace the colors assigned to the operations appear not to be sufficient in this particular case. One of the candidates suggested to implement advanced input methods like drawing shapes (rectangle, circle) to limit a certain object. This extension and the embedding of related functionality like the active contour [Kass et al., 1988] or Live Wire [Mortensen and Barrett, 1998] approach has to be postponed as future task.

Regarding the usability of the tool, the users added two more remarks. In order to reduce the mouse-clicks, an 'auto-set' functionality should be implemented. This way an operation is started implicitly once it has been selected, during evaluation phase the users sometimes forgot to start the operation recording mode. In addition, if no region is marked for inhibition from merging in the highest level, a warning should be displayed. This is because previously segmented regions in the final segmentation may be lost if not excluded explicitly from the process. Both aspects could be implemented with little effort. Disregarding these improvements, most of the users agree or even strongly agree that the tool provides all functionality for the purpose of creating image segmentations. At this point it is important to mention that not all responses given have been discussed. For detailed reference please see the score of the questionnaire attached to Section B.1.

CHAPTER 6

# Conclusion

According to the current problems shown in Chapter 1 and the assignment of tasks in Chapter 4, the approach of interactively modifying an irregular segmentation pyramid by defining merge and inhibit from merging operations on regions has been introduced. As it turned out, the interaction of the computational, visualization and data-representation part is very important during this process. The combinatorial map pyramid used in this work satisfies all the requirements without problems. The duality concept – region and boundary graphs are encoded in a single structure – is a key feature as well as the parent-child information of the merging tree.

This information – among the transformation from regions selected to edges to be processed – is required to guarantee consistency of the operations (within the structural representation) due to a possible mutual overlap of the corresponding regions' receptive fields. In conclusion, such complex situations require additional knowledge provided by the user as these conflicts cannot be resolved automatically by the framework. This supports the decision for a semi-automatic approach. Furthermore we require a data-structure at the base encoding all the information we need for processing. This applies to the calculation of the common starting level as well as the propagation of operations in the merging tree upwards.

Both hierarchical image segmentation and annotation requirements were considered, the user can select and combine regions from different levels of granularity. In contrast to other approaches we can access details on pixel-level as well and are able to produce any segmentation result. To be more precise, a stack of segmentations with the final image segmentation result in the topmost level. This requires advanced input methods like the brush mode. From the feedback provided in the questionnaire it shows that the functionality implemented is not yet optimal with respect to usability. Future developments need to concentrate especially on methods supporting the correction of boundaries that were not detected correctly in the beginning.

However, in opposite to going down to pixel-level precision, the user can start over again in the topmost level of the hierarchy and create custom levels of abstraction. The interactive extension developed supports different scenarios and does not impose certain strategies to be followed. In case a user cannot go his own way when creating segmentations, this needs to be addressed. The data recorded during the segmentation sessions show variations in the

levels/operations used, but reveals also the purpose of an operation during the process of creating a new hierarchy.

We successfully embedded interactive functionality into a previously fully automatic approach, without restricting the flexibility of the algorithm in merging other non selected (or not in the focus) regions autonomously. This is supported also by the promising results shown by the evaluation part. Without deep technical background knowledge, the candidates (beginners) were able to create image segmentations that satisfy their expectations. This goes together with the usability of the tool rated on average as good (school grade). However, the additional comments provided reveal limitations of the tool, some of them already have been discussed in Section 5.2. Apart from thinking about alternate input methods, the memory consumption and overall performance of the tool developed – especially when working with input images having high resolution – should be considered as a future task. From computational point of view, the structure encoded by combinatorial maps can be represented in several ways [Simon et al., 2006]. At the moment each level is represented explicitly and there is no need to have more than one segmentation level at a particular point time. The flat-pyramid concept of [Zankl, 2010] affects the memory consumption of the tool in a positive manner, however, the region information in between these levels is lost. This turns out to be a problem on segmenting thin structures.

During development of the interactive hierarchical segmentation tool using irregular pyramids, severals collaborations with other students at PRIP and the research group of Luc Brun (ENSICAEN, France) were established. Moreover, other colleagues already started to use the tool for their scientific work. Intermediate results during the origin of this diploma thesis have been published at the 8th IAPR TC-15 workshop on Graph-based Representations in Pattern Recognition [Gerstmayer et al., 2011].

APPENDIX A

# Interactive Segmentation Tool

The main purpose of this appendix is to provide an overview of the interactive segmentation tool. It contains an installation guide and a tutorial providing a step-by-step introduction.

## A.1 Installation Guide

### General information on the source code

Putting everything into numbers, this implementation of the interactive part overall counts approximately 6070 LOC[1] code including comment tags for better understanding. These comments can be used to create a documentation using Javadoc and Doxygen. Overall 5540 LOC belong to the User Interface written in Java programming language. This part contains the methods needed for visualizing the stack of segmentations created by the automatic method. On top of this, the user is able to record the set of interactive modification operations defined in Chapter 4. Furthermore this parts implements one phase of the conflict processing approach described in Section 4.3.

The data-structures to hold the segmentation information, I/O methods and basic image processing functionality are packed into a core-library belonging to the Java interface part. It may be re-used by other developers easily. As the two components (interactive extension and automatic segmentation framework) are designed to exchange data (see Figure 4.1) a common interface from the GUI to the command line segmentation tool had to be defined and implemented as well.

The remaining 630 LOC contain the methods to make the automatic segmentation framework (written in `C++` programming language) capable of processing this list of interactive operations. As a prerequisite for this functionality it is required to provide methods which enable the tool to serialize segmentation data (Section A.1) created by preceding runs. This way the segmentation-process can be restarted at any time and level.

[1] Lines of code

### Segmentation Framework (D,R)

The files located at `<drive:>/Code/mstpyr-merged` on the DVD-media provided, contain the Minimum Spanning Tree based Segmentation Framework modified for the use with interactive operations. This branch is merged with the one developed by [Zankl, 2010]. Regarding the environment where the tool can be installed, it is necessary to discern between the dependencies which are required to compile the source code (indicated using **D**evelopment) and the ones to run the compiled tools (indicated using **R**untime). This is necessary because most probably not all users want to modify the source code. However there are several operating-system dependencies which need to be satisfied in order to compile it successfully. For Linux based operating systems you need to install the following packages:

| **Ubuntu** $\geq 9.x$ | **CentOS** $\geq 5.x$ |
|---|---|
| xutils-dev (makedepend)<br>build-essential | Development Tools |
| gcc, g++, cpp, libstdc++6 $\geq$ version 4.4 | |
| libncurses5-dev<br>x11proto-*-dev<br>xserver-xorg-dev | ncurses-devel<br>xorg-x11-proto-devel |

For compilation, `GCC/G++` version 4.4 has to be used at least. Prior versions (like 4.1) will fail to compile the sources as certain language definitions and template arguments are not yet supported in these releases. In order to run the Linux binary, it has to be ensured that the same version of `libstdc++` as on the compilation system is installed. This is necessary because `libstdc++.so.6` needs to satisfy the dependencies of `GLIBCXX`: 3.4.9 and 3.4.11 within the library. When using `GCC/G++` version 4.4, this condition is fulfilled[2]. When using Microsoft Visual Studio ($\geq$ version 2003) no additional system libraries need to be provided.

#### Features Extension (D, R)

In order to use the features extension of the automatic segmentation framework developed by [Zankl, 2010] the following packages are required for compilation on Linux:

- `libblas, liblapack`: these dev-packages contain the libraries in order to use basic linear algebra routines in `C`-programming
- `lapackpp` [3]: library for high performance linear algebra
- `fortran` compiler: any version

These shared libraries need to be available on every PC running the tool, unfortunately it is not possible to link them statically to the MST segmentation binary. For Windows similar restrictions apply, therefore only a limited set of features is available here.

[2] `http://gcc.gnu.org/onlinedocs/libstdc++/manual/abi.html`
[3] The source to be compiled can be downloaded at: `http://lapackpp.sourceforge.net/`

**Boost Serialization (D)**

In this tool, the Boost[4] serialization library is used to write the pyramid data to the file system, i.e. combinatorial maps and the merging tree information are exported this way in order to use them again later. Boost provides free peer-reviewed and portable C++ source libraries, implementing best-practice methods. A major benefit of this approach is that several file containers are supported[5]. For performance reasons the combinatorial map is exported in text-format (non-human readable)[6], the level-to-level index changes are exported for interoperability reasons as XML. This way the latter information can easily be imported into programs based on Java, like our Interactive GUI. These libraries are available for all common operating systems and development environments. On Linux, precompiled Boost libraries are usually available in the distribution-repositories. The * (asterisk) is a placeholder for the version number which should be not lower than 1.39:

- libboost*-dev
- libboost-serialization*
- libboost-serialization*-dev

However these libraries are statically linked to the segmentation tool binary, therefore on the **R**untime-system it is not necessary to satisfy these dependencies. All files needed for compilation are shipped with the segmentation tool sources (header files and libraries are located at `<drive:>/Code/mstpyr-merged/boost`, therefore nothing has to be done here. A precompiled version of the Windows static runtime library is available[7], for Visual Studio 2010 all related files are provided on the media as well.

## User Interface

The files located at `<drive:>/Code/msti` on the DVD-media provided contain the interactive GUI to the Minimum Spanning Tree based Segmentation Framework called *MSTi*. In order to build and run the Java-based User Interface, Oracle Runtime Environment 1.6 together with Netbeans IDE should be used. It is important to use exactly this proprietary JVM, as other open source alternatives were – at the time the program was developed – not able to deal with the huge amount of data processed. This caused non-catchable runtime exceptions in the JRE itself which severely affect the stability of the tool. In order to run the interactive segmentation tool please ensure that the JRE is able to allocate sufficient heap memory, otherwise an "Out of Memory" exception is thrown. Hence the following command can be used to start the tool and providing sufficient heap memory:

```
java -Xmn100m -Xms512m -Xmx1024m -jar MSTi.jar
```

[4] Boost Framework: `http://www.boost.org/`
[5] Manual: `http://www.boost.org/doc/libs/1_45_0/libs/serialization/doc/`
[6] In order to produce a human-readable output use the `Dump()` method defined for each combinatorial map.
[7] Boost Windows Libraries: `http://www.boostpro.com/download/`

Doing it this way initially 512 MB (`Xms`) are allocated. Each time more memory is required, this is done in 100 MB steps (`Xmn`) until the maximum heap size of of 1024 MB. This is visualized in Figure A.1.

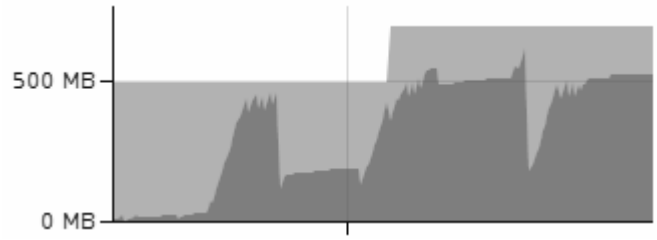



Figure A.1: History of heap memory allocated for a stack of segmentations (image size $500 \times 340$, 35 levels) plus related merging tree information and other data required for processing. The rising slope ($3\times$) corresponds to the initial load of the stack, on rebuild the previously used memory (y-axis) is released by the garbage collector.

In order to call the external MST pyramid segmentation tool, it has to be copied to the root folder of `MSTi` with the execute permission set on the binary.

**Custom and External Class Libraries (D)**

In order to handle/import the data created by the MST Pyramid based Segmentation Framework, a custom java package named *MSTCore* has been implemented. It is located at `<drive:>/Code/MSTcore` on the DVD-media provided and contains all necessary operations dealing with:

- Segmentation data I/O (implementing the interface to the automatic framework)
- Methods related to image segmentation
- Interactive operations and parsing of the merging-tree
- Conflict processing

These methods are documented properly within the code using Javadoc and can be re-used easily. The external libraries required in this Java project are:

- **JAI:** Java Advanced Imaging API, for advanced image manipulation[8]
- **Xerces:** Apache XML libraries, to read/write XML documents[9] like the level to level index changes serialized by the automatic segmentation framework (using Boost)
- **log4j:** Apache Logging[10]

All these additional packages are contained in the `lib/` subdirectory of the *MSTi* tool.

[8] `http://java.sun.com/javase/technologies/desktop/media/jai/`
[9] `http://xerces.apache.org/`
[10] `http://logging.apache.org/log4j/1.2/`

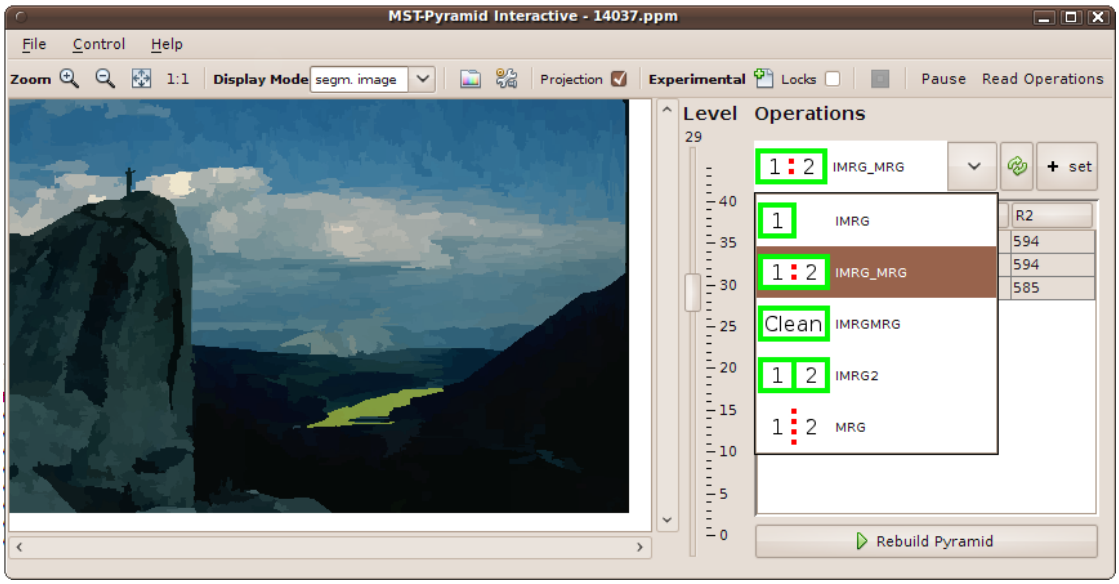



Figure A.2: User Interface, containing: the menu bar, toolbar and main panel. The latter is divided vertically into the image area, level chooser and operations panel.

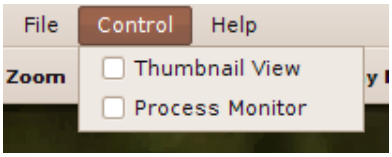
## A.2 Tutorial

### Interactive Image Segmentation Tool

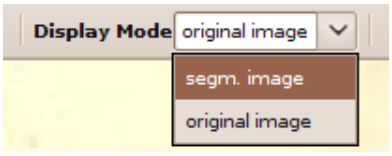
To start the process of interactively modifying a stack of segmentations execute `MSTi.sh` (Linux) or `MSTi.bat` (Windows) given as a supplement on the DVD. This shell script calls the tool with the JVM-arguments presented in the previous Subsection A.1. In the *File* menu, the user is asked to either create an initial/new or load an existing segmentation of an image. In both cases, the image file needs to be in the *Netpbm*[11] format, preferred is `.ppm`. You can easily convert existing files using ImageMagic[12] under (Linux) or Irfan View/Gimp on (Windows). However, in the first option the MST segmentation binary is called to create the stack of segmentations. Finally a file ending with `.sil` has to be provided, it contains the list of the segmentation image file names representing the stack. Once the stack of segmentations is loaded, the screen looks like in Figure A.2.

In the *Control* section of the menu-bar (Figure A.3a) a thumbnail view indicating the zoom area can be displayed (see Figure A.3c). Furthermore it is possible to switch to the process monitor which shows the output of the MST segmentation binary. For a sample output see the Listings 3.1, 3.2 or 4.2. The toolbar in Figure A.3d contains the sections described in the following.

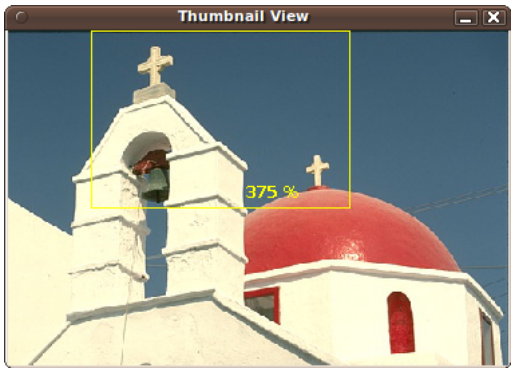
[11] `http://www.fileformat.info/format/pbm/egff.htm`
[12] Sample convert command: `convert testimage.jpg testimage.ppm`




(a) Control menu



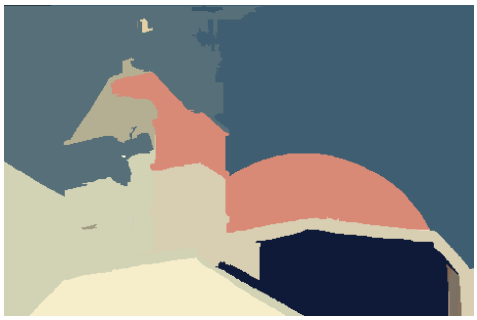

(b) Display modes




(c) Thumbnail view



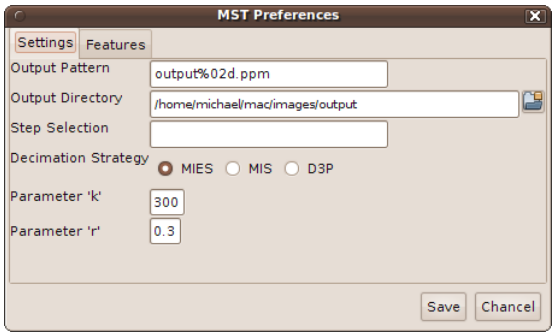

(d) MSTi application toolbar



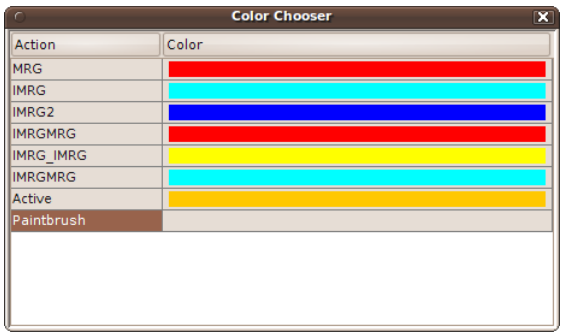
(e) Segmentation image



(f) Original image with labeling information




(g) Preferences Panel




(h) Color Chooser

Figure A.3: Controls and views in the Graphical User Interface.

**Zoom:** in order to work on details of high granularity, it is possible to enlarge the image display using zoom functionality (+, −, fit to screen, original size). If the mouse cursor is placed in the image area it is sufficient to scroll the mouse wheel to zoom in/out.

**Display Mode:** by default the segmentation image of the level selected is displayed (Figure A.3e). However it might be necessary to take a look on the input image again to match the segments with its correspondents in the highest level of detail. Therefore the user can switch to a display mode where the input image is visualized with the overlying labeling information of the corresponding level (Figure A.3f). The dedicated display mode chosen in the drop-down box (see Figure A.3b) is active for all levels in the hierarchy.

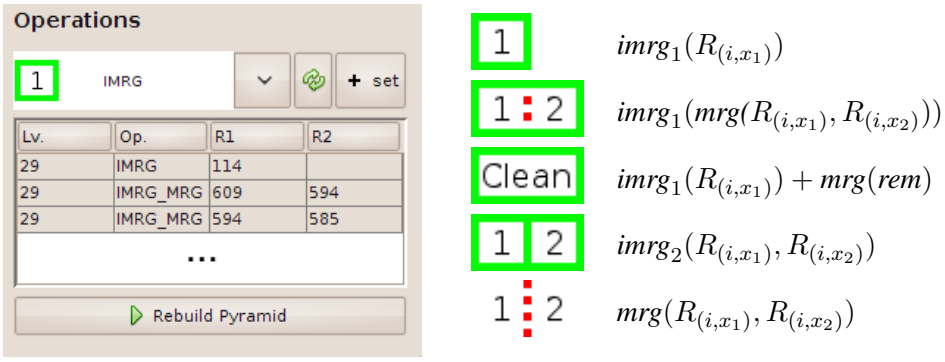
**Color Chooser:** to change the colors assigned to the operations the user can open at the color chooser panel (Figure A.3h ). If the color of a certain operation is similar to the one dominating in the image or region, it may be hard to detect which regions already have been processed and this consequently can be changed.

**Preferences:** to set others than the default arguments (RGB Int/Ext-Contrast feature, file/directory-names) passed on to the pyramid segmentation framework, click the options icon in the toolbar. In the preferences panel (Figure A.3g) one can set the same arguments as in the console-version of the segmentation framework. Since this version of the pyramid segmentation tool contains an extended feature-set (see [Zankl, 2010] for reference), a dedicated settings panel has been implemented for this extension as well.

**Projection and preview functionality:** using this checkbox, the visualization of the 'effect of operations' (preview functionality) can be turned on/off. This way the affected regions within the receptive field of an operation are colored or not. Note that only the operations placed in a level higher or equal than the current working level are displayed. Regions of operations placed in lower levels are not indicated here as they do not yet have a corresponding region in higher levels. This is because the result of merging or inhibition from merging operations only is available after a rebuild of the pyramid. Only then these regions are processed and the operation propagated to a higher level in the pyramid.

**Locks:** in this mode it is possible to create a custom hierarchy of segmentations on top of the stack below. Starting from the topmost level which contains the final segmentation, regions can be merged together again iteratively to create new levels of abstraction. By double clicking on the *Locks* label, these levels can be added to the new hierarchy. Note that it is no longer possible to select regions from lower levels than the new (topmost) starting level. To switch to the new hierarchy activate the corresponding checkbox and the pyramid level-slider range will change accordingly.

**Read Operations:** If you want to re-apply a list of interactive-operations already created by a previous run on the initial segmentation, you can load the operation history file having the pattern `<output_pattern>.ppm.actionlist_<timestamp>` into the tool again. This is the file which has been passed on to the segmentation tool at an earlier point of time. However this feature is experimental and no validity check is performed, therefore we strongly recommend to apply these files in the correct order.




(a) Panel

(b) Operations

Figure A.4: Operations Panel.

## Record Operations

To start recording an operation, first select an interactive operation in the drop-down menu shown in Figure A.4a. For a detailed explanation of the operations please refer to Section 4.2. Then click on the + set button, now the 'operation recording mode' is enabled. Consequently no other tasks can be performed in between, so most other GUI controls are locked as long as the operation is not complete. An operation is complete as soon as all regions it takes as argument – which is either one or two – are selected. This cardinality is given also in the icon assigned. So for each region has to be selected individually, of course they need to be adjacent as well. The recorded operation is finally added to the list shown in the center of Figure A.4a. Operations may also be deleted again from this list using the [DEL] key on the keyboard. A possible strategy here is to first find the ID of the region affected in the image and then to match this ID with the regions recorded in the list. In general – when the mouse cursor is moving over the image – the status panel in the bottom of the windows displays the labeling information. This is the current level, the region count in this level and the ID of the region where the mouse cursor is pointing at:

```
Lv.:  29 | Region#:  721 | Current:673
```

This region is also highlighted using the inverse color. For the purpose of recording operations – and their corresponding regions – in a more efficient way, a *multiple apply* switch has been implemented. The toggle button turns from green (inactive) to red once this mode is activated (caution!). Note that it has to be set before the recording is started. In case an operation takes two regions as input, the second one is automatically passed on as the first argument to the next operation. If an operation has only one region as argument, for each region selected an operation is automatically recorded. If the *multiple apply* switch is on, this information is displayed in the status bar as well. The operation recording mode has to be aborted manually using the [Esc] key.

Following the active-paintbrush method [Van Leemput et al., 1998], where it is sufficient to paint over region boundaries to initiate merging, a similar approach has been implemented. As this framework lacks the explicit representation of boundaries in the segmentation image, we

require two adjacent regions for the input. Once the *operations recording mode* is started using *multiple apply*, it is sufficient to double click on the first region to enable the *brush mode*. There the mouse pointer style switches to a crosshair cursor and it is no longer required to click on each region, brushing over a region is enough to record it (see Figure 4.5b). A double click on the last region ends the *brush mode* again.

In contrast to before when a double click activated the *brush mode* during operations' recording, a double click on a region in *normal* mode where the user can travel through the hierarchy starts the *explore mode*. This mode has already been addressed in Section 4.2. In short, with this method it is possible to divide a region into its components using parent-child relationship and to restore a particular detail. This region which is divided into its sub-components is called *explored region* and is indicated using a red boundary. In order to go to a different resolution level in this particular region, you need to press the [ALT] key and scroll the mouse-wheel up/down. The status panel now displays the region information of two levels, the current one and the one within the explored region. In this mode the list of operations available is extended by the *combined operations*. To leave the *explore mode* double click again the explored region.

Note that the operations recorded are not applied immediately. You collect a set of operations and after pushing the *Rebuild Pyramid* button the stack of segmentations is recalculated. If there are any problems or exceptions raised during this process, you can stop it manually using the button. The interactive segmentation tool and its data in the output directory chosen is then in an inconsistent state. This requires a manual cleanup of the segmentation files.

### Logging

To evaluate the created image segmentation, the tool automatically collects statistical information like application/segmentation specific events. See Listing A.1 below for an example:

```
INFO 2012-12-18 21:27:29,468 msti.MSTiApp - MSTi started
INFO 2012-12-18 21:27:46,633 msti.MSTiView - <user>: MST Pyramid started
INFO 2012-12-18 21:28:25,548 msti.MSTiView - <user>: image '14037.ppm' loaded
INFO 2012-12-18 21:28:25,549 msti.MSTiView - Level count: 44
INFO 2012-12-18 21:28:25,549 msti.MSTiView - Regions left: 5
INFO 2012-12-18 21:28:42,487 msti.MSTiView - Pause started
INFO 2012-12-18 21:29:00,751 msti.MSTiView - Pause ended after: 18.263 seconds
INFO 2012-12-18 21:29:19,645 msti.MSTiView - <user>: Rebuild triggered
INFO 2012-12-18 21:29:19,656 msti.MSTiView - <user>: MST Pyramid started
INFO 2012-12-18 21:29:32,208 msti.MSTiView - <user>: image '14037.ppm' loaded
INFO 2012-12-18 21:29:32,209 msti.MSTiView - Level count: 46
INFO 2012-12-18 21:29:32,209 msti.MSTiView - Regions left: 6
```

Listing A.1: MSTi Application Log (msti.log).

The `msti.log` contains information about the images loaded and the time a rebuild has been triggered. Once a stack of segmentation has been loaded successfully, the index of the highest level and its count of remaining regions will be added to the log file. This way it is possible to track the segmentation progress which corresponds to the amount of segments in the topmost level. During segmentation the user may want to take a break, therefore a Pause button has been added. Obviously we have to count the time spent doing nothing because this has also an effect on the evaluation result.

To get a more detailed overview of the operations performed, each time a rebuild is triggered, the collected statistical information is written to an XML file named `<imagename>.ppm.operationslog`. There we count – among others – the clicks and time consumed (in milliseconds) between the start of the operation and the time it is complete. As an example, see the statistical information belonging to the list of operations passed on to the segmentation framework in Listing A.2. The rebuild was triggered at 2012-12-18 21:29:19 due to `msti.log`, this matches also the value of the *rebuildStart* attribute in the root node.

```
<?xml version="1.0" encoding="UTF-8"?>
<operations imageName="14037.ppm" rebuildStart="20121218_212919">
   <operation action="IMRG" level="29">
      <regions>
         <region1>585</region1>
      </regions>
      <time>863</time>
      <clicks inputMethod="selection">1</clicks>
   </operation>
</operations>
```

Listing A.2: MSTi Statistical Information.

As this list can be quite long, we programmed a tool called `MSTiEval` which aggregates the information recorded and finally presents a list of key figures. It is located at `<drive:>/msti` on the DVD-media provided and can be started using the command `java -jar MSTiEval <imagename>.ppm.operationslog`.

APPENDIX B

# Evaluation Documents

This appendix contains the detailed results of the evaluation questionnaire and the declaration of consent. There the performance-data recorded during the sessions and its usage is listed explicitly. Due to data-protection reasons – the participation was anonymous and the declaration of consent contains a handwritten signature – only the form itself is attached. The signed documents will be provided upon request.

## B.1 Questionare Score

| | |
|---|---|
| Number of records in this query | 8 |
| Total records in survey | 8 |
| Percentage of total | 100,00 % |

### G1, Basic Information

**Q0001, Please select your gender:**

| Answer | Count | Percentage |
|---|---|---|
| Female (F) | 4 | 50,00 % |
| Male (M) | 4 | 50,00 % |

### G2, Image Segmentations

**Q0001, Enter the image ID of the first image segmentation run:**

| Answer | Count | Percentage |
|---|---|---|
| Answer | 8 | 100,00 % |
| No Answer | 0 | 0,00 % |

*Responses (ID, Session) added to comments in Q0002.*

**Q0002, How satisfied are you with the first image segmentation result?**

| Answer | Count | Percentage |
|---|---|---|
| (1) Excellent (A1) | 2 | 25,00 % |
| (2) Good (A2) | 5 | 62,50 % |
| (3) Satisfactory (A3) | 0 | 0,00 % |
| (4) Sufficient (A4) | 0 | 0,00 % |
| (5) Insufficent (A5) | 1 | 12,50 % |
| No rating (A6) | 0 | 0,00 % |
| Comments | 8 | 100,00 % |
| No answer | 0 | 0,00 % |

**If you are not satisfied, none of the options fits to your result or you just want to provide additional feedback, please use the comment box below:**

| ID | Session | Response |
|---|---|---|
| 1 | 25662 | Select Bird as foreground |
| 2 | 78B48 | wanted the horses to remain |
| 3 | 6C6DA | Photo of two horses: tried to keep the main contours just to the point where still can be determined that there are horses.Two different approaches: left horse -> merge regions in horse, right horse -> merge regions outside horse. Had some handling issues (rebuild twice without 'saving') regions. Tool got frozen in the end, so no satisfactory result. |
| 4 | BD498 | Vorhaben war, den Vogel möglichst genau freizustellen. Bei der Herausarbeitung der Flugfedern an der Flügelspitze war ich mit dem Endergebnis nicht vollends zufrieden, es war aber schlussendlich ausreichend. Positiv aufgefallen ist mir, dass die Trennung des Kopfes vom (ebenfalls weißen) Hintergrund gut funktioniert hat. Die Pinselfunktion empfand ich in diesem Fall als praktisch. Dass gewohnte, fast unbewusst verwendete |
| 5 | 4C584 | Tastenkombinationen funktionieren ist mir ebenfalls positiv aufgefallen (Mehrfachmarkierung in der Liste mittels Strg+Klick, Löschen mittels Entf, Beenden einer Mehrfachmarkierung mit Esc) |
| 6 | 3ED15 | Adler: hat gut funktioniert, Auswahl im feineren Pixelbereich etwas zeitaufwändig. |
| 8 | 2CDC2 | Fokus auf das Fohlen, auf Details geachtet |
| 9 | 2A4A6 | I wanted to get the mare and the foal and the pallor of both |

### Q0003, Enter the image ID of the second image segmentation run:

| Answer | Count | Percentage |
|---|---|---|
| Answer | 8 | 100,00 % |
| No Answer | 0 | 0,00 % |

*Responses (ID, Session) added to comments in Q0004.*

### Q0004, How satisfied are you with the second image segmentation result?

| Answer | Count | Percentage |
|---|---|---|
| (1) Excellent (A1) | 2 | 25,00 % |
| (2) Good (A2) | 2 | 25,00 % |
| (3) Satisfactory (A3) | 2 | 25,00 % |
| (4) Sufficient (A4) | 1 | 12,50 % |
| (5) Insufficent (A5) | 1 | 12,50 % |

| Answer | Count | Percentage |
|---|---|---|
| No rating (A6) | 0 | 0,00 % |
| Comments | 8 | 100,00 % |
| No answer | 0 | 0,00 % |

**If you are not satisfied, none of the options fits to your result or you just want to provide additional feedback, please use the comment box below:**

| ID | Session | Response |
|---|---|---|
| 1 | 3ABE4 | focus on the little horse in the foreground |
| 2 | C3D2D | set focus on different layers, difficult because of too similar elements. |
| 3 | 5814C | Tried to follow the snake's contour in brush mode, but had problems with this mode ('selected region is not a neighbour' when double click on a region) |
| 4 | 337D5 | Mein Ziel war es, das Bild grob in 4 Entfernungsintervallen aufzuteilen. Wiederum habe ich gerne die Pinselfunktion verwendet. Beim zweiten rebuild habe ich den zuvor bereits herausgearbeiteten Himmel glücklicherweise gleich zu Beginn wieder als Region markiert. Ansonsten wäre diese arbeit verloren gewesen. Dass dies nicht (optional) automatisch geschieht finde ich schade. Das häufige Klicken auf den Set-Button empfand ich nach längerem Arbeiten als etwas nervig. Ein Auslösen durch eine Taste wäre mir lieber. |
| 5 | 5DFB6 | Haupt, Flügel inklusive Federn |
| 6 | 12857 | Ast/Schleiche: Da das Bild sehr viele kleine Details hat, habe ich mich für den Ast entschieden. Anfangs erschien mir es schwer doch mit der Brush-Funktion hat es dann relativ flüssig funktioniert. |
| 8 | 54DFB | Fokus auf Ast rechts, SEHR detailliert, am Ende einmal vergessen zu schützen |
| 9 | 494B3 | Separate the sky, the mountains, the field and the wall. First I wanted to get all the arcs from the wall but because it was so difficult I selected only the four big parts. |

### Q0005, Enter the image ID of the third image segmentation run:

| Answer | Count | Percentage |
|---|---|---|
| Answer | 3 | 37,50 % |
| No Answer | 5 | 62,50 % |

*Responses (ID, Session) added to comments in Q0006.*

### Q0006, How satisfied are you with the third image segmentation result?

| Answer | Count | Percentage |
|---|---|---|
| (1) Excellent (A1) | 1 | 12,50 % |

| Answer | Count | Percentage |
|---|---|---|
| (2) Good (A2) | 2 | 25,00 % |
| (3) Satisfactory (A3) | 0 | 0,00 % |
| (4) Sufficient (A4) | 0 | 0,00 % |
| (5) Insufficent (A5) | 0 | 0,00 % |
| No rating (A6) | 0 | 0,00 % |
| Comments | 3 | 37,50 % |
| No answer | 5 | 62,50 % |

**If you are not satisfied, none of the options fits to your result or you just want to provide additional feedback, please use the comment box below:**

| ID | Session | Response |
|---|---|---|
| 1 | 452B2 | focus on the head of the tiger |
| 6 | B64DE | Landschaft mit Aquädukt: Fokus auf die Grasflächen. Die Funktionen sitzen jetzt schon relativ gut. Daher konnte ich die Flächen schnell markieren/ausschneiden. |
| 9 | 5FAD6 | I selected the snake, the branch and the small part of the fir branch |

### Q0007, Enter the image ID of the fourth image segmentation run:

| Answer | Count | Percentage |
|---|---|---|
| Answer | 1 | 12,50 % |
| No Answer | 7 | 87,50 % |

*Responses (ID, Session) added to comments in Q0008.*

### Q0008, How satisfied are you with the fourth image segmentation result?

| Answer | Count | Percentage |
|---|---|---|
| (1) Excellent (A1) | 0 | 0,00 % |
| (2) Good (A2) | 1 | 12,50 % |
| (3) Satisfactory (A3) | 0 | 0,00 % |
| (4) Sufficient (A4) | 0 | 0,00 % |
| (5) Insufficent (A5) | 0 | 0,00 % |
| No rating (A6) | 0 | 0,00 % |
| Comments | 1 | 12,50 % |
| No answer | 7 | 87,50 % |

**If you are not satisfied, none of the options fits to your result or you just want to provide additional feedback, please use the comment box below:**

| ID | Session | Response |
|---|---|---|
| 9 | FE38F | Select the eagle |

### Q0009, Enter the image ID of the fifth segmentation run:

| Answer | Count | Percentage |
|---|---|---|
| Answer | 1 | 12,50 % |
| No Answer | 7 | 87,50 % |

*Responses (ID, Session) added to comments in Q00010.*

### Q0010, How satisfied are you with the fifth image segmentation result?

| Answer | Count | Percentage |
|---|---|---|
| (1) Excellent (A1) | 0 | 0,00 % |
| (2) Good (A2) | 0 | 12,50 % |
| (3) Satisfactory (A3) | 1 | 12,50 % |
| (4) Sufficient (A4) | 0 | 0,00 % |
| (5) Insufficent (A5) | 0 | 0,00 % |
| No rating (A6) | 0 | 0,00 % |
| Comments | 1 | 12,50 % |
| No answer | 7 | 87,50 % |

**If you are not satisfied, none of the options fits to your result or you just want to provide additional feedback, please use the comment box below:**

| ID | Session | Response |
|---|---|---|
| 9 | AD4D6 | Eigentlich wollte ich den Tiger einzeln auswählen und dann zusätzlich noch den Fluss. Da ich aber mit der Brushfunktion nur die Umrisse ausgewählt habe, sind der Tiger und der Fluss miteinander verschmolzen. Das war eigentlich nicht mein wirkliches Ziel, aber trotzdem zufriedenstellend, so dass ich es gelassen hab. Die Brush Funktion verleitet dazu, dadurch dass sie so schnell ist, dass man kleinen Fehler ignoriert und einfach weiter macht obwohl man eigentlich dadurch nicht mehr so genau genau arbeitet, wie man zu Beginn eigentlich wollte. |

## G3, Segmentation Tool

### Q0001, How do you rate the usability of the interactive segmentation tool?

| Answer | Count | Percentage |
|---|---|---|
| (1) Excellent (A1) | 1 | 12,50 % |
| (2) Good (A2) | 6 | 75,00 % |
| (3) Satisfactory (A3) | 1 | 12,50 % |
| (4) Sufficient (A4) | 0 | 0,00 % |
| (5) Insufficent (A5) | 0 | 0,00 % |
| No rating (A6) | 0 | 0,00 % |
| Comments | 6 | 71,00 % |
| No answer | 0 | 0,00 % |

| ID | Response |
|---|---|
| 1 | simple, logical, useful |
| 2 | key for automatic activation of the operation "set" |
| 3 | -> I tended to forget the 'set' button (some key on the keyboard would be easier for me),<br>-> multi-merging: non-transitive (only direct neighbours and not neighbours of already merged regions)<br>-> no 'preview' mode on coarser scales when operations done on fine scales: got me confused<br>-> when mistakes done: an easier identification of the operation done to a region would be helpful to define the operation from the list which has to be deleted |
| 4 | Klicken auf set vor jeder Operation störend. Ansonsten gut, dass die Tasten Entf, Strg usw. ihre gewohnte Funktionalität haben finde ich gut. |
| 8 | Immer dazwischen "set" zu drücken ist umständlich, weil man mit der Mouse immer vom Bild wegfahren muss |
| 9 | Der Doppelklick, um in die Brushfunktion zu gelangen, ist in einer kleinen Auflösung kaum auszuführen. Das Programm möchte meistens zwei Flächen miteinander verbinden und gibt dann die Fehlermeldung, dass es keine Nachbarn sind. Geht man in eine höhere Auflösung, wo die Flächen größer sind, ist es deutlich einfacher durch den Doppelklick in die Brushfunktion zu gelangen.<br>Die Brush Funktion verleitet dazu, dadurch dass sie so schnell ist, dass man kleinen Fehler ignoriert und einfach weiter macht obwohl man eigentlich dadurch nicht mehr so genau genau arbeitet, wie man zu Beginn eigentlich wollte.<br>In einer sehr kleinen Auflösung muss bei sehr kleinen Feldern nicht die Spitze des Mauszeigers auf die zu markierende Fläche zeigen, sondern ein anderer Teil des Zeigers. Intuitiv würde ich aber mit der Spitze den Bereich auswählen. |

**Q0002, The state of the interactive segmentation tool was always clear to me and the process of segmenting images under control.**

| Answer | Count | Percentage |
|---|---|---|
| (1) Strongly agree (A1) | 2 | 25,00 % |
| (2) Agree (A2) | 5 | 62,50 % |
| (3) Neither/nor (A3) | 1 | 12,50 % |
| (4) Disagree (A4) | 0 | 0,00 % |
| (5) Strongly disagree (A5) | 0 | 0,00 % |
| Comments | 4 | 50,00 % |
| No answer | 0 | 0,00 % |

**If this was not the case or you want to give additional feedback please use the comment box below:**

| ID | Response |
|---|---|
| 1 | it became better the more pictures i did, sometimes explanation needed (because of no technical background) |
| 3 | Switching between scales/ rebuilding, I was not always sure which operations I had already done and which were still to be done (preview-mode possible?) |
| 4 | Dass ich auf den unteren Fensterrand blicken musste, um festzustellen, ob ich mich noch in der Auswahl befand, war nicht optimal. Evtl. könnte sich zusätzlich zum Hinweis zum Drücken der Esc-Taste auch die Form des Cursors ändern. |
| 9 | Durch die Möglichkeit, wenn man Felder falsch markiert hat, diese wieder zu entfernen, ist eine große Möglichkeit der Kontrolle gegeben. |

**Q0003, How was your working experience on fine-grained levels within the stack of segmentations (like regions of small size, high resolution close to pixel level)?**

| Answer | Count | Percentage |
|---|---|---|
| (1) Excellent (A1) | 1 | 12,50 % |
| (2) Good (A2) | 3 | 37,50 % |
| (3) Satisfactory (A3) | 2 | 25,00 % |
| (4) Sufficient (A4) | 1 | 12,50 % |
| (5) Insufficent (A5) | 1 | 12,50 % |
| No rating (A6) | 0 | 0,00 % |
| Comments | 7 | 87,50 % |
| No answer | 0 | 0,00 % |

**If you have additional feedback please use the comment box below:**

| ID | Response |
|---|---|
| 1 | with zoom function no problem |
| 3 | I am too clumsy and impatient to for these fine scales, even in zoom ;-), Small bug in brush mode (mode not possible). |
| 4 | Da ich gerne mit dem Pinsel arbeitete, kam ich auf feinem Niveau nicht so gut zurecht, allerdings war das Arbeiten auf diesen nur selten notwendig. |
| 5 | Nächstes Segment bei Federn schwer zu finden |
| 6 | Als erstes schwer, aber man bekommt nach einiger zeit ein Gefühl dafür schneller und effizienter zu Arbeiten. |
| 8 | Auf diesem Level ist es eher schwierig, die Flächen mit der Mouse zu treffen; man tippt oft daneben. Dafür detailgenaue Ergebnisse! |
| 9 | Es war in diesem Level schwieriger in den Brush-Modus zu gelangen und auch die Verbindung der einzelnen Flächen war deutlich komplizierter, weil sehr häufig die Fehlermeldung kam, dass die beiden Regionen keine Nachbarn waren. |

**Q0004 How was your working experience on coarse-grained levels within the stack of segmentations (like regions of big size, low resolution close to the uppermost levels)?**

| Answer | Count | Percentage |
|---|---|---|
| (1) Excellent (A1) | 6 | 75,00 % |
| (2) Good (A2) | 1 | 12,50 % |
| (3) Satisfactory (A3) | 1 | 12,50 % |
| (4) Sufficient (A4) | 0 | 0,00 % |
| (5) Insufficent (A5) | 0 | 0,00 % |
| No rating (A6) | 0 | 0,00 % |
| Comments | 4 | 50,00 % |
| No answer | 0 | 0,00 % |

**If you have additional feedback please use the comment box below:**

| ID | Response |
|---|---|
| 1 | very easy to handle |
| 4 | Schnell, gut und präzise |
| 6 | sehr Gut |
| 8 | Funktioniert prima! |

**Q0005 The interactive-segmentation tool provided all functionality you wanted to use for the purpose of creating image segmentations.**

| Answer | Count | Percentage |
|---|---|---|
| (1) Strongly agree (A1) | 4 | 50,00 % |
| (2) Agree (A2) | 3 | 37,50 % |
| (3) Neither/nor (A3) | 1 | 12,50 % |
| (4) Disagree (A4) | 0 | 0,00 % |
| (5) Strongly disagree (A5) | 0 | 0,00 % |
| Comments | 3 | 37,50 % |
| No answer | 0 | 0,00 % |

**If not, please provide also a short description of the missing function(s) in the comment box below:**

| ID | Response |
|---|---|
| 3 | -> Splitting regions mode?<br>-> Preview mode?<br>-> Undo? |
| 6 | Interessant wären Funktionen wie einige Standardformen (Rechteck, Kreis, Dreieck, Trapez) un alle segmente innerhalb der Form automatisch zu Markieren oder Zusammenzufügen. |
| 9 | Aber es wäre komfortabeler, wenn die Brushfunktion auch über einen Auswahl-Menü-Punkt erreichbar wäre. |

**Q0006, Which amount of experience/knowledge is necessary from your point of view to operate the tool and create image-segmentations efficiently?**

| Answer | Count | Percentage |
|---|---|---|
| Few (Beginner) (A1) | 5 | 62,50 % |
| Medium (Intermediate) (A2) | 3 | 37,50 % |
| Detailed (Advanced) (A3) | 0 | 0,00 % |
| No answer | 0 | 0,00 % |

**Q0007, In which part of the the segmentation framework do you see a need for improvement?**

| Answer | Count | Percentage |
|---|---|---|
| Design of the Graphical User Interface (SQ001) | 1 | 12,50 % |
| (Operation) Input Methods (SQ002) | 4 | 50,00 % |
| Performance (SQ003) | 1 | 12,50 % |
| Initial Segmentation (SQ004) | 2 | 25,00 % |
| Other | 0 | 0,00 % |

## G4, Skills

**Q0001, How do you rate your image-segmentation skills prior to this evaluation?**

| Answer | Count | Percentage |
|---|---|---|
| Beginner (A1) | 1 | 87,50 % |
| Intermediate (A2) | 4 | 12,50 % |
| Advanced (A3) | 1 | 0,00 % |
| No answer | 0 | 0,00 % |
| Other | 0 | 0,00 % |

**Q0002, How difficult was it for you to create the image segmentations?**

| Answer | 1 (easy) | 2 | 3 | 4 | 5 (hard) | No Answer |
|---|---|---|---|---|---|---|
| Count | 1 | 6 | 0 | 1 | 0 | 0 |
| Percentage | 12,50 % | 75,00 % | 0,00 % | 12,50 % | 0,00 % | 0,00 % |

| | |
|---|---|
| Sum (Answers) | 8 |
| Arithmetic mean | 2,13 |
| Standard deviation | 0,83 |

**Q0003, Did you notice an improvement of your skills while using the interactive-segmentation tool and creating image segmentations throughout the evaluation process?**

| Answer | Yes (Y) | No (N) | No Answer |
|---|---|---|---|
| Count | 7 | 1 | 0 |
| Percentage | 87,50 % | 12,50 % | 0,00 % |

**Q0004, Sufficient preliminary information was provided at the beginning of the interactive segmentation session.**

| Answer | Count | Percentage |
|---|---|---|
| (1) Strongly agree (A1) | 7 | 87,50 % |
| (2) Agree (A2) | 1 | 12,50 % |
| (3) Neither/nor (A3) | 0 | 0,00 % |
| (4) Disagree (A4) | 0 | 0,00 % |
| (5) Strongly disagree (A5) | 0 | 0,00 % |
| Comments | 1 | 12,50 % |
| No answer | 0 | 0,00 % |

| ID | Response |
|---|---|
| 1 | good explanations with examples (also for beginners)!!! |

**Q0005, Did you feel ready for the segmentation exercises after the introduction phase?**

| Answer | Count | Percentage |
|---|---|---|
| (1) Strongly agree (A1) | 5 | 62,50 % |
| (2) Agree (A2) | 3 | 37,50 % |
| (3) Neither/nor (A3) | 0 | 0,00 % |
| (4) Disagree (A4) | 0 | 0,00 % |
| (5) Strongly disagree (A5) | 0 | 0,00 % |
| Comments | 0 | 0,00 % |
| No answer | 0 | 0,00 % |

**If you have additional feedback please use the comment box below:**
No feedback provided,

## G5, Other

**Q0001, If you wish to give some further feedback not yet discussed in the questionaire, please use the comment box.**

| Answer | Count | Percentage |
|---|---|---|
| Answer | 1 | 12,50 % |
| No answer | 7 | 87,50 % |

| ID | Response |
|---|---|
| 9 | Während der Brushfunktion im niedrigen Level ist die rote Linie eher hinderlich, da man schnell den zuletzt markierten Bereich verliert. |

## B.2 Declaration of Consent

### Einverständniserklärung

für die Teilnahme an der Evaluation im Rahmen der Diplomarbeit zum Thema:

„Interactive Hierarchical Image Segmentation
using Irregular Pyramids“

Die Teilnahme an der Evaluation erfolgt auf freiwilliger Basis. Die TeilnehmerInnen können zu jedem Zeitpunkt die aufgezeichneten Daten einsehen oder das Experiment beenden.

Die Evaluation besteht aus zwei Phasen:

1) **Segmentierung:** Während der Bildsegmentierung werden folgende Daten zusätzlich zu den erstellten Segmentierungen aufgezeichnet: Liste der verwendeten Interaktiven-Operationen, betroffene Regionen, Level in der Pyramide, Maus-Klicks, Zeit/Operation, Gesamtzeit, Selektionsmodus

2) **Fragebogen:** Jedem Bild wird - um einen Bezug zu den korrespondierenden Fragen in der abschließenden Umfrage herstellen zu können - eine zufällige ID zugewiesen. Außer dem Geschlecht (Quote 50:50 unter den Probanden) werden in der Umfrage keine anderen persönlichen Daten abgefragt.

In beiden Phasen ist weder während der Aufzeichnung noch nach der Auswertung ein Personenbezug herstellbar.

Hiermit erkläre ich mich einverstanden, dass die während der Evaluation aufgezeichneten Daten im Rahmen der Diplomarbeit und darauf aufbauenden Publikationen wissenschaftlich ausgewertet und veröffentlicht werden dürfen.

Ort, Datum Unterschrift

# Bibliography

[Arbelaez et al., 2009] Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2009). From contours to regions: An empirical evaluation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2294–2301.

[Bister et al., 1990] Bister, M., Cornelis, J., and Rosenfeld, A. (1990). A critical view of pyramid segmentation algorithms. *Pattern Recognition Letters*, 11(9):605–617.

[Boykov and Jolly, 2001] Boykov, Y. and Jolly, M. (2001). Interactive graph cuts for optimal boundary amp; region segmentation of objects in n-d images. In *Proceedings of the Eighth International Conference On Computer Vision*, volume 1, pages 105 –112.

[Braquelaire and Brun, 1998] Braquelaire, J. P. and Brun, L. (1998). Image segmentation with topological maps and inter-pixel representation. *Journal of Visual Communication and Image representation*, 9(1):62–79.

[Brickell and Clark, 1970] Brickell, F. and Clark, S. (1970). *Differentiable Manifolds*. VNR.

[Brun and Kropatsch, 1999] Brun, L. and Kropatsch, W. G. (1999). Dual contraction of combinatorial maps. Technical Report PRIP-TR-54, Institute of Computer Graphics and Algorithms 186/3, Pattern Recognition and Image Processing Group, TU Wien, Austria.

[Brun and Kropatsch, 2003a] Brun, L. and Kropatsch, W. G. (2003a). Combinatorial pyramids. In *IEEE International conference on Image Processing*, volume 2, pages 33–36.

[Brun and Kropatsch, 2003b] Brun, L. and Kropatsch, W. G. (2003b). Construction of combinatorial pyramids. In *Graph Based Representations in Pattern Recognition, 4th IAPR International Workshop, GbRPR 2003, York, UK, June 30 - July 2, 2003, Proceedings*, pages 1–12.

[Brun and Kropatsch, 2003c] Brun, L. and Kropatsch, W. G. (2003c). Contraction kernels and combinatorial maps. *Pattern Recognition Letters*, 24(8):1051–1057.

[Brun et al., 2004] Brun, L., Vautrot, P., and Meyer, F. (2004). Hierarchical watersheds with inter-pixel boundaries. In *Image Analysis and Recognition: International Conference*, volume 1, pages 840–847, Proto (Portugal). Springer Verlag Heidelberg (LNCS).

[Burt et al., 1981] Burt, P. J., Hong, T.-H., and Rosenfeld, A. (1981). Segmentation and estimation of image region properties through cooperative hierarchial computation. *Systems, Man and Cybernetics, IEEE Transactions on*, 11(12):802 –809.

[Comaniciu and Meer, 2002] Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619.

[Culhane and Tsotsos, 1990] Culhane, S. M. and Tsotsos, J. K. (1990). An attentional prototype for early vision. In *European Conference on Computer Vision*, pages 551–560.

[Diestel, 2010] Diestel, R. (2010). *Graph Theory*. Springer, Berlin, 4th ed., electronic edition.

[Egenhofer, 1989] Egenhofer, M. J. (1989). A formal definition of binary topological relationships. In *FODO*, pages 457–472.

[Felzenszwalb and Huttenlocher, 2004] Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181.

[Gelasca et al., 2007] Gelasca, E. D., De Guzman, J., Gauglitz, S., Ghosh, P., Xu, J., Moxley, E., Rahimi, A. M., Bi, Z., and Manjunath, B. S. (2007). Cortina: Searching a 10 million + images database. Technical report, University of California, Santa Barbara.

[Gerstmayer et al., 2011] Gerstmayer, M., Haxhimusa, Y., and Kropatsch, W. G. (2011). Hierarchical interactive image segmentation using irregular pyramids. In Jiang, X., Ferrero, M., and Torsello, A., editors, *Graph-Based Representations in Pattern Recognition - 8th IAPR-TC-15 International Workshop, GbRPR 2011, Münster, Germany, May 18-20, 2011. Proceedings*, volume 6658 of *Lecture Notes in Computer Science*, pages 245–254, Muenster, Germany. Springer, Berlin Heidelberg New York.

[Glantz and Kropatsch, 1994] Glantz, R. and Kropatsch, W. G. (1994). Building irregular pyramids by dual graph contraction. Technical Report PRIP-TR-35, Institute of Computer Graphics and Algorithms 186/3, Pattern Recognition and Image Processing Group, TU Wien, Austria.

[Haxhimusa, 2007] Haxhimusa, Y. (2007). *The Structurally Optimal Dual Graph Pyramid and its Application in Image Partitioning*, volume 308 of *Dissertationen zur Künstlichen Intelligenz (DISKI)*. IOS Press.

[Haxhimusa et al., 2003] Haxhimusa, Y., Glantz, R., and Kropatsch, W. G. (2003). Constructing stochastic pyramids by mides - maximal independent directed edge set. In *Graph Based Representations in Pattern Recognition, 4th IAPR International Workshop, GbRPR 2003, York, UK, June 30 - July 2, 2003, Proceedings*, pages 24–34.

[Haxhimusa et al., 2002] Haxhimusa, Y., Glantz, R., Saib, M., Langs, G., and Kropatsch, W. G. (2002). Logarithmic tapering graph pyramid. In *DAGM-Symposium*, pages 117–124.

[Haxhimusa et al., 2005] Haxhimusa, Y., Ion, A., Kropatsch, W. G., and Brun, L. (May 2005). Hierarchical image partitioning using combinatorial maps. In Chetverikov, D., Czuni, L., and Vincze, M., editors, *Proceedings of the Joint Hungarian-Austrian Conference on Image Processing and Pattern Recognition*, pages 179–186.

[Haxhimusa and Kropatsch, 2004] Haxhimusa, Y. and Kropatsch, W. G. (2004). Segmentation graph hierarchies. In *Proceedings of Joint Workshops on Structural, Syntactic, and Statistical Pattern Recognition*, pages 343–351.

[Heimann et al., 2009] Heimann, T., van Ginneken, B., Styner, M., Arzhaeva, Y., Aurich, V., Bauer, C., Beck, A., Becker, C., Beichel, R., Bekes, G., Bello, F., Binnig, G., Bischof, H., Bornik, A., Cashman, P., Chi, Y., Cordova, A., Dawant, B., Fidrich, M., Furst, J., Furukawa, D., Grenacher, L., Hornegger, J., Kainmuller, D., Kitney, R., Kobatake, H., Lamecker, H., Lange, T., Lee, J., Lennon, B., Li, R., Li, S., Meinzer, H.-P., Nemeth, G., Raicu, D., Rau, A.-M., van Rikxoort, E., Rousson, M., Rusko, L., Saddi, K., Schmidt, G., Seghers, D., Shimizu, A., Slagmolen, P., Sorantin, E., Soza, G., Susomboon, R., Waite, J., Wimmer, A., and Wolf, I. (2009). Comparison and evaluation of methods for liver segmentation from ct datasets. *IEEE Transactions on Medical Imaging*, 28(8):1251–1265.

[Hug, 2000] Hug, D.-I. J. M. (2000). *Semi-Automatic Segmentation of Medical Imagery*. PhD thesis, Swiss Federal Institute of Technology Zürich.

[Ip and Lam, 1996] Ip, H. H.-S. and Lam, S. W.-C. (1996). Alternative strategies for irregular pyramid construction. *Image and Vision Computing*, 14(4):297–303.

[Jolion, 2003] Jolion, J.-M. (2003). Stochastic pyramid revisited. *Pattern Recognition Letters*, 24(8):1035–1042.

[Kass et al., 1988] Kass, M., Witkin, A. P., and Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331.

[Klava et al., 2009] Klava, B., Sumiko, N., and Hirata, T. (2009). Interactive image segmentation with integrated use of the markers and the hierarchical watershed approaches. In *Proceedings of the 4th International Conference on Computer Vision Theory and Applications*, pages 186–193.

[Korč and Schneider, 2007] Korč, F. and Schneider, D. (2007). Annotation tool. Technical Report TR-IGG-P-2007-01, Department of Photogrammetry, University of Bonn.

[Kropatsch, 1995] Kropatsch, W. G. (1995). Building irregular pyramids by dual-graph contraction. *IEE Proceedings on Vision, Signal and Image Processing*, 142(6):366–374.

[Lienhardt, 1991] Lienhardt, P. (1991). Topological models for boundary representation: a comparison with n-dimensional generalized maps. *Computer-Aided Design*, 23(1):59–82.

[Lienhardt, 1994] Lienhardt, P. (1994). N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal of Computational Geometry and Applications*, 4(3):275–324.

[Marfil et al., 2006] Marfil, R., Molina-Tanco, L., Bandera, A., Rodríguez, J. A., and Hernández, F. S. (2006). Pyramid segmentation algorithms revisited. *Pattern Recognition*, 39(8):1430–1451.

[Martin et al., 2001] Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the 8th International Conference On Computer Vision*, volume 2, pages 416–423.

[Meer, 1989] Meer, P. (1989). Stochastic image pyramids. *Computer Vision, Graphics, and Image Processing*, 45(3):269–294.

[Meine, 2009a] Meine, H. (2009a). Annotated contraction kernels for interactive image segmentation. In Torsello, A., Escolano, F., and Brun, L., editors, *Graph-Based Representations in Pattern Recognition, 7th IAPR-TC-15 International Workshop, GbRPR 2009, Venice, Italy, May 26-28, 2009. Proceedings*, volume 5534 of *Lecture Notes in Computer Science*, pages 273–282. Springer.

[Meine, 2009b] Meine, H. (2009b). *The GeoMap Representation: On Topologically Correct Sub-pixel Image Analysis*. PhD thesis, Department Informatik, Universität Hamburg.

[Meine and Köthe, 2005] Meine, H. and Köthe, U. (2005). The geomap: A unified representation for topology and geometry. In *Graph-Based Representations in Pattern Recognition, 5th IAPR InternationalWorkshop, GbRPR 2005, Poitiers, France, April 11-13, 2005, Proceedings*, pages 132–141.

[Meine et al., 2004] Meine, H., Köthe, U., and Stiehl, H. (2004). Fast and accurate interactive image segmentation in the geomap framework. In Tolxdorff, T., editor, *Bildverarbeitung für die Medizin 2004*, pages 60–65. Springer.

[Mičušić and Hanbury, 2006] Mičušić, B. and Hanbury, A. (2006). Automatic image segmentation by positioning a seed. In *Proceedings of the Nineth European Conference on Computer Vision*, volume 2, pages 468–480.

[Montanvert et al., 1991] Montanvert, A., Meer, P., and Rosenfield, A. (1991). Hierarchical image analysis using irregular tessellations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(4):307 –316.

[Mortensen and Barrett, 1998] Mortensen, E. N. and Barrett, W. A. (1998). Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, 60(5):349–384.

[Nacken, 1995] Nacken, P. F. M. (1995). Image segmentation by connectivity preserving relinking in hierarchical graph structures. *Pattern Recognition*, 28(6):907–920.

[Nesetril et al., 2001] Nesetril, J., Milková, E., and Nesetrilová, H. (2001). Otakar boruvka on minimum spanning tree problem translation of both the 1926 papers, comments, history. *Discrete Mathematics*, 233(1-3):3–36.

[Ossa, 2009] Ossa, E. (2009). *Topologie*. Vieweg+Teubner.

[Ren and Malik, 2003] Ren, X. and Malik, J. (2003). Learning a classification model for segmentation. In *9th IEEE International Conference on Computer Vision*, volume 1, pages 10–17.

[Shi and Malik, 2000] Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905.

[Simon et al., 2006] Simon, C., Damiand, G., and Lienhardt, P. (2006). nd generalized map pyramids: definition, representations and basic operations. *Pattern Recognition*, 39(4):527–538.

[Sonka and Fitzpatrick, 2001] Sonka, M. and Fitzpatrick, J. M. (2001). Handbook of medical imaging - processing and analysis i.n. *IEEE Transactions on Medical Imaging*, 20(3):249–250.

[Sonka et al., 2008] Sonka, M., Hlavác, V., and Boyle, R. (2008). *Image processing, analysis and and machine vision (3. ed.)*. Thomson.

[Tarjan, 1975] Tarjan, R. E. (1975). Efficiency of a good but not linear set union algorithm. *J. ACM*, 22(2):215–225.

[Torralba et al., 2010] Torralba, A., Russell, B. C., and Yuen, J. (2010). Labelme: Online image annotation and applications. *Proceedings of the IEEE*, 98(8):1467–1484.

[Van Leemput et al., 1998] Van Leemput, K., Maes, F., Vandermeulen, D., and Suetens, P. (1998). Automatic segmentation of brain tissues and mr bias field correction using a digital brain atlas. In *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 1222–1229.

[Vincent and Soille, 1991] Vincent, L. and Soille, P. (1991). Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598.

[Wertheimer, 1923] Wertheimer, M. (1923). Laws of organization in perceptual forms. *Psychologische Forschung*, 4:301–350.

[Zahn, 1971] Zahn, C. T. (1971). Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, 20(1):68–86.

[Zankl, 2010] Zankl, G. (2010). Robust image segmentation using irregular pyramids. Technical report, Institute of Computer Graphics and Algorithms 186/3, Pattern Recognition and Image Processing Group, TU Wien, Austria.