Haptic Image Exploration

by

David Lareau

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Biomedical Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

The haptic exploration of 2-D images is a challenging problem in computer haptics. Research on the topic has primarily been focused on the exploration of maps and curves. This thesis describes the design and implementation of a system for the haptic exploration of photographs. The system builds on various research directions related to assistive technology, computer haptics, and image segmentation. An object-level segmentation hierarchy is generated from the source photograph to be rendered haptically as a contour image at multiple levels-of-detail. A tool for the authoring of object-level hierarchies was developed, as well as an innovative type of user interaction by region selection for accurate and efficient image segmentation. According to an objective benchmark measuring how the new method compares with other interactive image segmentation algorithms shows that our region selection interaction is a viable alternative to marker-based interaction. The hierarchy authoring tool combined with precise algorithms for image segmentation can build contour images of the quality necessary for the images to be understood by touch with our system. The system was evaluated with a user study of 24 sighted participants divided in different groups. The first part of the study had participants explore images using haptics and answer questions about them. The second part of the study asked the participants to identify images visually after haptic exploration. Results show that using a segmentation hierarchy supporting multiple levels-of-detail of the same image is beneficial to haptic exploration. As the system gains maturity, it is our goal to make it available to blind users.

Acknowledgements

I would like to thank Dr. Jochen Lang for the moral support he has given me throughout the writing of this thesis, as well as his much-appreciated guidance during my research.

Thanks also to Vincent Levesque for sharing his experience in the research of assistive technologies for the blind, and Fan Zhang for his help during the implementation of the system for benchmarking interactive image segmentation.

This research was funded in part by Graphics, Animation and New Media Canada (GRAND).

Contents

1 Introduction **1**

1.1 Haptic Image Exploration 1
1.2 Hierarchical Segmentation 3
1.3 Thesis Statement 3
1.4 Major Contributions 4
1.5 Thesis Overview 5

2 Related Work **7**

2.1 Haptic Rendering 7
2.1.1 Traditional Relief Display 8
2.1.2 Haptic Devices 9
2.1.3 Domain of application 12
2.2 Segmentation 14
2.2.1 Automatic Segmentation 15
2.2.2 Interactive Segmentation 25
2.2.3 Segmentation Hierarchy 28
2.3 Summary 32

3 Haptic Rendering **33**

3.1 Contour Image 33
3.2 Navigation 36
3.3 Level-of-Detail 37
3.4 Texture 39
3.5 Implementation Details 41
3.6 Summary 43

4 Image Processing for Haptic Display **45**
4.1 Object-Level Hierarchy . . . 45
4.2 A Tool for Authoring Object-Level Hierarchy . . . 46
4.2.1 Examples of Object-Level Hierarchical Segmentations . . . 50
4.3 Merge-Event Timeline Data Structure . . . 52
4.3.1 Visualization Algorithm . . . 57
4.3.2 Comparison with UCM . . . 60
4.4 Vectorization . . . 61
4.5 Summary . . . 63

5 Interactive Segmentation by *Region Selection* **65**
5.1 Marker-based Interactive Segmentation . . . 65
5.2 Concept . . . 69
5.3 Evaluation . . . 74
5.3.1 Interaction Levels . . . 74
5.3.2 Greedy marker location . . . 83
5.4 Future Work . . . 85
5.5 Summary . . . 86

6 System Evaluation **91**
6.1 Experiment . . . 91
6.1.1 Experimental Design . . . 91
6.1.2 Experimental Procedure . . . 93
6.1.3 Questionnaire . . . 96
6.1.4 Data Collection . . . 97
6.1.5 Participants . . . 97
6.2 Individual Question Results . . . 98
6.2.1 The Car . . . 98
6.2.2 The Flower . . . 101
6.2.3 The Horse . . . 102
6.2.4 The House . . . 103
6.2.5 The Juggler . . . 104
6.2.6 The Clock Tower . . . 106
6.2.7 Mystery Image Omega . . . 109
6.2.8 Mystery Image Theta . . . 111
6.2.9 Mystery Image Zeta . . . 113

6.3 Analysis . . . 113
6.3.1 Experiment I . . . 113
6.3.2 Experiment II . . . 116
6.3.3 Questionnaire . . . 118
6.3.4 Subjective Comments . . . 118
6.4 Discussion . . . 119
6.5 Summary . . . 122

7 Conclusion **123**
7.1 Future Work . . . 124

A Instructions for members of the *LOD Textured* group **126**

B Glossary of Terms **128**

List of Tables

5.1 Summary of region coverage results after object extraction. 79
5.2 Summary of region coverage results after object extraction using our own markers. 85

6.1 Answer to the open question about the car photograph. 99
6.2 Correct number of answer to the location task in the car photograph. . . 99
6.3 Average time of exploration used to answer both questions correctly about the car photograph. 100
6.4 Answer to the open question about the flower photograph. 101
6.5 Correct number of answer to the location task in the flower photograph. . 102
6.6 Answer to the open question about the horse photograph. 103
6.7 Correct number of answer to the location task in the horse photograph. . 103
6.8 Answer to the open question about the house photograph. 104
6.9 Correct number of answer to the location task in the house photograph. . 105
6.10 Answer to the open question about the juggler photograph. 106
6.11 Correct number of answer to the location task in the juggler photograph. 107
6.12 Answer to the open question about the clock tower photograph. 108
6.13 Correct number of answer to the location task in the clock tower photograph. 109
6.14 Descriptive statistics on the percentage of correct answers per group during the first experiment of our study. 115
6.15 ANOVA table of Experiment I. 116
6.16 Descriptive statistics on the percentage of correct answers per group during the second experiment of our study. 116
6.17 ANOVA table of Experiment II. 117

List of Figures

1.1 PHANTOM 1.0 point contact device by SensAble Technology. 2
1.2 Steps in rendering an image with our haptic image exploration system. . 5

2.1 Mean-Shift segmentation. 16
2.2 Closest-Mode filter and its application in image segmentation. 17
2.3 Dominant-Mode filter and its application in image segmentation. 18
2.4 Statistical Region Merging (SRM) segmentation hierarchy built with different values for the Q parameter. 22
2.5 Image segmented with the SLIC algorithm. 24
2.6 The effect of colour indexing with 2-colour median cut on a photograph of peaches. 25
2.7 The OpenCV [Ope] watershed demo. 26

3.1 Holding the Phantom Device. 34
3.2 Front view of the groove line design. 35
3.3 Perspective view of groove line design. 36
3.4 Exploration at four different scales. 36
3.5 Segmentation hierarchy at three different levels-of-detail. 38
3.6 Two decaying sinusoidal waves generated with $F(t) = Ae^{-bt}sin(wt)$. . . . 40
3.7 HLAPI and OpenGL coordinates transform matrix. 42
3.8 Our haptic image exploration system running in debug mode. 44

4.1 Automatic hierarchy *object-level* assessment. 47
4.2 Automatic hierarchy accurate delineation assessment. 48
4.3 Screen-shots or our tool for authoring *object-level hierarchy.* 51
4.4 Authoring of an *object-level* hierarchy (man). 52
4.5 Authoring of an *object-level hierarchy* (runners). 53
4.6 Segmentation Hierarchy stored in a tree structure. 54

4.7 Segmentation Hierarchy stored with a merge-event timeline. 56
4.8 Building look-up table for using S_0 and the merge-event timeline. 58
4.9 Building the segment and colour look-up tables using S_0 and the merge-event timeline. 59
4.10 Smooth vectorization of a raster-based segmentation. 62
4.11 Smooth vectorization and surface simplification. 63

5.1 Segmentation results at different level of interaction. 67
5.2 Delineation results before and after providing additional interaction. . . . 68
5.3 Source image and three levels in the segmentation hierarchy. 69
5.4 Selecting multiple groups of pixels to form a selection mask. 71
5.5 Modifying the level-of-detail in the selection mask. 72
5.6 Dragging the mouse to select the remaining small regions. 73
5.7 Using pixel level controls. 75
5.8 Merging the Background by inverting the selection. 76
5.9 Sample of benchmark dataset from Zhao et al. [ZND+11] 76
5.10 Subset of a hierarchy produced by our Region Growing algorithm. 78
5.11 Subset of a hierarchy produced by the SLIC algorithm. 78
5.12 Subset of a hierarchy produced by the SRM algorithm. 78
5.13 Region coverage per levels using Region Growing for the the different categories of images. 80
5.14 Region coverage per levels using SLIC for the the different categories of images. 81
5.15 Region coverage per levels using SRM for the the different categories of images. 82
5.16 The first few interactions needed to extract the bear from the background. 84
5.17 Questionable ground truth segmentation for a butterfly in focus. 87
5.18 Region coverage per additional markers using Region Growing for the the different categories of images. 88
5.19 Region coverage per additional markers using SLIC for the the different categories of images. 89
5.20 Region coverage per additional markers using SRM for the the different categories of images. 90

6.1 The training photograph 94
6.2 The car photograph 98

6.3 The flower photograph 100
6.4 The horse photograph 102
6.5 The house photograph 104
6.6 The juggler photograph 105
6.7 The clock tower photograph 107
6.8 The first mystery photograph 109
6.9 Results for the Mystery Image Omega, and the other eight candidate images. 110
6.10 The second mystery photograph 111
6.11 Results for the Mystery Image Theta, and the other eight candidate images. 112
6.12 The third mystery photograph 113
6.13 Results for the Mystery Image Zeta, and the other eight candidate images. 114
6.14 Mean percentage of correct answers per group during first experiment of our study. 115
6.15 Mean percentage of correct answers per group during the second experiment of our study. 117
6.16 Mean (stddev) and median descriptive statistics on the feedback given in the questionnaire. 118

Chapter 1

Introduction

1.1 Haptic Image Exploration

In our research, we have developed a system for the exploration of digital photographs using haptic technologies in which users explore an image only with the sense of touch. We include features in our system that facilitate exploration and are suitable for rendering on an off-the-shelf haptic device. We have developed tools to prepare photographs into a format that contains the information needed by our haptic exploration system.

Our work is inspired by research done in assistive technology for the blind. Research attention in the literature is given to the haptic exploration of maps, schoolbook illustrations, and charts and curves in mathematics. Effort has been made to integrate haptic support in web browsers. The novel aspect of our research is that we seek to explore general photographs. The core features of our system are based on research done by Polly Edman [Edm92] on preparing relief images for the blind on physical displays. Relief displays are crafts made of material like felt that are meant to be explored with the fingertips. Edman proposes rendering the outline of the important objects in the image, and using different texture material to fill their interior. To deal with complex images, Edman recommends preparing step-by-step displays, *i.e.*, a series of relief displays of the same image at increasing levels-of-detail. Edman's ideas to prepare books and other physical material for the blind form the foundation of our computer system on haptic image exploration. Haptic devices exist in many forms, *e.g.*, the FEELit mouse from Immersion Corp, the IVEO touchpad [IVE], and the Phantom 1.0 point contact device by SensAble Technology [Tec] shown in Figure 1.1. Though the works related to our research are assistive technologies, haptics can benefit other domains. The vibration

mode of a mobile phone is a simple yet effective way to notify its user. Haptics have also made their way in video games as "rumbling" force feedback output and stylus-based input. Our research in haptic image exploration can be used as an assistive technology, but we believe the system and tools we have developed may find use in other domains. Our interactive tool for building object-level image segmentation may benefit some computer vision applications, and our haptic image exploration system may help research in psychology related to mental images.

Figure 1.1: PHANTOM 1.0 point contact device by SensAble Technology.

The main feature of our system is its support for rendering multiple levels-of-detail of the image being explored, effectively supporting the step-by-step display concept of Edman [Edm92]. We will show that exploring an image at multiple levels-of-detail aids exploration. Additional features of our system include encoding information in haptic texture as well as zooming and panning controls. The input format of our system is a vectorized segmentation hierarchy of the source image. An image segmentation algorithm transforms the image into contours, which we then render as haptic lines. The vector format enables rendering the contour image at any scale, and the hierarchy allows viewing the image at different levels-of-detail. Our implementation was tested on the Phantom 1.0, a 6 degree-of-freedom point contact device by SensAble Technology [Tec]. We chose to work with this device because it is able to render high frequency haptic texture, it is commercially available and has been used intensively in the literature.

1.2 Hierarchical Segmentation

An image segmentation is a partitioning of the image, and a contour image is its dual. The user of our haptic system can feel the shape of those contours to interpret the image, and the level-of-detail of the segmentation governs how much detail is felt. Too much detail may be confusing to the user, while not enough detail may give the user only superficial information. A segmentation hierarchy encodes multiple levels-of-detail for the same image simultaneously, but automatic hierarchical segmentation algorithms do not normally produce a meaningful *object-level* segmentation hierarchy. We have developed an authoring tool for building a segmentation hierarchy interactively, which can be used to prepare the hierarchy such that the multiple levels-of-detail encoded in it are all useful for haptic exploration. Our authoring tool uses segmentation algorithms to delineate the objects in the image at each level-of-detail. These algorithms benefit from user interaction when accurate delineation of the objects is desired. We found through experimentation that existing interactive segmentation methods are suitable for extracting a rough outline quickly but only reach an exact outline with a tedious amount of interaction. We have therefore developed a new interactive segmentation method that extracts objects with precise contours while keeping the interaction level manageable. We call our method interactive segmentation by *region selection*. However, in the event that work on automatic creation of *object-level* segmentation hierarchy matures, it could replace our authoring tool. The main focus of our research interest was to develop and evaluate a system for haptic exploration of general imagery.

1.3 Thesis Statement

It is possible to explore and understand an image using haptics, when the image is haptically displayed as a contour image. Access to multiple levels-of-detail of the same image during exploration helps the user understand the image better. An *object-level* segmentation hierarchy with precise delineation can be built for the purpose of haptic image exploration using the authoring tool and interactive segmentation by *region selection* that we have developed.

1.4 Major Contributions

We have built a system for haptic image exploration with a point-based haptic device, which supports changing the level-of-detail of the image that is being explored. A segmentation hierarchy is prepared from the image for the purpose of accessing multiple levels-of-detail of the contour image that we render using haptics. Our system also supports rendering haptic texture on the surface of each segments to convey additional information to the user. We have evaluated our system by conducting a user study designed to measure the benefit of using multiple levels-of-detail during exploration. The results of our study show that image understanding is possible using our system, and that changing the level-of-detail during exploration is indeed beneficial.

For the purpose of haptic exploration, it is important that the hierarchical segmentation be at an *object-level*. We have developed an interactive authoring tool to prepare an image into an *object-level* segmentation hierarchy. This tool recursively uses segmentation, making use of many existing segmentation algorithms. Haptic image exploration needs a segmented hierarchy with well delineated objects, and our new interactive segmentation method by *region selection* facilitates accurate segmentation of objects. We have adapted a benchmark system that measures the performance of marker-based interactive segmentation to also measure our interactive segmentation by *region selection* method. The results of the benchmark show that *region selection* is a practical solution for interactive segmentation.

In summary, the key contributions arising from our work are:

- a point-based haptic image exploration system with support for multiple levels-of-detail,
- the results of a user study conducted to measure the benefit of using multiple levels-of-detail,
- an authoring system for building *object-level* hierarchical segmentation interactively, and
- a novel method of interactive segmentation by *region selection* for precise delineation of objects, evaluated using objective performance measurements.

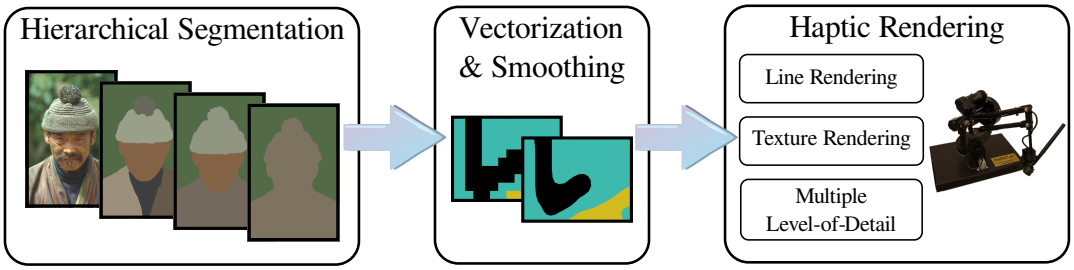


Figure 1.2: Steps in rendering an image with our haptic image exploration system.

1.5 Thesis Overview

In Figure 1.2 we give a high-level overview of our system in the form of the flow-chart depicting the process that we follow. The first two steps prepare an image for haptic exploration, and the last step renders the image haptically.

In our system, we begin by preparing a hierarchical segmentation from the source image. The hierarchy we prepare for haptic exploration has the following characteristics:

- The delineation of each of the segmented objects is precise.
- The parent-child relationship in the hierarchy is strict.
- Each level-of-detail in the hierarchy forms an *object-level* segmentation, where the objects in the image are defined at coarser level than their inner details.

Our interactive authoring tool for hierarchical segmentation is designed for building hierarchies that have these characteristics. In particular, our tool supports precise delineation efficiently when using our *region selection* method for segmenting. After the segmentation hierarchy is prepared, we perform a post-processing step in which the segmentation hierarchy is vectorized and smoothed using a vectorization method for segmented contours. The third and final step in our system renders the vectorized hierarchy for haptics exploration. We render haptic lines as 3-D grooves in an image plane using the contours defined in the segmentation hierarchy. Our groove design is a variation of the work of Ramloll et al. [RYB$^+$00] on rendering haptic lines with the Phantom. Ramloll et al. use grooves made of two surfaces forming a "V" shape. Our design uses an additional surface for the centre to give more space to the cursor, and one of the ramp is positioned lower than the other such that it is easier to climb back to the side the cursor originally entered the groove from. We also render haptic textures on the surface of each segment to convey ambient information to the user during exploration. We encode the size of the current

segment in the frequency of the haptic texture, such that it is possible to know if the segment is bigger or smaller relative to other segments. We tested the performance of our system with a user study of 24 sighted participants divided evenly into four groups, each group enabling or restricting access to the level-of-detail and haptic texture features. The study included two experiments, the first measuring how well participants could answer questions after 3 minutes of exploration of an images they were given a short description about. The second experiment tested how well participants could identify visually an image they had explored by touch for 1 minute. Our results confirm the benefit of using multiple levels-of-detail during exploration, but are inconclusive with regards to the impact of haptic texture on its own, and negative when textures are used in conjunction with multiple levels-of-detail.

Our research was inspired by related work done in assistive technology using haptics and by research done in image segmentation. In Chapter 2, we discuss fundamental principles in the development of a relief image, and related work on haptic rendering. In the same chapter we cover automatic and interactive methods for image segmentation. In Chapter 3, we present the details of our haptic rendering implementation. Chapter 4 introduces our authoring tool for building an *object-level* segmentation hierarchy from a source image. It also includes the implementation detail of the data structure we have designed to store the hierarchy in an efficient and raster-independent manner, and gives the procedure we follow to vectorize the hierarchy. Our authoring tool uses auxiliary segmentation algorithms to do the actual segmentation. We present our novel interactive segmentation by *region selection* method in Chapter 5. We have developed this interactive method as an efficient alternative to the interactive segmentation methods found in the literature for precise delineation. We have compared our method against marker-based interaction using an objective benchmark evaluation system and present our results in the same chapter. In Chapter 6 we present the user study we have conducted to test the overall performance of our haptic image exploration system. The results of the study confirm the beneficial effect of exploring an image at multiple levels-of-detail. We give our concluding remarks in Chapter 7.

Chapter 2

Related Work

In this chapter, we discuss previous work on image exploration using haptics and on the steps necessary to prepare digital images for haptic rendering. Our efforts build on various research directions related to assistive technology, computer haptics, and image segmentation. Section 2.1 introduces fundamental principles for the development of a relief image, as well as work done on computer assisted haptic image exploration. We discuss the many haptic devices used in research and discuss some of their strengths and weaknesses. Image segmentation algorithms used in the preparation of images are presented in Section 2.2. The discussion focuses mainly on popular algorithms, both automatic or interactive, and on solutions that build segmentation hierarchies. We give a summary of this chapter in Section 2.3.

2.1 Haptic Rendering

The goal of our research is to develop computer assisted technologies that aid a person in understanding a photograph using the sense of touch. There has been research in assistive technology for the blind to help explore maps [KMT+07, ICG+04, PWL10], graphs and curves [FB99, RYB+00], and schoolbook illustrations [PDL+08] using various haptic enabled devices. One novel aspect of our research is that we consider general imagery. We begin this section with an introduction on traditional relief display used in the blind community, after which we introduce work done on assistive technology using computers.

2.1.1 Traditional Relief Display

The American Foundation for the Blind published in 1992 a book on Tactile Graphics by Polly Edman [Edm92]. This book covers design principles and technical details on preparing material for visually impaired people. It does an extensive review of all aspects of design and conception. Separate chapters are dedicated to pictures, maps, and mathematics, graphs, and diagrams. The book only briefly mentions computer assisted technology due to its publication date. Nevertheless it is still considered a core reference in the field. In the book, a guideline when preparing haptic material involving line drawing, outlines, geometric figures and graphs is to use a single elevation above the background surface. For images, the important objects are segmented into outlines. The segments may have different texture material to fill their interior. Edman mentions that contrast in texture and elevation height are most important for the comprehension of the reader. The level-of-detail in the segmentation is left at the discretion of the person in charge of making the relief display. Edman warns against too many lines in a tactile graphic display, which would cause confusion. The guideline from the book is to keep the pictures simple, stating that only the most important part of the image should be portrayed. The following quote summarizes the solution to supporting multiple levels-of-detail suggested by Edman:

> "The problem of too much information in one display can be alleviated by breaking down the display into a step-by-step series of pictures. In step-by-step displays, an image is presented multiple times side-by-side at the same scale but with increasing levels-of-detail. This is to give time to the user to understand the core outline of objects before exploring the details within those same objects. Exploring the high levels-of-detail images at the start can lead to confusion of the image content." [Edm92]

Developing algorithms and tools that support this step-by-step solution in a computer system is one of the main contribution of our research. We do so while respecting the reference points defined in coarser levels-of-detail.

Rotard et al. [RTE08] have worked on assistive technology for rendering web pages using a tactile pin-matrix device. Graphics capable pin-matrix displays, like the tablets developed by the HyperBraille project [Hyp10], can render dots of Braille characters by lifting or lowering pins that are arranged in a matrix on a planar surface. The HyperBraille display is not limited to Braille, and can render any binary (monochrome) image on its surface. A pin-matrix display is convenient because the raised dot patterns

are visually identifiable for simple images. Though recognizable visually, a monochrome image is very difficult to recognize by touch, especially if it is dithered. Object contours on the other hand are more easily identifiable by touch.

In their work, Rotard et al. seek to give blind users a holistic alternative to linear screen readers when browsing web pages. They render the web page text and images on a pin-matrix device while trying to preserve the layout. Their research for rendering images is based on scalable vector graphics (SVG). By extracting the foreground objects, they can render the foreground and background using the two states of the pins. The nature of scalable vector graphics allows them to support zooming functionality. They have also implemented an edge filter that renders only the outline of the foreground as lifted pins, and the rest as lowered pins. To deal with raster images, which are the most common type of image, Rotard et al. explored converting images to binary (monochrome). The threshold for the conversion is controlled interactively. They have also implemented an alternative that uses edge detection using the Sobel operator to create a binary image. They explain their decision to use simple thresholding by pointing out that extracting a semantically adequate binary image would require algorithms that do not perform in real-time and that often only work in specific context. In our work, we have explored a more complex alternative to extract the outline of meaningful objects in an image. We use segmentation and vectorization to transform a raster image to a hierarchical vector graphics. We build a segmentation hierarchy to support the step-by-step solution proposed by Edman [Edm92] for exploring pictures rich in information. Our approach for raster image not only enables panning and zooming but displays smooth outlines and enables the interactive selection of the level-of-detail during image exploration. However, our steps in the segmentation of an image need interaction and therefore our method does not apply in the context of casual web browsing.

2.1.2 Haptic Devices

Haptic rendering of an image can be performed with a variety of commercial or research devices. We use the Phantom 1.0 which is a 6 degree-of-freedom point contact device by SensAble Technology [Tec]. Our Phantom has a stylus attached (pen-like handle) to provide point-based haptic feedback. Figure 1.1 shows the Phantom device that we work with in our lab. The workspace of the device is a volume in 3-D, but we render the image on a 2-D plane. The Phantom is used in many studies of haptic exploration [KWP02, Sjo01, FB99, KMT$^+$07, RYB$^+$00]. An ideal system would give

feedback to the user about the surface touched by the pen, much like if the user would be touching a real object with a real tool. Rendering an image with a point contact force feedback device like the Phantom [Tec] is not as straightforward as displaying the image on a pin-matrix display. The image has to be conveyed to the user as the user moves the haptic stylus through 3-D space. Photographs contain visual textures and features that can be transferred to haptic textures allowing the user to feel them. The haptic texture properties of segments can be rendered with various approaches. We experimented with the approach of Guruswamy et al. [GLL11] and Okamura et al. [ODH98]. Use of haptic texture in addition to contours ensures that the user has a sensation while inside a segment. By applying a different texture to adjacent segments, they will feel different. Norman et al. [NCNC08] have shown that we can reliably perform haptic recognition of 3-D objects that we have learned visually. However, they also found that we need rich cues for this task because of the cross-modal performance cost. Dopjans et al. [DWB09] observe a similar cost between vision and haptics for the particular domain of human faces recognition. Haptic exploration has a narrow field-of-view compared to visual exploration. Extra information about the global context of an image can be communicated to the user via ambient encoding. Global context information enhances the exploration experience by giving the user a larger field of view. We are using tactile textures to encode ambient information about the global context, and render then at a rate of 1kHz with our Phantom device.

There are many types of haptic devices in the literature. We have already mentioned the HyperBraille pin-matrix display [Hyp10] used by Rotard et al. [RTE08]. This device has the advantage of presenting a full "screen" of data simultaneously. The user is free to explore with both hands, even feeling with the palms, not only the fingertips. Edman [Edm92] noticed when working with blind children, that it is necessary for them to use both hands when they explore an object. Edman states that it is advantageous for the blind reader to use both hands when quickly scanning a relief display. While displaying images on a pin-matrix seems advantageous compared to rendering them with an haptic force-feedback device like the Phantom, limitations around cross-modal transfer, resolution, size and the lack of global context are largely the same. Haptic textures are also important according to Edman, but are not available on a pin-matrix display. We have chosen to research haptic textures, which is why we focus our attention on the Phantom.

Petit et al. [PDL+08] have used a device composed of a STReSS2 (Stimulator of Tactile Receptors by Skin Stretch) and a Pantograph [GCH05]. This device can produce

different kinds of tactile sensation under the finger by stretching the skin on the fingertip of a user. The STReSS2 device is mounted on the Pantograph device, which limits the movements of the finger on the STReSS2 device to a 2-D plane. This setup is very interesting, as haptic textures can be rendered under the fingertip. In comparison with the Phantom which gives 3-D force-feedback, the setup by Petit et al. is limited to 2-D. The surface area of the Pantograph is 11.3 cm wide by 6 cm deep whereas our Phantom workspace has larger dimensions of 25.4 cm wide by 17.8 cm high by 12.7 cm deep though not the whole volume is accessible.

The GRAB system developed by Iglesias et al. [ICG+04] is a multi-modal system that integrates audio, verbal commands and touch. It is designed with 3-D exploration in mind. GRAB uses a dual-finger haptic interface, which can be used with the thumb and the index, or two index fingers. The authors argue that such dual-finger system helps with orientation, spatial memory, locating objects, staying in touch with object and perceiving complex shape and size. They report from a user study that the second finger often acts like a reference point that allows users to better understand 3-D objects and helps in keeping their bearing. The workspace of their system is larger than the one of our Phantom. Each arm can navigate in a region 60 cm wide by 40 cm high by 40 cm deep. They mention a previous experience from one of their partners that showed that using two Phantom devices brings limitations in manipulation and shape recognition. Unfortunately, no detail is given as why that is the case.

Paladugu et al. [PWL10] have developed a system that presents maps to blind users in the form of audio-tactile maps. Their system fetches map and direction information from MapQuest. The person preparing the map manually adds tactile patterns to the map file, and prints it on sheets of paper to be placed on top of an IVEO touchpad [IVE]. This implies there is no zooming or panning available to the user unless they prepare and print a new map. They chose to focus on improving human computer interaction. The maps are printed by a specialized printer (embosser) that can print tactile patterns on a sheet of paper.

Sjostrom [Sjo01] experimented with the FEELit Mouse from Immersion Corp. This mouse has some force feedback capabilities. For example, it can send vibration signals to the user when they cross a window of the graphical user interface. Sjostrom proposes three search tools to help finding objects in the graphical user interface. They are defined as *cross* which sends a signal when the pointer lines up horizontally or vertically with an object, *magnet* which pulls the pointer towards the nearest object, and *ball* which makes the user feel objects at a distance but with less detail. Sjostrom prepared a user interface

with a radial menu instead of the traditional menu arranged in columns. He ran a small user study with two blind participants on the usability and usefulness of the device with the *cross* search tool and radial menu interface. Both users mentioned having difficulties with the small workspace of the device, and though they liked the radial menu, they were both reluctant in using. Sjostrom states that the user seems to prefer having an experience similar to the sighted over having a potentially more efficient but custom view. In other words, if the sighted have a menu arranged in a column, they prefer a column menu as well. The production of the FEELit Mouse has been discontinued.

2.1.3 Domain of application

The research in haptic image exploration has often been focused on practical applications. An important activity for the blind community is the understanding of maps. There has been extensive work on the subject, and we understand that the the goal was always map understanding and not particularly haptic exploration. Therefore, map systems for the blind are multi-modal systems usually involving audio and haptic display. Kostopoulos et al. [KMT+07] implemented a framework that extracts useful information from a 2-D map and presents the information gathered in a combination of audio cues and haptics. The map is explored in 3-D using a Phantom device. Using the suggestion of Ramloll et al. [RYB+00] on rendering haptic lines with the Phantom, the streets are rendered as grooved lines. An interesting feature from their work is that their input is a digital picture of conventional maps instead of a formatted map structure. The image of a map can be the only available data in certain scenarios. In order to supply street names to the user, they extract the names using character recognition. Iglesias et al. [ICG+04] built a system for city-map exploration which employs formatted map data (TeleATlas, ESRI). This map exploration system is one of three applications that were developed as proof of concept for their GRAB system. The multi-modal city-map exploration application supports zooming and panning capabilities and can report the name of the street or intersection where the user is located.

Another image domain that received attention from researchers in haptics is scientific or mathematical data shown as graphs and curves. Fritz and Barner [FB99] experimented with haptic points, lines and surfaces and established ground work to represent 1-D, 2-D and 3-D data using a Phantom device. They render 1-D points either using spheres for the points themselves, or using a sphere for the cursor. The user controls the radius of the cursor sphere. This allows the user to explore low resolution features using a

large radius, or high resolution features using a small radius. They suggest rendering 2-D lines as attraction lines, such that when the cursor is near the line, it is attracted to it like a magnet. The attraction force decreases as the cursor gets very close to the line in order to avoid the jitter that would happen as the cursor tries to position itself exactly on the line. They render 3-D data as a triangular mesh object and constrain the cursor to the surface of the object using the god-object method described by Zilles and Salisbury [ZS95]. Fritz and Barner argue for rendering virtual walls around a 3-D scene to prevent the user from falling away from the scene. They give the example of a 3-D plot where walls prevent the user from going beyond the limit of the axes. We use similar walls in our system to prevent the user from falling from the edges of the 2-D image that is being explored. Ramloll et al. [RYB+00] focused on making line graphs accessible to blind students using auditory and haptic media. They used the Phantom haptic device and they introduced a groove design to render haptic lines. In this design, the cursor falls into an actual 3-D groove when meeting a line. The groove constrains the user to follow the line in a passive manner that is intuitive. The line can be followed simply by pushing the device gently in the direction of least resistance. To get out of the groove the user lifts the cursor or applies enough force to climb the wall of the groove. We have integrated this groove design into our work to help the user follow the outline of objects. Ramloll et al. use speech output to report the cursor position. They also use a continuous audio tone of varying pitch to give ambient information about the y-position of the cursor. The user can request an auditory overview of the curve being explored, which plays the y-position pitch of the curve using the x-axis as time.

Petit et al. [PDL+08] developed a method to prepare school material for haptic exploration such as indoor building maps, bar graphs and world maps. They conducted a user study in which each participant answered questions about an image. The questions were designed to verify the understanding of the illustration explored by the participant. To prepare material for haptic exploration, the image is first simplified manually using an image editing software. The process produces a closed contour image. The simplification is done following guidelines developed by Trudeau [Tru], who recommends keeping only significant elements, increasing edge size, erasing patterns, and highlighting parts that are not apparent but important. Each segment formed by the contour image is assigned tactile and audio feedback. Once this labeling is done, the image is ready to be explored with their STReSS2 system. The user explores in 2-D with one index finger. Contours are identifiable by a distinct high frequency tactile sensation. Entering a segment triggers the audio and tactile feedback of that particular region. The tactile

feedback of interior regions feels like low frequency undulation to contrast with the high frequency of the outline. The image is scaled down to fit in the available working area of the Pantograph [GCH05] on which the STReSS2 device is mounted. The results of Petit et al. show that participants were able to answer questions about the given illustration correctly.

We are interested in general imagery and in the exclusive use of haptics for exploration, *i.e.*, no audio. Perhaps not as inherently practical as maps or school material, general imagery is the most common type of images on the world-wide-web, *e.g.*, photographs, drawings. The contributions of our research are for general imagery but specific image domains can also benefit from our novel exploration techniques.

Pai and Reissell [PR97] describe a method for rendering curves extracted from a general image. They use the Canny edge detector to extract the edges from the image. The edges are linked into a curve using hysteresis thresholding. The curve is stored with wavelets at multiple resolution such that ignoring high resolution information gives an approximation of the curve. They note that some resolution level act as denoising filters. To increase rendering speed, they propose rendering the curve using high resolution information only when the hand is moving slowly along it. They use a hierarchy of bounding boxes for fast collision detection with the curve. Pai and Reissell use a similar strategy to Zilles and Salisbury [ZS95] for constraining the cursor on the curve boundary. They remark that the user's perception of contact is greatly enhanced when giving visual feedback of the virtual cursor touching the curve boundary. The real cursor might penetrate the curve, but the virtual cursor that is shown to the user always respects the boundary constrain. The example included in their paper only has one curve. It is unclear how their multiple resolution encoding would behave in scenarios where two curves cross.

2.2 Segmentation

We transform digital photographs stored in a raster format into vector graphics by first segmenting the objects in the image to form outlines. In traditional relief display, the level-of-detail of the segmentation is left at the discretion of the person preparing the material [Edm92]. A low level-of-detail gives an preview of the image, tracing the outlines of the main objects. A high level-of-detail gives access to the details, but is confusing to explore. Breaking down a traditional relief display into a step-by-step series of picture is costly, but beneficial to the user. We have developed algorithms and data structures in our haptic image exploration system to support a segmentation hierarchy. The level-of-

detail of the display is controlled interactively during the exploration. In this section, we discuss work done on image segmentation and approaches for building a segmentation hierarchy. There are numerous algorithms for segmenting images automatically or interactively. A comprehensive review is beyond the scope of this thesis. We focus instead on methods that are widely used and that we have implemented in our system. For a more comprehensive tour of image segmentation algorithms the reader is encouraged to look at surveys by Haralick et al. [HS85] and Freixenet et al. [FMR+02], and look at a paper on automatic image segmentation by Arbeláez et al. [AMFM09] and at benchmark results published by Zhao et al. [ZND+11] about interactive image segmentation algorithms.

2.2.1 Automatic Segmentation

The Berkeley Computer Vision Group is very active in image segmentation. Their image segmentation database BSDS500 [AMFM] is very popular in the field [AMFM10]. Arbeláez et al. [AMFM09] identify three algorithms that seem to be the most commonly used: Mean Shift algorithm by Comaniciu and Meer [CM02], the graph-based method of Felzenszwalb and Huttenlocher [FH04], and the normalized cuts algorithm of Shi and Malik [SM00]. We begin this section on automatic segmentation by reviewing these three algorithms and their variants, followed by methods based on region merging and a method specialized in segmenting an image into superpixels. We close this section by discussing the relation between segmentation and edge detection and colour indexing.

The Mean Shift image filter proposed by Comaniciu and Meer [CM02] assigns each pixel of an image to the closest mode of its local density distribution in the spatial and range (colour) domain. The closest mode is found iteratively, and the method is guaranteed to converge. Comaniciu and Meer describe two applications for the Mean Shift filter: Discontinuity-preserving smoothing and image segmentation. Segmentation is achieved in two steps. In the first step, the image is smoothed with the Mean Shift filter which preserves the discontinuities (edges). In the second step, pixel assigned nearby modes in both space and range are given the same segment label. An optional third step eliminates segments smaller than a given size. Comaniciu and Meer do not give any suggestion on how to implement this optional step. Mean Shift is used by Zeng et al. [ZZXZ09] to get superpixels. A superpixel is a cluster of adjacent pixels of similar characteristics. Preprocessing images into superpixels is a standard step in computer vision as it reduces the complexity of the input, which helps complex tasks achieve better computational time performance. An implementation of the Mean Shift filter for image

segmentation can be found in the Edge Detection and Image SegmentatiON (EDISON) System [CMCM]. We have implemented the mean shift segmentation algorithm in our system because it is especially good for extracting the shape of small details accurately, *e.g.*, a flower pattern on clothing in Figure 2.1.

(a) Source Image (b) Mean-Shift

Figure 2.1: Mean-Shift segmentation ($s = 12, r = 6, M = 20$).

Kass and Solomon [KS10] presented a paper on Smoothed Local Histogram Filters. Their paper describes how to build local histogram filters with isotropic weighted neighbourhood and having a computational complexity independent of the neighbourhood size. Their technique can also compute the derivative and integral of the same histograms. The $O(1)$ complexity is made possible by the method of Deriche [Der93] for computing the 2-D convolution of a Gaussian kernel or its derivative. Kass and Solomon include details on implementing two edge-preserving smoothing filters using histograms called the Closest-Mode and the Dominant-Mode filters. The Closest-Mode filter is very similar to the Mean-Shift filter [CM02]. Using the derivative to find the modes, it assigns to each pixel the closest mode of its local histogram. Figure 2.2 shows the Closest-Mode filter and its application in image segmentation. The Dominant-Mode filter uses the integral of the histogram to find the population within each mode of the local histogram. The most populous mode is chosen as output instead of the closest mode. Figure 2.3 shows the Dominant-Mode filter and its application in image segmentation. These two filters are non-iterative alternatives to Mean Shift, which means they can work efficiently on

larger neighbourhoods. We achieve image segmentation by substituting Mean Shift with one of these edge-preserving filters in the image segmentation steps given by Comaniciu and Meer [CM02]. We consider the Closest-Mode to be an improvement on Mean Shift.

(a) Source Image (b) Closest-Mode Filter $\sigma = 6$

(c) Segmentation (filled and colored) (d) Segmentation (outline only)

Figure 2.2: Closest-Mode filter and its application in image segmentation.

Felzenszwalb and Huttenlocher [FH04] presented a greedy algorithm for image segmentation. Their design goal was to develop a method that is broadly useful in many computer vision tasks. They define a broadly useful image segmentation algorithm as being one that captures perceptually important regions and that is highly efficient, *i.e.*, $O(n)$, where n is the number of pixels. Their method is $O(nlogn)$. Their algorithm starts by interpreting the image as a graph. Each pixel is a node in the graph. They propose two ways to see the edges. The simplest form is to use the raster grid where edges connect each pixel with the 4 pixels in its neighbourhood, or with the 8 neighbours depending if diagonally connected segments are acceptable. Alternatively, they propose using a nearest neighbour graph, where each node is a feature vector, *e.g.*, (x,y,r,g,b) and edges connect each node to a fixed number of nodes that are the closest to it in this feature

(a) Source Image

(b) Dominant-Mode Filter $\sigma = 6$

(c) Segmentation (filled and colored)

(d) Segmentation (outline only)

Figure 2.3: Dominant-Mode filter and its application in image segmentation.

space. This second approach can create segments that are disjoint since edges may exist between pixels that are not neighbours in the raster grid. Independent of the graph representation, each node is initially in its own segment, and each edge is assigned a weight. The weight function can be as simple as the difference in grayscale intensity between its two nodes. The edges are sorted in decreasing order of weights. Considering each edge in that order, if the two components at each end of the edge are not in the same segment, the algorithm merges them into the same segment if the weight is small compared to the internal difference of both components. The internal difference of a component $Int(C)$ is computed as the maximum edge weight that is within it, but the internal difference of two components $MInt(C_1, C_2) = min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2))$, where $\tau(C) = k/|C|$, but $\tau(C)$ can be any threshold function. In their case, the threshold function is based on the size of the component, and k is a parameter. When k is set high, it favours the formation of larger segments. Two components are merged if the weight of the edge is lower than $MInt(C_1, C_2)$. Arbeláez et al. [AMFM09] say that this algorithm is typically used to create superpixels. We have implemented the algorithm because it is generally well regarded and it is a good choice to produce slight oversegmentations.

Shi and Malik [SM00] described a graph partitioning image segmentation algorithm. In graph partitioning methods, the image is viewed as a graph and segmented in a top-down direction. Initially, all nodes in the image, *e.g.*, pixels, are considered to be in the same segment. All internal nodes of this segment are connected by weighted edges. An objective function is responsible for computing the weights. The objective function generally is a function of similarity between the two nodes. The goal of the graph partitioning algorithm is to find the best cut that will partition, *i.e.*, bisect, the graph in two parts. The value of a cut is defined as the sum of the weights of the edges that are shared by both parts as:

$$cut(A, B) = \sum_{\substack{u \in A \\ v \in B}} weight(u, v),$$

where A and B would be the two parts formed after the cut. The algorithm tries to partition such that the cut chosen minimize the internal dissimilarity of the partitions. Shi and Malik cite Wu and Leahy [WL93] which have minimized this cut criterion for image segmentation, but acknowledge that the criterion favours cutting small sections away from the graph. To solve this problem Shi and Malik propose using a normalized cut criterion. Their normalized criterion computes the weight for each edge as a fraction

of the total edge connections in the graph, and is defined as:

$$N_{cut}(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)},$$

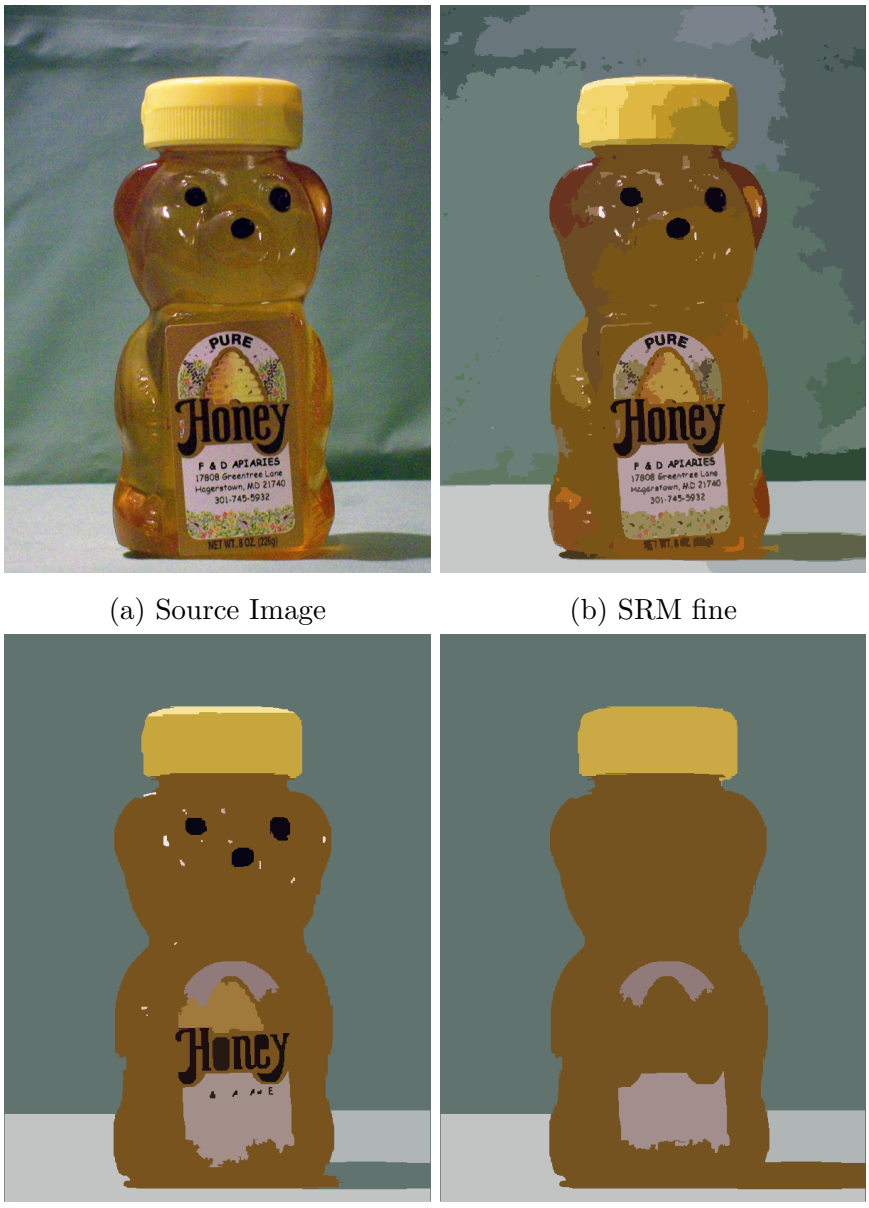
where $V = A \cup B$ and where $assoc(A,V)$ is the sum of all the weights between the nodes of A and all nodes in V, *i.e.*, $assoc(A,V)$ is actually the same formula as $cut(A,V)$ but with a different semantic. It is not trivial to find an efficient algorithm to solve this optimization problem given an arbitrary criterion. Not all criteria have an efficient solution. Shi and Malik mention that many interesting criteria have been rejected for not having any efficient algorithm to compute their global solution. They have found that minimizing a normalized cut exactly is NP-complete. When transforming the problem into the discrete domain, an efficient algorithm using eigenvector theory can approximate the solution. They use the Lanczos algorithm, having found that the graph of an image is normally very sparse, and that only the top few eigenvectors are needed for graph partitioning.

Aside the three algorithms mentioned by Arbeláez et al. [AMFM09], there are other automated image segmentation algorithms that are interesting to us. One approach is region merging which can produce state-of-the-art results when used with a statistical measure [NN04]. Region merging (or region grouping or growing) is often used to describe segmentation algorithms. In our definition, region merging starts with a set of fine segments. If the input is a digital photograph, each individual pixel in the image is considered a segment by itself. Every segment is connected to its spatial neighbours by weighted edges. The weight of the edges is a function of the similarity of the two segments they connect. The function can be as simple as the difference in mean colour between the two segments, or as complex as a variational criterion involving shape measures and the probability of an edge being present at the common boundary between the two segments. The region merging algorithm proceeds in a greedy fashion, merging the two segments connected with the lowest weight (merge cost). After merging the two segments into one, the weights of the affected edges are updated. The algorithm iterates until a stop criteria is satisfied or until there is only one segment remaining. The remaining segment then covers the whole image. The computational bottleneck of this method is in the update of the affected edges. Using a sorted tree structure, initially sorting the edges will be $O(nlog(n))$ where n is the number of edges. Updating edges after a merge will be $O(log(n))$. When working from a raster image where each pixel is initially a segment connected to its 4 neighbours, there are $2N$ edges, where N is the number of pixel. To improve performance of the described region merging

algorithm, it is common to preprocess the image into superpixels. Additionally, some region merging methods [FH04, NN04] bypass updating the edges by adding an extra condition before merging the two segments with the lowest weight. We have in our toolset a straightforward implementation of the region merging algorithm described here based on average colour distance between segments. The algorithm suffers from its bottom-up heuristic and does not find a global optimum given the cost function. However, a region growing segmentation algorithm implicitly builds a hierarchy. A segmentation hierarchy is a tree of membership relationship. The leaves of the hierarchy are the input segments, *e.g.*, pixels. Parent nodes in the tree represent a segment made of the union of its children. Each merge operation in the region growing algorithm can be translated as two children creating a parent node. A region growing algorithm therefore builds a hierarchy automatically. We discuss methods for building segmentation hierarchies in Section 2.2.3.

Nock and Nielsen [NN04] describe a statistical region merging algorithm (SRM) for image segmentation. In their method, they do not need to update the list of merging tests, *i.e.*, the edges are sorted only once. This is an adequate optimization that is possible when the weight function is designed to approximate an initial ordering that satisfies the following invariant defined by Nock and Nielsen: "when any test between two (parts of) true regions occurs, that means that all tests inside each of the two true regions have previously occurred". When greedily selecting the next lowest weighted edge, the merge between the segments is only performed if the two segments satisfy an additional constraint that enforces the invariant. Felzenszwalb and Huttenlocher [FH04] use a similar concept during segmentation. Nock and Nielsen have a parameter Q that governs the coarseness of the resulting segmentation. This Q parameter is similar in spirit to the parameter k of Felzenszwalb and Huttenlocher. The implicit hierarchy built by the sequence of merge operation is not very interesting when using the method of Nock and Nielsen or Felzenszwalb and Huttenlocher because the merge order does not faithfully represent a finer to coarser segmentation. Viewing the implicitly built hierarchy can still be valuable for debugging purposes. Nock and Nielsen instead suggest building the hierarchy at different scales by running the segmentation multiple times with increasing values of their parameter Q. Unlike the implicit hierarchy, this method for building the hierarchy does not guarantee that the segments of coarser levels are strictly made of segments of the next finer levels. Figure 2.4 shows a segmentation hierarchy built with SRM at different values for Q.

Achanta et al. [ASS+10] present a simple linear iterative clustering (SLIC) method



(a) Source Image (b) SRM fine

(c) SRM medium (d) SRM coarse

Figure 2.4: Statistical Region Merging (SRM) segmentation hierarchy built with different values for the Q parameter.

for finding high quality, compact, and nearly uniform superpixels as shown in Figure 2.5. Their method can be used for image segmentation, but is specialized in forming superpixels segments. The algorithm clusters the image in the spatial and colour domains, as is done in Mean-Shift [CM02]. Given N to be the number of input pixels, and K the number of output superpixels, the average size of the superpixels is N/K. If the superpixels are of uniform size and shape, then the centre of every superpixel is roughly positioned at every $S = N/K$ pixels. SLIC uses the CIELAB colour space, since it is considered to be perceptually uniform for small colour distances. Achanta et al. define a new distance metric that normalizes space distance before combining it with the colour distance. The normalization is needed because images vary in dimension whereas the CIELAB colour space is fixed, and the spatial distance can easily outweigh the perceptual colour distance limit if it is not normalized. Given D_{lab} to be the 3-D Euclidean distance between two CIELAB colours, and D_{xy} to be the Euclidean distance between two points in space, Achanta et al. define their normalized distance metric as:

$$D_s = D_{lab} + \frac{m}{S} D_{xy}$$

The parameter m controls compactness, *e.g.*, a larger m will amplify the contribution of the spatial distance in the combined distance, and therefore encourage more compact clusters. They mention using $m = 10$ because it is close to the empirical maximum perceptually meaningful CIELAB distance. The algorithm begins with initializing K clusters positioned uniformly in the image at every S spatial intervals. For each cluster center, a search area of $2S \times 2S$ is used to assign to this cluster pixels that are closer using their distance metric than to any other cluster. The centre of the clusters are moved according to the pixels that are assigned to them. This classification of pixels to clusters repeats until the change in cluster position is minimal. The algorithm does not enforce connectivity, so an extra step that relabels disjoint segments with the labels of the largest neighbouring cluster is done afterwards. The SLIC algorithm has a $O(N)$ complexity in computational cost and memory usage. SLIC offers parameters that directly control the number, size and compactness of the output segments in contrast to Mean-shift [CM02]. Achanta et al. argue that compact and highly uniform superpixels are desirable for many computer vision tasks, *e.g.*, Conditional Random Fields or SIFT.

We finish this section by mentioning some related image processing techniques to segmentation, *i.e.*, edge detection and colour indexing. The contours of segments (region boundaries) given by a segmentation algorithm typically match the edges of the original image. Segmentation and edge detection are therefore tightly related. However, segments

(a) Source (b) 200 superpixels (c) 500 superpixels (d) 1000 superpixels

Figure 2.5: Image segmented with the SLIC algorithm using various number of output superpixels. The photograph is copyrighted to 'The Fire Man Dave'.

are closed whereas edge detection does not usually have the restriction of producing closed contours. For example, the popular edge detection algorithm by Canny [Can86] gives a list of disconnected edges. Edge detection can aid a segmentation algorithm achieve its task. The watershed algorithm covered in section 2.2.2 uses edge detection to build a gradient image.

Colour indexing is a form of segmentation. The task consist of performing quantization on the colours used in one image. A simple threshold approach gives quick results when converting an image to binary (segmenting in two discontinuous segments). One implementation is to classify pixels in two groups depending on their brightness relative to half of the spectrum. A typical colour indexing task is to extract the best palette of 256 colours from a 24-bit RGB image that can be used to approximate the original image. In other words, finding groups of pixels that can be quantized to the same value while damaging a human observer's perception of the image the least. When looking at the task from the perspective of segmentation, the indices of the palette become the segment ids. The drawback of using colour indexing for segmentation is that the segments will generally not be contiguous. Dithering, which is a step commonly performed during indexing to improve the perceptual similarity of photographs, almost guarantees that the segments will not be contiguous and should be avoided when using colour indexing for segmentation. The median cut algorithm by Heckbert [Hec82] is the most widespread colour indexing algorithm. The algorithm works by looking at the RGB colour value of all the pixels of the input image in one 3-D box with axis for red, green and blue. The box containing the highest variance will be split evenly in two smaller box on the axis most responsible for its variance. If a box is empty, it is removed. The process continues until the number of boxes is equal to the desired quantization, *e.g.*, 256 boxes. Figure 2.6

shows the effect of colour indexing with median cut on a photograph of peaches.

(a) Source Image (b) 2-colour median cut

Figure 2.6: The effect of colour indexing with 2-colour median cut on a photograph of peaches.

2.2.2 Interactive Segmentation

Without human guidance, automatic segmentation will generally not match any particular goal for all images. The segmentation methods reviewed in 2.2.1 do require setting parameters, but work automatically without further interaction. Interactive segmentation solutions have been developed in order to get more relevant and controlled results for a given application. This is valuable to us since our needs require precise delineation of objects which cannot be reliably extracted with automatic segmentation.

Meyer [Mey92] describes a marker-based variant of watershed. Watershed works with the analogy of filling a topology with water from seed points. The waters originating from different seeds are prevented to mix by setting up dams at the points were the waters would mix. These dams are the boundaries of the resulting segments. There are as many segments as there are seed points. In the original watershed algorithm, the seed points are the local minima of the topology. The topology is normally a gradient image of the input image. A gradient image is generated by an edge detector, where the intensity of the gradients represent the likelihood of an edge being at that position. In the variation of Meyer, the seed points are labeled. Seed points can share labels, and if so will not create dams when their water are about to mix. Meyer acknowledges that finding appropriate seed points, which he calls markers, is task-dependent. This

watershed algorithm is implemented in the OpenCV library [Ope], a library of functions for real-time computer vision. In the watershed demo of OpenCV, the markers are given by the user in the form of mouse driven scribbles. Connected markers are given the same label. We built on this watershed demo to provide scribble-based watershed interactive segmentation in our system. Figure 2.7 shows the OpenCV watershed demo in action. In this demo, the markers are the contiguous white regions.

(a) Source image (b) Markers and Output

Figure 2.7: The OpenCV [Ope] watershed demo.

The Graph Cut algorithm by Boykov and Jolly [BJ01] utilize user given scribbles to perform segmentation. The scribbles act as hard constraints on the resulting segmentation, but soft constraints defined by the image data also exist. A graph is built from the image data, where weighted edge make connections between each nodes. The weights of the graph are used to run a min-cut/max-flow algorithms. One advantage of their method is that it applies not just to 2-D images, but can be generalized to N-D images. Rother et al. [CRB04] developed a popular marker-based algorithm called GrabCut that builds on the method of Boykov and Jolly. One of their improvement simplifies the interaction required for segmentation. They experimented with colour statistics stored as mixture of Gaussians to aid classification. An implementation for GrabCut is found in the OpenCV library [Ope].

Unger et al. [UPT+08] presented an interactive image segmentation technique based on a total variation optimization goal and where the interaction is also in the form of

scribbles for the foreground and background sections. They call their method TVSeg. Their segmentation method is followed by an image matting step which gives alpha values to the borders between foreground and background. The binary segmentation is done using a fast total variation minimization of the geodesic active contour energy with local constraints. Instead of using a graph-based approach for this optimization problem like Boykov and Jolly [BJ01], they use a projected gradient descent algorithm based on a dual formulation of the total variation. The constraints are given interactively by the user. They propose using two different types of brushes for this purpose. The first is the *hard* brush, which forces the specified section of the image to be classified as foreground or background. The second is the *sample* brush, which builds a 3-D colour histogram of the colours it covers to gauge the probability of a pixel to be part of a foreground or background. They call a third type of user interaction *draw and erase edges*. This interaction allows the user to influence the weights of the total variation used to minimize the geodesic active contour energy. Binaries for their GPU implementation are available online. We have not incorporated this method in our system, but it is a viable alternative to the other scribble-based algorithms we have incorporated.

A recent paper by Zhao et al. [ZND+11] benchmarks five state-of-the-art interactive segmentation algorithms. The five algorithms are Graph Cuts [BJ01], Geodesic matting [BS07], Random walker [Gra06], Power watersheds [CGNT09] and Structural Interactive Segmentation [NGC+08]. The benchmark was performed on a data set of fifty images that involves a broad variety of challenges in interactive segmentation. We have already covered Graph Cuts by Boykov and Jolly. The Geodesic matting by Bai and Sapiro is briefly mentioned by Unger et al. [UPT+08] in their TVSeg paper as a alternative to their framework. Like all methods discussed in this paragraph, it uses foreground and background scribbles as input to define seed points. It classifies pixels as foreground if the shortest path from that pixel and each seed is a path leading to a foreground seed. They use a geodesic distance for this purpose, defined as the smallest integral of a weight function over all the possible paths from two points. They have extended their framework to work with video containing dynamic backgrounds, and also use their approach to speed up matting algorithms. The Random walker method by Grady works by solving the problem that given a random walker starting at each unseeded location, what is the probability that it first reaches a specific seed point. The random walker is biased to avoid crossing sharp intensity gradients. The label of the most probable seed point is assigned to the starting point of the walker. A quality segmentation is obtained that respects object boundaries. The algorithm starts like many graph-based approach,

where each edge is assigned a weight. The weights correspond to the likelihood that a random walker will cross its corresponding edge. The walking is not actually simulated, but instead calculated. This is possible because the probability that a random walker first reaches a seed point is equivalent to a Dirichlet problem that can be solved. Power watersheds by Couprie et al. generalizes energy minimization algorithms like graph cuts, geodesic segmentation and random walker, and includes watershed in the generalized form. Their generalized model allowed them to discover a new family of segmentation algorithms they call *power watersheds*. They show that the new algorithms are improvement over the segmentation results of the watershed algorithm, while retaining its speed advantage. Structural Interactive Segmentation by Noma et al. uses graph matching to segment an image. An oversegmentation of the input image is built using watershed. It is assumed the oversegmentation delineate accurately the contour of the object to be extracted. Each segment becomes a node in what they call the *input graph*. Another graph called the *model graph* is build in a similar fashion but only the segments that intersect user given scribbles become nodes. The task becomes matching each node of the *input graph* with the nodes of the *model graph*. They also make use of a temporary graph called the *deformation graph* that ensures the final result matches the topology of the *model graph*.

These semi-automatic segmentation algorithm tend to give more satisfactory results than automated approaches, as the users will continue the interaction until they are satisfied. However, in certain cases the amount of interaction becomes taxing for users who desire precise delineation. In the worst case, the interaction in scribble-based approaches degenerates into complete free-hand delineation by the user. This is one motivation behind our alternative interactive segmentation that we present in Chapter 5. Our interactive segmentation solution does not use soft constraints, but instead enables the selection of precise regions directly. We believe this type of interaction is a viable alternative to scribbled-based methods, and is superior when the precision desired is not achieved by a minimal amount of scribbles. However, each approach has its strength and weaknesses. The system we have developed is flexible in the choice of segmentation algorithm used while building a segmentation hierarchy.

2.2.3 Segmentation Hierarchy

We have mentioned the benefits of step-by-step displays in haptic image exploration in Section 2.1.1. In order to get multiple levels-of-detail for an image, one can simply

segment the image at different coarseness. The coarsest segmentation outlines the main objects and is used to get comfortable with the image. The finer segmentation is used to comprehend the important details. Segmenting at different coarseness using an arbitrary method does not guarantee that the shapes of coarser level are still defined at the finer levels. In the haptic image exploration system we have developed, the user controls the level-of-detail (LOD) in real-time. Changing the LOD does not alter the current haptic position of the user. If the boundaries near the current position of the user move as the LOD changes, the points of reference are lost. It is important to retain the boundaries intact to avoid confusion. This explains our interest in a LOD solution that increasingly detail a scene without disturbing the lines defined by the coarser shapes.

Shi and Malik [SM00] mention that graph partitioning is inherently hierarchical. Instead of returning a single *flat* partition, they find it appropriate to return a tree structure corresponding to a hierarchical partition. Recall that their algorithm works from a top-down fashion, iteratively cutting the graph into more partitions. The root of the hierarchy is a single segment covering the whole image, and the leaves are the finest partitions. When combined, the finest partition reconstruct the full image. A parent node in the tree is strictly the combination of its children. A similar type of segmentation hierarchy can be built with a bottom-up algorithm. In that case the leaves are the initial segments, *e.g.*, each pixel. Initially, they form disjoint sets, *i.e.*, they are not connected in the partial hierarchy. A merge operation creates a parent node for the segments involved in the merge. If the algorithm is free to continue until only one segment remains, that segment is a root node of a tree structure representing the complete hierarchy of segments. In both these approach, parent nodes, *i.e.*, coarser levels, are strictly the combination of their children nodes, *i.e.*, finer levels. This algorithm respects our constraint to retain the shape of coarser segments as we increase the level-of-detail.

DeCarlo and Santella [DS02] draw from scale-space theory to build a segmentation hierarchy. They use the hierarchy in a non-photorealistic rendering system for stylizing and abstracting photographs. Their user interaction employs eye-tracking to flag high detail regions. This eye-tracking data is used to give more detail to these regions in their final output. They perform a segmentation at different scales of the input image, building a sequence of segmented image. Each segmentation is done with a fixed parameter in Mean Shift but this choice of segmentation algorithm can be substituted with another. They proceed to build a hierarchy from this sequence by giving sibling relationship to finer scale segments which are contained by the same segment at the next coarser scale. This method is based on work by Ahuja [Ahu96]. This method for building a hierarchy

does not guarantee that coarser segments are strictly made of whole segments at a finer level. Nock and Nielsen [NN04], in their statistical region merging algorithm (SRM), suggested a similar strategy to build a hierarchy of segmentation. Instead of changing the scale of the input image, they change the parameter of the segmentation method.

The gPb-owt-ucm algorithm described by Arbeláez et al. [AMFM10] builds a hierarchy using a contour detector followed by an oriented watershed transform and stores the result in an Ultrametric Contour Map (UCM). The oriented watershed transform (OWT) for image segmentation is introduced by Arbeláez et al. [AMFM09]. It builds an oversegmentation using a contour detector (edge detector) as input. The contour detector is expected to output the probability of an edge at location (x,y) and orientation (θ). Arbeláez et al. encourage using their gPb detector [MAFM08], but left their system generic so that other algorithms like the Canny edge detector [Can86] may be used. OWT works by running a watershed transform [Mey92] on the topology given by the maximum probability across the orientations for each (x,y) location. The seed location are the local minima of these maximum probability. Each catchment basin is a segment bounded by the watershed arcs around it. The advantage of Oriented Watershed is that each point forming the boundaries is given a strength value corresponding to the underlying contour detector probability at that location and orientation. The orientation chosen is the one that best match the tangent line at that point on the boundary. The output of OWT are weighted and closed boundaries stored in a UCM. Given a UCM, the values in the map represent the order of disappearance of the boundary at that position. A *flat* segmentation can be extracted from the hierarchy by using a threshold on the UCM. The threshold value controls the coarseness of the segmentation. Arbeláez and Cohen [AC08] describe how to exploit information from a hierarchy encoded in a UCM in order to allow interactive segmentation. The user inputs marker points. Each position is labeled with the marker that is closest to it in ultrametric distance. In other words, the output segments are the Voronoi cells of these marker points. The ultrametric distance is the maximum weight along the paths between two points. The weights in the map are the order of disappearance of the boundary at each location. The algorithm to compute each distance is based on a front propagation algorithms like Fast Marching or A*. We have developed an interactive segmentation algorithm in our toolset that is based on UCM. The interactive method with UCM is a good example of using a segmentation hierarchy as means to produce an object-level segmentation. In our case however, we are looking at building an complete object-level hierarchy, not simply one *flat* segmentation.

The discussed algorithms so far can produce high-quality results but each level in

the segmentation hierarchy may not follow object contours. Exploring an automatically generated hierarchy can greatly help to produce a desirable *flat* segmentation result but is constrained by the hierarchy of the automated segmentation. In our haptic image exploration system, the user will be exploring the image and controlling the level-of-detail, thus exposing the whole hierarchy. It is therefore important to us that the hierarchy be of object-level quality at every level. The work closest in goals to ours is the painterly rendering method by Zeng et al. [ZZXZ09] who extract a parse tree [ZM06] from an image in an interactive manner. In their method, the input is first processed by Mean Shift clustering [CM02] to get superpixels. The image composed of superpixels are segmented recursively by a scribble-based graph partitioning algorithm [BJ01]. The recursion continues until a certain resolution limit is reached. They extract the parse tree in order to get meaningful leaves which are then labeled using a list of categories. This interactive extraction of the hierarchy can produce the object-level hierarchy that we seek. The method that we propose is based on similar principles that drove Zeng et al. [ZZXZ09] in building their interactive system. We have developed an interactive tool for the authoring of segmentation hierarchy. Since our goal is more general, the method we propose is more flexible in the choice of segmentation algorithms and does not rely on an image grammar.

Image grammars, *e.g.*, spatial random trees [WPW$^+$06] and parse trees [ZM06] combine segmentation with higher-level knowledge. The use of image grammars instead of segmentation is very promising for non-photorealistic animation and rendering because these methods have the goal to derive semantic descriptions of an image. Unfortunately, the state-of-art of these methods still requires user interaction to correct the representation [ZZXZ09, LZL$^+$10]. Note that the painterly rendering method for video of Kagaya et al. [KBD$^+$11] may work without user interaction for video segmentation but it is only based on low and mid-level cues and does not segment at an object-level [BT09]. Also, video segmentation can draw from more features than image segmentation because of motion and the large amount of redundant information.

Our goal of generating and visualizing an object-level hierarchy is related to graph visualization [HMM00, CKB09]. Indeed many authors of publications on segmentation use tools developed for the visualization of hierarchical clustering. However, this is only effective for the top-level of the hierarchy and becomes quickly unusable for the large number of nodes present in an image. More fundamentally, during user interaction we would like to always present the user with an image representing their current effort and not with a tree view. A suitable visualization tool for multi-scale editing is MUSE [FZ98],

however, it is unclear how to adapt this method to hierarchical image segmentation when the goal is to edit the hierarchy itself as in our task.

2.3 Summary

In this chapter, we discussed related work on image exploration using haptics and the preparation needed for haptic rendering. Our effort draws mainly on assistive technology for the blind when it comes to haptic exploration. Section 2.1 covered fundamental principles when developing a relief image, and work done on computer assisted haptic exploration. That section also explains the haptic image features that facilitate comprehension, and puts importance on the benefit of step-by-step displays. We discussed various haptic devices used in the research community, and focused our attention on point-based exploration. The Phantom device, which is the point-based device we have chosen to work with, allows rendering of tactile texture through accurate force feedback. We have presented many image segmentation algorithms in Section 2.2. These algorithms are the basis of our interactive tool for the authoring of object-level segmentation hierarchies. The hierarchy contains the step-by-step image. We will show in this thesis that the system we have assembled allows haptic comprehension of digital images, and present our method for the preparation of these image for haptic exploration.

Chapter 3

Haptic Rendering

In this chapter we present the details regarding the implementation of the haptic rendering in our image exploration system. We begin by introducing in Section 3.1 how the image plane and the segment outlines are rendered. Section 3.2 describes how scaling and panning controls are implemented. The level-of-detail navigation controls are covered in Section 3.3. We explain the approach we have chosen for surface texture rendering in Section 3.4. Various implementation details with respect to our chosen platform are given in Section 3.5. We give a summary of this chapter in Section 3.6.

3.1 Contour Image

In our haptic image exploration system, we render the image on a 2-D plane in a 3-D environment. The user slides the haptic cursor on this plane to explore the image. We restrict the cursor inside a virtual box of dimensions equal to the image. The bottom of this box is the plane where the image lies. This box prevents the cursor from falling off the image at its edges. The box is inspired by the work of Fritz and Barner [FB99] who argue for rendering virtual walls around a 3-D scene to prevent the user from falling off. Figure 3.1 shows how the device is normally held during exploration. It is held like a pen, and pushed down on the virtual image plane. The input to our system is a segmentation hierarchy. Chapter 4 explains the process of building a segmentation hierarchy from a digital image.

The segmentation hierarchy contains the information necessary to render the outline of each of its segments. We store the outline as a polygon with edges encoded in a half-edge data structure. We follow the recommendation of Ramloll et al. [RYB$^+$00]

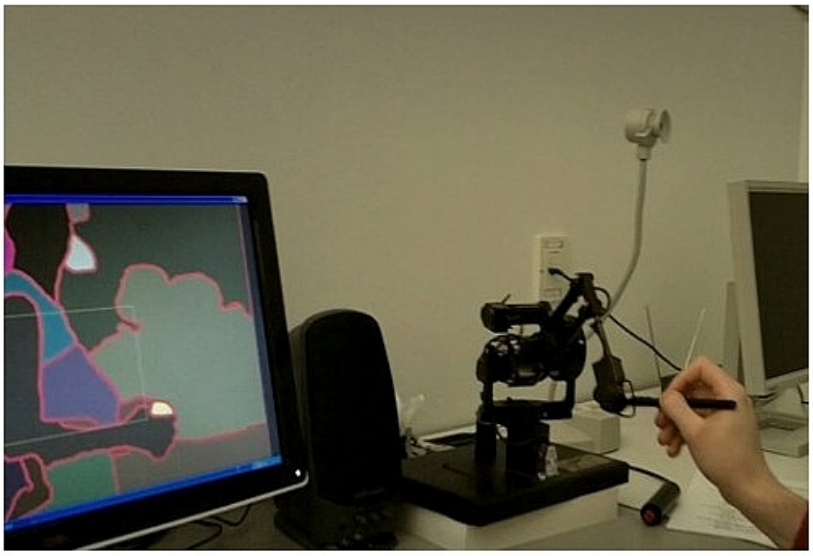


Figure 3.1: Holding the Phantom Device.

who suggest rendering haptic lines as grooves on the Phantom device. Even though we render the image on a 2-D plane, grooves take advantage of the 3-D nature of the Phantom device. When the user slides the haptic cursor on the image plane, the cursor will enter into the groove of a line segment. The groove acts like a track, which can be followed passively. This is in contrast with rendering lines with attraction forces (magnet) which actively attract the cursor and constrain it on its line [FB99]. To escape from a line rendered as a groove, the cursor can either be lifted away from the image plane or pushed away causing it to climb back up the ramp of the groove onto the image plane.

We have implemented the grooves using 3 planar surfaces, one for its bottom and one for each ramps. The shape resembles an aqueduct design, as shown in Figure 3.2 and Figure 3.3. Our original design was in the shape of a "V"' as Ramloll et al. suggest, but with only 2 surfaces, *i.e.*, the ramps, our experimentation revealed stability issues with restricting the cursor in tight corners.

We also chose to render the grooves above the image plane instead of carving them into it. The cursor can enter a groove freely because we do not render haptically the back face of the ramps. This implementation detail simplifies the geometry involved for creating the image plane. However, this choice means that the cursor will jump off the ramp once it is done climbing out of the groove. This should have only a minor influence on the exploration experience since the height of the ramp is negligible.

We have made the height of the two ramps different. The ramp going back into the current segment is lower and hence easier to climb. This reduces the chance of getting off track onto another segment by accident when following the groove.

The most important challenge when rendering haptic lines is the resolution. There is a finite amount of information one can fit into the limited haptic space. This challenge is also true for visual rendering. When an image is scaled down, the picture begins to merge, until the image vanishes in one single point. During haptic rendering, the ramps of two distinct edges can overlap and cross at low scales. This issue can cause the cursor to get stuck, and enter an unstable force feedback loop. Magnifying the image mitigates the severity of the overlapping ramps because we render the grooved lines at the same size no matter the scale.

We render only the grooves around the current segment, which means the overlap can only arise when edges from the same segment are close together. Rendering only the current segment also solves issues involving branching and crossroads within the groove path because a single path for the cursor to follow around the current segment is rendered. For this reason, falling from the opposite side into a grooved line guides the cursor around the opposite segment.

An alternative to our design would have been to use polygon offsetting. A polygon offsetting algorithm can compute a smaller inner polygon inside a segment that follows its shape from a specified distance. These inner polygon are rendered as islands separated by grooves. Computing the polygon offset is costly, *i.e.*, must be done offline for every segment, every supported level-of-detail and every supported scaling factor. We have not used this solution because its behaviour at smaller scale is also problematic. Instead of overlapping lines, segments would erode and finally disappear, changing the perceived shape of the segments. Wider sections in the groove would also appear for the same reason, and would not restrict the cursor in a given path well.

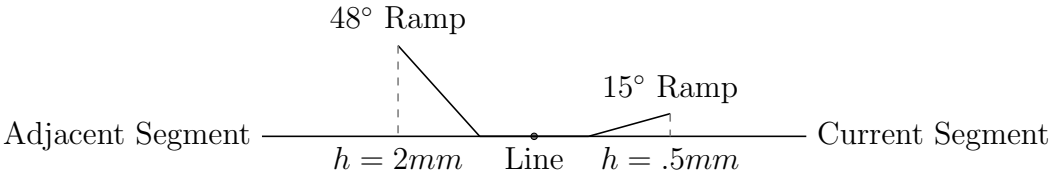


Figure 3.2: Front view of the groove line design.

As an alternative of rendering lines as grooves, we experimented with rendering vibrations when the cursor crosses an edge. Petit et al. [PDL+08] used this concept in their

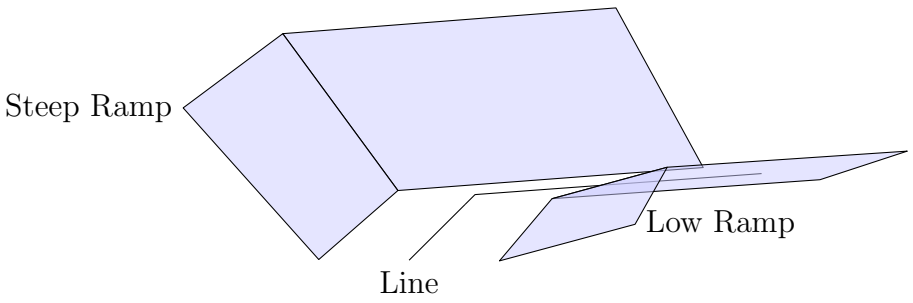


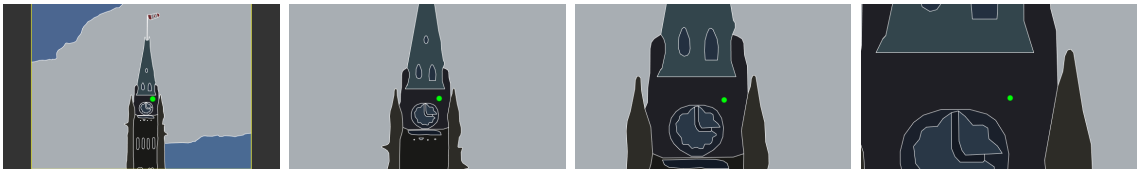
Figure 3.3: Perspective view of groove line design.

system for haptic exploration of schoolbook material. After testing, we prefer rendering lines as grooves because grooves make it easy to follow the outline of shapes, whereas a texture impulse simply tells the user that they have crossed a line.

3.2 Navigation

In order to explore large images, scaling and panning functionalities are necessary because the physical workspace of our device is limited.

We have implemented scaling navigation in our system. The purpose is to allow the user to magnify the image to a scale they are comfortable with. The image being explored can be scaled to fit completely in the haptic workspace. However, the outline of small segments is hard to follow at that scale. Figure 3.4 shows an image at four different scales, and only the content available in the haptic workspace is shown. Once magnified, small segments become bigger in the haptic workspace and their outline can be followed more easily.



(a) Fit to workspace (b) Magnified 2X (c) Magnified 4X (d) Magnified 8X

Figure 3.4: Exploration at four different scales. Only the content available in the haptic workspace is shown.

In our system, scaling controls are mapped to the up and down arrow keys of the keyboard. The scaling transformation is centered on the current position of the cursor,

such that the logical position of the cursor remains the same after scaling. This type of scaling introduces an shift when restoring back the scale to its original value after having moved the cursor. If the image is scaled to fit the boundary of the haptic workspace, and then is magnified followed by movement of the cursor, once the image is restored to its original scale its boundary will not fit the haptic workspace anymore. Scaling, moving, and scaling back will effectively pan the image. Even though this can create confusion, alternative solutions that would alter the logical position of the cursor during scaling are confusing as well. The scaling paradigm that preserves cursor position is found in many applications, *e.g.*, Google Maps (web mapping) or Inkscape (vector graphics editor). However, it is not a ubiquitous design, as Eye of Gnome (image viewer) favours centering the image when the scale makes the image smaller than the output frame instead of preserving the same relative position of the mouse pointer with the image displayed.

We have mapped panning controls to the button on the pen of our haptic device. Panning is achieved by holding the button on the stylus of the Phantom and moving the cursor. We have chosen to implement panning as a dragging action, similar to the way one can move a piece of paper by dragging it with one finger pressed on. While the button is held, moving the cursor no longer affects the logical cursor position but instead moves the image within the haptic workspace. The logical cursor position stays constant during panning. An alternative to this type of panning would be to move the image, but not the cursor along with it, which would change the logical position.

The largest amount of panning achievable in one action is the length of the haptic workspace itself. It it possible to get trapped in a narrow space if the image is completely dragged out of the haptic workspace since the cursor is constrained both by the box around the image and the physical limits of the workspace. This is a possibility with this design but it rarely occurs in practice and can be easily solved by a reset button.

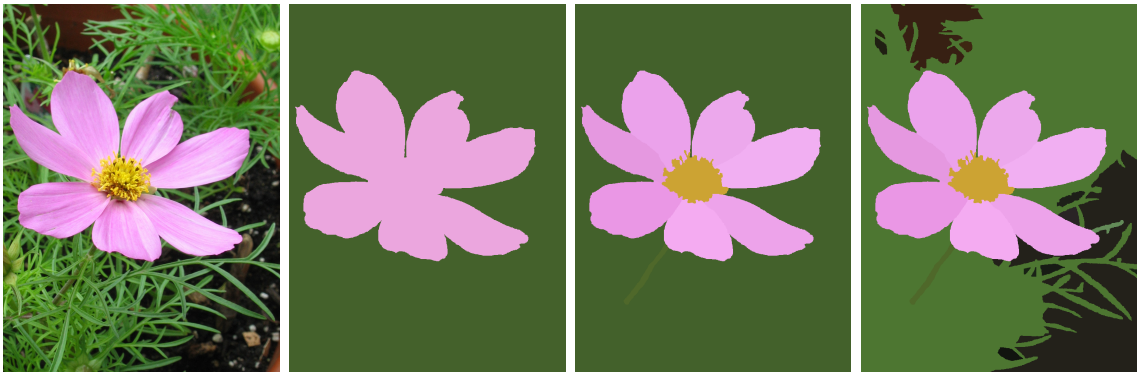
Preserving logical position during scaling or panning is especially important during haptic exploration because the exploration is done with a single point. The user would need to spend time exploring the new position's surrounding for familiar cues if the logical position would not be preserved.

3.3 Level-of-Detail

Exploring an image at different levels-of-detail is a key feature of our haptic image exploration system. The feature is inspired by the work of Edman [Edm92] regarding

step-by-step displays of physical relief images. We expose the segmentation hierarchy of the image to the user in the form of global levels-of-detail for the image. The lowest level-of-detail defines the outline of coarse segments representing the most visible objects in the scene. The highest level-of-detail defines very fine segments depicting the less important details in the image. Chapter 4 covers how we build the segmentation hierarchy.

The control we give the user over the level-of-detail is linear from coarse-to-fine. To reduce the chance of having situations where increasing the level-of-detail would not affect nearby segments, we have opted to use level-of-detail presets for a given image. The use of presets is in contrast to giving the user unit size controls over the level-of-detail , *i.e.*, one more segment at a time when increasing the level, or one less segment at a time when decreasing it. Appropriate presets for the level-of-detail are decided when preparing the image for haptic exploration. If presets are not supplied, we use a logarithmic scale for the number of segments in each level-of-detail for a given segmentation hierarchy. Figure 3.5 shows a segmentation hierarchy at three different levels-of-detail.



(a) Source (b) 2 segments (c) 10 segments (d) 26 segments

Figure 3.5: Segmentation hierarchy at three different levels-of-detail.

An alternative way to navigate the segmentation hierarchy is to allow the user to go down the hierarchy for the current segment node only. We have not experimented with this exploration paradigm. We argue that keeping track of which part of the image is at which level-of-detail is too complex for our image domain, *i.e.*, general imagery. The consequences of decreasing the level-of-detail for the current segment only is also harder to grasp than globally increasing the level-of-detail. This is why we have chosen to work with global discrete level-of-detail.

3.4 Texture

Edman [Edm92] remarks that contrast in texture in traditional relief display is important for comprehension. Haptic exploration has a narrow field-of-view compared to visual exploration. We seek to communicate to the user extra information about the global context of an image via ambient encoding. The extra information enlarges the field of view. In a traditional display for the blind, reaching a section that feels furry can instantly inform the user given a context that they have reached the top of the head of a person with hair. We are using tactile textures to encode ambient information. The ambient information we have chosen to communicate through texture is the relative size of the current segment.

We generate decaying sinusoidal waves to be used as tactile textures. Our design is inspired by the work of Okamura et al. [ODH98]. The formula for the wave is:

$$F(t) = Ae^{-bt}sin(wt).$$

We set the amplitude (A) to a constant, and the sampling period (t) to our haptic device period ($1/1KHz$). The period (w) of the wave tells the user how big the current image segment is relative to the whole image. In our system, larger segment have larger wave period (w).

We implement the wave with a second order infinite impulse response (IIR) filter. The formula for the filter is

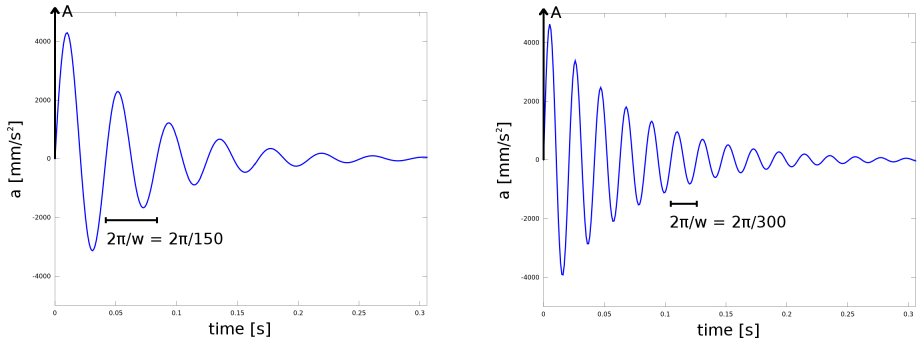
$$y_n = \left(\begin{array}{c} b_0x_n + b_1x_{n-1} + b_2x_{n-2} \\ -a_1y_{n-1} - a_2y_{n-2} \end{array} \right) / a_0,$$

where y_n is the output, x_n the input, x_{n-i} and y_{n-i} are the previous inputs and outputs respectively. The coefficients are computed given the decaying sinusoidal parameters A, b, w and t as follow:

$$\begin{aligned} b_0 &= 0 \\ b_1 &= Ae^{-bt}sin(wt) \\ b_2 &= 0 \\ a_0 &= 1 \\ a_1 &= -2e^{-bt}cos(wt) \\ a_2 &= e^{-2bt} \end{aligned}$$

We trigger the filters by feeding a 1 for the first input value, followed by zeros. We stop feeding zeros to the filter when the amplitude of the decaying sinusoidal has reached 1% of its original strength. We know this stop condition to happen when $n = -ln(0.01)/bt$.

The cursor triggers the IIR filter of the current segment when traveling on the surface of the image plane. The distance between impulses is another parameter. We set impulses further apart the larger the segment. To reduce the chance of having adjacent segment of similar size feel the same, we randomized the attenuation factor (b) for each segment. Figure 3.6 shows two decaying sinusoidal waves generated with different parameters.



(a) $w = 150$ (b) $w = 300$

Figure 3.6: Two decaying sinusoidal waves generated with $F(t) = Ae^{-bt}sin(wt)$.

The texture method we are using does not produce longitudinal forces, only perpendicular forces. In other words, there is no friction from the texture model. We have chosen to implement a semblance of friction to go along with the texture impulse force. We implemented friction by adding a force opposing the x/y movement direction. The magnitude of this force is a factor of the output magnitude of the texture model. Our texture rendering approach is an adaption of the method of Guruswamy et al. [GLL11]. In their method, they measure the acceleration profile of a physical texture, and reproduce the texture profile virtually. Though we chose to use procedurally generated decaying sinusoidal waves in our system, our IIR filter implementation is the same.

The parametric nature of procedural textures extends intuitively to encoding information. It is the reason we have chosen to use procedurally generated decaying sinusoidal waves over using scanned profiles.

We have experimented with a second approach to render textures using the HLAPI of the OpenHaptics API by SensAble [Tec], which supports texture properties for each rendered surfaces. The properties available are stiffness, damping and friction. Stiffness controls how hard the surface feels. A low stiffness allows the cursor (but not the proxy [ZS95]) to penetrate temporarily in the surface. Non-zero damping and friction parameters gave friction resistance to the surface but made our device produce high

pitch noises. The noises were heard even at low friction and damping values. Unlike our texture implementation, these forces seemed to be strictly longitudinal and did not feel realistic.

3.5 Implementation Details

We have used the OpenHaptics API by Sensable [Tec] to interface with our haptic device. OpenHaptics is organized into two parts: The Haptic Device API (HDAPI), and the Haptic Library API (HLAPI). The HDAPI gives low-level access to the device, *e.g.*, polling the physical position and button states of the device, or writing the desired output force. The HLAPI is a higher-level library that implements collision detection and collision response with shapes through the use of a proxy implemented with the god-object algorithm of Zilles and Salisbury [ZS95]. The design of OpenHaptics makes haptic rendering similar to graphics rendering with OpenGL. The geometry used by the HLAPI are actually given by OpenGL commands that are normally used for graphics rendering. Geometry clipping and back-face culling are example of features supported by the HLAPI leveraged from OpenGL.

The fixed-pipeline profile of OpenGL explicitly supports two transformation matrices for converting the coordinate spaces of the models to the coordinate space of the screen. The matrices are the MODELVIEW matrix, and the PROJECTION matrix. In HDAPI, two additional transformation matrices are supported to convert the coordinate space of the device to the coordinate space of the models. They are the TOUCHWORKSPACE matrix and the VIEWTOUCH matrix. Figure 3.7 shows the HLAPI coordinates transform matrices order.

The HLAPI offers utility methods to fit the haptic workspace into the visible scene. These method have to be used with care. For example, if the projection matrix uses perspective, then the haptic workspace will be mapped with perspective. This is not desirable because perspective projection is a visual phenomenon and does not apply to haptics. The rear end of a cube is not physically smaller than its front end, it only appears that way because of perspective viewing. Therefore orthographic rendering must always be used when calling the utility methods supplied by HLAPI.

The physical workspace of our Phantom device is not rectangular, but is a complex shape. We render artificial limits to constrain the physical position into a rectangular volume. We chose to render the image plane at a depth position that would maximize the available width and height. Our actual workspace dimension with artificial limits

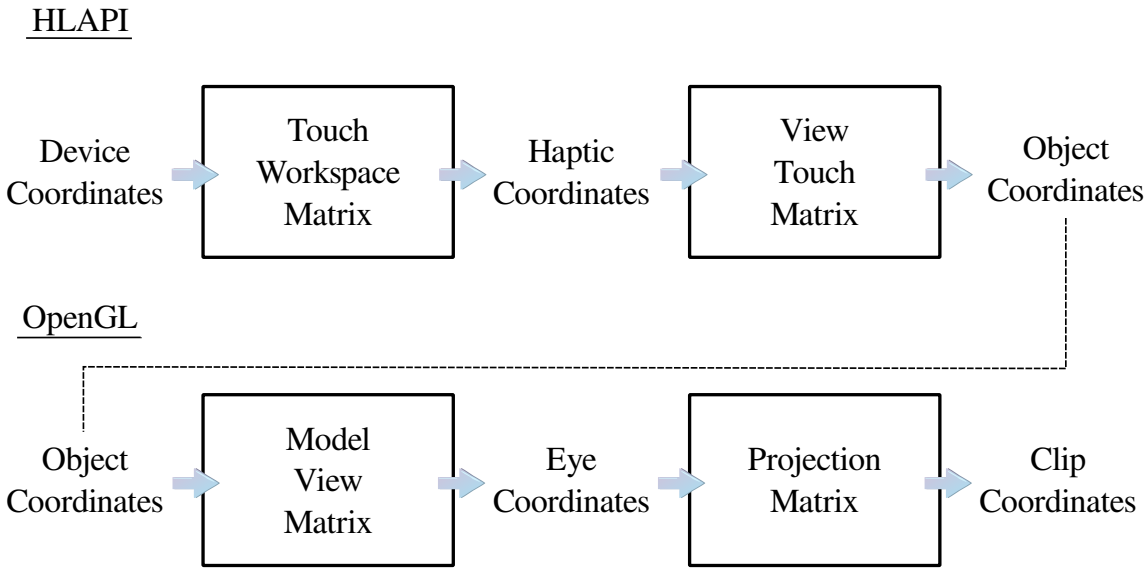


Figure 3.7: HLAPI and OpenGL coordinates transform matrix.

is 30.0 cm wide by 17.8 cm high. Note that this is wider than the marketed workspace width of our device of 25.4 cm. The depth range is not very important in our application, *i.e.*, only the grooves use depth at a negligible scale.

The HLAPI proxy implementation manages collision response for the image plane, artificial workspace limits and the grooves ramps. We add texture output to this force when textures are enabled. In order to protect the device we cap the output force amplitude with a maximum amount set with experimentation. If the force was indeed capped in the current haptic frame, we generate an audible warning.

When using OpenHaptics, one must be aware of the multiple threads that exist in the framework. A thread is created for the low level haptics (HDAPI). This thread must run at 1kHz for the haptic rendering to be smooth. Another thread is created for the collision detection (HLAPI). This thread normally runs at 100Hz. Finally, there is the application thread. It is critical that data common to multiple threads be shared carefully. The HLAPI information query functions are safe to be called from the application thread. For the haptic thread data, the OpenHaptics API supplies scheduling functionality to run tasks on the haptic thread. The application thread must schedule a safe data transfer time in order to receive data from or send data to the haptic thread. Failing to synchronize data transfer properly can result in undefined behaviours, *e.g.*, reading erroneous device position.

We use short duration audio signals to inform the user when an attempt is made

to navigate to an invalid state, *i.e.*, increasing the level-of-detail when already at the maximum preset, or zooming out when already at the minimum scale. The frequency of the beep is different compared to the signal sent to notify the haptic force output value was capped. These audio beeps would normally be sent from the application thread asynchronously. However, multi-threading issues arise with respect to the haptic thread on our Windows API platform when doing so. In order to ensure a stable 1kHz haptic thread loop, we send audio signals from the application thread synchronously. The application thread is responsible for sending new geometry, but since our scene changes infrequently, and our audio signals are short, the application thread in this scenario is the most appropriate thread to do synchronous calls, *i.e.*, blocking calls.

To ensure good product quality, we have implemented several runtime visualization tools. In debug mode, thread loop speed indicator warn us of any anomalies. We display a force graph to visualize the amplitude of the output force of our device. Several parameters, *e.g.*, height of grooves, are changeable at runtime to experiment with different settings. Figure 3.8 shows a screen-shot of our haptic image exploration system in debug mode. In this mode, the image is rendered visually as well as haptically. The yellow graph overlaid at the bottom of the screen shows the amplitude of the force feedback output. The graph is noisy because textures are enabled. The amplitude value is mostly in response to the force applied by the user on the image surface. The red text displays the frame per second of each threads, the cursor position before and after transformation, the proxy position, the workspace limits, the current translation and scaling amount and the status and parameters for various features supported by our system. The yellow lines delimit the haptic workspace.

3.6 Summary

In this Chapter we have presented how we render images in our haptic image exploration system. The system renders the image on a 2-D plane, but uses 3-D to render grooves on the outline of the segments defined by the segmentation hierarchy. We support scaling and panning to explore large and detailed images, and linear control of the level-of-detail of the image. The segments have textures applied on their surface which are procedurally generated with information regarding the size of the segment. We concluded the presentation of the system with various implementation details who could help someone to build a similar system. We present and give the results of a user study we have conducted to evaluate our system in Chapter 6.

Figure 3.8: Our haptic image exploration system running in debug mode. In this mode, the image is rendered visually as well as haptically. The yellow graph shows the amplitude of the force feedback output. The red text displays the frame per second of each threads, the cursor position before and after transformation, the proxy position, the workspace limits, the current translation and scaling amount and the status and parameters for various features supported by our system. The yellow lines delimit the haptic workspace.

Chapter 4

Image Processing for Haptic Display

In this chapter we present the procedures in the processing of digital photographs for haptic rendering. We begin by introducing in Section 4.1 the concept of *object-level hierarchy*. Section 4.2 follows by presenting the tool we have developed for the authoring of an *object-level hierarchy*. The design of the data structure we use to store the hierarchy for haptic image exploration is covered in Section 4.3. We list in Section 4.4 the steps we follow to transform a raster-based segmentation hierarchy into smooth vector graphics before haptic image exploration. We give a summary of this chapter in Section 4.5.

4.1 Object-Level Hierarchy

Based on the description of Edman [Edm92] on step-by-step displays that we mentioned in Section 2.1.1, we seek to produce a coarse-to-fine sequence of segmentations for a given image. The outline of the main objects in the scene should be apparent at coarser levels, while minor details of the scene should only be apparent at finer levels. Introducing the main outline of an object at a coarser level than its inner details is important for comprehension. The goal of step-by-step displays is to break down the exploration task in multiple abstraction levels. The coarse levels are used to locate the core objects and discover their relative location in the image. The finer levels are used for grasping the inner details in each object. A hierarchy that respects these guidelines is called an *object-level hierarchy*. For the objects to be recognizable, their outline must also be traced with precision. The point of reference defined at a coarser level must be present across the finer levels in the sequence for the step-by-step display to be congruent. In this chapter, we extend the definition of *object-level hierarchy* to also mean a hierarchy with precise

delineation of objects at all levels, and with a strict parent-child relationship, *i.e.*, the region covered by a parent segment completely and only covers the regions of its children.

We have introduced segmentation algorithms that can generate a hierarchy automatically in Section 2.2. When the intended output of an algorithm is a single *flat* segmentation, the algorithm may still build a hierarchy as a processing step. In the case of a region growing algorithm, the output of the algorithm is the coarsest level computed. In our system, creating the hierarchy is a goal in itself because the hierarchy will be exposed to the user during haptic exploration. The hierarchy is not simply a means to achieve a *flat* segmentation.

Segmentation at the *object-level* is not trivial. Figure 4.1 shows a UCM map generated by the gPb-owt-ucm algorithm [AMFM10], and two *flat* segmentation at different threshold. One may argue that Figure 4.1 (c) is an acceptable *object-level segmentation*, but (d) which shows a coarser level-of-detail in the same UCM hierarchy is inadequate as an *object-level segmentation*. It is inadequate because the hula skirt of the boy is visible without the boy. The hula skirt is a detail of the boy, therefore the boy should appear first in the hierarchy. Figure 4.1 (e) was generated differently and presents what is acceptable as an *object-level segmentation* at a coarser level.

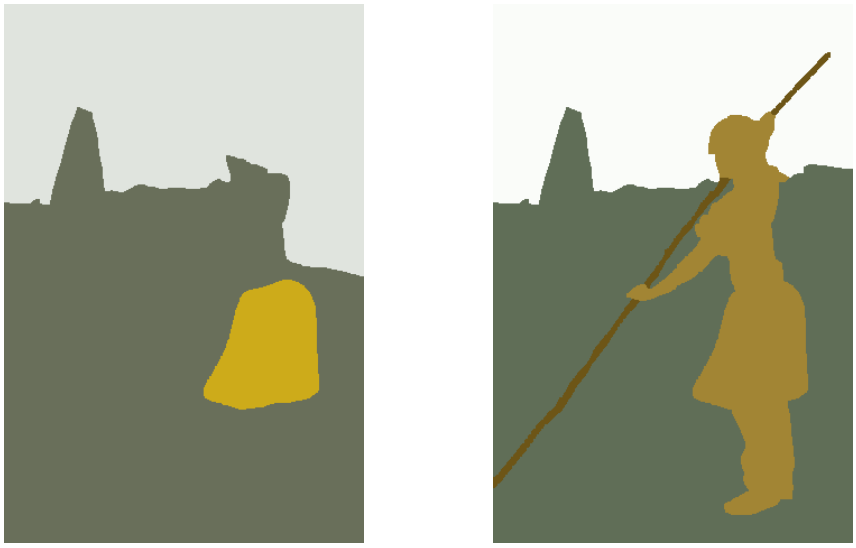
We give an example of an hierarchy that does not delineate objects accurately on all levels in Figure 4.2. The hierarchy was build automatically using statistical region merging [NN04], which is not designed to keep a strict parent-child relationship in the hierarchy. In Figure 4.2 (b), we show one level in the hierarchy where the core outline of the horses is acceptable. We show another level in the same hierarchy that poorly defines the contour of the same horses in Figure 4.2 (c). Note that in this examples, the inner details are not very meaningful.

An *object-level hierarchy* introduces the outline of objects at coarser levels than the outline of their inner details, and with our extended definition is strict and has precisely delineated objects at every level. These two examples show how difficult it is for an automatically generated hierarchy to be *object-level* across all its levels.

4.2 A Tool for Authoring Object-Level Hierarchy

Segmenting an image at an *object-level* is an ill-posed problem. This has been shown by the human delineated ground truth in the Berkeley Segmentation Data Set [AMFM], which varies with the individual subject. Inconsistent level-of-detail between subjects is one factor to explain the different results and motivates us to to seek an *object-level*

(a) Source (b) UCM (c) Medium-Fine Level



(d) Coarse Level (e) Object-Level Coarse Level

Figure 4.1: Automatic hierarchy *object-level* assessment.
(c) An acceptable *object-level* fine level-of-detail. (d) Coarse level-of-detail that is not *object-level*. (e) What would have been a *object-level* coarse level-of-detail.

(a) Source (b) Fine Level (c) Coarse Level

Figure 4.2: Automatic hierarchy accurate delineation assessment.
(b) An acceptable delineation of the horses. (c) A poor delineation of the horses.

hierarchy. We propose an interactive authoring tool to give each individual user means to achieve their view of an appropriate hierarchical *object-level segmentation* for a given image. The interactive flow of our tool shares similarity to the one described by Zeng et al. [ZZXZ09] discussed in Section 2.2.3. Zeng et al. extract a parse tree interactively from an image with the end goal to produce painterly rendering.

The tool we have developed follows a top-down approach. The first step in our method is to extract the main objects in the image, e.g., foreground objects and a background object. Working from those objects, the principal interaction performed by the user is to recursively select coarse segments that need more details. The details of the selected segment will emerge with further segmentation. The core steps during our proposed interaction are:

1. Perform a coarse *object-level segmentation* of the image.
2. Recursively add detail to objects by segmenting them at a finer *object-level*.
3. Optionally fall back to an automatically generated hierarchy for specific regions to reach a fine level-of-detail.

In our tool, the choice of the segmentation algorithm is available to the user at each step. This allows the user to choose an interactive segmentation algorithm for precise generation of details, or a fully automatic segmentation algorithm when the author is confident that the resulting details will correspond to the author's goals. We believe that giving the choice of the segmentation algorithm to the user improves segmentation results. Marker-based interactive segmentation approaches are powerful, but not always the best answer to every situation. Our proposed interactive segmentation method by *region selection* presented in Chapter 5 is efficient when precision is not achieved easily

with markers. Automatic segmentation is well suited for situations where getting similar results using interactive segmentation would require too much user effort.

The system described by Zeng et al. [ZZXZ09] aims at labeling the leaves into categories and hence a user would stop recursively splitting segments when they reach a certain level. In our case, we support the author in continuing the recursive segmentation until a very fine level-of-detail if desired. In order to quickly include very fine levels, we give the option of generating an hierarchy automatically for a selected segment. At a fine enough level-of-detail, automated segmentation results are not easily distinguishable from interactive results. In our application of haptic image exploration we are satisfied with moderate amounts of details, and we do not need very fine levels of detail. We designed our tool keeping in mind a broader class of application because a quality *object-level hierarchy* can be beneficial in many computer vision application.

In our tool, the segmentation algorithm chosen at each step will process only the region covered by the selected segment. The corresponding region in the source image is used as input to the algorithm, though optionally the source content may have been preprocessed into superpixels. Since the selected region has an arbitrary shape, our segmentation routines supports non-rectangular input. For arbitrary shaped input, a solution we use is to fill a rectangular bounding box with an extreme flat colour, *e.g.*, bright pink, before copying-in the data of the region we want to segment. The idea is that the extreme colour used for out-of-bounds pixels will discourage the algorithm to merge those pixels in. This solution worked also for an external library algorithm that we use (the OpenCV watershed implementation), but it may not work with other libraries depending on the algorithm.

We use a tree data structure to represent the hierarchy in memory while it is being built by the user. The root node of the tree is the parent of the segments created in the first step of interaction, *i.e.*, coarsest object-level segmentation. Nodes in the tree may have multiple children since we do not restrict the number of segmented details that can be added when segmenting further. When the user is done building the hierarchy, the leaves of each branch of the tree are the segments that are visible at the finest level-of-detail. As we will describe in Section 4.3, we will be flattening the tree structure into a merge-event timeline data structure. We record the order in which each segment was further segmented and store it in this data structure.

Our tool for authoring *object-level hierarchy* encourages using an effective segmentation algorithm for each segmentation task. This is most effective if the user knows which algorithm is the most adequate for a given situation. We acknowledge that the

user must be an expert in segmentation to make the best choice of algorithm and setting its respective parameters correctly. We argue that this is also true for other simple tasks in general image manipulation programs. Rescaling an image in such software is simple, but expert knowledge about colour modes, interpolation algorithms and image format quickly becomes necessary when the control over the final result is critical. To mitigate this problem, we suggest typical default values for each parameters of the segmentation algorithms that we support, and allow the user to undo the most recent segmentation.

Future plans to improve our tool include giving the users advanced control on the tree. They would be allowed to cut off branches (random-access undo), merge siblings (if they are spatially connected), lift up child nodes (convert a child to sibling), and modify the order used when flattening the tree into a linear structure. The *object-level hierarchy* we have generated in Section 4.2.1 and in Chapter 6 were authored without these advanced features.

Figure 4.3 gives screen-shots of our tool. We show in Figure 4.3 (b) the result of the first coarse segmentation. The segments serve as the base for recursively adding details. In Figure 4.3 (c), the foreground segment containing the women and her baby was chosen for further segmentation. The result of this segmentation is shown in Figure 4.3 (d). Figure 4.3 (e) shows more details in the other foreground objects. Finer details were added using automatic segmentation to the other foreground objects, and the results are shown in Figure 4.3 (f). The face of the women was selected for further segmentation in Figure 4.3 (g).

4.2.1 Examples of Object-Level Hierarchical Segmentations

We give two example results of *object-level* hierarchical segmentations. In our first example shown in Figure 4.4, we are authoring an *object-level hierarchy* for image 302022 of the BSDS500 [AMFM] shown in (a). In Figure 4.4 (b), we show the coarse *object-level segmentation* that serves as the top level of our hierarchy. This outline was extracted using the interactive marker-based watershed algorithm [Mey92]. Figure 4.4 (c) through (e) show the detail that were added recursively to objects by segmenting them at a finer *object-level* with different algorithm at each step. The details in Figure 4.4 (c) were extracted using the region selection interaction we present in Chapter 5. In Figure 4.4 (d), we used Mean Shift to extract the details of the shirt. This is an example where automated segmentation can identify many segments that would be tedious for a user to segment interactively. The new details in Figure 4.4 (e) were extracted using the same

(a) Source (b) Base

(c) Recursive segmentation (d) Segmentation results

(e) Segmentation results (f) Fine details (g) Recursive segmentation

Figure 4.3: Screen-shots or our tool for authoring *object-level hierarchy*.

watershed algorithm used in the initial step.

(a) Source (b) Foreground / Background (c) Head / Shirt (d) Shirt Pattern (e) Face / Hair

Figure 4.4: Authoring of an *object-level* hierarchy (man). (b) Foreground extraction using Watershed. (c) Shirt separation using region selection interactive segmentation. (d) Shirt pattern extraction using Mean Shift. (e) Separate hair from face using Watershed.

Figure 4.5 shows a second example. In this example we are authoring an *object-level hierarchy* for image 226043 of the BSDS500 [AMFM] shown in Figure 4.5 (a). The image was first preprocessed into superpixels. We have used interactive segmentation using region selection at each step. We began by performing a coarse *object-level segmentation* of the scene. Figure 4.5 (b) shows the result of the coarse *object-level segmentation* that served as the top level of our hierarchy. In Figure 4.5 (c), we show the coarse details that we added to the front runner. We chose to start by separating the cloth and hair from the body. Figure 4.5 (d) shows the same kind of details for all the runners. We continued to recursively add detail. We show in Figure 4.5 (e) the facial details of the front runner. More detail was added to the clothes of each runner as shown in Figure 4.5 (f) and (g).

4.3 Merge-Event Timeline Data Structure

Hierarchical structures are commonly stored as trees. Shi and Malik [SM00] suggest returning a tree structure representing the segmentation hierarchy created with their normalized cut algorithm. In Section 2.2.3 we have discussed how to build a tree to store a segmentation hierarchy. A tree structure offers the advantage of non-linear exploration of the hierarchy. For a simpler linear exploration, a tree can be explored in level order.

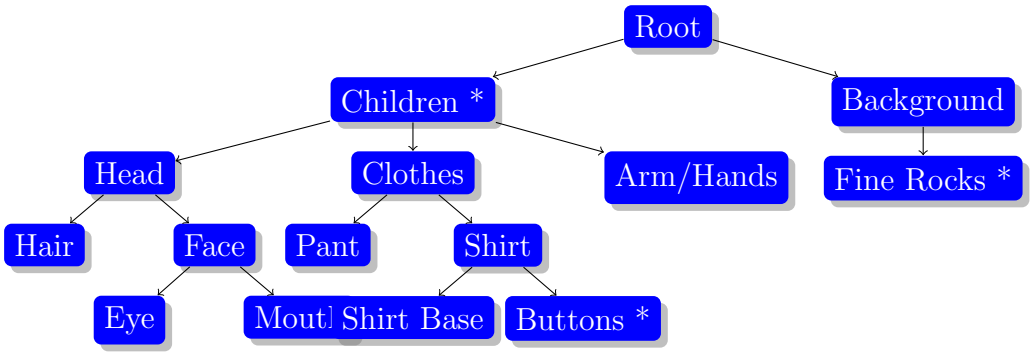
(a) Source (b) Runner's Outline (c) Skin / Clothes (d) Skin / Clothes *

(e) Face detail (f) Bib detail (g) Details *

Figure 4.5: Authoring of an *object-level hierarchy* (runners).

However, the level order of a segmentation tree structure is not always interesting. We give an example of this in Figure 4.6. Because the background is simple in this example, the finer detail of the background are closer to the root than coarse details in the foreground objects. The level order traversal of the tree will show those fine details early, at a level-of-detail that is supposedly coarse.

(a) Source



(b) Hierarchy

Figure 4.6: Segmentation Hierarchy stored in a tree structure.

We have designed an alternative data structure to store a segmentation hierarchy. In addition to interactive-rate linear exploration and raster independence that the tree structure can provide, our design goals also include the ability to specify the linear exploration order during authoring of the hierarchy. Our solution is a merge-event timeline.

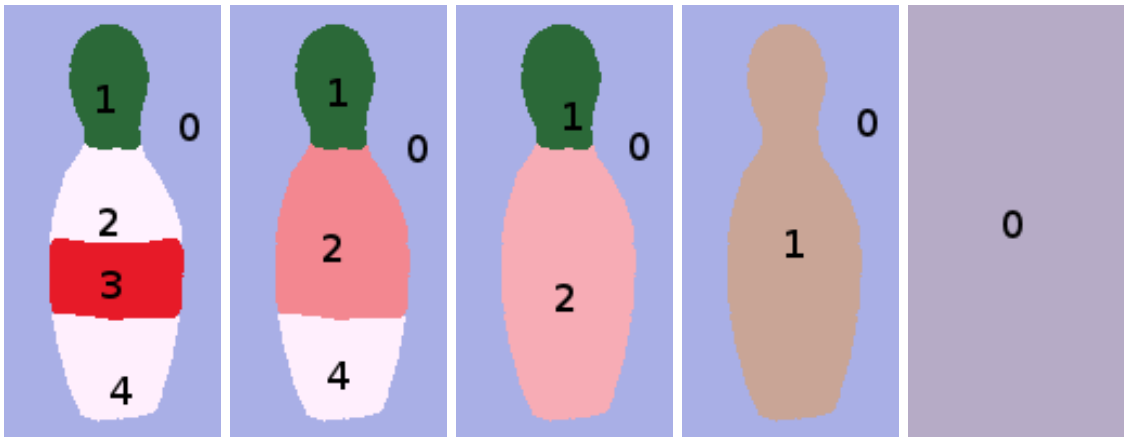
At each iteration of a region growing algorithm two segments are merged together. Storing this merge information is enough to store the hierarchy. We initialize the merge-event data structure with an original segmentation S_0 containing n segments indexed through $[0, n)$. Typically, n is the number of pixels in the image, unless the image is already segmented to some degree, e.g., grouping similar colored pixels into superpixels as in the approach by Arbeláez and Cohen [AC08] and others. For our data structure the segments in S_0 may be stored in any form, e.g., pixels of a raster image or vector graphics, as long as the segments are identified by indices.

We record the two segments that are merged at each step as governed by a segmentation algorithm. We use discrete steps to represent the order of operations which we call a timeline. There can be a maximum of $n - 1$ merge operations since there are n original segments. The merge-event data structure must therefore keep track of up to $n - 1$ discrete steps. We store each merge operation in an array of that size where the index represents the step a specific merge occurred. We do not create new ids for parent nodes formed by merging two child nodes. We instead arbitrarily choose the segment id of one of the children as the id for the parent as well. This keeps the structure efficient. The data structure then tracks the segment id for the parent and child based on the fact that the id of the child will never be referenced again after the merge step. For convenience we chose to record the colour of the newly formed segment as well although this is not absolutely necessary since the colour could be recalculated using S_0. We store this extra information as the rendering properties of the segment.

Figure 4.7 shows a simple hierarchy stored in a merge-event timeline structure. The numbers drawn on Figure 4.7 (a) to (e) are the updated segment ids at that time. Only S_0 shown in Figure 4.7 (a) and the merge-event table shown in (f) are stored.

The merge-event timeline data structure retains the characteristic of the tree data structure and contains no spatial information regarding the original segments. This means that the segments in S_0 do not have to be pixels on a raster. For example, we store S_0 as polygonal outlines in a half-edge data structure with sub-pixel precision in our file format for haptic image exploration. The segments in S_0 can be converted to different formats after the hierarchy has been formed, as long as the identifier for the segments are not altered. We employ this property to vectorize at sub-pixel precision the outline of a raster-based hierarchy. In contrast to the tree, the merge-event timeline has a specific linear traversal order. A tree structure needs additional information if linear traversals in level-order, in-order, pre-order, post-oder, etc. are not adequate. However, the merge-event timeline does not support non-linear traversal.

The following algorithm describes the steps necessary to convert a n-tree structure to our merge-event timeline. The leaves of the tree act as the initial segments of S_0. Sibling leaves are merged together in pairs to make parent nodes. Which group of siblings are merged is selected arbitrarily. If there are more than two children under the same parent, the order the siblings merge is also chosen arbitrarily. The merge order forms the merge-event timeline data structure. The complexity of the conversion is $O(n)$. A region growing segmentation algorithm directly produces a merge-event timeline. The structure encodes efficiently the order of importance of the merges. In Section 4.2, we



(a) S_0 (b) S_1 (c) S_2 (d) S_3 (e) S_4

t	0	1	2	3
parent	2	2	1	0
child	3	4	2	1
color				

(f) Merge-event timeline

Figure 4.7: Segmentation Hierarchy stored with a merge-event timeline. Only (a) and (f) are stored.

mentioned that we store the user defined hierarchy in memory using a tree structure. We flatten this tree into a merge-event timeline using the steps we just described. In this case, when flattening the tree, we choose the group of siblings in the reverse order that they we created by the user such that the linear traversal will be identical to the order it was built.

4.3.1 Visualization Algorithm

The merge-event timeline data structure allows efficient rendering for interactive visualization of the hierarchy. We compute different look-up tables to visualize the hierarchy corresponding to a specific time step t. Using these tables, we can get the segment index and its rendering properties at step t for any original segment of S_0 and produce an image showing the segments of S_t. The data necessary to build the look-up tables are the merge-event timeline and the finest segmentation level S_0 . An example merge-event timeline is shown in Figure 4.7 (f). We build a segment look-up table by traversing backwards from $[t-1, 0]$ benefiting from dynamic programming. In this look-up table, each segment points to its oldest ancestor instead of simply to its direct parent segment. Only one pass is required to calculate this look-up table. We also build a rendering properties look-up table. We first initialize the rendering properties table using the colour values from S_0. Then we overwrite the colour of the parent segment at each step traversing from $[0, t-1]$ with the blended colour value that we stored earlier during construction of the hierarchical segmentation. Note that this is comparable to building lookup tables using pre-order and post-order traversal with a tree data structure. The difference is, as was shown in Figure 4.6, that building look-up tables to visualize a given level-order is not always pertinent.

Figure 4.8 illustrates the steps in building the two look-up tables necessary for rendering S_2 from S_0 and the merge-event timeline of Figure 4.7. First the segments are initialized with self-referencing pointers. We iterate from $t-1$ to 0. At iteration $t-1 = 2-1 = 1$ in the timeline, segment 2 is merged with segment 4, and segment 2 is chosen as parent. Now segment 4 points to segment 2. We update the table accordingly. At iteration $1-1 = 0$ in the timeline, segment 2 is merged with segment 3, and segment 2 is chosen as parent. Now segment 3 points to segment 2. After updating the table, it is ready to be used to render S_2. For the colour look-up table, it is initialized with the colours of S_0. This time we iterate from 0 to $t-1$; At iteration 0 in the timeline, segment 2 is the parent of a merge and gets a new colour. It used to be white, now it is dark pink. At

iteration $t-1$ in the timeline, segment 2 is the parent of a merge and gets yet another new colour. It used to be dark pink, now it is light pink. The colour look-up table is now ready to be used for rendering S_2. Figure 4.9 shows pseudocode for building the lookup tables.

0	1	2	3	4
0	1	2	3	4

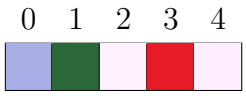
(a) The segments are initialized with self-referencing pointers.

0	1	2	3	4
0	1	2	3	2

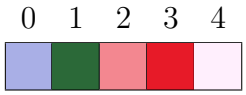
(b) At $t-1=2-1=1$ in the timeline, segment 2 is merged with segment 4, and segment 2 is chosen as parent. Now segment 4 points to segment 2.

0	1	2	3	4
0	1	2	2	2

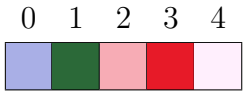
(c) At $1-1=0$ in the timeline, segment 2 is merged with segment 3, and segment 2 is chosen as parent. Now segment 3 points to segment 2.



(d) The colors are initialized with S_0.



(e) At time 0 in the timeline, segment 2 is the parent of a merge and gets a new color. It is now dark pink.



(f) At time $1=t-1$ in the timeline, segment 2 is the parent of a merge and gets a new color. It is now a slightly lighter pink.

Figure 4.8: Building look-up table for S_2 using S_0 and the merge-event timeline of Figure 4.7.

Using the look-up tables, we can render S_t. The following steps are for rendering on a raster, when S_0 is stored also in raster form. For each pixel on the output raster, we find its colour by:

```
1   void buildSegmentLookup(int lookup[], int t) {
2
3     // initialize table to self-referenced segments
4     for (int segment = 0; segment < getS0SegmentCount(); segment++) {
5       lookup[segment] = segment;
6     }
7
8     // traverse backward from time t to 0 & update segment reference
9     for (int i = t - 1; i >= 0; i--) {
10      lookup[child[i]] = lookup[parent[i]];
11    }
12
13  }
14
15  void buildColorLookup(int lookup[], int t) {
16
17    // initialize table using original colors in S0
18    for (int segment = 0; segment < getS0SegmentCount(); segment++) {
19      lookup[segment] = getS0Color(segment);
20    }
21
22    // update color until time t
23    for (int i = 0; i < t; i++) {
24      lookup[parent[i]] = color[i];
25    }
26
27  }
```

Figure 4.9: Building the segment and colour look-up tables using S_0 and the merge-event timeline. The timeline in the pseudocode consist of the three tables *parent*, *child* and *color*.

1. Fetching the segment id for this pixel's position in S_0.
2. Retrieving the segment id for this position in S_t using the fetched id of S_0 and the segment look-up table.
3. Get the output colour from the colour look-up table with the retrieved S_t id.

These steps can render efficiently a hierarchy based on a raster. We use the same look-up tables to render a hierarchy based on vector graphics. The rendering proceeds with the set of contour primitives for each segment of S_0. During rendering, edges shared by two polygons that belong to the same segment at time t are ignored.

Browsing through different levels-of-detail of a segmentation hierarchy in real-time allows novel exploration techniques. Chapter 5 describe how we use real-time linear exploration for interactive non-hierarchical segmentation. Exploration can also assist in the analysis of the underlying algorithm that generated the hierarchy. Limitations with the merging criterion or implementation bugs may be identified this way.

Analysis

Storage Our implementation stores S_0, and three values per step. These values are the child and parent segment id and the blended colour. There can be up to $n-1$ steps so in total we use $3n-3$ storage besides S_0. The rendering uses look-up tables for segments and render properties. The look-up tables both have a size of n each. Therefore, the storage during rendering is $O(n)$.

Time Computing one look-up table takes t time, where t is the time we are interested in visualizing, which at worst can be $n-1$. For both table we use $2n-2$ time which is $O(n)$. The performance for viewing the result also depends on the format of S_0 and of what it is we are rendering. For a typical raster-based scenario where S_0 is a 2-D array that implements $getSegment(x, y)$ in $O(1)$, and where we are rendering each pixels, it takes $W \times H$ look-ups, where W and H are the width and height of the map. If S_0 is a contour image based on vector graphics, the additional rendering time is a function of the number of edges.

4.3.2 Comparison with UCM

Arbeláez [Arb06] describe how to store a hierarchy in an Ultrametric Contour Map (UCM). Different levels-of-detail in the UCM can be explored using thresholding, in a

similar fashion that we can explore our merge-event timeline using t. Arbeláez and Cohen [AC08] show how to use a UCM for marker-based interactive segmentation. The UCM map is essentially a raster. It stores the occurrence of a boundary and the boundary's strength between neighbouring pixels. A UCM is tightly connected to a source image. This source image must be on a raster because the UCM is a raster. In our haptic image exploration system, we use a merge-event timeline data structure because our hierarchy is defined as a vector graphics.

If the merge-event timeline data structure represents a segmentation hierarchy for a raster image, it can be converted to UCM. One can use the merge step index between segments as the scale of disappearance for the corresponding edges in the UCM. The merge step index is however not as meaningful a scale as the boundary strength used by Arbeláez.

The interactive segmentation with UCM presented by Arbeláez and Cohen [AC08] is marker-based, and possess the advantage of this style of interaction. In Chapter 5 we describe an innovative style of interactive segmentation that uses a segmentation hierarchy as input.

4.4 Vectorization

Once a suitable segmentation hierarchy is built from a digital photograph, it is not yet ready for haptic image exploration with a point-based haptic device like the SensAble Phantom. Zooming in the segmented image reveals that the shape of the segments are blocky because the segments are made of pixels, *i.e.*, lines are rendered as stairs made from pixels. We transform the raster-based segments into smooth vector graphics using vectorization. In this section we discuss the related work that form the building blocks that we use to achieve this goal.

The process of vectorization converts a raster image into geometric primitives like lines, curves and shapes. The classic goal in vectorization is to create a vector image that looks like the input raster image while keeping the vector data simple. Lecot and Levy [LL06] developed an algorithm for converting raster images to vector graphics. In their algorithm, they take advantage of the gradient capability of vector graphics and simplify the image by estimating such gradients. They also take full advantage of splines and curves available in vector formats. Their result are of high visual quality, and yet of simple geometry. This system is a great solution to the task of vectorizing photographs. Since we seek to vectorize an image that has been segmented, their solution does not

directly apply to us. For more methods on generic raster image vectorization, we refer the reader to papers on gradient meshes by Sun et al. [SLWS07] and Lai et al. [LHM09] and a paper on a patch-based vectorization method by Xia et al. [XLY09]. Our input is a segmented image, and therefore our task is different than vectorizing a generic image.

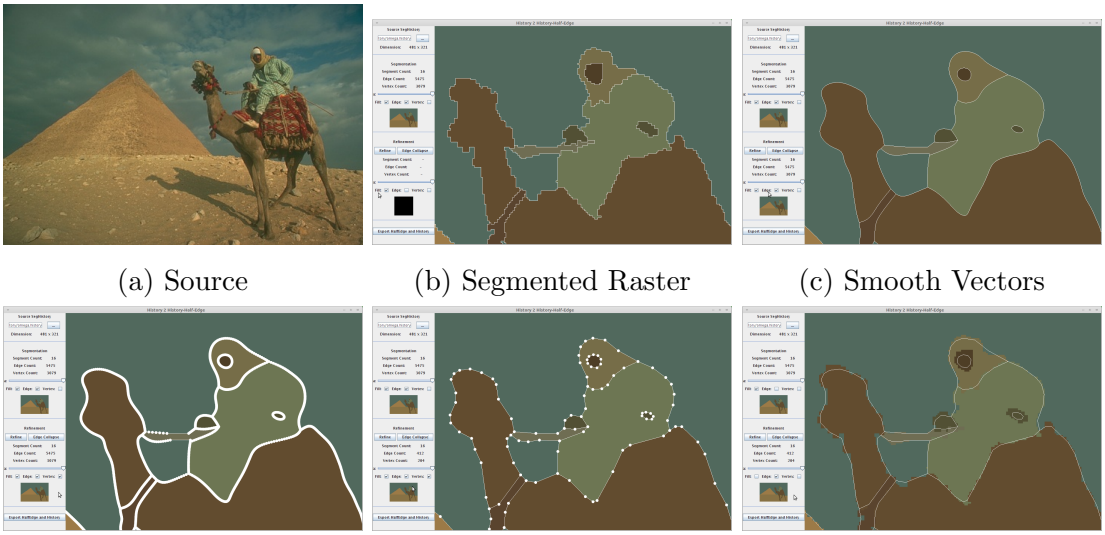
We seek a vectorization solution that is specialized in dealing with a segmented image as input. McDonald and Lang [ML08] have developed a vectorization method given a segmented image as input, as well as the original non-segmented source image. The algorithm not only looks at the geometric configuration of the segments, but also at the image data in order to vectorize and adjust with sub-pixel precision the shapes forming the segments. It is based on the active contour framework of Kass et al. [KWT88]. Active contours minimize an energy function consisting of two parts. The first part of the energy function defines the internal energy governed by the regularity of the contour shape, *i.e.*, its tension and rigidity. The internal energy pulls each vertex to the centroid of its neighbouring vertices. The second part of the energy function defines the external energy coming from the image data. McDonald and Lang employ an external energy term derived from competitive forces exerted by adjacent regions on a contour point. It is based on the region competition principle of Rosin [Ros98] which states that at the precise boundary of two true regions, the domain will be equally representative of each. The method adjust the boundary at sub-pixel precision in the appropriate direction to balance the regions. Figure 4.10 shows our vectorization results using the method of McDonald and Lang [ML08].

(a) Raster (b) Smooth Vectors (c) Overlay View

Figure 4.10: Smooth vectorization of a raster-based segmentation.
We see here a magnified view on the bib of the front runner of the segmented image found in Figure 4.5 (a).

Many vertices are redundant after sub-pixel refinement, and can be removed using edge collapse methods. Garland and Heckbert [GH97] presented a method to approx-

imate a 3-D surface mesh from a high definition original while minimizing geometric error. The approximation is done through a series of edge and vertex contraction. Each possible contraction is given a cost that represents the error introduced if this contraction would be executed. At each iteration, the contraction with the lowest cost is performed. Garland and Zhou [GZ05] later generalized the approach to any dimension. In our 2-D segmented image context, we use their method to simplify the segment of the vectorized hierarchy. We perform this simplification of the segmented surfaces to improve rendering speed during haptic exploration. Figure 4.11 shows the results of our implementation of quadric-based simplification on a smooth vector graphics derived from a raster.



(a) Source (b) Segmented Raster (c) Smooth Vectors

(d) Vertices before simplification (bold white) (e) Vertices after simplification (f) Overlay of Raster and vectorized edges after surface simplification

Figure 4.11: Smooth vectorization and surface simplification.

4.5 Summary

The strength of our approach for authoring *object-level hierarchies* from images is its flexibility. We use the most effective segmentation algorithm for the task at hand. Our method falls back on automatic hierarchy generation if very fine details must be included in the results. The hierarchy itself is stored in a structure that is independent of the segmentation format, e.g. this freedom permits storing vectorized segments.

We have designed a new merge-event timeline data structure to store a segmentation hierarchy. The structure supports linear exploration in a specific order. It is minimalist, storing only what it needs and does not rely on a raster representation. We have shown an algorithm for efficient exploration of the hierarchy it contains. In comparison, a tree does not contain by itself a specific traversal order, but it supports non-linear exploration. UCM relies on raster representation but does have a marker-based interactive segmentation algorithm designed to work with it. The next chapter describes a novel interactive segmentation method based on real-time linear exploration of a hierarchy.

Chapter 5

Interactive Segmentation by *Region Selection*

In this chapter we propose a new style of user interaction for image segmentation. We begin by exposing some drawbacks of marker-based interactive methods in Section 5.1. We introduce our novel interactive segmentation by *region selection* method in Section 5.2. We evaluate this new type of interaction in Section 5.3 using an existing benchmark dataset and measure [ZND+11]. All source images used in this Chapter are from this benchmark dataset but are credited to the Lotus Hill Institute (LHI) database [BYZ07]. We give a summary of this chapter in Section 5.5.

5.1 Marker-based Interactive Segmentation

Interactive segmentation has received much research attention, and we have reviewed some of the more influential and popular interactive segmentation methods in Section 2.2.2. While these methods are powerful, we find that they focus on achieving good segmentation while minimizing interaction. We would like an interactive method that gives the controls necessary to segment with a good level of accuracy quickly, while supporting reaching extreme accuracy levels efficiently as well.

The most popular type of interaction in the literature is marker-based interaction and its extension to scribbles [Mey92, BJ01, UPT+08, AC08]. The scribbles act as hard constraint on the resulting segmentation, but each algorithm also uses soft constraints defined in the image data. Unger et al. [UPT+08] suggest using a more complex selection of brushes and interactions, *e.g.*, a sample colour brush that builds an histogram used to

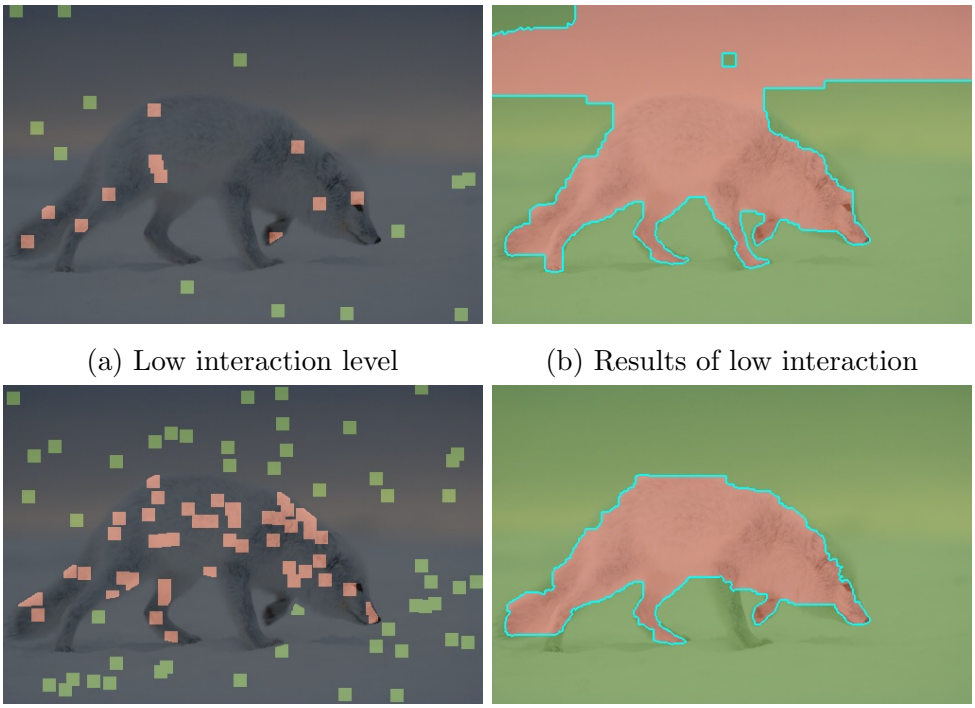
compute the probability of a pixel to be part of the foreground based on its colour. A variant to scribbles is placing a bounding box around the object to extract, and letting the algorithm work out a more precise delineation [CRB04]. Instead of a bounding box, a closer delineation can be given with a magnetic lasso tool, which is a selection mode that clings to nearby edges found in the image given a threshold value. The process of finding a better delineation given an approximation of the solution is sometime referred to as matting.

One major drawback with these methods is their unpredictable outcome caused by using soft constraints. It is nearly impossible to foresee with accuracy where an algorithm will set the segment boundaries, even with intimate knowledge of the algorithm. It is fair to assume that a user would place markers at given locations no matter which marker-based segmentation algorithm is used under the hood.

Another inconvenience in existing interaction methods is the free-hand nature of interaction. To achieve a very precise delineation, the brush placing the markers is used to delineate free-hand the desired boundary. We argue, that this can degenerate into manual segmentation, where the benefit of automation to assist the user is lost. Figure 5.1 shows the segmentation results of different level of interaction on a difficult image using Graph Cuts [BJ01]. The image is difficult to segment because the colour distribution of the background and the foreground are similar, *i.e.*, the white wolf blends into the snowy surroundings. We can see that even with a large amount of interaction, the wolf in the image is not well delineated, *e.g.*, it is missing a leg and its forehead. Markers defining the complete and precise boundary of the wolf and the background would be required in order to extract it accurately, but then there would be no difference to manual segmentation interaction.

Another downside with existing marker-based interaction methods is the potential loss of progress towards precise delineation when providing additional interaction. Figure 5.2 show the segmentation results of a watershed algorithm [Mey92] before and after additional interaction. In the example, we would like to fix in Figure 5.2 (c) the dark spot in the background that gets classified as foreground with additional interaction. We show in Figure 5.2 (d) that the correction introduces a new error, this time classifying parts of the coat as background.

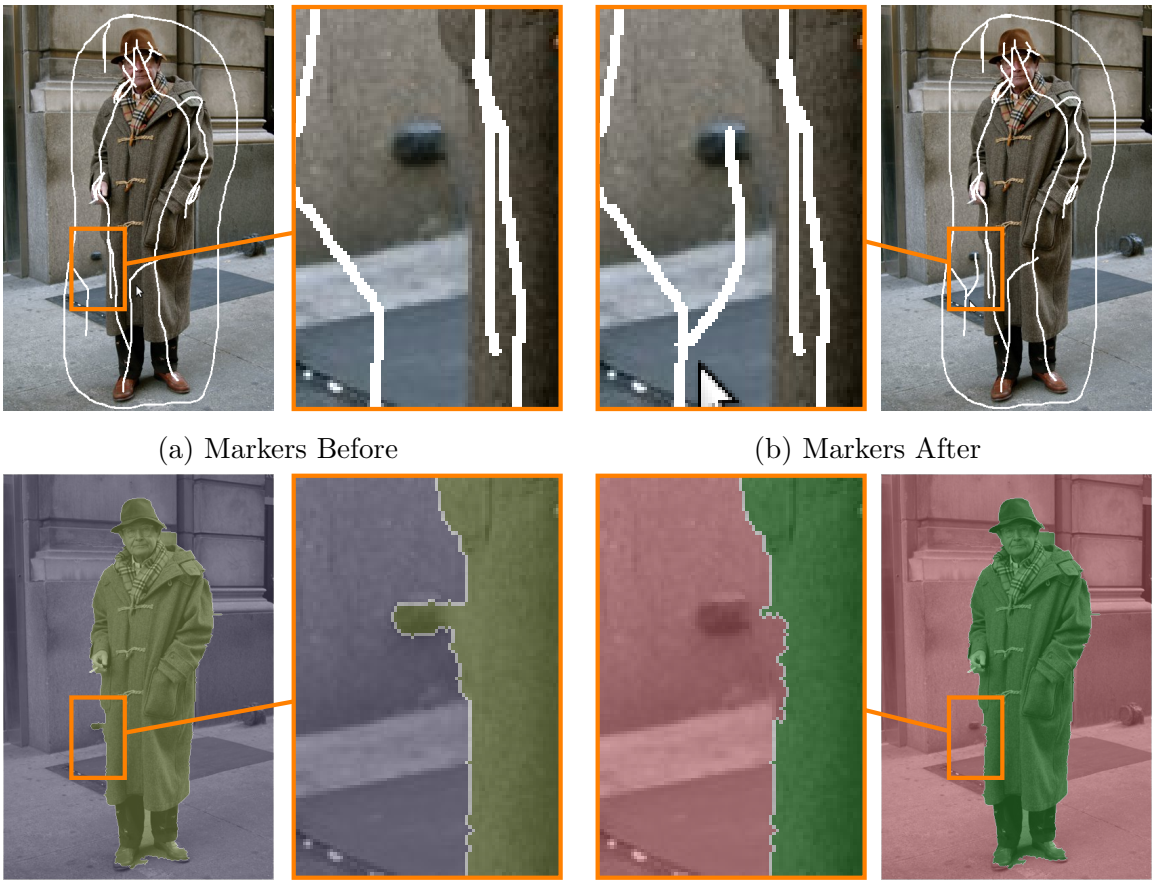
The limitations we have just described motivated us to develop an alternative interactive segmentation method, one that would overcome these challenges. The solution we present avoids using soft constraints, which is at the root of these disadvantages.



(a) Low interaction level (b) Results of low interaction

(c) High interaction level (d) Results of high interaction

Figure 5.1: Segmentation results at different level of interaction on a difficult image using Graph Cuts [BJ01]. These are publicly available results given by Zhao et al. [ZND+11].



(a) Markers Before (b) Markers After

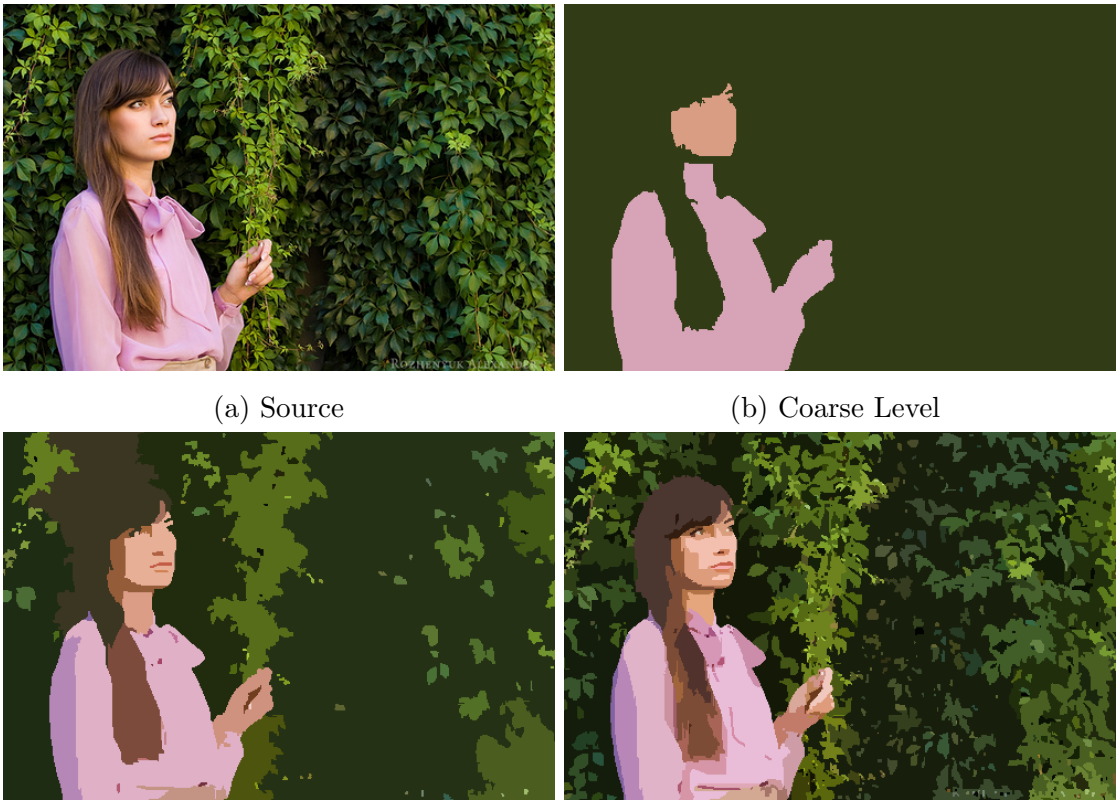
(c) Delineation Before (d) Delineation After

Figure 5.2: Delineation results before and after providing additional interaction.

5.2 Concept

The interaction we propose is to browse a segmentation hierarchy generated automatically and select contiguous pixel regions to build the output segments. We call our concept interactive image segmentation by *region selection* .

Before interaction starts, a segmentation hierarchy is built from the source image. We leave the choice of algorithm to build the hierarchy open. The segmentation hierarchy does not have to be strict nor *object-level*. In essence, we build from the source image a sequence of segmentation at increasing levels of coarseness. Figure 5.3 shows the source image we will be using throughout the examples of our method, as well as three levels from the hierarchy that we built automatically with SRM [NN04].



(a) Source (b) Coarse Level

(c) Medium Level (d) Fine Level

Figure 5.3: Source image and three levels in the segmentation hierarchy.

The goal of the user is to segment the source image. The user will begin by browsing linearly through the sequence of segmentations levels stored in the hierarchy. In the event that one of the segmentation levels suits the exact need of the user, the interaction

can end simply by selecting this level. Throughout this section, Figure 5.3 to Figure 5.8 will show all the steps needed to extract the women from the background. Normally, the user will find a level that is close to his/her needs, and will continue interaction by forming a selection mask delineating the object to be extracted into a segment.

The selection mask is formed by selecting multiple groups of pixels. We have implemented selection such that it works like the colour selection wand found in image manipulation programs, with a colour distance threshold set to zero. The level of detail can be coarse such that the regions are not tedious to select. To make things more convenient, multiple regions can be selected by dragging the mouse cursor over them. The user can select or deselect groups of pixels until he/she is satisfied with the mask. The selection can be inverted, such that if it was more convenient for the user to select the opposite mask, he/she can do so. When the selection mask is complete, the user merges all the selected pixels together, and starts over with another mask. To merge, we use the source image to compute the average colour of the pixels in the mask, and set the colour of those pixel to that average. When the interaction is complete, segments are created with each contiguous pixel regions sharing the same colour. Figure 5.4 shows the selection mask after selecting multiple regions, and the results after merging the mask. A technical detail of our implementation is that we give the user the liberty to pick a colour manually with a colour swatch widget instead of using the average. This is to workaround the implicit merge that happens when two adjacent segments share the exact same average colour.

Merging regions from a single level-of-detail in the hierarchy does not give enough precision control to the user. In our method, it is possible to view multiple simultaneous levels-of-detail for different regions in the same image. We use clipping in the implementation of this feature. The user forms a selection mask for the regions that need a distinct level-of-detail. Instead of merging, the user browses through the sequence of segmentations levels stored in the hierarchy, and only the pixels within the selection mask will be affected. This form of interaction allows the user to change the level-of-detail of specific regions without affecting the progress elsewhere in the image. An interactive segmentation session typically involves working with multiple levels-of-detail. Each of the objects in the image should be segmented at the level-of-detail that is most convenient for the user. Figure 5.5 shows an example of using multiple simultaneous levels-of-detail.

Individual pixel selection is an option that allows full control over the mask being formed. At a given level-of-detail, a segment may bleed into an adjacent segment. Pixel-level controls can be used to patch the bleeding region. We show in Figure 5.7 an example

(a) Selecting a single region

(b) Selecting multiple regions

(c) Resulting selection mask

(d) Merging the selection mask

Figure 5.4: Selecting multiple groups of pixels to form a selection mask, and merging the mask. The blue regions represents the selection mask, the green line represents the positions when dragging the mouse.

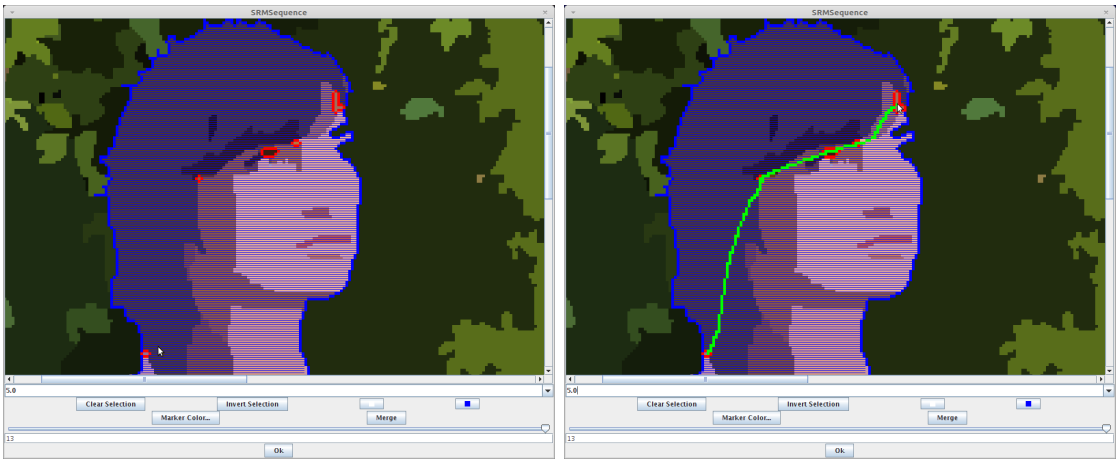
(a) Selecting a region

(b) Increasing the level-of-detail for the selection

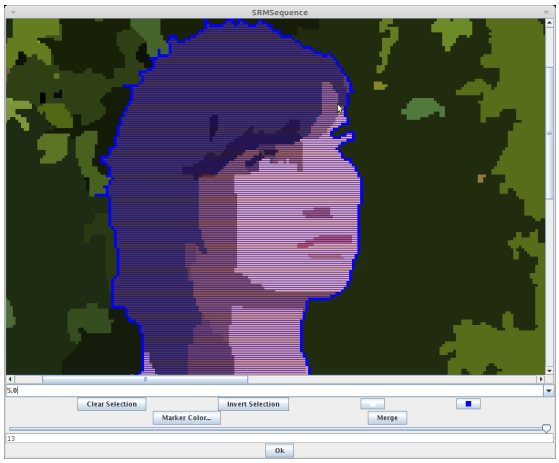
(c) Selecting multiple regions

(d) Result of selection

Figure 5.5: Modifying the level-of-detail in the selection mask. The blue region represents the selection mask, the green line represents the positions when dragging the mouse, red points highlight the outline of small regions that are out of the selection mask.



(a) Zoomed view (b) Selecting multiple regions



(c) Results of selection

Figure 5.6: Dragging the mouse to select the remaining small regions. The blue region represents the selection mask, the green line represents the positions when dragging the mouse, red points highlight the outline of small regions that are out of the selection mask.

where pixel control is used. The background has bled into the left eye and eyebrow of the women in the picture. To select the regions belonging to the face, we first add to the selection mask a series of individual pixels that close the opened regions. We continue by selecting the now closed eyebrow and eye regions. Individual pixel control to form the mask is necessary to achieve precise segmentation if superpixels with potential delineation errors are used as finest level in the hierarchy.

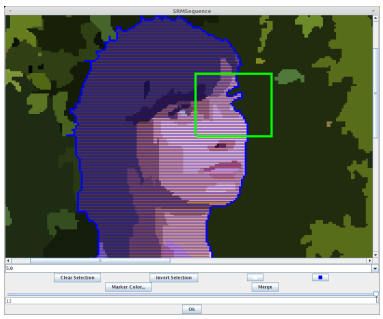
The interactive segmentation by *region selection* we have described possess the following advantages compared to marker-based interaction: The interaction outcome is predictable, *i.e.*, the user knows that selecting a region will add all the pixels of same colour to the selection mask. The user knows what to expect when merging the selection mask, there are no hidden or soft constraints. When used carefully, the interaction incrementally works toward a goal without loss of progress. Our method also has some properties that may be viewed as disadvantages in some applications. If only approximate results are desired, our method may require more interaction than marker-based approaches depending on the quality of the hierarchy. If the segmentation hierarchy does not delineate an object properly, the interaction can degenerate into manual segmentation for that region, similar to the corresponding predicament for marker-based approaches.

Our interactive segmentation method effectively uses an automatically generated segmentation hierarchy to assist the user in creating a *flat* segmentation. We believe our *region selection* interactive segmentation method is more convenient than existing approaches when a high quality segmentation is required. We present in the next Section an evaluation of our method that supports this claim.

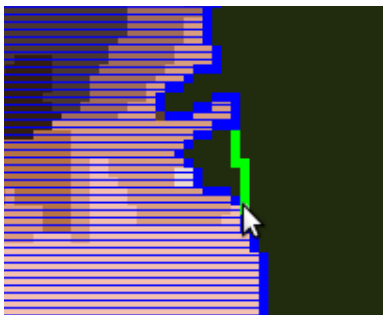
5.3 Evaluation

5.3.1 Interaction Levels

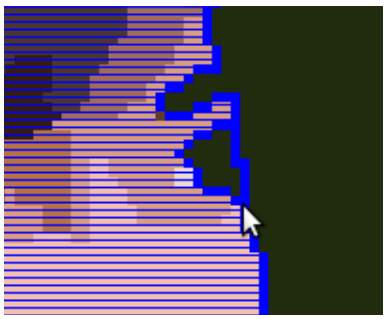
We evaluate our *region selection* method using an existing application benchmark developed by Zhao et al. [ZND+11]. Zhao et al. have prepared an objective evaluation method for comparing interactive image segmentation algorithms. For marker-based approaches, the evaluation process involves running the segmentation algorithm with predefined markers and comparing the result segmentation with a ground truth delineation that was prepared manually. Their dataset includes a variety of photographs classified in five categories: animal, artifact, building, human and plant. Each photograph has an



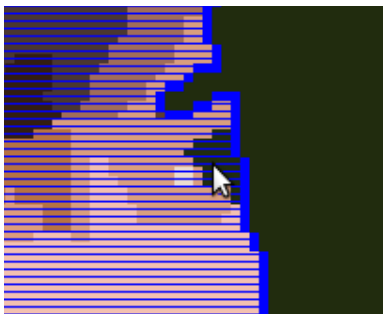
(a) Selection Mask so far



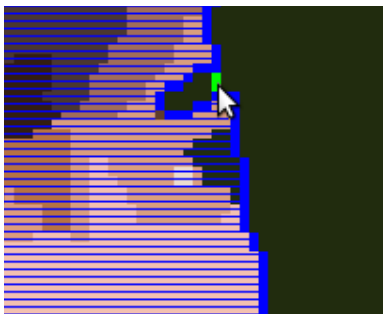
(b) Pixel control to close the open eye region



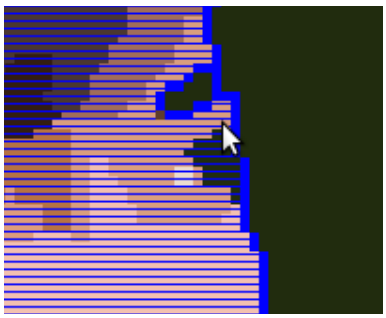
(c) Result of selection



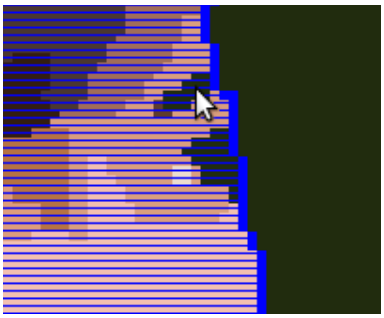
(d) Selecting the closed eye region



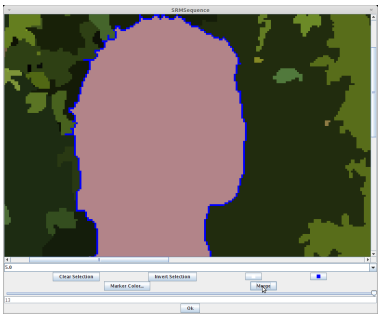
(e) Pixel control to close the open eyebrow region



(f) Result of selection

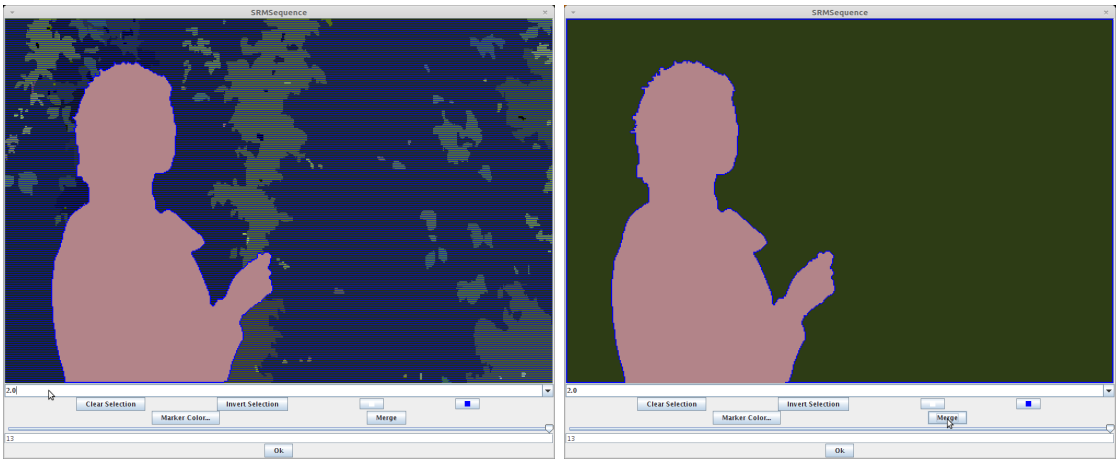


(g) Selecting the closed eyebrow region



(h) Merging the selection mask

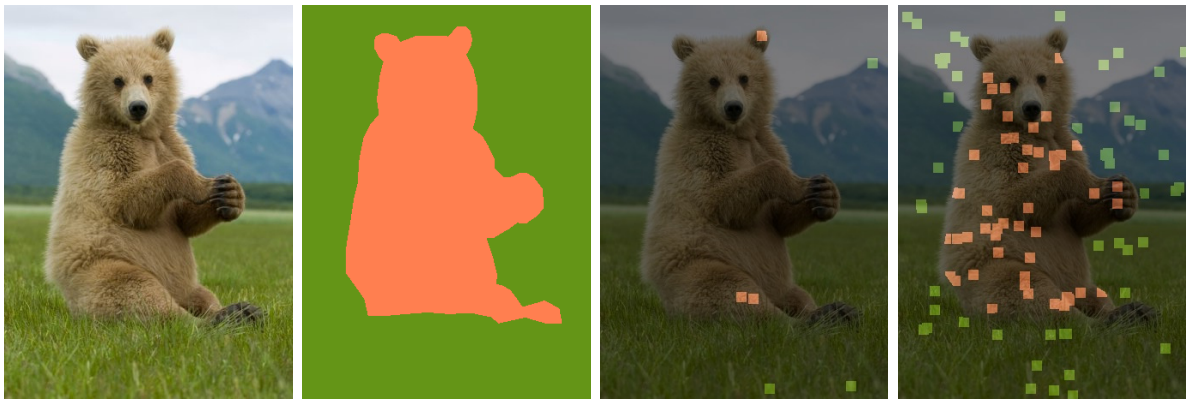
Figure 5.7: Using pixel level controls.



(a) Invert the selections (b) Merge the selection mask

Figure 5.8: Merging the Background by inverting the selection.

associated ground truth segmentation. Zhao et al. include in their dataset 4 levels of predefined marker positions. Figure 5.9 shows a sample from the benchmark dataset of Zhao et al. Their current work evaluates marker-based interactive image segmentation algorithms. With some adaptation, we use their dataset to run our benchmarks on our interactive segmentation by *region selection*. We chose to use the dataset of Zhao et al. because it allows us to measure and compare objectively our method against other interactive segmentation algorithms.



(a) Source (b) Ground Truth (c) Interaction Level 1 (d) Interaction Level 4

Figure 5.9: Sample of benchmark dataset from Zhao et al. [ZND+11]

When using the benchmark with a marker-based method, all the interactive markers are given simultaneously as input to the algorithm. Our *region selection* interaction

is iterative, and therefore we have designed a special procedure to use *region selection* interaction with the interactive markers supplied in the benchmark dataset of Zhao et al. We first build a segmentation hierarchy using the algorithm we want to benchmark. For each interactive marker, we simulate a selection at that location for each available level-of-detail in the segmentation hierarchy and keep the selection that brings the selection mask the closest to the ground truth of this image. We use the region coverage quantitative evaluation criterion provided by Zhao et al. to compute a score against the ground truth. They use the Jaccard index to compute region coverage, with the two following sets: the foreground pixels of the ground truth, and the pixels forming the selection mask, *i.e.*, the segmentation candidate. The Jaccard index is the ratio between the size of the intersection of these two sets, and the size of the union of these two sets.

$$J(A,B) = \frac{\mid A \cap B \mid}{\mid A \cup B \mid}$$

Because *region selection* interaction is iterative, extracting the foreground object is not done with the same interaction as extracting the background object. In our results we have measured how well our algorithm performs using only foreground extraction, only background extraction, and the extraction of the most appropriate region depending on the image. The Jaccard index used in the benchmark to measure the score of the candidate segmentation with the ground truth does not give the same value when the task is inverted into background extraction. That is because the size of the background is not the same as the size of the foreground in general. Typically, rough segmentation of bigger regions will yield a higher Jaccard index. The metric is only reciprocal when the segmentation perfectly matches the ground truth, or completely fails to match it. To remove errors when comparing scores of background extraction and foreground extraction of the same image, we always use the Jaccard index of the foreground object region independent of the method of the extraction.

We ran the benchmark with *region selection* using the segmentation sequence generated by 3 different automatic segmentation hierarchy algorithms. We call the first algorithm Region Growing, which uses simple colour distance as merge criterion. We described it in Section 2.2.1 when giving our definition of a standard region merging algorithm. Figure 5.10 shows a sample hierarchy produced by the Region Growing algorithm on the source image in Figure 5.9 (a). The two other algorithms we have used in our benchmark are SLIC [ASS+10] and SRM [NN04]. We show a sample hierarchy produced by both these algorithms in Figure 5.11 and Figure 5.12.

We present the results of the benchmark in Table 5.1. Extracting the foreground

(a) Coarse (b) Medium (c) Fine

Figure 5.10: Subset of a hierarchy produced by our Region Growing algorithm.

(a) Coarse (b) Medium (c) Fine

Figure 5.11: Subset of a hierarchy produced by the SLIC algorithm.

(a) Coarse (b) Medium (c) Fine

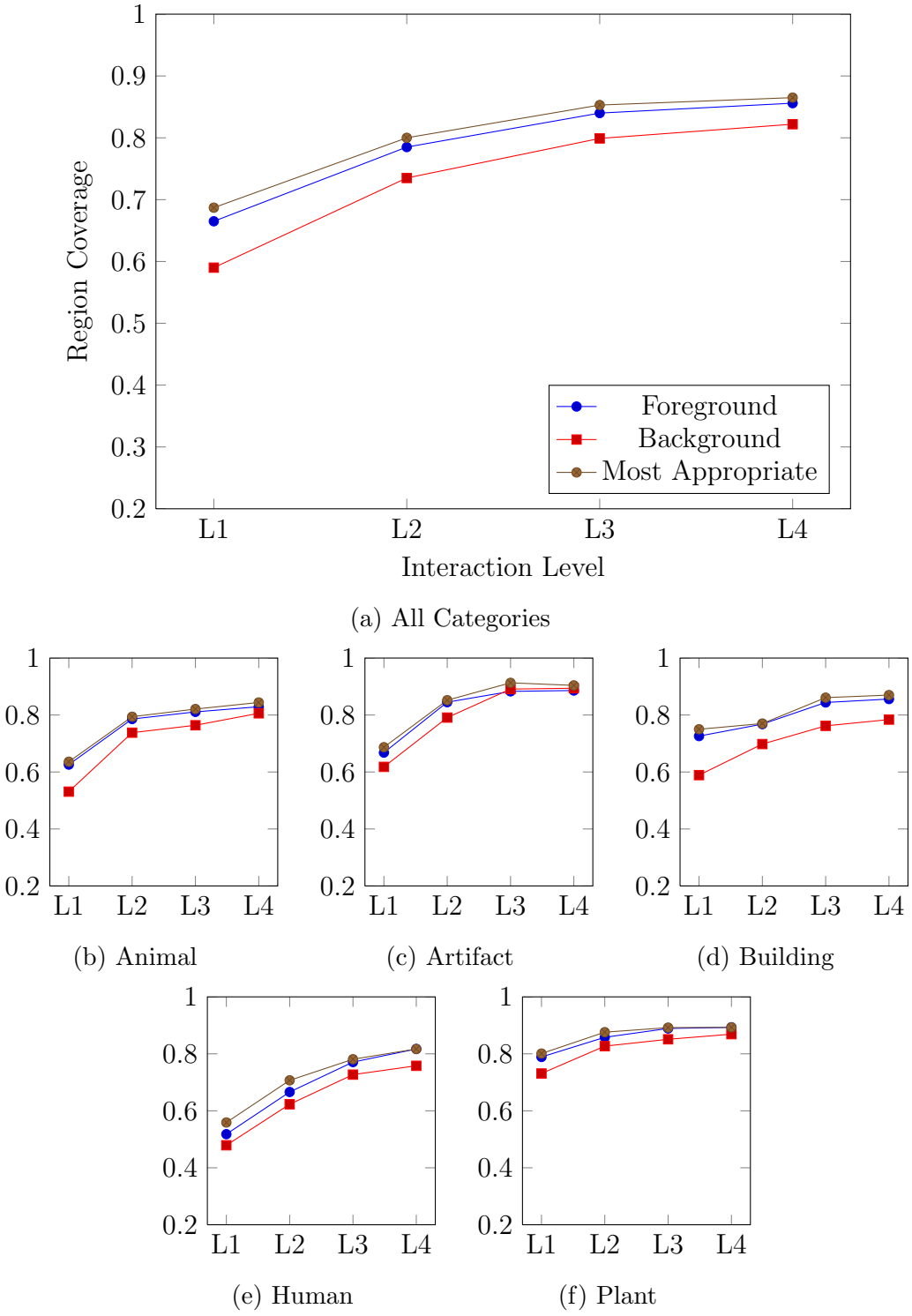
Figure 5.12: Subset of a hierarchy produced by the SRM algorithm.

or extracting the background with our method is not the same task, because of the iterative nature of interactive segmentation by *region selection*. The table includes how the algorithms performed for foreground extraction, background extraction and when choosing the simplest object to extract between foreground and background. We included the results of Zhao et al. in the five last rows of the table for comparing our results with the marker-based algorithms they have benchmarked in their study. We present the results of our study for each of the image categories in the image dataset developed by Zhao et al. in Figure 5.13, 5.14, and 5.15.

Method	Point-Process Level			
	L1	L2	L3	L4
Region Growing (foreground)	.67	.78	.84	.86
Region Growing (background)	.59	.74	.80	.82
Region Growing (best of fg/bg)	.69 (36%)	.80 (36%)	.85 (42%)	.87 (32%)
SLIC (foreground)	.69	.82	.89	.90
SLIC (background)	.51	.76	.85	.88
SLIC (best of fg/bg)	.70 (18%)	.83 (22%)	.89 (26%)	.90 (32%)
SRM (foreground)	.71	.82	.89	.90
SRM (background)	.60	.81	.85	.87
SRM (best of fg/bg)	.74 (30%)	.85 (32%)	.90 (26%)	.91 (28%)
Graph Cuts [BJ01]	.41	.63	.80	.82
Geodesic matting [BS07]	.36	.53	.74	.80
Power watersheds [CGNT09]	**.50**	**.72**	**.84**	**.88**
Random walker [Gra06]	.46	.71	**.84**	**.88**
Structural IS [NGC$^+$08]	.49	.69	.82	.87

Table 5.1: Summary of region coverage results after object extraction. The value in parenthesis indicates the proportion of background extraction used when using best strategy between foreground or background extraction. The last five rows are results published by Zhao et al. [ZND$^+$11] on marker-based interactive segmentation algorithms.

Our results are generally good, even for our naïve Region Growing hierarchical segmentation algorithm, though it scores lower than SLIC and SRM at every level, which can be expected. Comparing the results of *region selection* with the five marker-based algorithms benchmarked by Zhao et al. we notice that *region selection* gives considerably better results at low levels of interaction compared with marker-based approaches. Our



(a) All Categories

(b) Animal

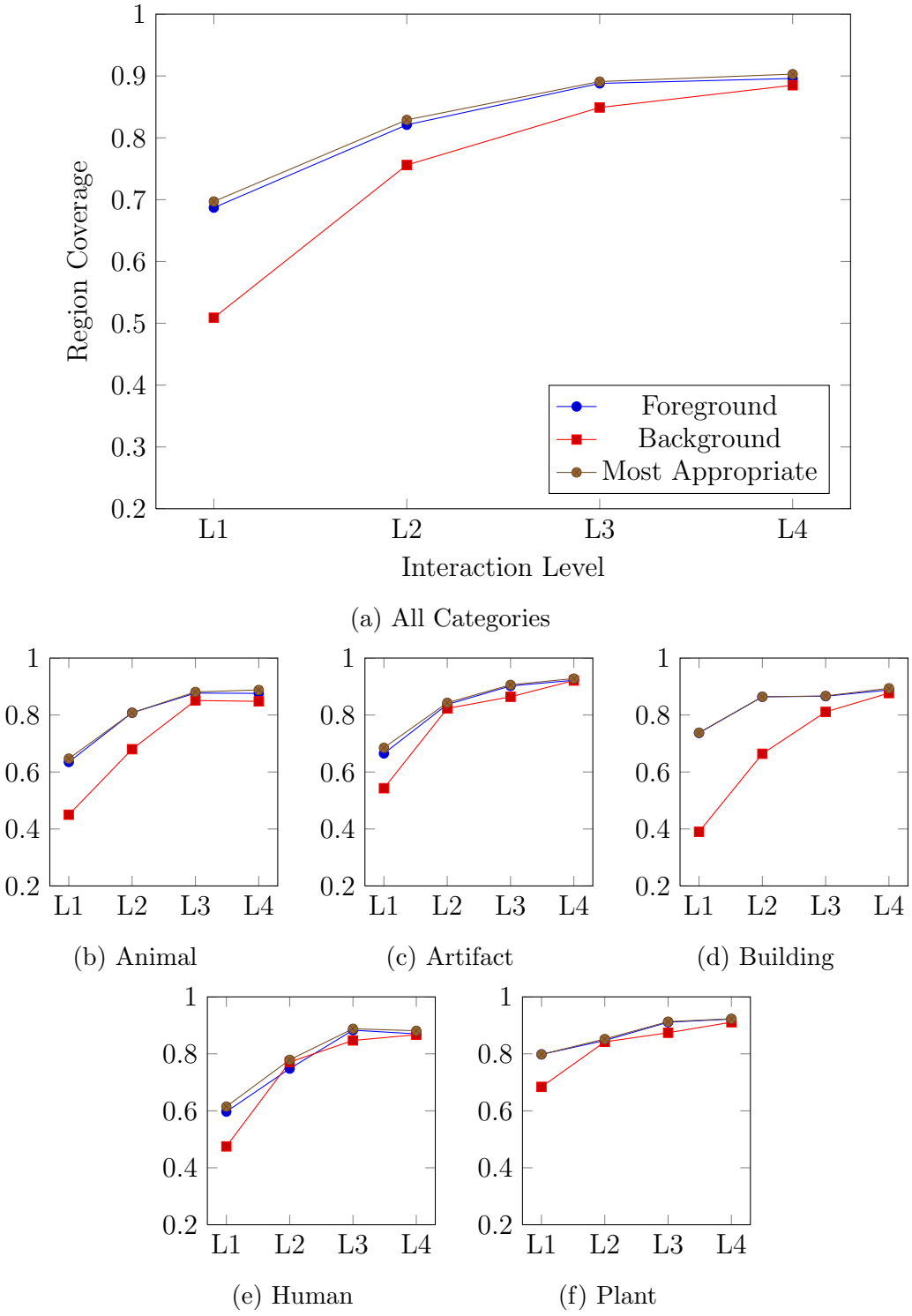
(c) Artifact

(d) Building

(e) Human

(f) Plant

Figure 5.13: Region coverage per levels using Region Growing for the the different categories of images.



(a) All Categories

(b) Animal

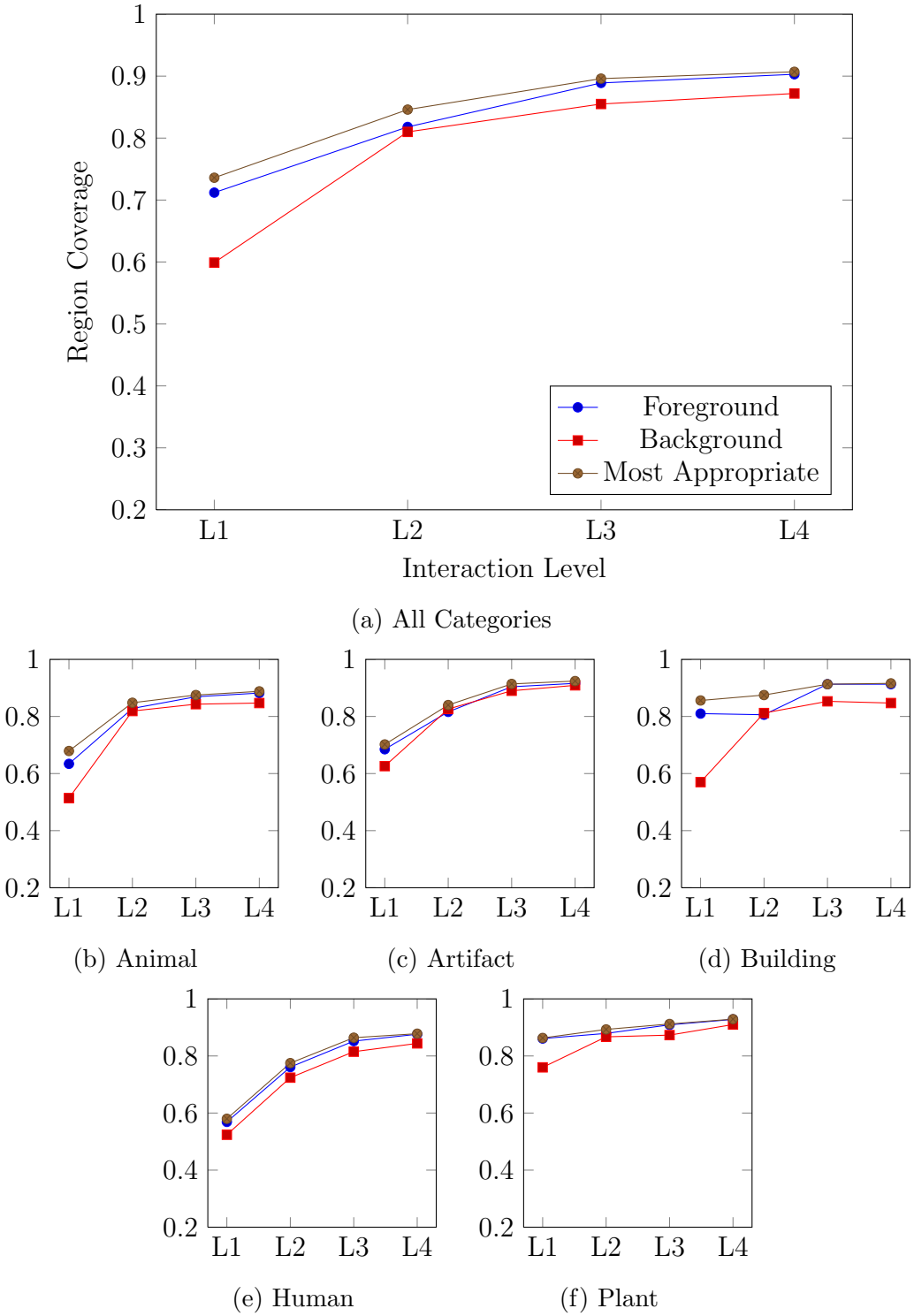
(c) Artifact

(d) Building

(e) Human

(f) Plant

Figure 5.14: Region coverage per levels using SLIC for the the different categories of images.



(a) All Categories

(b) Animal

(c) Artifact

(d) Building

(e) Human

(f) Plant

Figure 5.15: Region coverage per levels using SRM for the the different categories of images.

results support the idea that *region selection* is a viable solution to interactive image segmentation, and performs best with a high quality automatic segmentation algorithm. Note that we have ignored some markers when running the benchmark if they did not improve the score when used. This happens when the regions under a foreground marker is already part of the selection mask, or vice versa for a background marker. For our three algorithms, more than 50%, 75%, 80% and again 80% of the markers were ignored in the first, second, third and fourth levels of interaction respectively. A technical detail about this statistics is that we have weighted each marker by their size since they do not all have the same size, and some overlap and are treated as a single bigger marker.

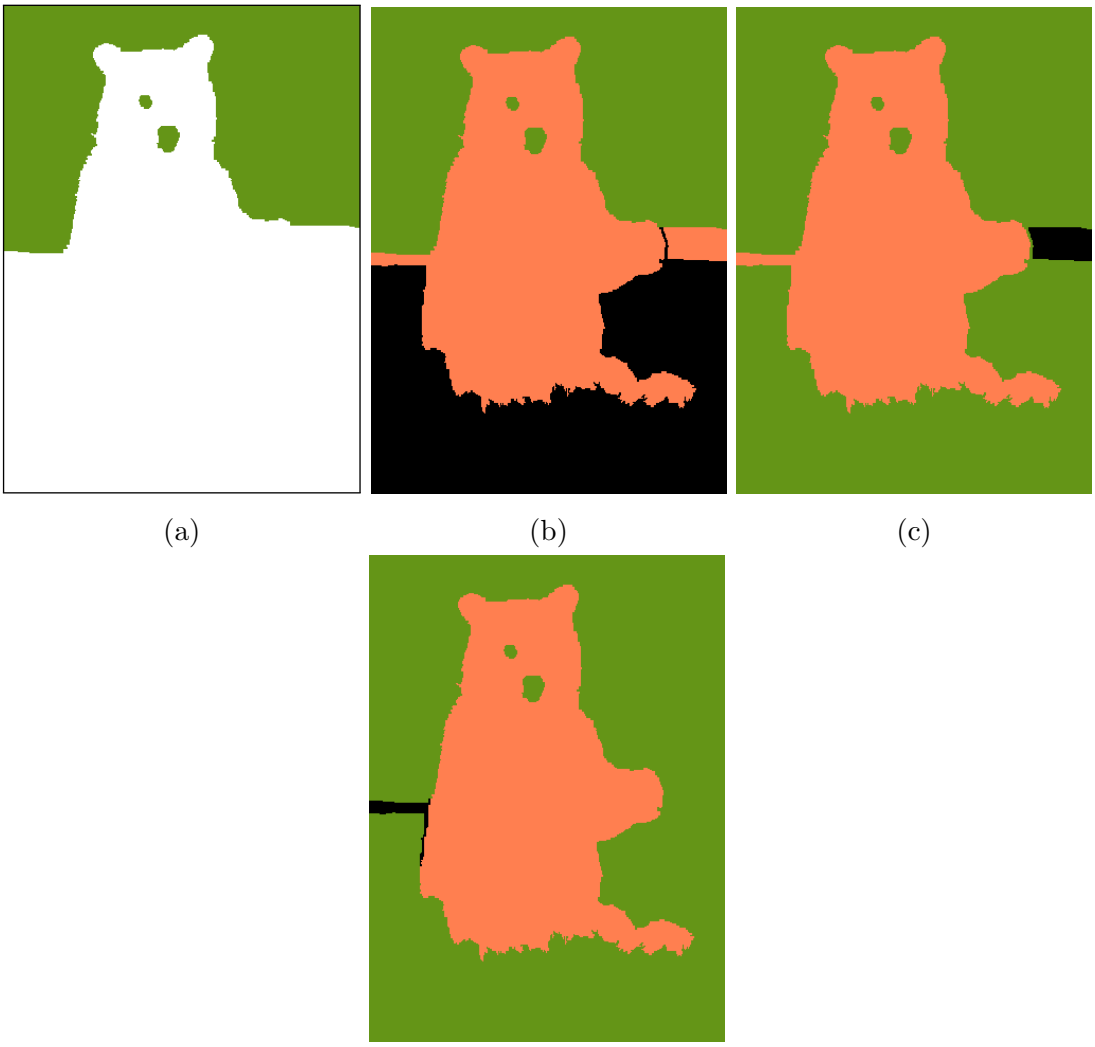
5.3.2 Greedy marker location

We argue that the predefined marker positions included in the benchmark dataset are optimized for marker-based algorithms. Zhao et al. describe using a k-means colour space clustering algorithm to find the most important colours in the image. They make sure to select marker positions that cover the characteristic colours. This heuristic is beneficial to the five algorithms that they have benchmarked. A neutral approach would have been to select positions randomly.

We have designed another benchmark experiment that uses our own interactive marker location instead of the one supplied in the benchmark dataset. Using our own markers also gives us the possibility to evaluate our method at a finer level of interaction granularity, *i.e.*, each individual additional markers. Like in the earlier experiment, we first compute the segmentation hierarchy automatically. For each additional marker, we find the region across all levels-of-detail that if selected optimizes the region coverage quantitative evaluation criterion against the ground truth. This algorithm is greedy, and selects the region that increases the score the most, but it is not an optimal algorithm. A better score may be achieved in some cases by selecting several small regions sequentially, rather than a large region, but our greedy algorithm will always select the largest region.

We illustrate the methodology used in this new experiment in Figure 5.16. White regions represent the regions added to the selection mask at that step. Black regions represent the regions removed from the selection mask. In Figure 5.16 (a) the greedy algorithm computes that adding a large coarse region to the selection mask will best improve the score against the ground truth at this step. From Figure 5.16 (b) to (d) the algorithm chisels away regions at increasingly finer levels-of-detail. The greedy algorithm will eventually add fine regions to complete the object if adding regions is found to be

better than chiseling the excess away. We find that using our own optimal markers versus using the dataset markers is a better simulation of how a human user would perform segmentation with *region selection*.



(a) (b) (c)

(d)

Figure 5.16: The first few interactions needed to extract the bear from the background. White regions represent the regions added to the selection mask at that step. Black regions represent the regions removed from the selection mask.

We present our results using our own markers in Table 5.2 alongside the results of Zhao et al. [ZND+11]. We show the scores when using 6, 13, 26, 50, 62 and 100 markers to be able to compare with the results of Zhao et al. which use 6, 26, 62, and 100 of their own markers. We have run our experiment from 1 to 50 markers, and present full results

in graphs form in Figure 5.18 to 5.20. Our results show clearly that *region selection* only needs a low amount of markers to reach good region coverage compared to marker-based methods. Our scores using 6 markers with *region selection* are similar to the scores with marker-based interaction that use 100 markers. The strength of our method is how it leverages the use of an automatically generated segmentation hierarchy. We acknowledge that *region selection* interaction involves selecting a level-of-detail for each markers, and that this interaction is not accounted for in the benchmark results. It would be interesting to run a subjective user study with human participants given the task to segment different images with different methods, but we have chosen a more objective approach because it is reproducible, and other researchers can therefore objectively compare their results with ours in the future, just like we did with the marker-based results of Zhao et al. We believe *region selection* to be a great alternative to marker-based interaction not just for its lower marker count for similar region coverage, but for offering the advantages described in Section 5.2.

Method	Number of Markers					
	6	13	26	50	62	100
Region Growing	.82	.87	.90	.91		
SLIC	.86	.91	.93	.93		
SRM	.88	.91	.92	.93		
Graph Cuts [BJ01]	.41		.63		.80	.82
Geodesic matting [BS07]	.36		.53		.74	.80
Power watersheds [CGNT09]	.50		.72		.84	.88
Random walker [Gra06]	.46		.71		.84	.88
Structural IS [NGC$^+$08]	.49		.69		.82	.87

Table 5.2: Summary of region coverage results after object extraction using our own markers. The last five rows are results published by Zhao et al. [ZND$^+$11] using 6, 26, 62, and 100 of their own markers.

5.4 Future Work

The benchmark dataset developed by Zhao et al. is the only objective benchmark system we are aware of for comparing interactive segmentation algorithms. The Jaccard index

used in the benchmark is biased by the size of the foreground object relative to the background. The rough segmentation of relatively big region will yield a higher Jaccard index. We have worked around this issue to be able to compare foreground extraction scores with background extraction scores. We believe the Jaccard index is a poor choice of metric, especially if the scores are aggregated among multiple images as it with the results of this benchmark, because the object to extract in each image do not all have the same relative size. We would like to research normalized alternatives to the Jaccard index for scoring a segmentation against a ground truth.

We have identified minor improvement we would like to see in the dataset of Zhao et al. Their markers are 10×10 regions, but sometime the markers are fragmented. We would have preferred a simple 1×1 region for simplicity. To work with their bigger markers, we simulate up to a hundred selections per markers. We treat overlapping predefined markers as one big marker.

Since our method is iterative, we would have preferred if the markers would have been given in a specific order of importance. We understand that marker-based interactive segmentation algorithms use all the markers simultaneously so the order does not matter. At least, we would have expected the four levels of interaction given in the study to be incremental, but they are not, *i.e.*, the markers found in lower levels are not present in the high levels. We have experimented with heuristics for ordering the markers, *e.g.*, interlace the use of background and foreground markers, but found that random order was as good as any ordering we have tried.

We found the ground truths themselves to be of questionable accuracy. This is normal, since segmentation is subjective, but in this case the ground truths are very liberal as is shown in Figure 5.17 where the antennas and legs of the butterfly subject are completely ignored in the ground truth. We disagree with this type of ground truth which rewards rougher segmentation.

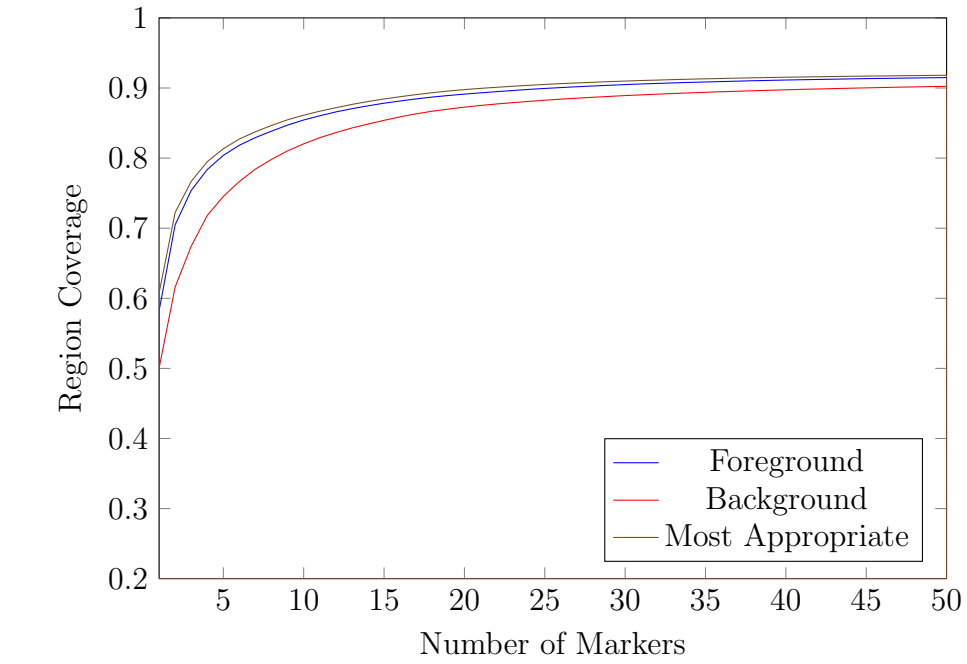
5.5 Summary

In this section we have discussed some drawbacks of using marker-based interactive segmentation method when desiring precise segmentation, notably the unpredictable impact of interaction on the results. We presented a novel method for accurate segmentation we call *region selection* that leverages the use of an automatically generated segmentation hierarchy for interactive segmentation. We have presented the positive results of our method in an existing objective evaluation benchmark for interactive image segmentation

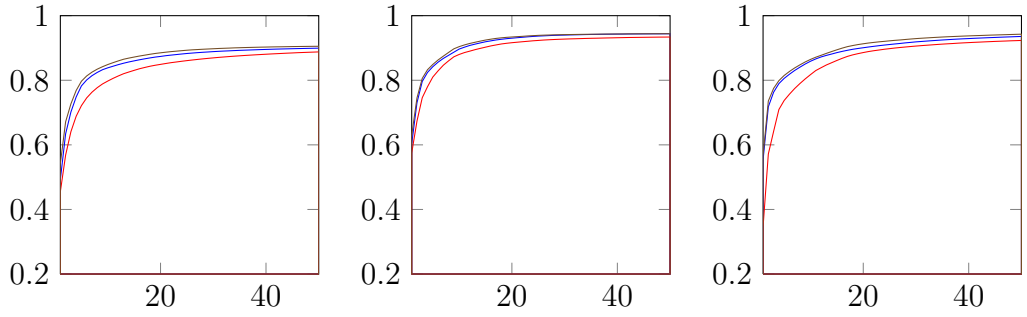
(a) Source (b) Ground Truth

Figure 5.17: Questionable ground truth segmentation for a butterfly in focus.

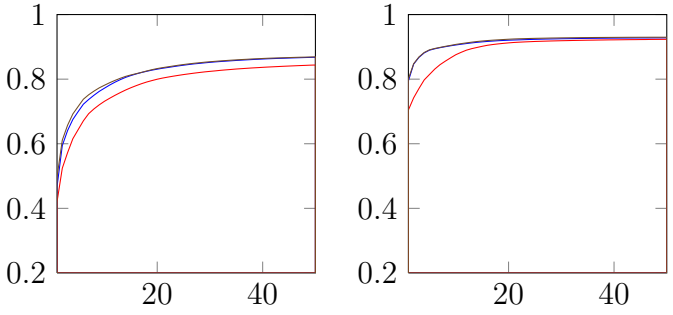
methods developed by Zhao et al.



(a) All Categories

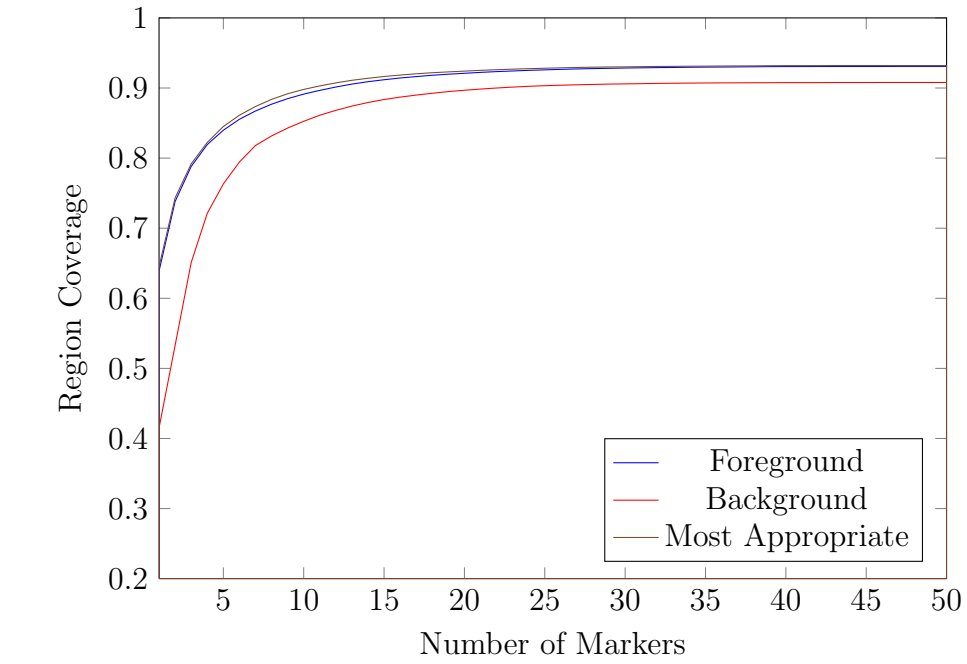


(b) Animal (c) Artifact (d) Building

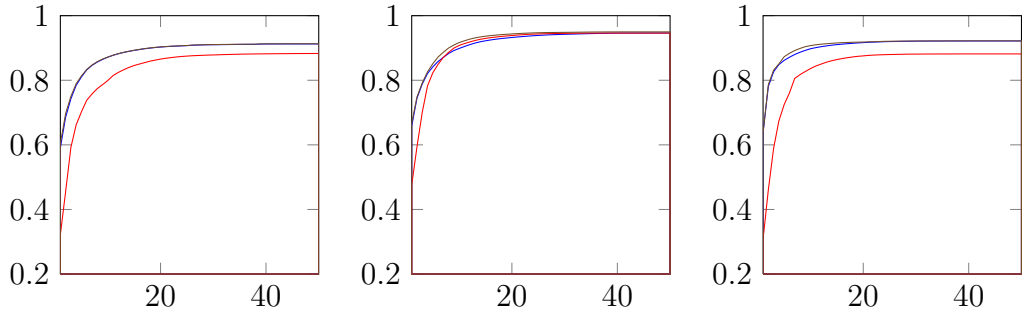


(e) Human (f) Plant

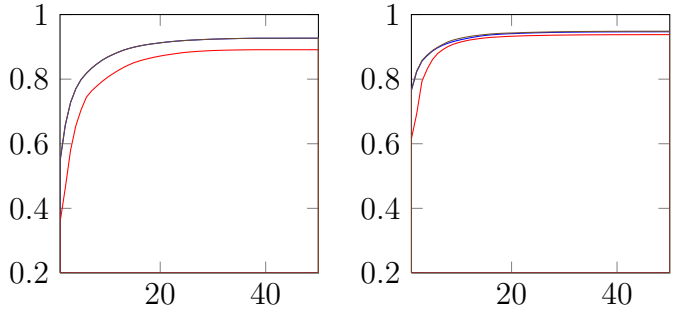
Figure 5.18: Region coverage per additional markers using Region Growing for the the different categories of images.



(a) All Categories

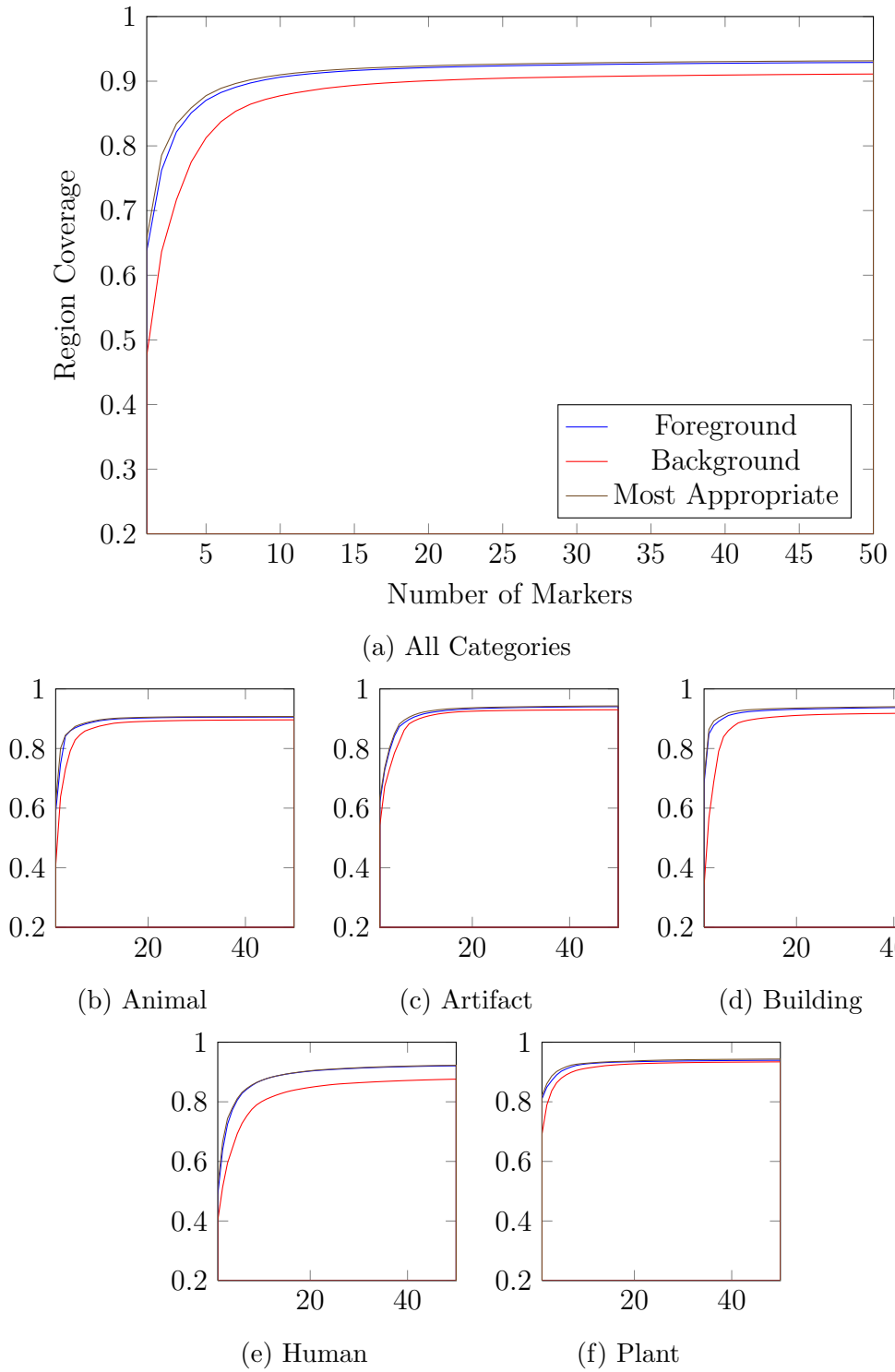


(b) Animal (c) Artifact (d) Building



(e) Human (f) Plant

Figure 5.19: Region coverage per additional markers using SLIC for the the different categories of images.



(a) All Categories

(b) Animal

(c) Artifact

(d) Building

(e) Human

(f) Plant

Figure 5.20: Region coverage per additional markers using SRM for the the different categories of images.

Chapter 6

System Evaluation

In this chapter we present the user study we have conducted to measure the effectiveness of our haptic image exploration system. We begin in Section 6.1 by describing the design of our experiment and the details of its execution. Section 6.2 shows each of the images used in our study, and reports the aggregated results for each individual question. We present the analysis of the results in Section 6.3 and summarize the comments given by our participants. We follow with a discussion in Section 6.4. We give a summary of this chapter in Section 6.5.

6.1 Experiment

6.1.1 Experimental Design

Our haptic image exploration system implements the haptic rendering methods we describe in Chapter 3, and uses a strict *object-level* segmentation hierarchy as input extracted from an image with the authoring tool presented in Chapter 4.

We have designed and conducted a user study to compare how well a user can get an understanding of an image using only haptics given different exploration conditions. The features that we have evaluated are access to multiple levels-of-detail of the same image and access to textured surface. We divided the participants into four groups identified in this document as *No Texture*, *Textured*, *LOD No Texture* and *LOD Textured*. The groups without *LOD* in their name had access to only the finest level-of-detail for each of the images. For the groups with *No Texture* in their name, rendering was done with smooth surfaces for all of the segments. Comparing the performance of the participants in these four groups will help us measure the possible benefits of the level-of-detail feature,

and of encoding segment size information in the texture for image understanding. Our first expectation is that our support for multiple levels-of-detail is an effective way to render the step-by-step displays of Edman [Edm92] on a computer, and that is should therefore have a positive impact on image exploration. Our second expectation is that ambient information about the size of the current segment conveyed through haptic texture will give the user a larger field of view, and therefore also have a positive impact on exploration. Overall, we expect participants of the *LOD Textured* group to perform better than members from the other groups.

We made our participants explore real-life photographs, and not artificial pictures of simple geometry because our goal is to develop and measure a system for haptic exploration of general imagery. The source images were all photographs of moderate complexity, *e.g.*, a single subject and background noise. We chose photographs of subjects that are familiar to anyone, *e.g.*, of a car, of a flower or of a horse. Our selected images for the experiment include only one main subject each, but also include background noise, occlusions and inner details in the objects.

We divided the study in two experiments and a initial training period. In the first experiment the participant must answer questions about an image. In the second experiment the participant performs cross-modal identification of image from haptics to visual. The first experiment is inspired by the methodology of Petit et al. [PDL+08]. In their study, Petit et al. asked participants to explore schoolbook material haptically using their STReSS2 device supported by audio cues, and answer questions about them. We want to test if exploring an *object-level hierarchy* with our display method is sufficient for understanding the image. In our first experiment we ask the participant to explore six images. A short description of the current image is given, as well as two questions about it. The participants acknowledge having read the description and questions before they beginning their exploration. The user is given 3 minutes to explore each of the images. We chose this duration because it is an amount of time participants can work without needing rest. We wanted to given as much time as necessary to the participants, while keeping the total duration of the study under 30 minutes per participants to mitigate the effect of fatigue in later questions. The duration was tested during a small pilot study we have conducted to adjust such parameters. The description, questions, and timer are shown on the screen during exploration, but nothing else is shown, *e.g.*, no cursor, scale or LOD status. The participants may rest between questions if they wish. One question is answered verbally, and recorded by the experimenter on paper. I have conducted the experiments for all participants, and refer to myself as the experimenter in this Chapter.

We have the participant answer verbally because they can do so without having to pause their current activity. The second question for each image always asks to find a specific region in the image and is answered with the space bar by the participants when they believe the cursor is in the region. The images and questions are shown in Section 6.2.

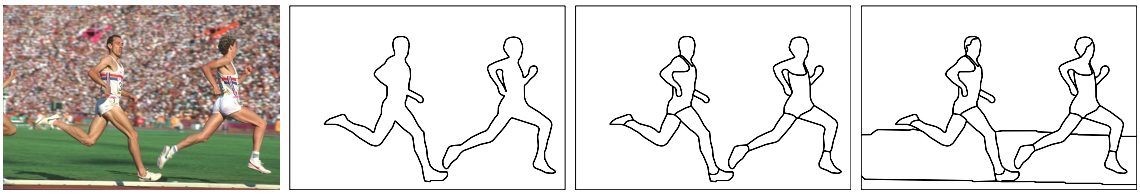
The cross-modal identification experiment is inspired by the work of Dopjans et al. [DWB09]. In their work, they study the cross-modal transfer between visual and haptic face recognition. Their study involved only human faces, and throughout the study the users became more familiar with haptic cues for facial recognition. We are interested to know if the images learned using our system also transfer to visual recognition. We seek to answer if the user of our system can recognize visually an illustration which they have only been exposed to haptically. We ask the participants to explore three images haptically the same way they have done in the previous experiment. However, for this experiment no descriptions of the images are given, and the timer is set to 1 minute instead of 3 minutes. We chose a shorter duration for because we believed it was enough to find the information in the image needed for visual recognition. Because the whole minute is used, we do not measure the effect of the available duration, and this is a limitation of our study. After the user has explored one image, the experimenter ask him/her to identify visually the image among 9 source photographs, *i.e.*, original images that are not segmented. The participants are informed beforehand about this task. The goal of this specific experiment is to measure how well the mental images built using haptic exploration with our system transfers to visual recognition of the same image. In other words, the goal was to see if one can form a distinct enough mental image when exploring an image haptically without description with our system. If so, then transfer should be successful.

6.1.2 Experimental Procedure

The participants are asked to read an instruction sheet before the experiment begins. The sheet explains the various controls available to them during exploration, *i.e.*, zooming, panning and if applicable changing the level-of-detail. The sheet also covers controls to advance to the next question, and general guidelines for handling the device. A copy of the instructions for the *LOD Textured* group, which encompass all features, is found in Appendix A.

After reading the instructions, the participants are given time to familiarize themselves with the software and hardware of our system. During this training period, the

user can see the cursor over the segmented image on the computer screen. The image is a level in the segmentation hierarchy (Figure 6.1 (b),(c) and (d)) of the photograph of two runners shown in Figure 6.1 (a). The participants see the segmented view of the image on the screen only during training. We chose to expose the concept of segmentation to the participants in order for them to understand that they will be following the contours of the segmented objects during exploration. The experimenter informs the participants that the image was originally a photograph and that it has been segmented. An alternative would have been to show the original photograph during training. We inform them that all images they will be exploring will also be from photographs, and therefore shapes will not be as smooth or simple compared to what could be expected of an artist's drawing.



(a) Source (b) Coarse LOD (c) Medium LOD (d) Fine LOD

Figure 6.1: The training photograph
The training photograph and its multiple levels-of-detail.

During training, the participants are asked to read the image description and questions and press ENTER to begin exploration. The description is: "Runners on a track". The example of an open question is: "How many runners are there"? The example of a region location task is: "Find the head of a runner". At the beginning of the exploration, the participants are asked to acknowledge that they can feel the image plane. They are asked to show the experimenter that they can follow the grooves in the image. At this point they are encouraged to experiment following grooves from different sides so that they understand that the grooves trace the contour of the current segment only. Members of the *LOD* groups are shown how to change the level-of-detail, and instructed that at the beginning of each image the level will be at its finest level. Members of outside the *LOD* groups are not made aware about the existence of this feature not available to them. Edman [Edm92] recommends exploring coarser levels first, but we chose the finest level as default such that the starting condition is the same for all groups.

Initially, we opted not to inform the participants of the *Textured* groups about the existence and meaning of the haptic textures. We felt the participants had received

enough instruction already, and that teaching this additional concept would draw away their attention from the other instructions. It was our hope that this feature would be learned intuitively. We conducted a small pilot study with two participants, to help us adjust details of the experiment. The pilot participant with texture enabled was not instructed about the feature, and we found that the texture confused the participant to the point that texture impulse were misinterpreted as walls. With the results of our pilot study in mind, members of the *Textured* groups are informed about the vibration that occurs while dragging the cursor on the surface of a segment. They are asked to feel the difference between the texture of smaller segments versus bigger segments. Members of *No Textures* groups are not informed about the existence of this feature not available to them.

Participants are shown how to zoom in and out, and told that by default the image will be scaled to fit the available workspace. They are informed of the panning that happens when zooming in, moving the cursor, and zooming out. Because they can see the training image and the haptic workspace limits on the screen, they can understand the potential downside of zooming. Participants are shown how to pan using the button on the device. They are encouraged to experiment with panning and zooming without looking at the screen.

The experimenter asks the participants to feel the difference between the haptic workspace virtual walls, and the image plane virtual wall. The haptic workspace limits are bounded by a walls that feel elastic. They are informed the difference can help them judge if they should scale the image or pan when reaching a wall. The participants are told how to answer the questions, *i.e.*, verbally for the open question, and with the space bar for the region location task. Our pilot study showed that having a separate control for advancing to the next image, and for answering the location task was problematic. Because we want to track the total time of exploration, we found it simpler to advance to the next question once the location task is completed. This is because the two participants of our pilot study had a tendency to take a break after answering the questions, but before pressing the key to advance to the next image. We prefer to advance to the next image when the location task is completed so that the time recorded is accurate, even though this means the participants must answer that question last. The participants are informed about this order, and told that the experimenter will encourage them to answer the location task when 30 seconds are left on the timer (since some participants may have their eyes closed when exploring and may not see the timer). We inform the participants that the images are presented in random order, and that this means the

images are not ordered in increasing difficulty.

Finally, the participants are instructed on how to resume exploration in the eventuality that the device get stuck on overlapping lines, a resolution problem described in Chapter 3. The workaround is to zoom in and zoom out quickly. This occurred with about half of the participants, for some of which it happened many times. Handling the device gently seemed to avoid the problem altogether.

After the first experiment is over, and before the cross-modal identification experiment begins, we instruct the participants that they will have less time and no description of what they are exploring for the remaining images. They are told that after the time runs out, 9 images will appear and that they will be asked to identify the image they believe is the one they were exploring. If they so desire, the participants are debriefed at the end of the experiment on their individual test results.

6.1.3 Questionnaire

After both experiments are complete, the participants are asked to answer a Likert questionnaire and to write comments about their experience. We used a five degrees Likert scale: "Strongly disagree", "Disagree", "Neither agree nor disagree", "Agree" and "Strongly Agree" shown in this order. The questions were:

1. I am confident of my answers to the questions for each images.
2. Answering the questions was easy.
3. Controlling the level of detail was useful (asked of LOD group members only).
4. Moving the image was useful.
5. Zooming was useful.
6. Keeping track of my position in an image was easy.

The original design of our study did not include explicit explanation regarding the texture feature to members of the *Textured* groups. This is the reason no questions about textures are in the questionnaire. After the pilot study we added explanation on the texture feature in the training period, but we have not updated the questionnaire accordingly.

6.1.4 Data Collection

We recorded on paper (and later in a database) the answers to each open questions, Likert questionnaire answers and comments. A log of the cursor position, scaling factor and level-of-detail was recorded during exploration for each of the images at every 100 milliseconds or when a state change occurred. The log is timestamped for every entry and contains the position marked by the participant to answer the location question.

6.1.5 Participants

Twenty-seven participants volunteered to take part in our study, 23 males and 4 females, aged between 18 and 40. All participants were sighted, and had no handicaps to report aside one participant being blind in one eye. Though we avoided selecting photographs including pronounced perspective or complex occlusions in our study, we wanted to verify our proof of concept with sighted participants before contacting blind candidates such that we may improve our system before exposing it to users with special needs. Systems developed specifically for the blind are normally multi-modal, *e.g.*, include haptics and audio, and avoid complex occlusion and pronounced perspective in images. It is beyond the scope of this first study to measure the impact of perspective and occlusion on a blind participant, or measure the difference in strength and weaknesses between blind and sighted participants using our system.

The user study sessions were held between July 4 and August 3, 2011 (31 days). We relied on participants making a best effort during the experiments. While conducting the user study, three participants showed signs of having lost interest in the activities. Each of these participants were in *LOD* groups. Besides our impression of their behaviour in the study, their activity log also presented evidence of a very low usage of the level-of-detail feature compared to their peers in the same groups. This evidence supports our decision in discarding their data, though we acknowledge that no similar measure exists for members outside the *LOD* groups. We have made the decision to reject the three participants, 1 male and 2 females, prior complete analysis of the data. The results that we present in this chapter use the data of the twenty-four other participants, six in each of the groups.

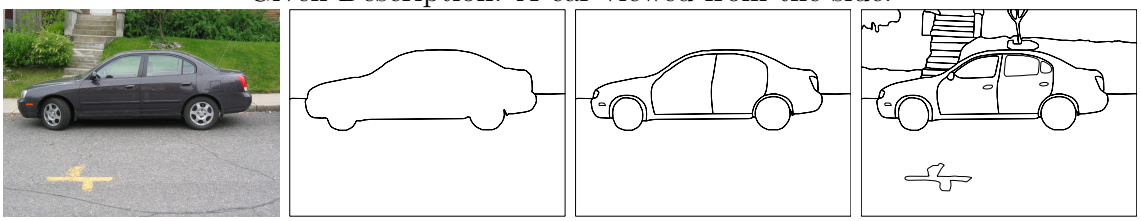
Each participant was assigned to a group without bias, except the last three candidates who were assigned to the respective groups of the rejected participants. This was done to have a balanced number of participants in all groups. Participants were not told about the existence of other groups, except during debriefing after the experiment.

6.2 Individual Question Results

In this section we show each of the images used in our study and report the aggregated results for their respective questions.

6.2.1 The Car

Given Description: A car viewed from the side.



(a) Source (b) 3 segments (c) 12 segments (d) 47 segments

Figure 6.2: The car photograph
The car photograph and its multiple levels-of-detail.

We have prepared a segmentation hierarchy from the photograph of a car. The source image and the levels-of-detail of the hierarchy exposed to the participants are shown in Figure 6.2.

The participants were given the short description: "A car viewed from the side" before beginning exploration. The open question asked was: "Would you describe the car as a sedan (normal car, not a bus/truck)?" and the location task statement was: "Find the rear wheel".

The original question did not include the clarification on the meaning of the word "sedan". The clarification was added after observing that most participants did not understand its meaning, or had a different definition. One participant verified with the experimenter before starting exploration if he needed to count the number of doors, because his own definition of the word "sedan" was a four doors car. We saw no problem in updating the question for clarity during the study because the early participants could verify their understanding of the question with the experimenter before the timer began.

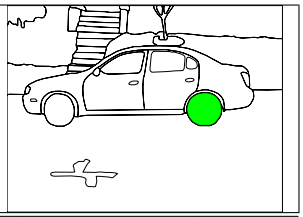
Comments from the participants and our cursor movement log reveals that the upper front curve of the car was useful in identifying both the type of the car, and its orientation. Table 6.1 lists the answers given by the participants in each groups for the open question. Table 6.2 reports the number of correct, incorrect and absent answers in each group for the location task.

Table 6.3 shows the average time of exploration used to answer both questions correctly. Our hypothesis was that access to multiple levels-of-detail aids exploration greatly, and that ambient global context encoded in the textures also improves exploration. It is interesting to see that the time necessary to answer accurately both questions decreases when multiple levels-of-detail are available, and decrease even more when both level-of-detail and textures are available. Also, no participants from the group without access to any of the additional features were able to give the right answer to both questions. Our hypothesis is supported by these observation, but as shown in Section 6.3, this kind of results is not visible when all images are considered. The car image was easier than the other images, and a third of the participants were able to find the good answer to the questions. Unfortunately, the number of participants that were able to answer correctly both questions for the other images was too low for us to compute averages per groups.

Question: Would you describe the car as a sedan (normal car, not a bus/truck)?
Correct Answer: yes.

Answer	No Tex.	Textured	LOD No Tex.	LOD Tex.	Total
yes	1	4	5	5	15
no	5	2	0	1	8
(no answer)	0	0	1	0	1

Table 6.1: Answer to the open question about the car photograph.

Location Task	Find the rear wheel.	

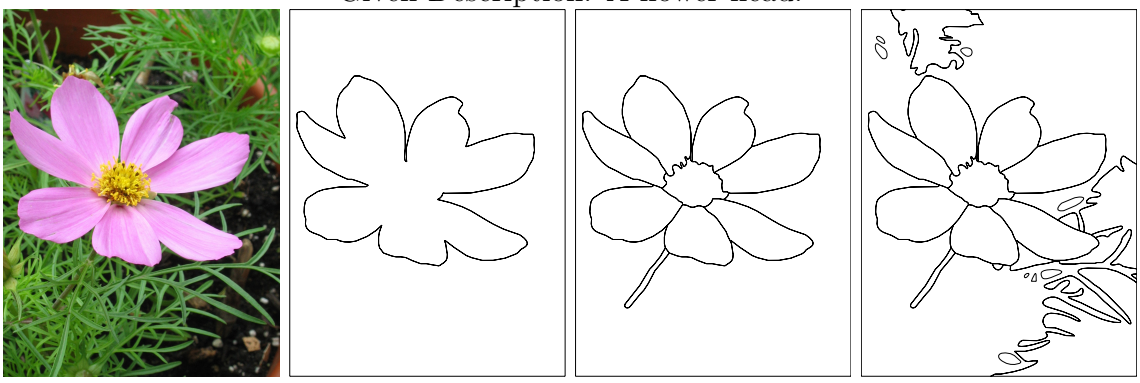
	No Tex.	Textured	LOD No Tex.	LOD Tex.	Total
Found	1	2	4	2	9
Incorrect	4	3	1	4	12
(no answer)	1	1	1	0	3

Table 6.2: Correct number of answer to the location task in the car photograph.

	No Tex.	Textured	LOD No Tex.	LOD Tex.	All Groups
Average Time (s)	-	162	122	95	125
stddev (s)	-	6	47	15	40
Number of samples	0	2	4	2	8

Table 6.3: Average time of exploration used to answer both questions correctly about the car photograph.

Given Description: A flower head.



(a) Source (b) 2 segments (c) 10 segments (d) 26 segments

Figure 6.3: The flower photograph
The flower photograph and its multiple levels-of-detail.

6.2.2 The Flower

We have included the segmentation hierarchy of a flower in our study. Figure 6.3 shows the source image and the levels-of-detail of the hierarchy.

The short description given was: "A flower head". The open question asked was: "How many petals are there?" and the location task statement was: "Find the flower centre".

The segmentation hierarchy we have prepared for this image was prone to the overlapping lines limitation described in Section 3.1. The lines defining the contour of the flower centre are too close at the default scale in the upper-left region.

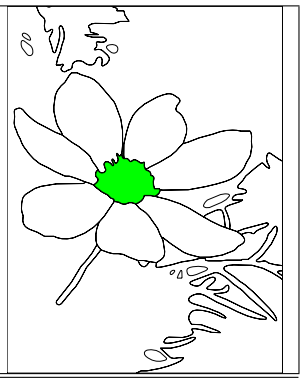
Though it was explained to each of the participants that all images were prepared from photographs, many participants expressed surprise when not being able to find a perfect circle where they expected the centre to be. The results are shown in Table 6.4 and Table 6.5. It was difficult for the participants to count the petals. Feedback from the participants with no access to multiple levels-of-detail mention the difficulty of managing the background, or simply not being certain if a petal was already counted or not. One participant decided to measure the arc length of one petal and estimate a petal count after having given up on counting the petals one by one. His answer was 8 petals, which would have been the correct answer if the flower did not have gaps between some of its petals.

Question: How many petals are there?

Correct Answer: 7.

Answer	No Tex.	Textured	LOD No Tex.	LOD Tex.	Total
3	0	0	2	2	4
4	1	1	0	1	3
5	1	3	0	1	5
6	1	1	1	1	4
7	1	0	1	0	2
8	0	1	1	0	2
9	0	0	1	0	1
10	1	0	0	1	2
no answer	1	0	0	0	1

Table 6.4: Answer to the open question about the flower photograph.

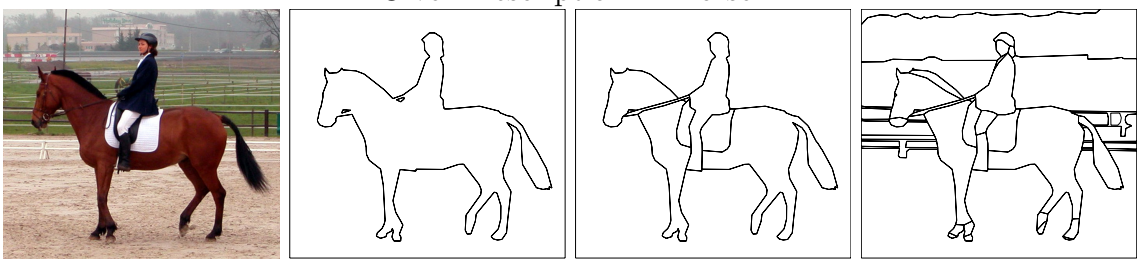
Location Task	Find the flower centre.	

	No Tex.	Textured	LOD No Tex.	LOD Tex.	Total
Found	0	2	4	2	8
Incorrect	5	4	2	4	15
(no answer)	1	0	0	0	1

Table 6.5: Correct number of answer to the location task in the flower photograph.

6.2.3 The Horse

Given Description: A horse.



(a) Source (b) 4 segments (c) 10 segments (d) 57 segments

Figure 6.4: The horse photograph
The horse photograph and its multiple levels-of-detail.

We had the participants explore the image of a horse. The source image and the levels-of-detail of the hierarchy exposed to the participants are shown in Figure 6.4. The photograph was taken by Karoly Lorentey, and can be found in Wikimedia Commons under the name "Horse riding left".

We presented the image as: "A horse". The open question asked was: "Is there a rider on the horse?" and the location task statement was: "Find the tail".

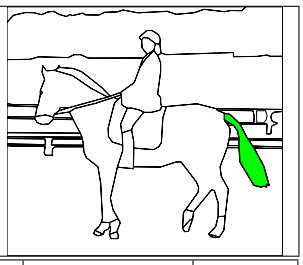
The results for this question are given in Table 6.6 and Table 6.7. Several participants mentioned expecting the horse to face right. This incorrect assumption can explain the

poor results for the finding task. The source of the assumption is unclear though one participant argued that on race tracks, horses are running facing right when viewed from the closest side of the crowd.

Question: Is there a rider on the horse?
Correct Answer: yes.

Answer	No Tex.	Textured	LOD No Tex.	LOD Tex.	Total
yes	4	2	3	2	11
no	2	4	3	4	13

Table 6.6: Answer to the open question about the horse photograph.

Location Task	Find the tail.	

	No Tex.	Textured	LOD No Tex.	LOD Tex.	Total
Found	0	0	4	0	4
Incorrect	6	5	2	5	18
(no answer)	0	1	0	1	2

Table 6.7: Correct number of answer to the location task in the horse photograph.

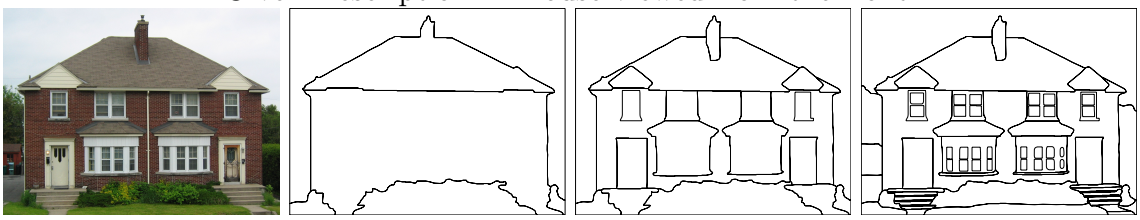
6.2.4 The House

We have prepared a segmentation hierarchy from the photograph of a house. We show the source image and the levels-of-detail of the hierarchy in Figure 6.5.

The image was given the short description: "A house viewed from the front". The open question was: "How many windows would you say there are?" and the location task was: "Find the chimney".

We list answers given by the participants in each groups for the open question in Table 6.8, and report the number of correct, incorrect and absent answers in each groups for the location task in Table 6.9. The window counting question was very difficult, mainly because the house is not the typical house one would expect to see, *e.g.*, there is

Given Description: A house viewed from the front.



(a) Source (b) 6 segments (c) 20 segments (d) 80 segments

Figure 6.5: The house photograph
The house photograph and its multiple levels-of-detail.

more than one door. Arguably, it is also hard to count the windows visually, considering the ambiguity of the necessity of counting individual window panels or just the main frames. Finding the chimney was relatively easy compared to counting the windows. Overall, we have made a poor choice of photograph. The idea was to have some images more difficult than others in order to gather useful results, but in retrospect this image was too hard, and easier images like the car in Section 6.2.1 were difficult enough.

Question: How many windows would you say there are?
Accepted Answers: 6 or 28.

Answer	No Tex.	Textured	LOD No Tex.	LOD Tex.	Total
2	4	2	2	3	11
3	1	2	2	0	5
4	0	1	2	0	3
5	0	1	0	2	3
(no answer)	1	0	0	1	2

Table 6.8: Answer to the open question about the house photograph.

6.2.5 The Juggler

We built a segmentation hierarchy from the image of a juggler. The image and the exposed levels-of-detail of the hierarchy are shown in Figure 6.6. The photograph is copyrighted to 'The Fire Man Dave', whom granted us permission to use the image in our work.

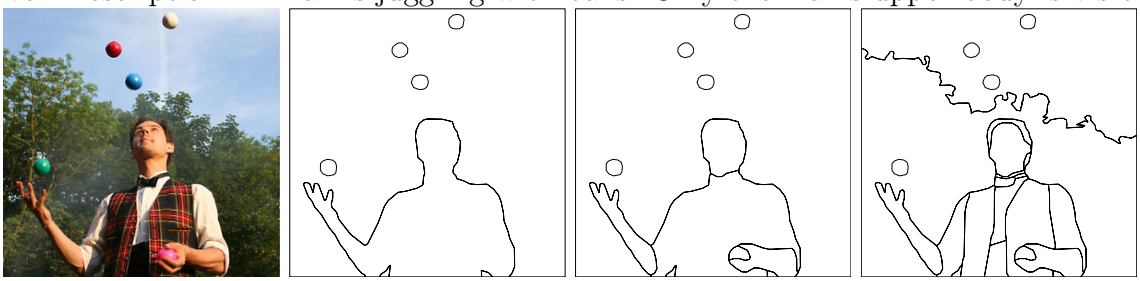
The participants were given the short description: "A man is juggling with balls. Only the man's upper body is visible". The question and task were: "How many balls

Location Task	Find the chimney.	

	No Tex.	Textured	LOD No Tex.	LOD Tex.	Total
Found	2	5	2	4	13
Incorrect	3	1	4	1	9
(no answer)	1	0	0	1	2

Table 6.9: Correct number of answer to the location task in the house photograph.

Given Description: A man is juggling with balls. Only the man's upper body is visible.



(a) Source (b) 8 segments (c) 12 segments (d) 23 segments

Figure 6.6: The juggler photograph
The juggler photograph and its multiple levels-of-detail.

is the man juggling with?" and "Mark the position of a ball".

This contour image we have prepared was prone to the overlapping lines limitation described in Section 3.1. The lines defining the background trees are too close together at the default scale.

Table 6.10 lists the answers given by the participants in each groups for the open question. Table 6.11 reports the number of correct, incorrect and absent answer in each groups for the location task. It was difficult for the participants to count the balls. The background trees were a big distraction at the finest level-of-detail, and finding small objects in an mostly empty space with a cursor is inherently hard. Some participants felt it was appropriate to report counting just one or two balls, even though that would hardly be considered juggling. Some participants expressed interest in looking for balls in the hands of the juggler during exploration. Because the ball held in one of the hand is not well defined due to the hand occlusion, we chose to accept answers that did not include it.

Question: How many balls is the man juggling with?
Accepted Answers: 4 or 5.

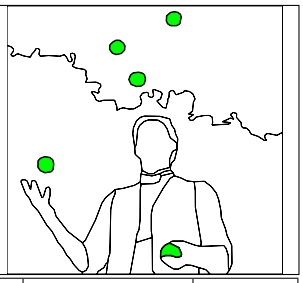
Answer	No Tex.	Textured	LOD No Tex.	LOD Tex.	Total
1	0	1	0	1	2
2	1	0	0	3	4
3	2	2	5	0	9
4	1	1	0	1	3
5	2	1	0	1	4
6	0	1	1	0	2

Table 6.10: Answer to the open question about the juggler photograph.

6.2.6 The Clock Tower

We took a photograph of the clock tower of the Parliament of Canada and prepared a segmentation hierarchy from it. The photograph and the levels-of-detail of the hierarchy are displayed in Figure 6.7.

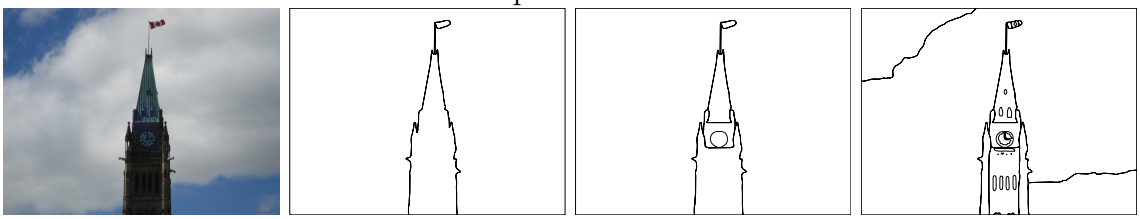
We simply gave the participants the short description: "A clock tower", and asked them "What time is it on the clock (15 min precision)?". We also asked asked them to "Mark the position of the clock".

Location Task	Mark the position of a ball.	

	No Tex.	Textured	LOD No Tex.	LOD Tex.	Total
Found	3	1	5	4	13
Incorrect	3	5	1	2	11
(no answer)	0	0	0	0	0

Table 6.11: Correct number of answer to the location task in the juggler photograph.

Given Description: A clock tower.



(a) Source (b) 2 segments (c) 7 segments (d) 33 segments

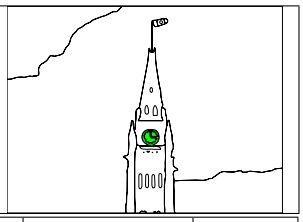
Figure 6.7: The clock tower photograph
The clock tower photograph and its multiple levels-of-detail.

The participants results for this image are presented in Table 6.12 and Table 6.13. The cursor movement logs tell us that the vast majority of participants spent their time on the outside of the tower. For the participants with access to multiple levels-of-detail, they spent most of their time at a too coarse level where the clock is not visible. Many participants answered 6 o'clock, or 12:30, which represents a straight line. Only one participant found the clock and read the correct time. The participant was part of the *LOD No Texture* group, which according to our analysis in Section 6.3 was the best performing group. For this question, the participant chose not to decrease the level-of-detail. This question was the fourth one in the participant's randomly assigned order of questions. The logs of the previous images he had completed show that he had lowered the level-of-detail for all three previous images. After 80 seconds of exploration, he was able to find the clock. He zoomed in and took an additional 80 seconds to carefully read the clock hands. We knew this image would be harder than the others, but did not expect it to be this problematic for the majority of the participants. From the log of the one participant correctly answering the questions for this image, we know that his answer was not simply luck.

Question: What time is it on the clock (15 min precision)?
Correct Answer: 3 o'clock.

Answer	No Tex.	Textured	LOD No Tex.	LOD Tex.	Total
1:45	1	0	0	0	1
2:00	1	0	0	0	1
3:00	0	0	1	0	1
4:00	0	1	0	0	1
5:45	0	1	0	0	1
6:00	2	1	0	1	4
6:15	0	0	0	1	1
7:00	1	0	0	0	1
9:30	0	0	1	0	1
12:00	0	0	1	0	1
12:15	0	0	0	1	1
12:30	0	2	1	0	3
(no answer)	1	1	2	3	7

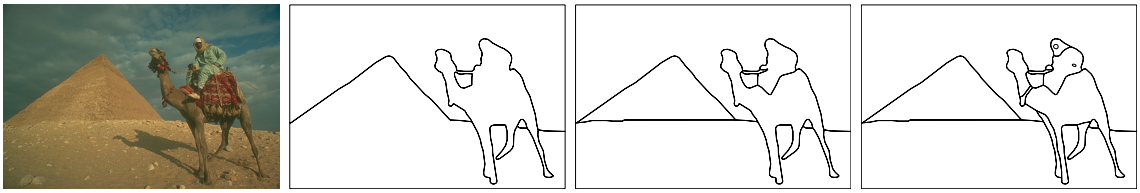
Table 6.12: Answer to the open question about the clock tower photograph.

Location Task	Mark the position of the clock.	

	No Tex.	Textured	LOD No Tex.	LOD Tex.	Total
Found	0	0	1	0	1
Incorrect	6	4	2	5	17
(no answer)	0	2	3	1	6

Table 6.13: Correct number of answer to the location task in the clock tower photograph.

6.2.7 Mystery Image Omega



(a) Source (b) 5 segments (c) 8 segments (d) 16 segments

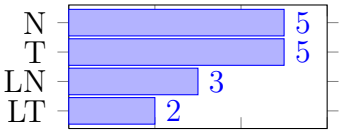
Figure 6.8: The first mystery photograph
The mystery image omega photograph and its multiple levels-of-detail.

This question is part of the second experiment, where participants are asked to visually identify an image they have explored using haptics. We have prepared a segmentation hierarchy from the photograph of a camel in the desert, and chosen eight other photographs to be candidates during the visual identification task. All photographs used for the cross-modal experiment are part of the BSDS500 [AMFM]. Figure 6.8 shows the source image and the levels-of-detail of the hierarchy exposed to the participants during exploration.

The participants were given the short description: "Mystery Image Omega", and instructed that they would only have one minute to explore this image and be asked to identify it visually among 9 candidate images.

Figure 6.9 lists the answers given by the participants in each groups, as well as showing the candidates images. The majority of the participants were able to correctly identify the image. The cursor movement logs show that many participant spent time reviewing

Camel



Across Groups: 15/24 (62.5%)

(a) Number of correct answers per groups

No Texture (N)	*Textured (T)*	*LOD No Texture (LN)*	*LOD Textured (LT)*
Car: 1	Car: 1	Desert: 1 Fireman: 1 Javelin: 1	Javelin: 2 Astronauts: 1 Car: 1

(b) Number of other answers per groups

(c) Car

(d) Astronauts

(e) Javelin

(f) Race

(g) Ski

(h) Ski Jump

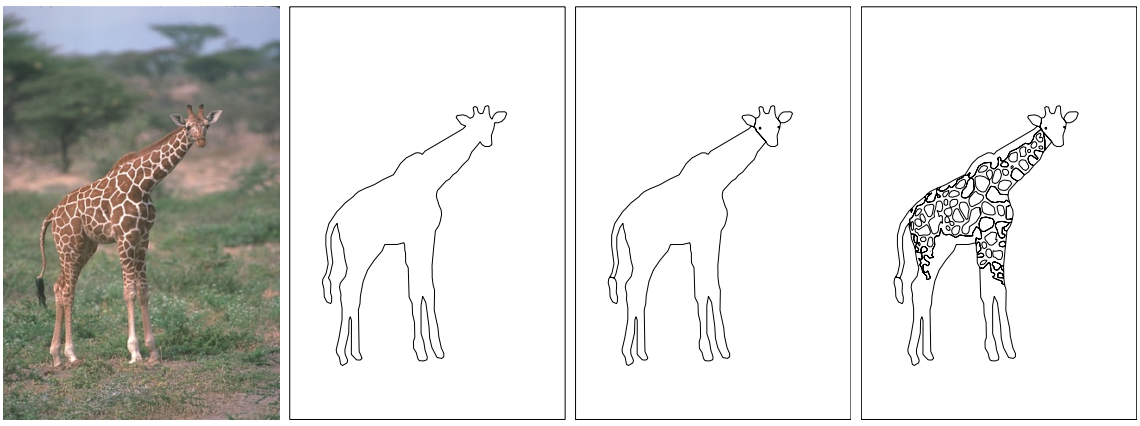
(i) Fireman

(j) Desert

Figure 6.9: Results for the Mystery Image Omega, and the other eight candidate images.

the pyramid shape. One participant mentioned before looking at the nine candidate images that he would be looking for a triangle shape object positioned in the left region of the image, empty space in the upper region, and a complex shape in the right. I thought this was a good description of what can be pictured mentally during exploration of the unknown image. Common mistakes were choosing the javelin picture or the car. The legs of the javelin thrower form a similar shape to the pyramid, though they do not have a base. The car has straight lines, which can also feel similar to the pyramid lines.

6.2.8 Mystery Image Theta

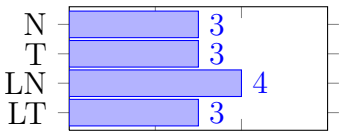


(a) Source (b) 2 segments (c) 8 segments (d) 83 segments

Figure 6.10: The second mystery photograph
The mystery image theta photograph and its multiple levels-of-detail.

We have built a segmentation hierarchy from the photograph of a giraffe. We called this image: "Mystery Image Theta". We show the source image and the levels-of-detail of the hierarchy exposed to the participants in Figure 6.10.

The participants had to pick their answer among 9 images. Figure 6.11 lists the results for this question and shows each candidate images. About half the participants were able to identify visually the image explored haptically. Common mistakes were the "Horses Field" candidate, which has a similar overall shape. The "Boys Sitting" candidate image was also chosen often. One participant mentioned selecting it because she remembered the shape of the heads of the children from exploration. This is erroneous, but perhaps the spots in the giraffe were what the participant felt. Measuring the aspect ratio of the image makes the identification task much easier. It was a oversight on our part to



Giraffe Across Groups: 13/24 (54.2%)

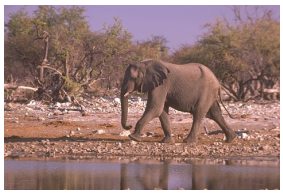
(a) Number of correct answers per groups

No Texture (N)	*Textured (T)*	*LOD No Texture (LN)*	*LOD Textured (LT)*
Boys Sitting: 1	Boys Sitting: 1	Boys Sitting: 2	Horses Field: 2
Horses Field: 1	Horses Field: 1		Boys Sitting: 1
Penguin: 1	Gorilla: 1		

(b) Number of other answers per groups

(c) Boys Sitting

(d) Boys Standing



(e) Elephant

(f) Rhinoceros

(g) Horses Gate

(h) Horses Field

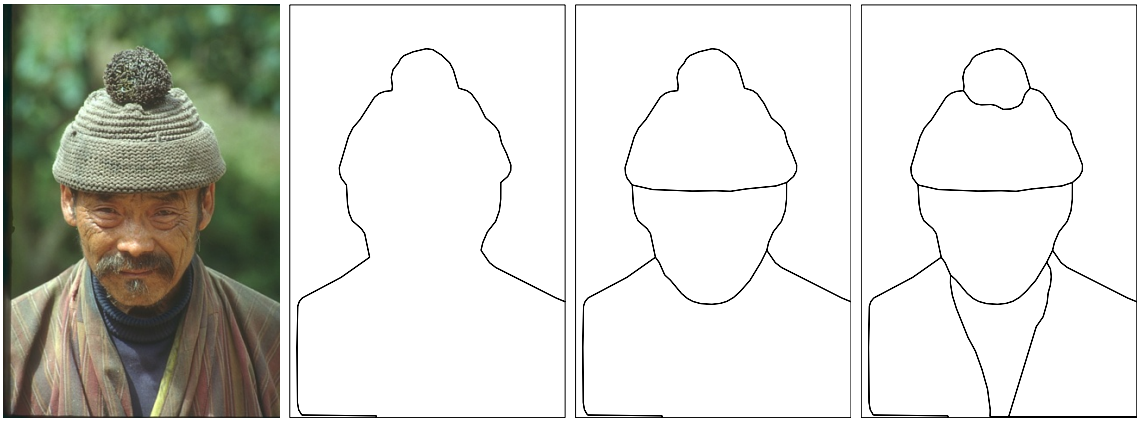
(i) Gorilla

(j) Penguin

Figure 6.11: Results for the Mystery Image Theta, and the other eight candidate images.

present images of different aspect ratio, especially in this case since only two candidates fit the portrait orientation of the image to identify. It is unknown how many participants have used aspect ratio to answer this question.

6.2.9 Mystery Image Zeta



(a) Source (b) 2 segments (c) 4 segments (d) 7 segments

Figure 6.12: The third mystery photograph
The mystery image zeta photograph and its multiple levels-of-detail.

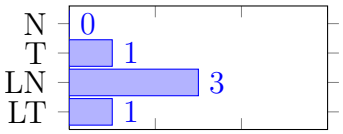
We included in our experiment a segmentation hierarchy built from the photograph of a man wearing a tuque, with the description "Mystery Image Zeta". The source image and the levels-of-detail of the hierarchy are shown in Figure 6.12.

The results for this question are displayed in Figure 6.13. This picture was misidentified by the majority of the participants. This came as a surprise, since even the finest level-of-detail was simple to explore compared to the other two images. This may indicate that exploring finer details can help build a mental image.

6.3 Analysis

6.3.1 Experiment I

A factorial design analysis of variance (ANOVA) pairing the two texture options with the two access to level of detail options was used to test differences between means for significance. The details of the analysis are found in Table 6.15.



Tuque Across Groups: 5/24 (20.8%)

(a) Number of correct answers per groups

No Texture (N)	*Textured (T)*	*LOD No Texture (LN)*	*LOD Textured (LT)*
Girl: 3	Straw Hat: 2	Boy: 2	Cylinder Hat: 2
Stick: 2	Boy: 1	Girl: 1	Boy: 1
Straw Hat: 1	Girl: 1		Elephant: 1
	Grocery: 1		Grocery: 1

(b) Number of other answers per groups

(c) Elephant

(d) Guy

(e) Girl

(f) Boy

(g) Stick

(h) Grocery

(i) Cylinder Hat

(j) Straw Hat

Figure 6.13: Results for the Mystery Image Zeta, and the other eight candidate images.

Group	mean	stddev	median
No Texture	0.22	0.14	0.21
Textured	0.29	0.05	0.29
LOD No Texture	0.44	0.15	0.46
LOD Textured	0.29	0.10	0.33

Table 6.14: Descriptive statistics on the percentage of correct answers per group during the first experiment of our study.

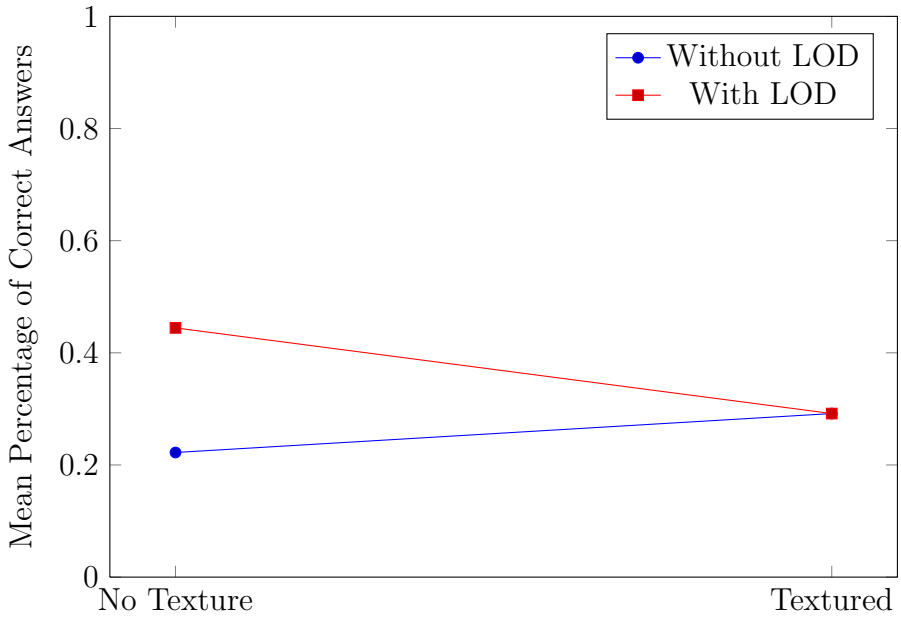


Figure 6.14: Mean percentage of correct answers per group during first experiment of our study.

It was found that groups with access to multiple levels of detail performed better overall than the ones that did not have this feature enabled, $F(1, 20) = 5.66, p = 0.0274$. The difference is visible in the median values of Table 6.14, and can be seen somewhat in Figure 6.14.

An inspection of Figure 6.14 reveals the negative effect of enabling textures when access to multiple levels of detail is also enabled. This impact is reflected in a significant Texture x LOD interaction, $F(1, 20) = 5.66, p = 0.0274$. Our data is inconclusive with regards to the effect of enabling textures in general or enabling textures when LOD is not present.

Source	SS	df	MS	F	$Prob > F$
Texture	0.01042	1	0.01042	0.8	0.3828
LOD	0.07407	1	0.07407	5.66	0.0274
Interaction	0.07407	1	0.07407	5.66	0.0274
Error	0.26157	20	0.01308		
Total	0.42014	23			

Table 6.15: ANOVA table of Experiment I.

6.3.2 Experiment II

Group	mean	stddev	median
No Texture	0.44	0.27	0.50
Textured	0.50	0.35	0.50
LOD No Texture	0.56	0.34	0.67
LOD Textured	0.33	0.37	0.33

Table 6.16: Descriptive statistics on the percentage of correct answers per group during the second experiment of our study.

We analyzed results of the cross-modal identification experiment with a factorial design ANOVA to test differences between means for significance the same way we have done for the first experiment. The details of the analysis are found in Table 6.17.

Though Figure 6.15 is similar in shape to Figure 6.14, the method found no statistically significant effect in the factors. This can be explained by the higher variance of

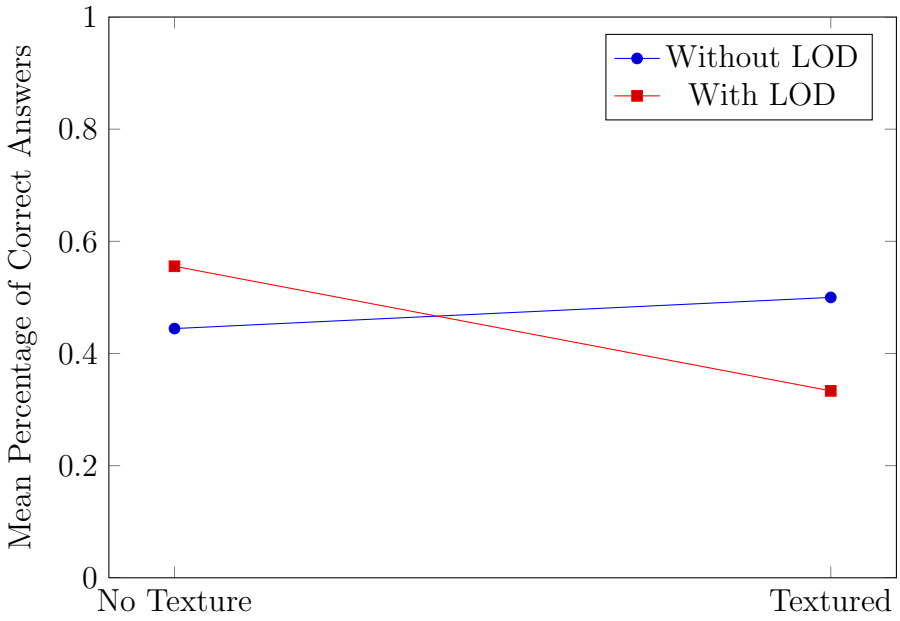


Figure 6.15: Mean percentage of correct answers per group during the second experiment of our study.

the data as seen in Table 6.16.

It would be interesting to know if the higher variance is simply due to having too few tests in the experiment, or if it is caused by a more complex factor like the shorter time available for image exploration in this experiment, the lack of a general description of the image, the nature of the task itself, or the discrepancy in the level of difficulty of "Mystery Image Zeta" compared to the other two images.

We verified that the members of the groups with access to multiple levels-of-detail used the feature as often as they did in the previous experiment. The usage ratio was indeed similar.

Source	SS	df	MS	F	$Prob > F$
Texture	0.04167	1	0.04167	0.37	0.5488
LOD	0.00463	1	0.00463	0.04	0.841
Interaction	0.11574	1	0.11574	1.03	0.3216
Error	2.24074	20	0.11204		
Total	2.40278	23			

Table 6.17: ANOVA table of Experiment II.

6.3.3 Questionnaire

Feedback	No Texture	Textured	LOD No Texture	LOD Textured
Confident	2.7 (0.8) 2.5	1.7 (0.8) 1.5	2.7 (1.2) 2.5	2.0 (0.9) 2.0
Easy	2.0 (0.0) 2.0	1.8 (0.8) 2.0	2.8 (0.8) 3.0	2.2 (0.8) 2.0
Pan	2.7 (0.8) 2.5	2.0 (0.9) 2.0	2.3 (1.2) 2.5	2.2 (1.2) 2.0
Zoom	4.0 (1.1) 4.0	2.2 (0.8) 2.0	2.7 (1.4) 3.0	2.7 (1.4) 3.0
Position	3.8 (1.0) 4.0	3.2 (1.3) 4.0	3.2 (1.0) 3.5	2.7 (1.2) 2.5
LOD			4.0 (1.1) 4.0	3.8 (1.2) 4.0

Figure 6.16: Mean (stddev) and median descriptive statistics on the feedback given in the questionnaire.

The overall feedback scores in our Likert-type questionnaire are low aside from the high feedback scores given to the usefulness of multiple levels-of-detail. The low scores reflect the high difficulty of the experiment.

Participant without textures enabled reported more confidence in their answers than the ones that had textures enabled, $F(1, 20) = 4.63, p = 0.0438$. This suggest ambient texture-encoded information was a source of confusion.

Participants with access to multiple levels of detail found it easier to answer questions, $F(1, 20) = 4.8, p = 0.0404$. The exploration behaviour of members of LOD groups normally involved reducing the level-of-detail right at the beginning of exploration. This strategy was expected, and is an effective way to familiarize oneself with the main objects in the image before exploring at a higher level-of-detail.

No other differences between means were found to be significant, except for an appreciation for the zooming feature from the group with texture disabled, $F(1, 20) = 3.67, p = 0.0699$. This might be explained by the fact that this group only had the zooming feature to help in the exploration.

6.3.4 Subjective Comments

The comments from the participants hint that a different approach should have been used for training. Participants suggested training with more than one image with varying degree of difficulty. Some mentioned that it would have been beneficial to have a trial image after training, and then more time in training before beginning the experiment. All suggestions were good ideas, but they all involve a longer training time, which would

also increases fatigue. We believe the training period we have done was too short to teach the complex concepts of the system to our participants. A practical solution could be to send the participants material that demonstrates the concepts they will practice during the training period before they attend the study. We imagine this material to be in the form of an instructive video or text with figures. The objective is to change the focus of the training period to be more about practicing instead of learning.

A recurring feature request was resetting the image pan and scale to fit the haptic workspace. We had disabled this feature because of a limitation with our implementation of the artificial walls. The device may be positioned outside the allowed region delimited by the image boundary when resetting the image to fit the haptic workspace. Our current implementation kicks back the device into place, as if it had penetrated the wall with high force when reseting pan and zoom. A solution would be to ignore the wall constraints until the user moves the device back inside the image boundaries.

Some comments talked about the difficulty of knowing how much detail is in each of the accessible level-of-detail. This is an important point. The images were all prepared by the same person, *i.e.*, myself, but no specific standard was followed to decide on the amount of information in each level-of-detail. To the opposite, we purposely prepared images with different standards. The finest level-of-detail for the "Mystery Image Zeta" image presented in Section 6.2.9 is very coarse, *e.g.*, the eyes, the nose and the moustache are not visible. We interpret the poor results for this question as an indication that a finer level-of-detail is indeed necessary. It is unfortunate that we chose an image that had a relatively coarse highest level-of-detail for the training image as well, *e.g.*, the background was not very noisy. The scaling factor has a similarly unknown level, though no comments were done about it most likely due to the low use of the zooming feature.

Two participants reported not being able to build a good mental picture of any of the images even when concentrating. This comment tell us that the differences between participants even within the same group can be large, depending on their personal ability to create mental pictures.

6.4 Discussion

The results for the first experiment give us valuable information, that is, the level-of-detail is indeed useful to better understand an image given a short description of the image, and that the feature was appreciated by the participants that had it. This is an important result, it supports the finding of Edman [Edm92] about step-by-step displays,

and tells us that using a strict segmentation hierarchy for this feature is a good solution. It was possible for some users to gain a level of understanding of the images for them to answer our basic questions, even though the images ended up being more difficult than expected.

The experiment failed to demonstrate the usefulness of the segment size information encoded in the texture. In fact, it suggests the opposite when level-of-detail is also present. We believe this might be more due to the overwhelming amount of learning required to participate in the *LOD Textured* group. Having both level-of-detail and texture enabled meant a longer sheet of instructions, and more concepts to learn during the training period. During development, we had found texture to be useful to haptic image exploration. Since we knew the images well, it was impossible for us to evaluate ourselves the potential of textures for haptic exploration. However, we have implemented a game for ourselves were the image to explore haptically was chosen randomly, and the goal was to identify which image was being explored with nothing else than haptics cues. Though not a scientific study, it was only until we enabled textures that we were able to answer accurately and with confidence. We are expert users with our own system, and this game experiment may have only given positive results because we knew very well how to listen to the texture cue. Perhaps using a limited amount of textures that are easily distinguishable instead of procedural textures to convey size information would lower the learning curve to users not familiar with the system. Another thought is to encode something different than segment size information in the haptic texture, something like the smoothness degree in the image data at that position. For example, an object with an high frequency dash pattern observable visually could have a high frequency haptic texture rendered on its surface while a smoother more uniform visual pattern would be rendered by a smooth haptic texture.

The quality of the segmentation hierarchy, the complexity of each task, and the nature of the image itself affects greatly the results. It is unclear what was wrong with the clock tower image of Section 6.2.6. Perhaps the jagged shape of segments extracted from photograph was the problem, and if so brings the question if a complete subjective reinterpretation of the image instead of accurate segmentation would be better for understanding.

With the knowledge we gathered from the study, if we would have to do the cross-modal transfer experiment again we would adjust its design. A repeated-measures design (within-subject) would allow us to separate the differences among subjects from error. Such design normally asks participants to repeat the same task under different conditions.

We cannot design a robust within-subject experiment because participants would already be familiar with an image after the first time they encounter it, and images are not easily forgotten. However, if all images have the same level of difficulty (hard to prove), one can argue that the experiment would be a valid repeated-measures design. We propose varying the time allowed for exploration, and varying access to multiple levels-of-detail. Since our texture implementation had a negative effect, we would not include texture as is in this new study. We only had our participants explore and identify 3 images in this experiment, and as mentioned in Section 6.2.9 one of them was flawed. A larger number would increase the power of the statistical method used during analysis. From the feedback received, we would also including a trial images with each of the conditions during training for a within-subject experiment, and emphasis the fact that the images are prepared from photographs.

The car image presented in Section 6.2.1 had interesting results about the time needed to answer both its questions correctly. Given images of similar level of difficulty to the car image, and a higher maximum time available per image, we could infer more conclusions about the impact of the different features. In retrospect, the level of difficulty of the questions in our study was higher than it needed to be. A moderate level of difficulty would have been ideal for statistical analysis. We could not use statistics about the length of time needed by participants for each questions because that length was often equal to the maximum time allotted. Some of our questions had potentials for greater analysis if minor modifications would had been done to them. The juggler image of Section 6.2.5 comes to mind, where the participants were asked to count the balls and mark the location of one. If we would have asked to mark the location of all the balls they could find once, we could measure the rate of marking the same ball multiple times, which would give us more insight on the challenges of counting similar objects using our system.

There are two limitations with the implementation of our system that would need to be addressed before more experimentation should be done. The main one is about the device getting stuck or becoming unstable when two lines forming the outline of a segment cross at a given scale. There is no easy solution to this problem, but we could develop tools to detect problematic regions in the hierarchy given a minimum scale level which would help the person preparing the images. The second limitation was about the rendering of lines at the image boundary. Several participants mentioned that while moving along boundaries, they could sometime detect lines from one direction and not from the other. This limitation is related to rendering the outline of the current segment

only. At boundaries the current segment can be the outside segment that frames the image, and gives the illusion that there is no object touching the frame of the image. Some participants reported having a hard time trusting the device was functioning properly. The limitations we have just described, and the fact that our device has open wires and feels fragile are the most likely reasons for this low confidence level.

6.5 Summary

In this chapter we have presented the user study conducted to measure the effectiveness of our haptic image exploration system. We have presented its design and execution details and shown results that indicate that the level-of-detail feature aids in haptic image exploration. This feature is inspired by the step-by-step displays of Edman [Edm92], and the positive results of our study confirm that our solution using a segmentation hierarchy is effective.

We could not show the benefit of encoding the current segment size in the textures. On the contrary, our results suggest that our texture implementation had a negative impact. The texture feature lowered the confidence of the study participants. However, based on our own experience with the system, we suspect that may change for expert users.

Users had valuable input that we can use to improve the system and future studies. As our system gains maturity, we believe it could find use in adaptive technology for the blind.

Chapter 7

Conclusion

In this thesis, we presented our haptic image exploration system that supports exploration of an image at multiple levels-of-detail. We have evaluated this system with a user study, and shown that participants were able to answer questions by exploring a segmented image. It is important to realize that in our system the exploration is based on image contours, where the user explores an image by tracing with a single point of contact the contours of the segmented objects. The results of our study show that access to multiple levels-of-detail eases the exploration task. The level-of-detail support is possible with the use of a hierarchical segmentation of the image being displayed. We argue that for exploring the hierarchy without confusion, the segmentation must be at an *object-level* throughout, each object must be well delineated, the parent-child relationship of nodes in the hierarchy must be strict, and an object must be defined at a coarser level in the hierarchy than its details.

User interaction is necessary to create an *object-level* contour image. We have designed and evaluated a new interactive segmentation method by *region selection* for the purpose of efficiently creating accurate *object-level* contour images. The results of our benchmarks show that this new method is a viable alternative to marker-based interactive segmentation algorithms. Interactive segmentation by *region selection* is done on top of an automatically generated intermediate segmentation hierarchy. The intermediate segmentation hierarchy needs to be traversable linearly, but does not need to be strict, or at an *object-level*, or contain coarse-to-fine segmentation levels. When preparing an image for haptic exploration, we proposed building the hierarchy itself in an interactive manner to ensure that the order of appearance of the objects in the hierarchy is coherent with their importance. This hierarchy is not to be confused with the intermediate one built

automatically when using interactive segmentation by *region selection*.

We present our method of interactive segmentation by *region selection* and our tool for authoring segmentation hierarchies in this thesis, but one can be used without the other. We believe both to be useful for computer vision systems that need either a segmentation or a segmentation hierarchy with specific requirements. Our need was to have an accurate *object-level* strict segmentation hierarchy, and it motivated us to build these tools. Our authoring tool will always build a strict hierarchy, but the other requirements are only enforced by the user of the tool.

Assistive technologies were our inspiration for our haptic image exploration system. Our user study was conducted with sighted participants, because it was an early proof of concept prototype, and we would prefer to have a more mature system before including blind participants. A system for the blind needs rigorous input and feedback and comprehensive navigation features. We believe our study results and the feedback from the sighted participants can be used to improve our system to be ready for blind users.

The image exploration system we have developed is not only meant to be useful for the blind. We believe different research domains might be interested in this haptic technology. For example, the study of mental images in psychology might benefit from using a system like ours.

7.1 Future Work

The cross-modal identification experiment in our user study was for the most part inconclusive. We would like to redo the experiment with a larger number of images, and different conditions within-subjects to reach stronger conclusions. The effects on cross-modal identification of the duration of exploration and access to multiple levels-of-detail could be answered this way.

We would be interested in rendering the segmentation hierarchy we prepare on different types of haptic devices. Rendering the contour image on a pin-matrix display would be possible, as well as changing the level-of-detail, zooming and panning, though haptic textures cannot be rendered on current pin-matrix devices. The learning curve of using a pin-matrix display for image exploration should be lower than using the Phantom. That is because blind user and even sighted users are already familiar with the concept of raised dots because of Braille characters. Comparing an image exploration system using a pin-matrix display with our system could help measure how larger the field of view truly is with multi-points exploration.

It would be worthwhile to revisit the use of haptic texture for image exploration. In our system, the texture on the surface of each segment conveyed information about its size. We are interested in studying the effect of encoding a different type of information instead in the texture. For example, the frequency of the colour in the original image could be encoded in the haptic texture to communicate to the user more information, even ignoring segment boundaries while doing so.

A major bottleneck in our system is the resources required to produce a segmentation hierarchy for an image. It would open up new applications if the images could somehow be prepared automatically, so that for example the user may explore the images while browsing the web. The development of a system that can prepare precise and *object-level* hierarchical segmentation automatically is a difficult task. Object recognition could help decide which objects should appear at coarse levels, and help in their accurate delineation. Object recognition is still an important challenge in computer vision. Creating a database of models of segmentation hierarchies using our tools might be a step in the development of an *object-level* automatic hierarchical segmentation, or object recognition systems.

The user interface of the research prototype of our interactive segmentation by *region selection* could be enhanced. Our interface supports linear level-of-detail access, as well as positive and negative selection of regions, inverting the selection, and a mode for pixel-level controls. It is possible to lose some work if one is not careful when changing the level-of-detail. Undo operations while forming the selection mask, merging the regions covered by the mask or changing the level-of-detail would be definite improvements to the user interface needed for commercialization of the technology.

We would like to improve the user interface of our authoring tool for building segmentation hierarchies as well. Giving the user advanced control on the tree representing the hierarchy would ease its creation. Some tree operations that we think would improve the tool are: cutting off branches, merging siblings and lifting up child nodes. Exporting the hierarchy as a tree, in addition to our current export to a merge-event timeline format would make the tool more versatile. Explicit controls over the order used when flattening the tree into the merge-event timeline would also be a useful improvement to our tool.

Appendix A

Instructions for members of the *LOD Textured* group

Take your time to read the image description and questions. Once you are ready, press ENTER to start exploring. The time you take to complete each section will be recorded. You should try to be as quick as possible while remaining accurate. You can see the time remaining for each section on the screen. A training image will be used in the first section to familiarize yourself with the system.

The SPACEBAR is used to answer the questions that require finding an object by marking its position. The section ends once this question is answered, or the time limit expires. Between each section, the device will move to the top left corner of the image automatically.

Zooming is done with the UP/DOWN ARROW KEYS. A low tone beep will notify you when zooming has reached its limits.

Panning is done by pressing the DEVICE BUTTON and dragging.

The level of detail is controlled with the LEFT/RIGHT ARROW KEYS. A low tone beep will notify you when the level of detail has reached its limits. Some details necessary to answer some questions are not visible at low level of detail.

If you follow an object contour from the exterior, it is easier to fall away from the object. If you follow an object contour from the interior, it is easier to fall inside it.

The surface of the image is textured (rough, bumpy). The parts that form the image have different texture. In general, smaller parts have a higher frequency of bumps.

When following small details, the device might get stuck in a corner. Zooming in will get you unstuck.

Be gentle with the device. A high tone beep will warn you when too much force is applied.

Hold the device when pressing ENTER so that it does not fall.

Appendix B

Glossary of Terms

1-D One Dimensional

2-D Two Dimensional

3-D Three Dimensional

N-D N Dimensional

ANOVA Analysis of Variance

API Application Programming Interface

BSDS500 Berkeley Segmentation Data Set 500

CIELAB International Commission on Illumination L*a*b* Colour Space

GPU Graphics Processing Unit

HLAPI Haptic Library API

HDAPI Haptic Device API

IIR Infinite Impulse Response

LOD Level-of-Detail

NP Nondeterministic Polynomial Time Complexity

$O(n)$ Linear Complexity

$O(1)$ Constant Complexity

OpenCV Open Source Computer Vision

OpenGL Open Graphics Library

OWT Oriented Watershed

RGB Red, Green and Blue

SLIC Simple Linear Iterative Clustering

SRM Statistical Region Merging

STReSS2 Stimulator of Tactile Receptors by Skin Stretch

UCM Ultrametric Contour Map

Bibliography

[AC08] P. Arbeláez and L. Cohen. Constrained image segmentation from hierarchical boundaries. In *Proc. CVPR*, Anchorage, AK, US, 2008.

[Ahu96] Narendra Ahuja. A transform for multiscale image segmentation by integrated edge and region detection. *IEEE T-PAMI*, 18:1211–1235, 1996.

[AMFM] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Berkeley segmentation data set 500 (BSDS500). `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html`.

[AMFM09] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. In *Proc. CVPR*, pages 2294–2301, Miami, FL, USA, 2009.

[AMFM10] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE T-PAMI*, Aug 2010. Pre-Print.

[Arb06] Pablo Arbeláez. Boundary extraction in natural images using ultrametric contour maps. In *Proc. Computer Vision and Pattern Recognition Workshop*, New York, USA, 2006.

[ASS+10] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. SLIC Superpixels. Technical report, EPFL, June 2010.

[BJ01] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *Proc. 8th ICCV*, pages 105–112, Vancouver, BC, Canada, 2001.

[BS07] Xue Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1 –8, oct. 2007.

[BT09] W. Brendel and S. Todorovic. Video object segmentation by tracking regions. In *Proc. IEEE 12th ICCV*, pages 833–840, Kyoto, Japan, 2009.

[BYZ07] Xiong Yang Benjamin Yao and Song-Chun Zhu. Introduction to a large scale general purpose ground truth dataset: methodology, annotation tool, and benchmarks. In *EMMCVPR*, 2007.

[Can86] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8:679–698, November 1986.

[CGNT09] C. Couprie, L. Grady, L. Najman, and H. Talbot. Power watersheds: A new image segmentation framework extending graph cuts, random walker and optimal spanning forest. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 731 –738, 29 2009-oct. 2 2009.

[CKB09] A. Cockburn, A. Karlson, and B.B. Bederson. A review of overview+detail, zooming, and focus+context interfaces. *ACM Comput. Surv.*, 41:2:1–2:31, January 2009.

[CM02] D. Comaniciu and P. Meer. Mean Shift: A robust approach toward feature space analysis. *IEEE T-PAMI*, 24(5), 2002.

[CMCM] Bogdan Georgescu Christopher M. Christoudias and Peter Meer. Edge detection and image segmentation (edison) system. Rutgers School of Engineering Robust Image Understanding Laboratory.

[CRB04] V. Kolmogorov C. Rother and A. Blake. GrabCut: Interactive foreground extraction using iterated graph cuts. *ACM ToG*, 23:309–314, 2004.

[Der93] Rachid Deriche. Recursively implementing the gaussian and its derivatives, 1993.

[DS02] Doug DeCarlo and Anthony Santella. Stylization and abstraction of photographs. *ACM ToG*, 21:769–776, 2002.

[DWB09] L. Dopjans, C. Wallraven, and H. H. Bülthoff. Cross-modal transfer in visual and haptic face recognition. *IEEE Transactions on Haptics*, 2(4), 2009.

[Edm92] P. Edman. *Tactile Graphics*. American Foundation for the Blind Press, May 1992.

[FB99] J.P. Fritz and K.E. Barner. Design of a haptic data visualization system for people with visual impairments. *IEEE Transactions on Rehabilitation Engineering*, 7:372–384, September 1999.

[FH04] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. of Computer Vision*, 59(2):167–181, 2004.

[FMR+02] J. Freixenet, X. Muoz, D. Raba, J. Mart, and X. Cuf. Yet another survey on image segmentation: Region and boundary information integration. In *In ECCV*, pages 408–422, 2002.

[FZ98] G.W. Furnas and X. Zhang. MuSE: a multiscale editor. In *Proc. 11th ACM UIST*, pages 107–116, 1998.

[GCH05] Qi Wang Gianni Campion and Vincent Hayward. The pantograph mk-ii: A haptic instrument. In *Proc. 2005 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pages 723–728, 2005.

[GH97] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 209–216, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[GLL11] V. Guruswamy, J. Lang, and W.-S. Lee. Iir filter models of haptic vibration textures. *IEEE T. on Instrumentation and Measurement*, 60(1), 2011.

[Gra06] L. Grady. Random walks for image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(11):1768 –1783, nov. 2006.

[GZ05] Michael Garland and Yuan Zhou. Quadric-based simplification in any dimension. *ACM Trans. Graph.*, 24:209–239, April 2005.

[Hec82] Paul S. Heckbert. Color image quantization for frame buffer display. *Computer Graphics*, 16:297–307, 1982.

[HMM00] I. Herman, G. Melancon, and M.S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE TVCG*, 6(1):24–43, 2000.

[HS85] R M Haralick and L G Shapiro. Image segmentation techniques. *CVGIP*, 29:100–132, 1985.

[Hyp10] HyperBraille Consortium. Hyperbraille. `http://www.hyperbraille.de/?lang=en`, May 2010.

[ICG+04] R. Iglesias, S. Casado, T. Gutierrez, J.I. Barbero, C.A. Avizzano, S. Marcheschi, and M. Bergamasco. Computer graphics access for blind people through a haptic and audio virtual environment. In *The 3rd IEEE International Workshop on Haptic, Audio and Visual Environments and Their Applications*, pages 13–18, October 2004.

[IVE] IVEO. Iveo software/touchpad. `http://www.viewplus.com`.

[KBD+11] M. Kagaya, W. Brendel, Q. Deng, T. Kesterson, S. Todorovic, P.J. Neill, and E. Zhang. Video painting with space-time-varying style parameters. *IEEE TVCG*, 17(1):74–87, 2011.

[KMT+07] Konstantinos Kostopoulos, Konstantinos Moustakas, Dimitrios Tzovaras, Giorgos Nikolakis, Cline Thillou, and Bernard Gosselin. Haptic access to conventional 2d maps for the visually impaired. *Journal on Multimodal User Interfaces*, 1(2):13–19, October 2007.

[KS10] Michael Kass and Justin Solomon. Smoothed local histogram filters. In *ACM SIGGRAPH 2010 papers*, SIGGRAPH '10, pages 100:1–100:10, New York, NY, USA, 2010. ACM.

[KWP02] Rick Komerska, Colin Ware, and Matthew Plumlee. Haptic interface for center-of-workspace interaction. In *Proceedings of the 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, HAPTICS '02, pages 352–, Washington, DC, USA, 2002. IEEE Computer Society.

[KWT88] Michael Kass, Andrew P. Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331, 1988.

[LHM09] Y.-K. Lai, S.-M. Hu, and R.R. Martin. Automatic and topology-preserving gradient mesh generation for image vectorization. *ACM Trans. Graph.*, 28:85:1–85:8, July 2009.

[LL06] G. Lecot and B. Levy. ARDECO: Automatic region detection and conversion. In *Proc. Eurographics Symp. on Rendering*, pages 349–360, 2006.

[LZL+10] L. Lin, K. Zeng, H. Lv, Y. Wang, Y. Xu, and S.-C. Zhu. Painterly animation using video semantics and feature correspondence. In *Proc. 8th Symp. on NPAR*, pages 73–80, 2010.

[MAFM08] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1 –8, june 2008.

[Mey92] F. Meyer. Color image segmentation. In *Proc. ICIP*, pages 303–306, 1992.

[ML08] D. MacDonald and J. Lang. Bitmap to vector conversion for multi-level analysis and visualization. In *SVG Open*, August 2008.

[NCNC08] J. F. Norman, A. M. Clayton, H. F. Norman, and C. E. Crabtree. Learning to perceive differnces in solid shape through vision and touch. *Perception*, 37:185–196, 2008.

[NGC+08] Alexandre Noma, Ana Beatriz V. Graciano, Luís Augusto Consularo, Roberto M. Cesar, and Isabelle Bloch. A new algorithm for interactive structural image segmentation. *CoRR*, abs/0805.1854, 2008.

[NN04] R. Nock and F. Nielsen. Statistical region merging. *IEEE T-PAMI*, 26(11):1452–1458, 2004.

[ODH98] Allison M. Okamura, Jack T. Dennerlein, and Robert D. Howe. Vibration feedback models for virtual environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 674–679, 1998.

[Ope] OpenCV. Open source computer vision. `http://opencv.willowgarage.com/wiki/`.

[PDL+08] Grégory Petit, Aude Dufresne, Vincent Levesque, Vincent Hayward, and Nicole Trudeau. Refreshable tactile graphics applied to schoolbook illustrations for students with visual impairment. In *Proc. 10th Int. ACM SIGACCESS Conf. on Computers and Accessibility*, Assets '08, pages 89–96, New York, NY, USA, 2008. ACM.

[PR97] D. K. Pai and L.-M. Reissell. Haptic interaction with multiresolution image curves. *Computers and Graphics*, 21(4):405–411, 1997.

[PWL10] Devi Archana Paladugu, Zheshen Wang, and Baoxin Li. On presenting audio-tactile maps to visually impaired users for getting directions. In *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems*, CHI EA '10, pages 3955–3960, New York, NY, USA, 2010. ACM.

[Ros98] Paul L. Rosin. Refining region estimates. *International Journal of Pattern Recognition and Artificial Intelligence*, 12:841–866, 1998.

[RTE08] M. Rotard, C. Taras, and T. Ertl. Tactile web browsing for blind people. *Multimedia Tools and Application*, 37(1):53–69, March 2008.

[RYB+00] Rameshsharma Ramloll, Wai Yu, Stephen Brewster, Beate Riedel, Mike Burton, and Gisela Dimigen. Constructing sonified haptic line graphs for the blind student: first steps. In *Proc. 4th Int. ACM SIGACCESS Conf. on Computers and Accessibility*, Assets '00, pages 17–25, New York, NY, USA, 2000. ACM.

[Sjo01] Calle Sjostrom. Designing haptic computer interfaces for blind people. In *Sixth International Symposium on Signal Processing and its Applications*, volume 1, pages 68–71, 2001.

[SLWS07] Jian Sun, Lin Liang, Fang Wen, and Heung-Yeung Shum. Image vectorization using optimized gradient meshes. *ACM Trans. Graph.*, 26, July 2007.

[SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 2000.

[Tec] SensAble Technologies. Phantom 1.0. `http://www.sensable.com/`.

[Tru] N. Trudeau. Analyse visuelle du livre, de la page et de l'image du manuel de l'élève enjeux et découvertes. Final report of quinquinal research project, Montréal, 2000-2005.

[UPT+08] M. Unger, T. Pock, W. Trobin, D. Cremers, and H. Bischof. Tvseg - interactive total variation based image segmentation. In *British Machine Vision Conference*, Leeds, UK, 2008.

[WL93] Zhenyu Wu and Richard M. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:1101–1113, 1993.

[WPW+06] W. Wang, I. Pollak, T.-S. Wong, C. A. Bouman, M. P. Harper, and J. M. Siskind. Hierarchical stochastic image grammars for classification and segmentation. *IEEE T-Image Processing*, 15(10):3033–3052, 2006.

[XLY09] Tian Xia, Binbin Liao, and Yizhou Yu. Patch-based image vectorization with automatic curvilinear feature alignment. *ACM Trans. Graph.*, 28:115:1–115:10, December 2009.

[ZM06] S.-C. Zhu and D. Mumford. A stochastic grammar of images. *Found. Trends. Comput. Graph. Vis.*, 2:259–362, 2006.

[ZND+11] Yibiao Zhao, Xiaohan Nie, Yanbiao Duan, Yaping Huang, and Siwei Luo. A benchmark for interactive image segmentation algorithms. In *Person-Oriented Vision (POV), 2011 IEEE Workshop on*, pages 33 –38, jan. 2011.

[ZS95] C. B. Zilles and J. K. Salisbury. A constraint-based god-object method for haptic display. In *Proceedings of the International Conference on Intelligent Robots and Systems-Volume 3 - Volume 3*, pages 3146–, Washington, DC, USA, 1995. IEEE Computer Society.

[ZZXZ09] Kun Zeng, Mingtian Zhao, Caiming Xiong, and Song-Chun Zhu. From image parsing to painterly rendering. *ACM ToG*, 29:2:1–2:11, 2009.