

PH.D THESIS

Image Segmentation, Evaluation, and Applications

by

Kevin McGuinness, B.Sc. (Hons)

School of Electronic Engineering

Supervisor:

Prof. Noel E. O'Connor

November 6, 2009



Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ph.D in Electronic Engineering is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: _____ ID No.: _____
Kevin McGuinness

Date: _____

Acknowledgements

I wish to express my deepest thanks to everyone that helped and supported me throughout my Ph.D and when compiling this thesis. In particular, I want to thank my supervisor, Noel O'Connor, for his continual reassurance and sound advice; Alan Smeaton for taking me on as an intern back in 2005, and giving me the opportunity to pursue postgraduate studies; Tomasz Adamek for inspiring my original interest in image processing; and everyone at the Center for Digital Video processing: your friendship and advice have been invaluable.

I also wish to express my gratitude to the European Commission for funding my research under contract FP6-027026 (K-Space), and Science Foundation Ireland under grant 07/CE/I1147 (CLARITY). Many thanks, also, to the Disney corporation for giving me the opportunity to intern with them in 2008.

On a personal note, I want to thank my family for their constant support and encouragement: my parents Gertrude and Tony, my brother Michael, and my sister Martina. Finally, I would like to especially thank my girlfriend Agnieszka, for her unending support, encouragement, and patience.

Abstract

This thesis aims to advance research in image segmentation by developing robust techniques for evaluating image segmentation algorithms. The key contributions of this work are as follows. First, we investigate the characteristics of existing measures for supervised evaluation of automatic image segmentation algorithms. We show which of these measures is most effective at distinguishing perceptually accurate image segmentation from inaccurate segmentation. We then apply these measures to evaluating four state-of-the-art automatic image segmentation algorithms, and establish which best emulates human perceptual grouping. Second, we develop a complete framework for evaluating interactive segmentation algorithms by means of user experiments. Our system comprises evaluation measures, ground truth data, and implementation software. We validate our proposed measures by showing their correlation with perceived accuracy. We then use our framework to evaluate four popular interactive segmentation algorithms, and demonstrate their performance. Finally, acknowledging that user experiments are sometimes prohibitive in practice, we propose a method of evaluating interactive segmentation by algorithmically simulating the user interactions. We explore four strategies for this simulation, and demonstrate that the best of these produces results very similar to those from the user experiments.

List of Publications

- K. McGuinness and N. E. O'Connor. "A comparative evaluation of interactive segmentation algorithms," Pattern Recognition, March 2009.
- I. Gonzalez-Diaz, K. McGuinness, T. Adamek, N. E. O'Connor, and F. Diaz-de-Maria. "Incorporating spatio-temporal mid-level features in a region segmentation algorithm for video sequences," in IEEE International Conference on Image Processing, 2008.
- K. McGuinness, N. E. O'Connor. "The K-Space segmentation tool set," in Semantics And Digital Media Technology, 2008.
- P. Wilkins, T. Adamek, D. Byrne, G. Jones, H. Lee, G. Keenan, K. McGuinness, N. E. O'Connor, A. F. Smeaton, A. Amin, Z. Obrenovic, R. Benmokhtar, E. Galmar, B. Huet, S. Essid, R. Landais, F. Vallet, G. Papadopoulos, S. Vrochidis, V. Mezaris, I. Kompatsiaris, E. Spyrou, Y. Avrithis, R. Morzinger, P. Schallauer, W. Bailer, T. Piatrik, K. Chandramouli, E. Izquierdo, M. Haller, L. Goldmann, A. Samour, A. Cobet, T. Sikora, and P. Praks. "K-Space at TRECVid 2007," In Text Retrieval Conference TRECVID Workshop, November 2007.
- K. McGuinness, O. Gillet, N. E. O'Connor, and G. Richard. "Visual analysis for drum sequence transcription," in Proceedings of the 15th European Signal Processing Conference, September 2007.
- K. McGuinness, G. Keenan, T. Adamek, and N. E. O'Connor. "Image segmentation evaluation using an integrated framework," in Proceedings of the IET 4th International Conference on Visual Information Engineering, July 2007.

- K. McGuinness, G. Keenan, T. Adamek, and N. E. O'Connor. "Framework for integrating and evaluating automatic region-based segmentation algorithms," in *Semantics And Digital Media Technology*, December 2006.
- P. Wilkins, T. Adamek, P. Ferguson, M. Hughes, G. Jones, G. Keenan, K. McGuinness, J. Malobabic, N. O'Connor, D. Sadlier, A. F. Smeaton, R. Benmokhtar, E. Dumont, B. Huet, B. Merialdo, E. Spyrou, G. Koumoulos, Y. Avrithis, R. Moerzinger, P. Schallauer, W. Bailer, Q. Zhang, T. Piatrik, K. Chandramouli, E. Izquierdo, L. Goldmann, M. Haller, T. Sikora, P. Praks, J. Urban, X. Hilaire, and J. Jose. "K-Space at TRECVID 2006," in *Text Retrieval Conference TRECVID Workshop*, November 2006

Abbreviations and Acronyms

API	Application programming interface
BCa	Bootstrap bias corrected accelerated interval
BGM	The bipartite graph matching index
BPT	The binary partition tree segmentation algorithm
CIE	International Commission on Illumination
CIELAB	A CIE color-opponent space designed to be perceptually uniform
CIELUV	A CIE color space designed to be perceptually uniform
CIE76	The Euclidean distance in CIELAB space
CIEDE2000	An updated color difference measure for the CIELAB space
FMI	The Fowlkes-Mallows index
FN	The number of false negatives
FP	The number of false positives
GCE	Global consistency error
GIF	The graphics interchange image format
GIMP	The GNU image manipulation program
HD	The Huang-Dom performance measure
HDI	The Huang-Dom index (error measure)
HTML	Hypertext markup language
IGC	The interactive graph cuts algorithm

JAI	The Java advanced imaging library
JI	The Jaccard index
JPEG	The joint photographic experts group image format
LCE	Local consistency error
LHE	Proposed measure combining LCE and HDI
MI	Mutual Information
MPEG	Motion picture experts group
MRSST	The modified recursive shortest spanning tree algorithm
MSHIFT	The mean-shift segmentation algorithm
NCUTS	The normalized cuts segmentation algorithm
NMI	The normalized mutual information index
PNG	The portable network graphics image format
PNM	The portable anymap image formats
RAG	Region adjacency graph
RGB	The red/green/blue color space.
RI	The rand index
RSST	The recursive shortest spanning tree segmentation algorithm
SIOX	The simple interactive object extraction algorithm
SRAG	The spatiotemporal region adjacency graph segmentation algorithm
SRG	The seeded region growing algorithm

SRM	The statistical region merging algorithm
TN	The number of true negatives
TP	The number of true positives
VI	The variation of information index
WMV	The windows media video format
<i>c.i.</i>	Confidence interval
<i>sd</i>	The biased corrected standard deviation
\mathcal{A}_B	Binary boundary accuracy measure
$\tilde{\mathcal{A}}_B$	Fuzzy boundary accuracy measure
\mathcal{A}_O	Object accuracy measure
$\mathcal{P}r$	Precision
$\mathcal{R}e$	Recall

Contents

1	Introduction	1
1.1	Overview	1
1.2	Motivation	5
1.3	Objectives	8
1.4	Structure of Thesis	8
2	Image Segmentation: A Review	10
2.1	Introduction	10
2.2	Taxonomy	11
2.2.1	Application-Centric Classification	12
2.2.2	Algorithm-Centric Classification	17
2.3	Grouping Cues	19
2.3.1	Gestalt Laws	20
2.3.2	Low-level Features	23
2.4	Automatic Segmentation	30
2.4.1	Region Adjacency Graphs	30
2.4.2	Statistical Region Merging	37
2.4.3	Normalized Cuts for Image Segmentation	41
2.4.4	Mean-shift Analysis for Image Segmentation	47
2.4.5	Other Approaches	54
2.5	Interactive Segmentation	56

2.5.1	Seeded Region Growing	58
2.5.2	Interactive Graph Cuts	62
2.5.3	Interactive Segmentation using Binary Partition Trees	66
2.5.4	Simple Interactive Object Extraction	68
2.5.5	Other Approaches	71
2.6	Discussion	73
3	Segmentation Evaluation: A Review	75
3.1	Introduction	75
3.2	Taxonomy	76
3.3	Supervised Evaluation	81
3.3.1	Evaluation Metrics Based on Clustering	82
3.3.2	Local and Global Consistency Error	88
3.3.3	The Huang-Dom Evaluation Measure	93
3.3.4	Other Approaches	95
3.4	Unsupervised Evaluation	97
3.4.1	Entropy Based Evaluation	99
3.4.2	Visible Color Distance Based Evaluation	104
3.4.3	Other Approaches	105
3.5	Discussion	106
4	Evaluating Automatic Segmentation	109
4.1	Introduction	109
4.2	Algorithms	110
4.3	Measures	112
4.4	Dataset	113
4.5	Software	114
4.5.1	Features	115
4.5.2	Architecture	117

4.6	Method	120
4.7	Experiments	123
4.7.1	Experiment 1: Examining the Evaluation Measures	123
4.7.2	Experiment 2: Evaluating the Algorithms	135
4.8	Discussion	144
5	Evaluating Interactive Segmentation	147
5.1	Introduction	147
5.2	Algorithms	149
5.3	Evaluation	151
5.3.1	Human Factors	151
5.3.2	Evaluation Measures	152
5.3.3	Boundary Accuracy	153
5.3.4	Object Accuracy	156
5.3.5	Choosing Sigma	157
5.3.6	Other Measures	158
5.4	Experiment	158
5.4.1	Software	159
5.4.2	Ground Truth	161
5.4.3	Setup	162
5.4.4	Deployment	165
5.5	Results and Analysis	166
5.5.1	Object and Border Accuracy	166
5.5.2	Perceived Accuracy	174
5.5.3	User Feedback	174
5.5.4	Validation	178
5.6	Conclusion	179

6 Automating Interactive Segmentation Evaluation	181
6.1 Introduction	181
6.2 Automation	182
6.2.1 Strategy 1	187
6.2.2 Strategy 2	191
6.2.3 Strategy 3	192
6.2.4 Strategy 4	196
6.3 Evaluation	197
6.4 Analysis	199
6.4.1 Experiments	202
6.4.2 Correlation and Validation	206
6.5 Conclusion	211
7 Conclusion	214
A Evaluation Dataset	219
Bibliography	229
List of Figures	246
List of Tables	249

Chapter 1

Introduction

This objective of this chapter is to provide a general introduction to the thesis subject, our motivations for this research, and the overall research objectives. The next section introduces image segmentation, discusses the various definitions of image segmentation found in the literature, and outlines some applications of segmentation. We also introduce the topic of image segmentation evaluation, and describe why such evaluation is becoming increasingly important. We subsequently discuss the specific motivations behind our work. We then describe the objectives of the research, and finally, outline the thesis structure.

1.1 Overview

Image segmentation is critical for many computer vision and information retrieval systems, and has received significant attention from industry and academia over the last 30 years. Despite notable advances in the area, there is no standard technique for selecting a segmentation algorithm to use in a particular application, nor even is there an agreed upon means of comparing the performance of one method with another. This deficiency is likely a result of the inherent ambiguity in what is understood as the purpose and scope of segmentation itself.

Like many complex computer vision problems, image segmentation is ill-defined. A common, if rather unconstrained, definition of segmentation is that it is the process of partitioning the set of pixels in an image into several disjoint subsets, according to a set of predefined criteria. Although this definition admits and conforms to almost all other definitions found in the literature, the criteria itself is usually a source of debate.

Cheng et al. [Cheng et al., 2001] define image segmentation as the process of dividing an image into different regions such that each region is, but the union of any two adjacent regions is not, homogeneous. Similarly, Morris et al. [Morris et al., 1986] describes segmentation as the process of partitioning an image into regions that are in some sense homogeneous, but different from neighboring regions. Skarbek and Koschan [Skarbek and Koschan, 1994] opt for a simpler interpretation: the identification of homogeneous regions. All these definitions use the concept of homogeneity, which usually corresponds to identifying regions containing features that are relatively nearby according to a prescribed distance measure.

Segmentation may also be considered as an algorithmic attempt to mimic a human interpretation of an image, known as perceptual grouping. Considering segmentation in this way substantially increases the scope and complexity of the problem. Fu and Mui [Fu and Mui, 1981] assume this viewpoint, stating that “the image segmentation problem is basically one of psychophysical perception, and therefore not susceptible to a purely analytical solution.” Martin et al. [Martin et al., 2001] also imply this interpretation in their work on comparing automatic segmentation algorithms with human generated ground truth. Martin et al. [Martin et al., 2001], and Salembier and Garrido [Salembier and Garrido, 2000] both argue that perceptual grouping is hierarchical in nature, and consequentially a flat partitioning of an image is insufficient for representing a perceptual segmentation.

Other authors admit both viewpoints are valid. Adamek et al. state that segmentation is the process of partitioning an image into a set of semantic entities *or* homogeneous regions [Adamek et al., 2005]. Shi and Malik concede that the final goal is perceptual; however, they imply that the scope of segmentation should be limited to robust homogeneity-based hierarchical grouping of low-level features, and that high-level reasoning can be used at a later stage to identify the semantic entities [Shi and Malik, 2000]. Nock and Nielsen [Nock and Nielsen, 2004] suggest a more physical definition; if an image is thought of as an observed phenomena induced by physical objects and lighting conditions, then segmentation is a process of attempting to infer some knowledge about the model that caused the observed patterns, i.e., retrieving a model of the image generation process.

It is clear from the above that there is considerable variation in what is understood to be the scope and definition of the image segmentation problem. Image segmentation is usually one of several components in a larger information processing system, and the variation observed in the definition of image segmentation is mirrored in the variation in requirements on the image segmentation algorithms in these systems. For multimedia information retrieval systems, image segmentation algorithms capable of producing homogeneous regions usually suffice, since the purpose of image segmentation in such systems is often simply to create a set of localized features. Object recognition systems, on the other hand, usually require semantic objects from which features can be extracted and processed by a pattern recognition engine (a support vector machine, for example). In some cases, a priori information about the object is available, or can be fed back into the segmentation algorithm; in other cases, no such information is available, and the segmentation algorithm is required to produce regions or objects based on the image data alone.

When developing a system in which image segmentation is an integral component, the choice of algorithm used can directly affect the performance of the system as a whole. For example, if the segmentation phase in a object recognition system produces a very inaccurate object boundary, then the shape features for this object may change dramatically, ultimately causing the recognition to fail. If the object selection algorithm in a photo-editing application consistently fails to locate a satisfactory object boundary, the user becomes frustrated. If, given two similar images, the segmentation algorithm used by a content based retrieval system produces very different segmentations, then the features extracted for the regions in these segmentations will likely be very dissimilar, and the performance of the retrieval engine will suffer. Choosing the appropriate segmentation algorithms for a particular system is therefore critical.

Arguably the most effective way to select the most appropriate segmentation algorithm for a particular application is to implement each available segmentation algorithm, integrate each into the system in turn, test the system to measure its performance, and select the algorithm that gives the best overall performance. Given the number of segmentation algorithms in the literature, and their complexity, such a study is usually infeasible. Furthermore, repeatedly testing some systems, particularly interactive systems, requires many time consuming user experiments. In practice, only a small subset of the available algorithms can be tested when developing such systems. It is necessary, therefore, to select a subset of algorithms based on known properties and performance characteristics that can be measured in advance, and that are independent of the target application.

Segmentation evaluation is concerned with measuring and comparing, in a reasonably generic way, the performance and characteristics of segmentation algorithms. The objective is to measure and compare attributes of segmentation algorithms that are likely to be pertinent to a wide range of applications. If a segmentation evaluation technique can demonstrate that one segmentation

algorithm is significantly better than some others, for a given set of assumptions, then a systems designer can reasonably exclude the other algorithms from further consideration, provided the assumptions hold for their particular application. If the evaluation is sufficiently generic, then evaluating existing segmentation algorithms can potentially make designing a whole range of systems significantly easier. Research efforts in segmentation can then focus on optimizing, modifying, or generalizing techniques shown to be effective for a wide range of applications. New algorithms can be justified by comparing them against the existing state-of-the-art using structured and well understood evaluation techniques.

1.2 Motivation

Segmentation is an important component in many systems. In multimedia analysis systems, image segmentation can be used to partition images into regions that are in some sense homogeneous, or have some semantic significance. This provides subsequent processing stages with high-level information about scene structure. From regions we can derive geometric features, shape features, texture features, and contextual features (like spatial arrangement); individual pixels admit far less semantic information. In pattern classification systems, image segmentation can be used to delineate objects and provide features to the classifier. In photo-editing applications, segmentation can be used to isolate semantic objects so that they can be independently manipulated. From a computer vision standpoint, the ability to reliably obtain a good segmentation implies a useful model of human perceptual grouping. The diverse requirements of systems that use segmentation have led to the development of segmentation algorithms that vary widely in both algorithmic approach, and the quality and nature of the segmentation produced. Some applications simply require the image to be divided

into coarse homogeneous regions, others require rich semantic objects. For some applications precision is paramount, for others speed and automation.

TRECVID is an annual event, sponsored by the National Institute of Standards and Technology (NIST), that encourages researchers to develop innovative video information retrieval engines through coordinated evaluation and comparison procedures. Our group frequently participates, and our retrieval engines often use segmentation as a core step in the feature extraction process. From our membership in the K-Space Network of Excellence in 2006 to 2008, we had available to us implementations of various state-of-the-art segmentation algorithms, contributed by several K-Space partners. We therefore needed to select, in some way, one of these algorithms to use for our TRECVID system. Our research into methods for evaluating and comparing segmentation algorithms revealed that several, relatively recent, methods for evaluating automatic region segmentation algorithms had indeed been proposed. However, it was clear that the research was still in its incipency, that the values produced by the proposed measures were not yet widely understood, and that relatively few segmentation algorithms had been actually evaluated using these measures.

At the same time, other members of the K-Space network were interested in performing semantic reasoning on multimedia objects, and required ways to delineate and annotate these objects. The research required annotating meaningful objects; automatic segmentation is, in general, incapable of locating such semantic entities without high-level guidance. Interactive segmentation provides a solution by invoking the aid of a human operator to provide this guidance. As with automatic segmentation, there are also several interactive algorithms to choose from. Again we were faced with the issue of how to select the most appropriate algorithm for our application; we required a method to evaluate and compare the available algorithms. Unfortunately, we discovered that no research had yet been done into how to evaluate interactive segmentation algorithms. The

only literature on the subject that we located ([Mao et al., 1999]) was targeted specifically at medical image segmentation. Furthermore, the algorithm that was evaluated in this paper was not interactive in the sense that it accepted input and provided feedback, but more semi-automatic, in the sense that it accepted a single seed point and produced a final segmentation. In interactive systems the user iteratively refines the segmentation, and therefore the relationship between effort and accuracy is important.

Evaluation is not yet standard practice when proposing segmentation algorithms. Authors typically give anecdotal evidence of a segmentation algorithm's performance, in the form of a few sample images and the corresponding segmentations. It is difficult to compare segmentation algorithms based solely on sample images. The algorithms themselves are often complex and computationally demanding; implementing them efficiently and correctly requires significant effort. Because of this, systems designers often choose the segmentation algorithm that is closest to hand, or easiest to implement, instead of the one that is most suitable. In some instances, as with interactive segmentation, standard evaluation techniques are unavailable. In others, as with automatic segmentation, the evaluation techniques are relatively new and not well understood. Researchers are, therefore, often reluctant to apply them to their proposed algorithms.

There have been several significant advances in image segmentation, but it is far from a solved problem. We believe that developing methods for evaluating image segmentation is key to advancing the state-of-the-art. Effective segmentation evaluation techniques benefit application designers, who need to select the best algorithm for their system. They benefit researchers, who can use them to justify new algorithms.

1.3 Objectives

The objectives of this thesis are as follows. First, to produce a comprehensive and up-to-date review of the state-of-the-art in image segmentation and segmentation evaluation techniques. Second, to experimentally examine, in detail, the properties of existing segmentation evaluation techniques and determine which are most effective at judging segmentation accuracy. Third, to apply the most effective of these techniques to evaluate the performance of several state-of-the-art segmentation algorithms, and compare it with human perceptual grouping. In doing so we should like to establish (1) if any of these algorithms can produce segmentations that have a higher accuracy on average than random, (2) if any of these algorithms can produce segmentations that are as accurate on average as those produced by humans, and (3) which of these algorithms is the most effective at emulating human perceptual grouping. Fourth, to develop measures, tools, and techniques for evaluating interactive segmentation, with a particular focus on segmentation of natural scenes. Finally, to apply these new techniques to evaluating existing interactive segmentation algorithms, and in doing so establish which of these algorithms is most effective.

1.4 Structure of Thesis

The remainder of this thesis is organized as follows. Chapter 2 reviews and discusses the state-of-the-art in image segmentation. We begin by classifying segmentation algorithms according to their applicability and algorithmic properties, and follow with a discussion of the psychological principles underlying perceptual grouping. We then discuss how these principles are usually applied in practice: through the use of low-level image features. We conclude the chapter by reviewing eight well-known segmentation algorithms that we believe to be repre-

sentative of the state-of-the-art: four automatic algorithms, and four interactive algorithms.

Chapter 3 reviews the literature in segmentation evaluation. Again, we begin the discussion by classifying the methods for segmentation evaluation, and follow with a review of specific evaluation techniques. We review several measures from the two major classes of segmentation evaluation techniques: supervised evaluation techniques, which compare a machine segmentation against a predefined ground truth, and unsupervised evaluation techniques, which endeavor to measure the general quality of a segmentation independent of a particular ground truth.

Chapter 4 focuses on evaluating four automatic segmentation algorithms using existing segmentation evaluation measures. Our objective is to examine the properties of these evaluation measures, to establish which are the most effective for evaluating an image segmentation algorithm's ability to emulate human perceptual grouping, and then to use these measures to compare the performance and characteristics of the selected algorithms.

Chapter 5 investigates how interactive segmentation algorithms can be evaluated. In particular, we outline how evaluating interactive segmentation is different from evaluating automatic segmentation, and develop a set of measures and a methodology for evaluating interactive segmentation using user experiments.

Since user experiments can sometimes be prohibitive in practice. Chapter 6 considers some strategies for automating the process, and investigates their effectiveness. Finally, Chapter 7 presents our conclusions and outlines some avenues for future research.

Chapter 2

Image Segmentation: A Review

2.1 Introduction

The purpose of this chapter is to review and discuss the state-of-the-art in image segmentation technology. The research in the area is vast: a profusion of papers, studies, and reports have been published investigating new methods for image segmentation and their applications. The nature of the problem has inspired diverse algorithms, drawing from fields of research that include statistics, machine learning, graph theory, and psychology. The applications of image segmentation are as varied as the algorithms; such applications include: bio-medical image analysis, multimedia information retrieval, image understanding, and machine vision.

Our first task is to comprehend this extensive body of work. In the next section we take our first step toward this; we investigate and characterize the different approaches to segmentation. The objective is to develop a useful means to classify segmentation algorithms, and limit the scope of our investigation if necessary. Following this, we investigate grouping cues: the criteria segmentation algorithms use to determine which pixels belong in each region. This necessitates a discussion on human perceptual grouping, upon which many of these grouping

cues depend. We then individually introduce region segmentation and interactive segmentation and discuss specific algorithms in detail. We conclude the chapter with a discussion on the limitations of the state-of-the-art, and how we believe they can be addressed by the research reported in this thesis.

2.2 Taxonomy

Image segmentation algorithms can be classified in various different ways, and several useful taxonomies already exist. Some of these taxonomies are general, others are tailored for a specific application domain (for example, the taxonomy in [Pham et al., 2000] focuses on medical image segmentation). It is difficult to produce a single taxonomy that encompasses all aspects of image segmentation; there exists considerable variety and crossover in the proposed techniques.

We do not try to devise a single taxonomy. Instead, we consider two separate classification viewpoints, and produce two independent classification strategies. The first strategy is to classify each algorithm from the perspective of how it can be used. We call this the application-centric classification. The idea here is to produce a classification scheme that focuses solely on an algorithm’s applicability, as opposed to its method: what an algorithm does, not how it does it. The second strategy is from the opposite viewpoint: it focuses on the underlying method of the algorithm itself.

Together these viewpoints give a clear indication of a segmentation algorithm’s applicability and its algorithmic properties. We shall use both of these viewpoints to classify each of the segmentation algorithms that we investigate later in this chapter. Of course, other facets such as algorithm complexity and adaptability should also be considered when developing, selecting, or evaluating segmentation methods. Note that several other authors have proposed alternate classification

schemes, for instance: [Adamek and O'Connor, 2006], [Lucchese and Mitra, 2001], and [Adams and Bischof, 1994].

2.2.1 Application-Centric Classification

Image segmentation algorithms are often designed with specific application domains in mind. The target application domain limits the scope of the algorithm's applicability. Of course, this does not mean that an approach designed for a particular application domain is useful only within that domain. In general, however, an application domain does imply certain requirements that may be incompatible with other domains. The objective behind our application-centric classification scheme is to examine the aspects of a segmentation algorithm that most substantially effect its applicability.

Consider an image segmentation algorithm designed for a photo-editing application. The purpose of such an algorithm is to allow a user to easily select an object in a photograph so it can be independently changed: copied, moved, removed, touched-up, etc. The application implies a set of requirements. Suitable algorithms are usually interactive, and partition the image into two non-overlapping regions: the region of interest, and everything else. A fully automatic image segmentation algorithm, designed to partition an image into regions, might be ideal for a multimedia information retrieval engine, in which automation is essential and errors are tolerated. However, such an algorithm may be useless in a photo-editing application. Similarly, an interactive segmentation algorithm designed for a photo-editing application is unlikely to be suitable for a multimedia retrieval application. This does not mean that an image segmentation method designed for one application, say, photo-editing, can only be used for photo-editing; it might be suitable many more tasks. For example, such an algorithm might be ideal for

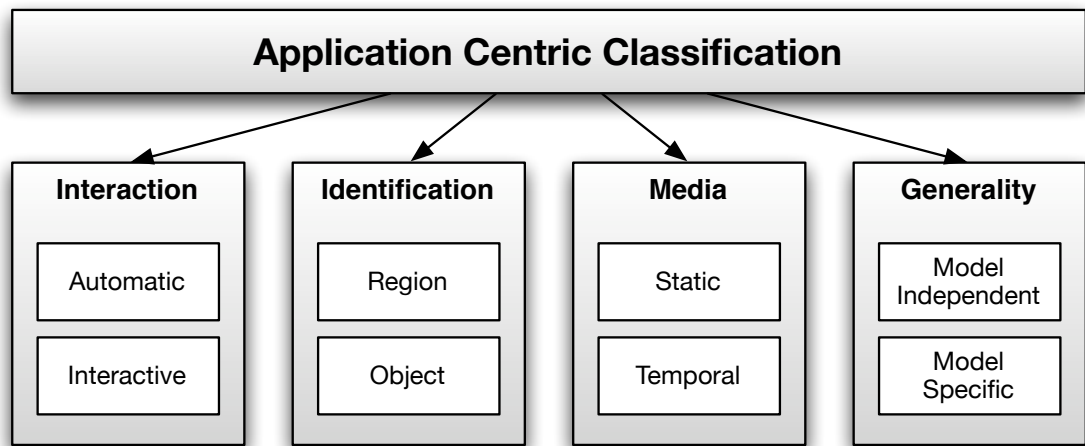


Figure 2.1: Application-centric classification of segmentation algorithms

generating HTML image maps for rich web content (the software described in Section 5.4.1 is capable of generating such content).

From the above discussion, we can already identify two aspects that affect the applicability of a segmentation algorithm: (1) whether the algorithm requires interaction or is automatic, and (2) whether it identifies regions or objects. Further consideration reveals four specific facets that significantly influence the domain in which a segmentation algorithm can be used. These are illustrated in Figure 2.1, and addressed in the following sections.

Interaction

The first facet is the amount and granularity of *interaction* required to segment an image. Using the level of user interaction as a criteria, segmentation algorithms can be roughly divided into two categories: automatic and interactive. Automatic segmentation algorithms require no user interaction to compute a segmentation (although many require a set of initial parameters to be selected). Automatic algorithms are especially useful when the quantity of data prohibits interaction. Multimedia information retrieval is a typical application, often using automatic

segmentation to partition images into regions, thereby enabling local feature extraction. Examples of multimedia IR engines that use automatic image segmentation include Blobworld [Carson et al., 2002] and the 2007 K-Space TRECVid engine [Wilkins et al., 2007].

Automatic segmentation algorithms are an effective solution for applications that require quick, coarse, region-based segmentation. Other applications require more accurate semantic objects. When such objects are necessary fully-automatic segmentation is typically impossible; some high-level information is needed to traverse the so called “semantic-gap” between homogeneous regions and perceived objects.

Interactive segmentation algorithms¹ provide a solution by invoking the aid of a human operator. This operator supplies the high-level information needed to detect and extract semantic objects through a series of interactions. Typically, operators mark areas of the image as object or background, and the algorithm updates the segmentation using the new information. By iteratively providing more interactions, the user can refine the segmentation. The goal of interactive segmentation is thus to provide a means of extracting semantic objects from an image quickly and accurately.

Identification

Segmentation techniques also differ in the type, form, and quantity of objects that they identify. We can roughly divide image segmentation algorithms into two categories based on what they identify: object-based, and region-based. Object-based algorithms partition the image into two distinct parts: object (foreground) pixels and non-object (background) pixels. Region-based algorithms partition the image into an arbitrary number of parts (regions). Unlike object-based algorithms,

¹Also referred to herein as semi-supervised or semi-automatic segmentation algorithms

no semantic significance is assigned to these regions. The aim is typically to create regions that are, in some way, coherent, or homogeneous.

Most thresholding techniques [Sezgin and Sankur, 2004] can be considered simple object-based segmentation algorithms. Many interactive segmentation algorithms are also object-based (see Section 2.5). The majority of automatic segmentation algorithms, especially those designed for use on natural scenes, are region-based: it is very difficult to detect objects without some high-level information about the scene ².

Region-based techniques do not directly imply what regions are part of the foreground, and indeed, most semantic objects are composed of several regions in the segmentation. It is clear, therefore, that the type of identification performed by a segmentation algorithm directly effects its applicability: if an application requires semantic objects, a region-based algorithm will not suffice.

Media

When deciding what segmentation technique to apply, it is often important to know if the algorithm was intended to be used on video sequences or static images. The majority of classic segmentation algorithms, such as Watershed segmentation [Vincent and Soille, 1991], and the iterative split and merge algorithms [Horowitz and Pavlidis, 1974], were originally proposed for static image segmentation. Recently some of these methods have been extended [Bailer et al., 2005], and other new techniques have been proposed [Galmar and Huet, 2006], specifically for segmenting moving pictures.

The main challenge faced when considering video segmentation is that of producing regions or objects that are consistent across groups of frames. If a static image segmentation algorithm is applied to each frame in a video sequence,

²The most notable exception to this is video segmentation algorithms, especially those used in surveillance applications, that use background modeling and motion information to automatically extract semantic objects. This thesis is focused primarily on image segmentation.

usually the phenomenon of regions or objects abruptly disappearing and reappearing occurs. Algorithms designed specifically for video segmentation typically incorporate techniques for dealing with such phenomena.

Video segmentation is a large research area. In this thesis, we will limit our focus exclusively to image segmentation; a thorough treatment of video segmentation would require another dissertation. It is, however, worth noting that some of the algorithms that we discuss have been extended to work with video, and we shall indicate this where possible.

Generality

The final facet to consider when choosing a segmentation algorithm for a particular application domain is the algorithm's *generality*; that is, how general, or how specific, is the problem that the algorithm was designed to address? We refer to algorithms that are designed for a particular domain as model-specific, and general purpose algorithms as model-independent. Clearly whether an algorithm is model-specific affects the algorithm's applicability; if an algorithm is designed specifically to segment the brain mass from an MRI scan, it's unlikely to be very good at extracting flowers or trees from natural images. In general, model-specific algorithms perform better in their respective domains, whereas model-independent algorithms can be used for more applications.

When the application domain is sufficiently restrictive, model-specific algorithms are often a better choice. In medical image analysis, for example, the kinds of objects that are required are restrictive enough for model specific techniques to perform quite well: atlas guided segmentation [Kikinis et al., 1996], deformable models, and active contours [Kass et al., 1988, Caselles et al., 1995] are a few examples of such techniques. Similarly, model-driven algorithms are suitable for segmenting regular shapes, like lines, curves, and ellipses. The Hough-transform

and its derivatives [Duda and Hart, 1972, Yang et al., 1997] are well known examples of model-driven algorithms for segmenting lines and curves.

If, however, the application requires segmentation of more general scenes then model-independent algorithms are necessary. This thesis will focus primarily on model-independent algorithms.

2.2.2 Algorithm-Centric Classification

Classifying segmentation algorithms according to the previous criteria is useful in determining the different application domains in which they can be used. Another interesting way of considering segmentation algorithms is in terms of the general properties of the algorithms themselves. This algorithm-centric view, although perhaps less useful for the purposes of selecting an algorithm for a particular application, is useful when developing and evaluating segmentation algorithms. Figure 2.2 illustrates the algorithm-centric classification. From this viewpoint we consider more general algorithmic properties: the problem the algorithm is trying to solve (*model*), how it considers the data (*perspective*), and the level at which it attempts to compute the solution (*scale*).

Perspective

Most segmentation algorithms (aside from thresholding techniques), can be considered either region-based, edge-based, or a hybrid of these. Region-based algorithms are identified as methods that attempt to produce coherent regions by clustering together groups of pixels based on some homogeneity condition, such as the Euclidean distance in a particular color space. Edge-based algorithms begin by estimating the discontinuities in the image, usually by approximating the first or second derivatives, and proceed by linking together these edge pixels in some manner to form closed regions. Hybrid algorithms use both

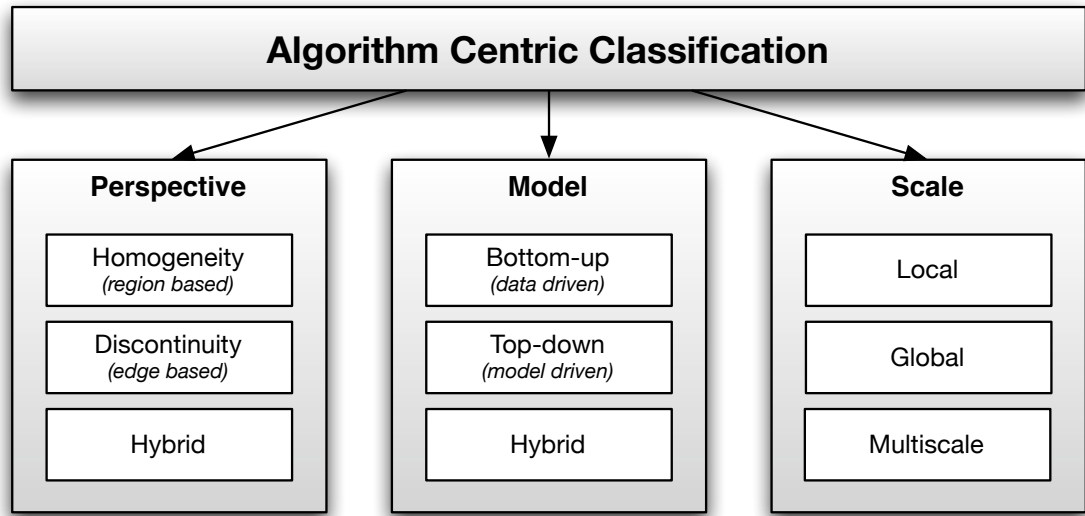


Figure 2.2: Algorithm-centric classification of segmentation algorithms

the homogeneity and discontinuity information to perform the segmentation. The EDISON system, described in [Christoudias et al., 2002], is an example of a hybrid algorithm. There is evidence to suggest that both discontinuity and homogeneity play important roles in low-level human and animal vision (see [Marr and Hildreth, 1980, Lee et al., 1998]).

Model

The model used to form a segmentation is closely related to the algorithm's generality in our application-centric classification. Bottom-up algorithms, also referred to as data-driven algorithms, attempt to infer regions or objects using low-level visual cues, like color, texture, and geometric information. Thus, these data-driven algorithms tend to make up the set of more general segmentation techniques. Top-down, or model-driven algorithms, begin with certain assumptions about the nature of the scene, i.e. they have a specific model of the objects that are to be segmented, and proceed by attempting to locate and extract these objects. Notable examples of top-down techniques include face extraction, line detection, and

thresholding. The model view of segmentation algorithms is also considered in [Adamek and O'Connor, 2006], where the authors use it as their primary method of classifying algorithms. They also consider semi-automatic and interactive segmentation algorithms to be model-driven, where in this case the model is created using markup supplied by the user.

Scale

Another important facet to consider is the level at which algorithms process information in the scene. Local algorithms are characterized by only processing a certain subset of the pixels in the scene at each step. Usually this subset is a central pixel and a prescribed set of neighbor pixels. Examples of local algorithms include: mean-shift [Comaniciu and Meer, 2002], seeded region growing [Adams and Bischof, 1994], statistical region merging [Nock and Nielsen, 2004], region adjacency graphs [Garrido et al., 1998], and recursive shortest spanning tree [Morris et al., 1986]. Global algorithms, on the other hand, consider the entire image when performing the segmentation. Often this involves (recursively) partitioning the image into regions while optimizing some predefined criteria. Popular global algorithms include the normalized cuts algorithm [Shi and Malik, 2000], the interactive graph cuts algorithm [Boykov and Jolly, 2001], and algorithms that are based on the Hough transform [Duda and Hart, 1972]. Multi-scale algorithms, in general, only consider local features, but do so at several different spatial scales.

2.3 Grouping Cues

All segmentation techniques require criteria to determine which pixels belong to which region. These criteria are known as grouping cues. Knowing which pixels to group into regions, and eventually into objects, requires some knowledge of how the human visual system works.

Our current understanding of the human visual system suggests that we perform perceptual grouping and structure extraction using many complex and interacting mechanisms [Grossberg and Raizada, 2000]. A simplified model of perception often used in computer vision research involves three stages. First, the visual information from the eye is filtered using structures in the visual cortex. These filters extract low-level features that describe the information in different parts of the scene: the color, texture, edges, geometry, and spatial layout. Next, these features are grouped into a more compact representation of the scene using perceptual grouping principles. The study of these perceptual grouping principles is known as Gestalt psychology [Koffka, 1935, Wertheimer, 1997], and the principles themselves are called Gestalt laws. Finally, this representation is processed by high-level mechanisms in which complex learned associations guide the discovery of structure.

The final stage in the above scheme is the least well understood, and the most difficult to model. Indeed developing an algorithm to model this stage involves addressing core artificial intelligence problems: learning, knowledge representation, and reasoning. Nevertheless, several groups have tackled the problem, and produced some interesting results [Athanasiadis et al., 2005, Athanasiadis et al., 2006, Papadopoulos et al., 2006], albeit in limited domains.

In the remainder of this section, we address the first two stages of this model, first discussing the principles of perceptual grouping, and then the low-level features to which they are typically applied.

2.3.1 Gestalt Laws

The Gestalt Laws are descriptive principles in Gestalt psychology that specify the way in which the human brain performs perceptual grouping. They have found extensive application, not only in computer vision, but as guiding principles in

visual interface design [Mullet and Sano, 1995, Chang et al., 2002] and in designing educational material [Moore and Fitz, 1993]. The main principles that govern perceptual grouping, according to Gestalt theory, are as follows:

Similarity

The mind groups together similar entities. This is perhaps the most widely applied Gestalt law in image segmentation. In computer vision applications it usually defined in terms of distance between low-level image features like color and texture, or higher level shape features such as size and shape.

Proximity

Entities that are spatially or temporally nearby are grouped together by the mind. This principle is implicit in any algorithm that considers pixel neighborhoods, either in the temporal or spatial domain, and so is particularly important in image segmentation.

Closure

The mind fills in missing gaps to complete entities. This principle has also received attention in image processing, for example, in edge-based image segmentation algorithms and Hough transforms. However, the principle of closure is often particularly difficult to formulate, as one often is required to first detect the object in order to rectify missing pieces that occur due to noise or occlusion. Figure 2.3 illustrates the Gestalt law of closure; despite the missing parts, we still easily perceive a circle and a square.

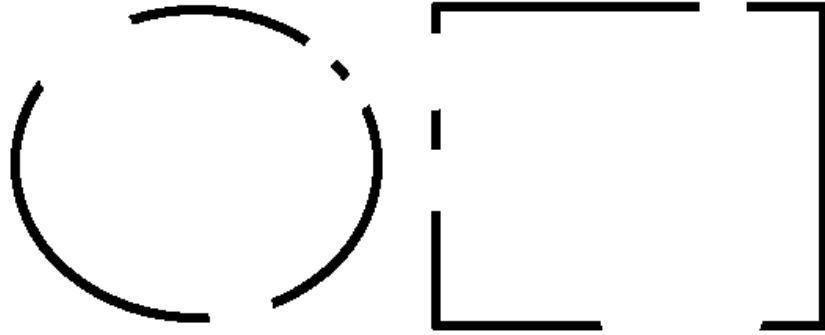


Figure 2.3: The Gestalt law of closure

Symmetry

The mind can easily interpolate missing pieces in symmetric entities, and studies such as [Attneave, 1954] give an indication of the capacity of the mind for perceiving symmetry. Attneave's study further investigates the significant informational redundancy that occurs as a result of the Gestalt principles, an indication that the mind exploits these principles to reduce the amount of information it is required to process and store by capitalizing on inherent redundancies in visual forms. This has a direct impact on perception.

Simplicity (Prägnanz)

The mind organizes or reduces reality into its simplest possible forms in terms of complexity, regularity, symmetry, etc. As an example, consider Figure 2.4. Although the entity is relatively complex, the mind immediately recognizes that it is composed of simple geometric primitives, that of a circle, rectangle, and triangle. Attempts to incorporate the simplicity principle into image segmentation have been made in [Adamek et al., 2005] and [Bennstrom and Casas, 2004].

Note that there are several other principles of Gestalt perception, such as the principle of common-fate, and the principle of continuity, that also have

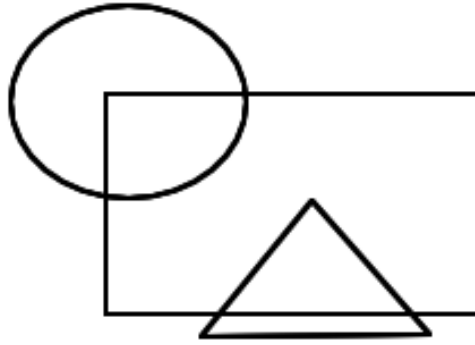


Figure 2.4: The Gestalt law of Prägnanz

applications in machine vision and segmentation. The interested reader is referred to [Koffka, 1935] and [Wertheimer, 1997] for a more thorough exploration.

The Gestalt principles discussed above are descriptive, not explanatory; they do not tell us how or why humans perceive entities in this way. Moreover, the degree of effect of any particular Gestalt principle on our interpretation of a scene is difficult to quantify: in Figure 2.3, is our interpretation based more on the closure principle, or more on symmetry? Such difficulties have lead to researchers focusing their efforts mainly on the proximity and similarity principles, as these can be formulated in terms of low-level image features.

2.3.2 Low-level Features

To perform perceptual grouping, the Gestalt principles tell us that we should group pixels that are in some way close to each-other (proximity), and that are also in some way alike (similarity). The most common way in computer vision of deciding how similar pixels or distinct groups of pixels are is by using low-level image features, and defining distance measures between these features. The purpose of these low-level features is to encapsulate information about the pixels they describe. The purpose of distance measures is to determine how similar

two sets of pixels are, based on the low-level features that describe them. Some of these low-level features are available directly from the image pixels; others require pre-filtering or pre-processing to extract. The following is an overview of the most commonly used low-level features in image segmentation.

Spatial Distance

According to the Gestalt law of proximity, entities should be grouped if they are near one another. Proximity is usually measured in terms of spatial distance for images, or spatiotemporal distance for video. Almost every image segmentation algorithm uses proximity in some way to guide the segmentation.

In segmentation, proximity is usually either implicitly observed using a hierarchical neighborhood system, or explicitly, by clustering pixels based on some distance metric in a joint feature space that incorporates both the spatial location of the pixel and its value (such as its color, or luminance). In the first case, common in local graph-based algorithms like statistical region merging [Nock and Nielsen, 2004], each pixel is assigned a neighborhood, usually consisting of the four or eight points that surround the pixel in the image grid. Grouping is then performed hierarchically, by merging similar neighbors to form regions. Observation of the proximity principle is therefore implicit since, at each step, only adjacent pixels or regions can be merged. Other algorithms, like normalized cuts [Shi and Malik, 2000] and mean-shift segmentation [Comaniciu and Meer, 2002], use spatial distance explicitly, clustering pixels based on (among other things) their spatial Euclidean distances.

Color

Color is by far the most common feature that is used to determine similarity, as color information is directly available from image pixels. The color space

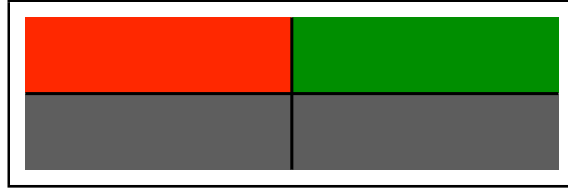


Figure 2.5: An illustration of the loss of information when converting to grayscale. The top two colors are $\mathbf{c}_1 = [1 \ 0 \ 0]^T$, and $\mathbf{c}_2 = [0 \ 0.5 \ 0]^T$. Their grayscale values, shown below, are indistinguishable.

and color distance measure used, however, vary substantially from algorithm to algorithm, and effect the computational complexity and overall performance.

Many earlier algorithms simply use absolute gray level difference as a color distance measure, for computational efficiency and simplicity of implementation. Although this leads to fast implementations, it discards color information, which can often negatively impact segmentation. As an example, consider two colors \mathbf{c}_1 and \mathbf{c}_2 in a RGB space, the first bright-red and the second green. We can represent these colors as three dimensional column vectors $\mathbf{c} = [r \ g \ b]^T$, where $r, g, b \in [0, 1]$ describe the brightness of each of the color channels. Let $\mathbf{c}_1 = [1 \ 0 \ 0]^T$, and $\mathbf{c}_2 = [0 \ 0.5 \ 0]^T$. The standard color to grayscale transform is a dot product with the vector $\mathbf{t} = [0.3 \ 0.59 \ 0.1]^T$. Most would perceive \mathbf{c}_1 to be different from \mathbf{c}_2 , yet their grayscale values are indistinguishable: $\mathbf{c}_1 \cdot \mathbf{t} = 0.2999$, $\mathbf{c}_2 \cdot \mathbf{t} = 0.2949$ (see Figure 2.5).

Image segmentation algorithms can use information from all three color channels to avoid problems like this. The issue now becomes how to compare two different colors. A first attempt might be to use the Euclidean distance between two RGB color vectors. Unfortunately, however, the distance between two points in RGB space is, in general, unrelated to the perceptual distance. There have been several attempts to design perceptually uniform color spaces, that is, color spaces in which the spatial Euclidean distance between two points reflects the perceived difference in the colors. Two of note are the CIELUV color space and

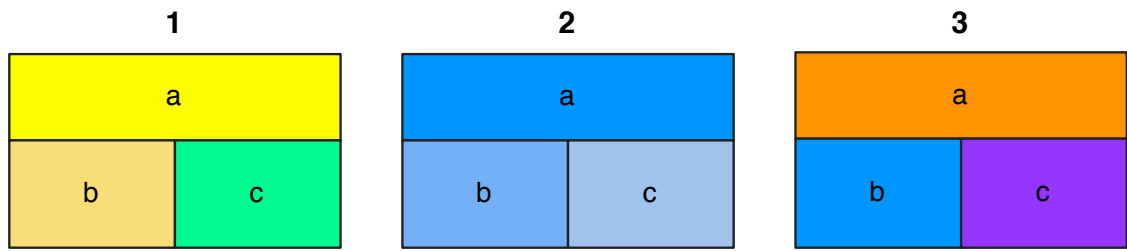


Figure 2.6: The saturation effect for large color differences. In swatch 1 and 2, most would agree that color *b* is more similar to color *a* than is color *c*. For swatch 3 the answer is less clear; the colors are simply judged to be “just different.”

the CIELAB color space [Wyszecki and Stiles, 2000, Schanda, 2007]; both were designed to be perceptually uniform with respect to the Euclidean metric. More accurate, non-Euclidean distance measures have since been developed for the CIELAB space (for instance CIEDE2000 [Johnson and Fairchild, 2003]).

The color spaces and distance measures described above are designed to model perceived color difference; they can, however, only do this with reasonable accuracy for small color differences [Shevell, 2003a]. This is not so much a problem with the models but more due to a saturation effect in the visual system: it is easy for us to tell which colors are more similar when the differences are small. When the differences are large, they become “just different.” Figure 2.6 illustrates the effect.

There are other complications when dealing with color perception. For instance, the human visual system compensates for differences in illumination conditions across a scene, allowing us to perceive different color stimuli as the same physical color. This effect is known as color constancy [Ebner, 2007]. The famous gray square optical illusion illustrates the effect (Figure 2.7).

Color science and colorimetry are fascinating research areas, and play a key role in image segmentation. For more information on the subject, there are several comprehensive books, including: [Wyszecki and Stiles, 2000], [Shevell, 2003b],

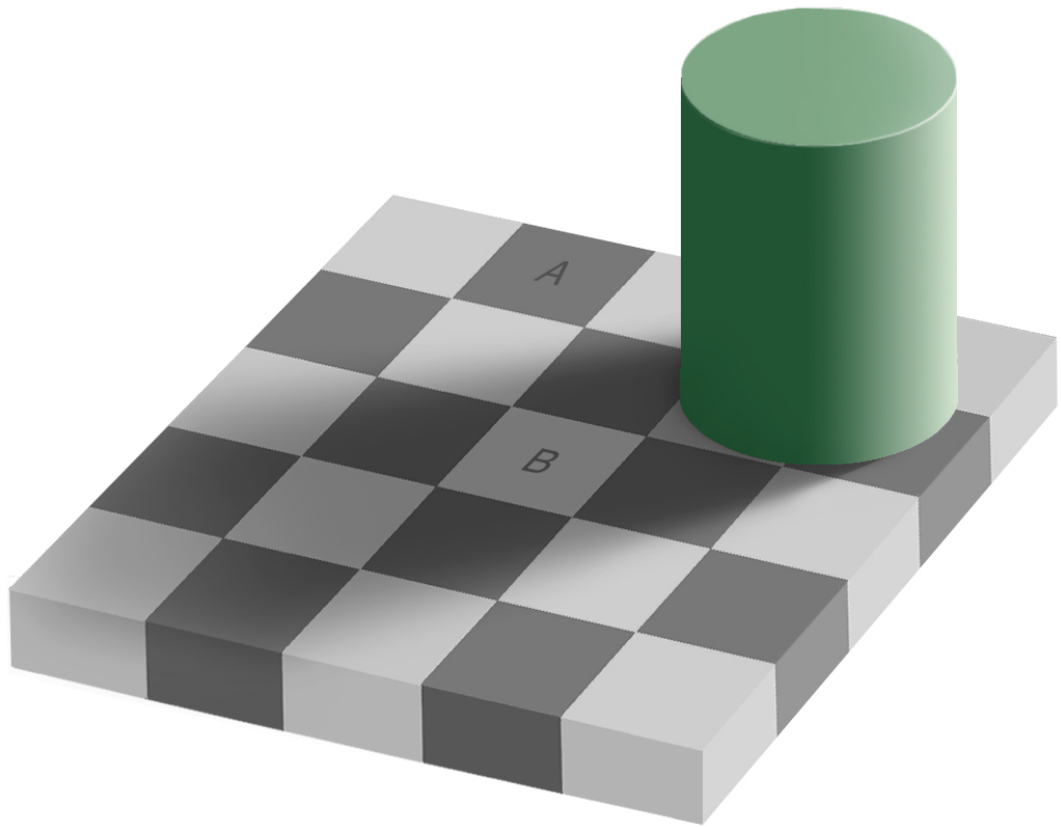


Figure 2.7: The gray square optical illusion. We perceive square marked 'A' to be darker than the square marked 'B;' in reality, they are exactly the same color. (Image due to Adelson [Adelson, 1995]; this is best seen on a computer screen as color reproduction may be inaccurate in the print version).

and [Ohta and Robertson, 2005]). More information on color science as applied to image segmentation can be found in the survey papers by Skarbek and Koschan [Skarbek and Koschan, 1994], and Cheng et al. [Cheng et al., 2001].

Edges

Abrupt changes in brightness or color in an image are known as edges, and have long been known to play an important role in recognition and perceptual grouping [Marr and Hildreth, 1980]. They have been used for image segmentation in edge linking algorithms [Farag and Delp, 1995], and hybrid algorithms [Sabera'b et al., 1997].

Edges can be extracted in many different ways. First-order edge detectors identify edges as the peaks in the first order spatial derivative of the image. Examples of first order edge detectors include the Sobel [Sobel and Feldman, 1968], Prewitt, and Roberts cross operators. The widely used Canny operator [Canny, 1986] is essentially a first order edge detector with some post processing steps. Second-order edge detectors identify edges as the zero crossings in the second derivative; the most well know example is the Marr edge detector [Marr and Hildreth, 1980]. Another, entirely different approach to edge and corner detection is the SUSAN operator [Smith and Brady, 1997], which uses a moving circular window to detect the presence of edges.

Texture

Humans find it easy to distinguish between textures. We have an intuitive notion of what texture is, yet a formal definition is elusive. Petrou and Sevilla [Petrou and Sevilla, 2006] state that texture is variation of data at scales smaller than the scale of interest. Their definition is subjective but instructive; useful texture features should capture the high-level structure of image detail but suppress the detail itself.

Texture is an aggregate feature. It makes no sense to analyze the texture of an individual pixel; texture features must be defined on groups or blocks of pixels. For image segmentation applications, extracting texture features usually involves analyzing the local spatial context of a pixel, and creating a vector of values that describe the variation in this context. Many methods have been devised to do this.

Two popular methods are local binary patterns and oriented multi-scale filter banks. Local binary patterns [Ojala and Pietikainen, 1999] produce a descriptor from the eight neighbors of each individual pixel. Since they operate at such a small spatial scale, they can be computed very quickly, but are only useful for

capturing micro-textures. Oriented multi-scale filter banks [Pietikäinen, 2000], on the other hand, are computationally expensive, but can distinguish texture at multiple scales and orientations. Various types of filters can be used to for these banks; Gabor filters and Wavelets are common choices. However, designing an optimal filter bank for a particular application, i.e. selecting an appropriate set of filters and filter parameters, can be challenging, and has been the subject of considerable research (for example, see: [Randen and Husoy, 1999, Clausi and Jernigan, 2000])

We have only glossed over the many ways of analyzing and creating texture features here; there is much literature on the subject. The interested reader should refer to the collection of papers in [Pietikäinen, 2000] for more information on filter banks and local binary patterns, and the survey paper by Materka and Strzelecki [Materka and Strzelecki, 1998] for a more in-depth review of texture analysis methods. The book by Petrou and Sevilla [Petrou and Sevilla, 2006] is also a good reference, with practical advice and many worked examples.

Geometry

The final family of low-level features we discuss are the geometric features. By geometric features we refer to the shape and configuration of spatial entities in an image. Several algorithms make use of geometric features, including many model-driven segmentation approaches, such as the Hough transform [Duda and Hart, 1972] and ellipse segmentation techniques. Geometric features are, however, more difficult to use as grouping cues in bottom-up segmentation as they are not available directly from the image. Some fully automatic segmentation algorithms [Adamek and O'Connor, 2006, Adamek et al., 2005] have made use of syntactic visual features [Bennstrom and Casas, 2004] at an intermittent stage in the segmentation process to merge together image regions so as to produce objects that have a lower overall complexity. For the interested reader, the book by Costa

and Cesar [da F. Costa and Cesar, 2001] provides an excellent overview of shape features.

2.4 Automatic Segmentation

This section examines four specific algorithms for automatic image segmentation: segmentation that does not require any user interaction. There are many such algorithms in the literature; we investigate a small subset that we believe to be representative of the state-of-the-art.

We first discuss region adjacency graphs. This is not so much a segmentation algorithm in itself, but a family of algorithms. A good deal of the segmentation algorithms from the literature fall into this category (though this is not always recognized in the papers describing them). Next we discuss the statistical region merging algorithm, a recent instance of a region adjacency graph type algorithm. Following this we describe two interesting algorithms that are not based on region adjacency graphs: the normalized cuts algorithm, and the mean-shift algorithm. In the final subsection we mention other noteworthy algorithms from the literature, and some of the traditional algorithms that still enjoy popularity.

2.4.1 Region Adjacency Graphs

Image segmentation algorithms based on region adjacency graphs, also known as *region merging* algorithms, have existed for some time (for instance, the algorithm proposed by Brice and Fennema [Brice and Fennema, 1970]). Garrido et al. [Garrido et al., 1998] noted that many image segmentation algorithms can be reformulated using region adjacency graphs, and hence completely specified using three specific criteria: the *merging order*, the *merging criteria*, and the *region model*. Examples of such algorithms include the statistical region merging algorithm, which we will visit in the next section, the recursive short-

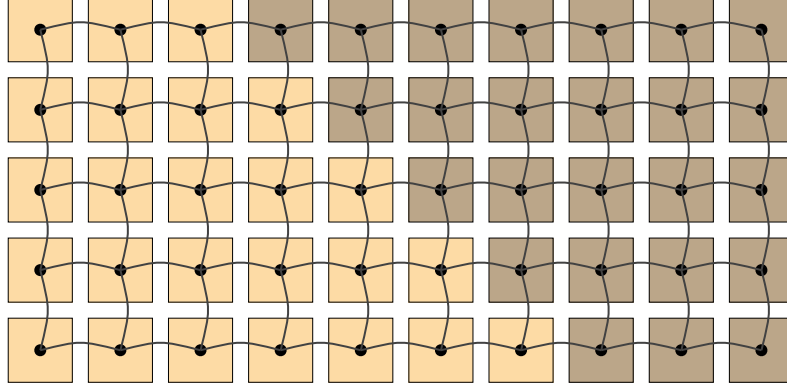


Figure 2.8: A 4-connected region adjacency graph for a small section of an image. The black dots are the nodes in the graph, and the colored squares beneath are the pixels they represent. Each node is connected to its four neighboring nodes.

est spanning tree algorithm [Morris et al., 1986], the split and merge algorithm [Horowitz and Pavlidis, 1974, Horowitz and Pavlidis, 1976], and the watershed algorithm [Vincent and Soille, 1991].

In the remainder of this section we review the region adjacency graph formulation of image segmentation, and discuss the three criteria that fully specify any region adjacency graph algorithm. We also outline a specific instance of a region adjacency graph algorithm, proposed by Garrido et al. in [Garrido et al., 1998].

Theory

An raster image is essentially a rectangular lattice of pixels. Each pixel representing a color. To transform an image into a region adjacency graph, we map each pixel in the image onto a node in the graph, and connect each node to its neighbors. More formally, given an image I with pixels $p_{ij} \in I$, we construct the graph $G = (V, E)$ to have vertices $p_{ij} \in V$. The set of edges are given by:

$$E = \{(p, q) : p, q \in V \wedge q \in \mathcal{N}_p\}$$

where \mathcal{N}_p is the set of four or eight neighbors of pixel p . Figure 2.8 illustrates the concept using a four connected neighbor system for a simple 5 row by 10 column image.

Given this region adjacency graph, a region merging algorithm is then defined as any algorithm that operates on this graph by iteratively removing edges and merging the corresponding nodes, eventually terminating when the segmentation is complete. Any such algorithm can be specified using three criteria:

1. *The merging order.* This defines the order in which the edges in the graph are considered for merging. The merging order is usually given by a similarity measure between two nodes. For example, the merging order might be defined by color similarity: nodes that exhibit high color similarity should be merged before those with low color similarity.
2. *The merging criteria.* When each edge is visited, the merging criteria decides if it is to be merged or not. In essence, the merging criteria defines the stopping criteria for the algorithm.
3. *The region model.* When two nodes are merged, the region model decides how to represent their union. The region model might, for example, dictate that when two nodes are merged, the average color of the two nodes should represent the new node. Figure 2.9 illustrates the effect of merging two nodes in a region adjacency graph algorithm.

The general region merging algorithm is specified in terms of the above criteria as follows:

- *Step 1.* Initialize the region adjacency graph G from the image I using the region model. Add all edges from the region adjacency graph to the edge queue.

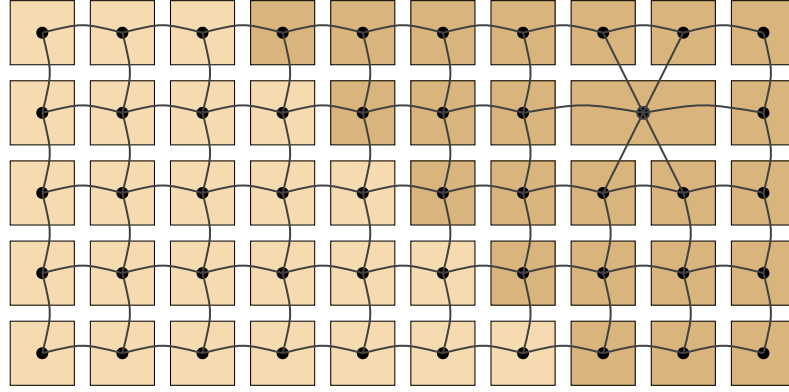


Figure 2.9: Merging nodes in a region adjacency graph algorithm

- *Step 2.* If the edge queue is empty go to step 4. Otherwise order edge queue according to the merging order.
- *Step 3.* Remove first edge from the edge queue and test it for merging using the merging predicate. If the merging predicate succeeds, merge the regions according to the region model. Return to step 2.
- *Step 4.* Stop the merging and output the segmentation.

Algorithm

What remains is to specify the three criteria: the merging order, the merging predicate, and the region model. Garrido et al. focus on the segmentation of gray level images and suggested a whole range of these criteria in [Garrido et al., 1998]. We discuss a few presently, refer to [Garrido et al., 1998] for more information.

Region Model: Let $f(p)$ be the gray level at pixel p in the image. The region adjacency graph is initialized so that the initial model for each node p equals the gray level in the image: $M_p = f(p)$. Two ways are suggested to update the model when nodes are merged. The first is to combine the models using a weighted

average; nodes p and q are combined as:

$$M_{pq} = (N_p M_p + N_q M_q) / (N_p + N_q)$$

where N_p is the number of pixels represented by node p . The other suggestion is to use the median:

$$M_{pq} = \begin{cases} M_p & N_p > N_q \\ M_q & N_p < N_q \\ \frac{1}{2}(M_p + M_q) & N_p = N_q \end{cases}$$

which they assert to be more robust than the mean.

Merging order: Garrido et al. define several potential merging orders. The simplest is to order the edges using the average squared error between the models of the regions. That is, the ordering is defined by a function $O(p, q)$:

$$O(p, q) = ||M_p - M_q||^2$$

Also suggested is ordering based on mean squared error. If $R(p)$ is the set of all pixels in node p then this is given by:

$$O(p, q) = \sum_{x \in R(pq)} \frac{(f(x) - M_{pq}(x))^2}{N_p + N_q}$$

They note that in practice these orderings both produce a few large regions surrounded by many small regions, and suggest the following ordering as a compromise:

$$O(p, q) = N_p(M_p - M_{pq})^2 + N_q(M_q - M_{pq})^2$$

which is the squared difference between the current model and the proposed model, weighted by area.

Merging criteria: The last thing to specify is the merging criteria $C(p, q)$, which will control when the algorithm terminates. Garrido et al. suggest two merging criteria: area merging criteria, and contrast merging criteria. The area merging criteria will merge two nodes if the number of pixels in either is below a preset threshold: $C_A(p, q) = N_p < T_A \vee N_q < T_A$. This criteria ensures the final segmentation only contains regions of area larger than T_A . The contrast merging criteria merges nodes if the square difference between the models is less than a preset threshold: $C_C(p, q) = \|M_p - M_q\|^2 < T_C$.

Implementation

To efficiently implement a region adjacency graph algorithm the data structure used to store the edge queue needs to be carefully chosen. In particular, it must be possible to perform fast insertion and ordered removal. Garrido et al. suggest the use of balanced binary partition trees (for example red-black trees [Cormen et al., 2001]). Using such trees the complexity is $O(n \log n)$ for the initial sort, and $O(\log n)$ for each update. Storing the edges in a heap data structure [Cormen et al., 2001] gives similar complexity.

Analysis

A classification of the region adjacency graph algorithm according to the criteria set forth in Section 2.2 is shown in Table 2.1. Note that the application-centric classification given here is specific to the instance described in [Garrido et al., 1998]; the region adjacency graph model of segmentation is quite general. It can easily be applied to video, for example, or used for interactive segmentation [Salembier and Garrido, 2000].

The algorithm has some disadvantages. First, using only luminance for segmentation discards important color information. This can negatively impact

Application-centric classification

Interaction:	Automatic
Identification:	Region
Media:	Static
Generality:	Model independent

Algorithm-centric classification

Perspective:	Homogeneity
Model:	Data driven
Scale:	Local

Table 2.1: Classification of the region adjacency graph algorithm

segmentation performance (see Section 2.3.2). Garrido et al. state that color information can easily be incorporated into the scheme, using a linear combination of the channel differences to define the merging order; in practice color perception is not so straightforward.

Three issues need to be carefully considered to meaningfully incorporate color. First, is how to determine the similarity between two colors. A reasonable solution here would be to use an appropriate color space and color difference measure. The CIELAB space and the CIEDE2000 difference measure are good candidates. These only give meaningful results, however, for small color distances (see Section 2.3.2). Second, it is necessary to determine how to represent the union of two regions. This is not trivial: is the average of two colors perceptually meaningful? Finally, it is necessary to determine the merging criteria. Since the human visual system can only accurately compare small color differences, the saturation point at which colors become “just different” would seem a reasonable choice. There are no studies, however, that explicitly indicate this point.

In addition, the algorithm is a local scale algorithm. These tend to be more sensitive to small variation and noise than their global scale counterparts. The algorithm is also heuristic; no specific criteria is optimized.

Finally, Garrido et al. make no attempt to formally evaluate the performance of their region adjacency graph algorithm against the state-of-the-art. Anecdotal evidence is given, but performance is difficult to judge. Nevertheless, the region adjacency graph formalism is extremely useful: it provides a common framework in which many other algorithms can be recast and analyzed.

2.4.2 Statistical Region Merging

The statistical region merging algorithm (SRM) is another region adjacency graph based segmentation algorithm [Nock and Nielsen, 2004]. In their paper, Nock and Nielsen formulate image segmentation as a statistical inference problem, and derive a simple merging order and merging predicate that can achieve, with high probability, a low error in segmentation. There are two points of note about the algorithm: First, unlike the previously described algorithm SRM incorporates (in a limited way) color information. Second, it can be implemented to have a $O(n)$ computational complexity, making it one of the fastest color image segmentation algorithms available.

Theory

The statistical region merging algorithm operates on a four connected region adjacency graph. It suffices, therefore, to specify the merging criteria, the merging order, and the region model in order to completely specify the segmentation. Note that Nock and Nielsen refer to the merging criteria as the *merging predicate*, and define the region model implicitly. We will consider the region model explicitly, since it is primarily what makes a linear time algorithm possible.

Let us first turn to the merging criteria. Nock and Nielsen define the merging criteria so as to provide a quantitative bound on the segmentation error as follows.

Given a region R in an image I , let

$$b(R) = g \sqrt{\frac{1}{2Q|R|} \ln \left(\frac{1 + |R|}{\delta} \right)} \min(g, |R|) \quad (2.1)$$

where g is the range of the color band (usually $g = 256$) and $\delta = 1/(6|I|^2)$. The Q parameter controls the scale of the segmentation: the size and number of regions produced. The merging predicate is then defined as:

$$\mathcal{P}(R, R') \Leftrightarrow |\bar{R}' - \bar{R}| \leq \sqrt{b^2(R) + b^2(R')} \quad (2.2)$$

where \bar{R} is a the value of a color channel, and the predicate $\mathcal{P}(R, R')$ must be true for all color bands.

The merging order is specified by an invariant A , defined as follows: If a test between any two true regions occurs, it implies that all tests between pairs of regions contained within these regions has occurred previously. For RGB color images the invariant is realized by sorting the links to be tested according to a weight W_{ij} equal the maximum absolute difference between each of the color bands:

$$W_{ij} = \max(|R_i - R_j|, |G_i - G_j|, |B_i - B_j|) \quad (2.3)$$

and testing the merging predicate on links in the implied order.

The merging order is defined in terms of the individual pixels; it does not change after pixels have been merged to form regions. The region model is therefore the set of all pixels that form the internal boundary of a region. Merging two regions results in the region model being updated to be the internal boundary of the union of these two regions.

The implication is that because the merging order is fixed it can be calculated by sorting the edges between the nodes in the original region adjacency graph.

This sort only needs to be performed once: a balanced binary tree is unnecessary, implying each merge runs in constant time, as opposed to logarithmic.

Algorithm

The SRM algorithm proceeds as follows: The region adjacency graph is built in the usual way and each edge is weighted according to W_{ij} defined above. These edges are then sorted according to their edge weight. A single pass through the edges is then performed, merging the corresponding regions if the merging predicate is satisfied.

Implementation

From the above discussion, it may appear that the algorithm still requires $O(n \log n)$ time; after all, the edges still need to be sorted, and the lower bound complexity for comparative sorting is $O(n \log n)$. For the integral RGB color space, however, a comparative sort can be neatly avoided by noting that $W_{ij} \in \mathbb{Z}$ and $0 \leq W_{ij} \leq 255$, and using the bucket sort algorithm [Cormen et al., 2001]. Bucket sort is $O(n)$, thus the complete algorithm runs in linear time.

The basic algorithm as described will often tend to produce a few large regions and many small regions. As a workaround, a post-processing stage in which regions with an area less than a prescribed threshold are merged with their nearest neighbor is usually added.

Analysis

There are two major advantages of the statistical region merging algorithm. First, since it requires only $O(n)$ time, it is very fast. Second, it depends only upon the single parameter Q in Equation (2.1) that controls the scale of the segmenta-

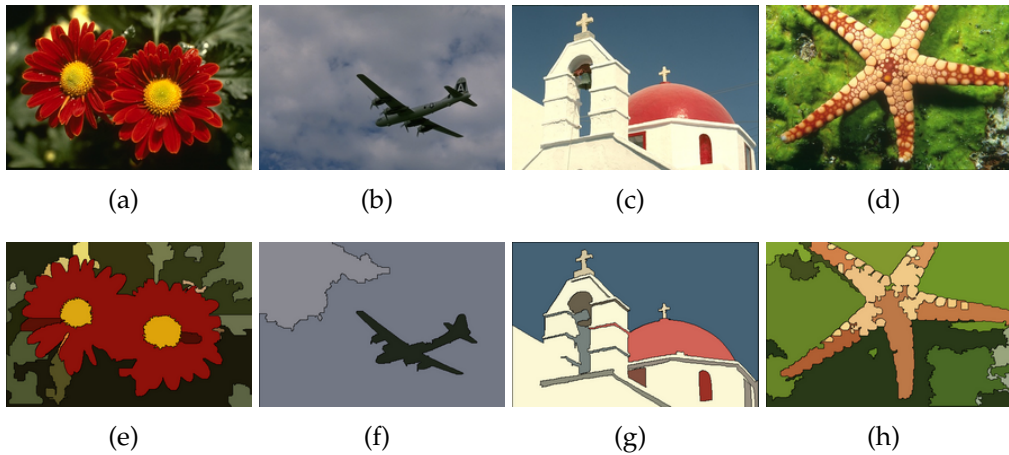


Figure 2.10: Examples of the statistical region merging algorithm

tion. This can be beneficial if we wish to apply a machine learning algorithm to determine the optimal Q for a given application.

There are, however, some limitations. The algorithm makes no attempt to be perceptually accurate: it treats each color band separately. The separation of bands also inhibits extending the algorithm to other color and feature spaces. If the color space used is not composed of integral channels, the complexity degenerates to $O(n \log n)$. As a region adjacency graph algorithm, the algorithm operates on a local scale, and suffers the same problems as other local algorithms. Nock and Neilson do, however, demonstrate the algorithm to be reasonably robust to random noise.

Table 2.2 outlines the algorithmic properties and applicability of the algorithm. Although the technique is classed as static with respect to the media it operates on, extending the algorithm to temporal media appears possible, although this idea was not explored in [Nock and Nielsen, 2004]. Figure 2.10 shows some sample images and the corresponding output of the algorithm for these images.

Application-centric classification

Interaction:	Automatic
Identification:	Region
Media:	Static
Generality:	Model independent

Algorithm-centric classification

Perspective:	Homogeneity
Model:	Data driven
Scale:	Local

Table 2.2: Classification of the statistical region merging algorithm

2.4.3 Normalized Cuts for Image Segmentation

The image segmentation algorithms we have discussed thus far have all been based on performing bottom-up merging on region adjacency graphs. In their 2000 paper, Shi and Malik propose an entirely different approach [Shi and Malik, 2000]. Their algorithm is known as normalized cuts; like the previous algorithms, it is graph based. The difference is that instead of performing bottom-up merging of regions, the normalized cuts algorithm begins with a single region: the whole image, and performs recursive top-down splitting to form the final segmentation. The process used to split a region is known as spectral clustering, and is a global scale algorithm.

Theory

Given a two dimensional image \mathbf{I} , assume a function $\mathbf{v} = f(\mathbf{x})$ that maps each location $\mathbf{x} \in \mathbb{Z}^2$ in \mathbf{I} to an n dimensional feature vector $\mathbf{v} \in \mathbb{R}^n$, which describes local properties of \mathbf{I} at \mathbf{x} . We associate the image \mathbf{I} with a weighted undirected graph $G = (V, E)$ such that each feature vector $\mathbf{v}_i \in V$ is a node in the graph and every pair of nodes is connected by an edge $\{i, j\} \in E$. Each edge is weighted

with a weight w_{ij} equal to a function of the similarity between the feature vectors \mathbf{v}_i and \mathbf{v}_j .

Using G we can construct a segmentation of I by recursively bipartitioning the graph into disjoint subsets A, B such that $A \cup B = V$, by removing edges from the graph. The sum of weights of the removed edges in such a partitioning is known as a cut of the graph.

$$cut(A, B) = \sum_{\mathbf{v}_i \in A, \mathbf{v}_j \in B} w_{ij}$$

The optimal partitioning that minimizes this cut value is known as the minimum cut, and there exist fast algorithms to compute it, such as max-flow algorithms [Boykov and Kolmogorov, 2004]. However, as noted by Wu and Leahy [Wu and Leahy, 1993], constructing a segmentation by recursively computing the minimum cut of a graph usually leads to over-segmentation: that is, an excessive number of regions in the segmentation. This is because if $|A|$ is significantly different to the $|B|$ then there are less weights in the summation $cut(A, B)$.

To inhibit the bias of the minimum cut for over-segmentation, Shi and Malik proposed that the cost of the cut be normalized by the total edge weight between nodes in A and B , and all other nodes in the V . The new normalized cut measure is defined as:

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

where $assoc(X)$ is the associativity between a subset of nodes $X \subset V$ with all nodes in V :

$$assoc(X) = \sum_{\mathbf{v}_i \in X, \mathbf{v}_j \in V} w_{ij}$$

Although finding the exact minimum normalized cut is NP-complete (a complete proof is given in [Shi and Malik, 2000]), it is possible to approximate a solution in polynomial time. Let $N = |V|$ be the number of pixels in the image.

Let \mathbf{D} be an $N \times N$ diagonal matrix such that the diagonal elements $\mathbf{D}_{ii} = \sum_j w_{ij}$ equal the total connection between v_i and all other nodes in V . Also, define \mathbf{W} to be an $N \times N$ symmetric matrix where $\mathbf{W}_{ij} = w_{ij}$. The minimum normalized cut value can then be approximated by solving the generalized eigenvalue problem:

$$(\mathbf{D} - \mathbf{W})\mathbf{y} = \lambda\mathbf{D}\mathbf{y} \quad (2.4)$$

The eigenvector \mathbf{y} corresponding to the second lowest eigenvalue λ in the solution to (2.4) contains N elements, and can then be used to bipartition G such that $\mathbf{v}_i \in A \iff \mathbf{y}_i < 0$. The proof of this result can be found in [Shi and Malik, 2000].

Algorithm

Using the above discussion, it is now possible to specify the recursive normalized cuts segmentation algorithm as follows:

1. Given an image, construct a graph representation $G = (V, E)$, where the edge weights w_{ij} correspond to a measure of similarity between nodes v_i and v_j . From this, derive the \mathbf{D} and \mathbf{W} matrices.
2. Solve the generalized eigensystem $(\mathbf{D} - \mathbf{W})\mathbf{y} = \lambda\mathbf{D}\mathbf{y}$ for the second smallest eigenvalue λ .
3. Bi-partition the graph using the eigenvector \mathbf{y} corresponding to the second smallest eigenvalue as an indicator vector.
4. Check the stability of the cut to decide if the current partition should be sub-divided. Recursively sub-divide if necessary.

Implementation

A few details need to be mentioned in order to make an implementation of the normalized cuts algorithm practical. First, the matrix \mathbf{W} above will be prohibitively

large, even for moderately sized images. To avoid this we can define our feature similarity function in such a way that, as the spatial distance between two feature points increases, then $w_{ij} \rightarrow 0$. Most of the elements of \mathbf{W} are now very near zero and can then be truncated. The resulting matrix is sparse and can be efficiently stored and processed [Saad, 1992]. Allowing $w_{ij} \rightarrow 0$ is reasonable, because it reflects the gestalt *proximity* grouping principle.

We also need a way of efficiently solving the now sparse symmetric eigenvalue problem in (2.4). Fortunately, several solvers exist for exactly this type of problem, including the Implicitly Restarted Arnoldi/Lanczos Method [Sorensen, 1996]. However, even with such solvers, the eigenvalue computation is still significant, typically requiring $O(N^{\frac{3}{2}})$ operations, which equates to approximately 2-3 minutes of computation time even for moderately sized images on a desktop computer. If an approximation is acceptable, then a significant speedup can be achieved using a multi-scale approach, such as the fast multi-grid inspired optimization in [Sharon et al., 2000].

Because equation (2.4) only allows us to approximate the solution to finding the minimum normalized cut, the elements of \mathbf{y} will take continuous values, instead of simply $\{-1, +1\}$. Thus, often the partitioning can be improved by choosing a value other than zero as the split point, usually by computing the *ncut* cost at several discrete intervals between $\{-1, +1\}$ and choosing the optimal one as the split point.

Of course, a similarity function is also required. Shi and Malik suggest using the following joint exponential similarity function:

$$w_{ij} = \begin{cases} \exp \frac{-\|\mathbf{v}_i - \mathbf{v}_j\|^2}{\sigma_v^2} \exp \frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma_x^2} & \|\mathbf{x}_i - \mathbf{x}_j\|^2 < r \\ 0 & \text{otherwise} \end{cases} \quad (2.5)$$

where r is some spatial neighborhood radius. The parameters σ_v and σ_x are feature and spatial-distance bandwidth parameters, and are typically set to values of between 10 and 20 percent of the maximum distance between any two features. Note that both the neighborhood radius r and the bandwidth parameters have a direct effect on the execution time of the method (by affecting the number of non zeros in the sparse matrix \mathbf{W}) as well as the coarseness and quality of the result.

Finally, we need to determine the stability of a cut, as this will act as the stopping criteria. Given an eigenvector solution \mathbf{y} from Equation (2.4), when the values of the eigenvector resemble a continuous function rather than an discrete indicator vector, it implies that a stable split point does not exist for the graph. Thus, a simple way of determining the stability of the split involves computing a histogram representation of the eigenvector, followed by determining the ratio between the minimum and maximum values in the bins. A higher value of this ratio, denoted $\mathcal{R}(\mathbf{y})$, indicates a stable cut. In their paper, Shi and Malik suggest using a threshold of $\mathcal{R}(\mathbf{y}) > 0.06$ in combination with a threshold on the cost of the cut as a stopping criteria for the process.

Analysis

Table 2.3 classifies the normalized cuts algorithm according to the criteria discussed at the beginning of the chapter. From this, there are two notable advantages of the normalized cuts algorithm. First, it is one of the few model-independent region segmentation algorithms that is also global in scale. Global algorithms, in general, are more robust and more resilient to noise: the optimization criteria encapsulates information about the relationships among all pixels in the image simultaneously. As such, decisions are based on the relationships among all pixels in the image, rather than just the relationships among a small subset of the pixels. Second, the normalized cuts algorithm can easily be extended to handle different feature spaces and distance measures, simply by modifying the similarity function

Application-centric classification

Interaction:	Automatic
Identification:	Region
Media:	Static/temporal
Generality:	Model independent

Algorithm-centric classification

Perspective:	Homogeneity
Model:	Data driven
Scale:	Global

Table 2.3: Classification of the normalized cuts algorithm

in Equation (2.5). Third, the method can be extended in a straightforward way to video, although this significantly increases the complexity.

The algorithm also has some limitations. It has a tendency to over-balance segments, a problem that is particularly visible in the first few partitions of natural images. These partitions can appear unnatural and often conflict with human intuition. The problem becomes less pronounced as the partitioning continues, visible only as an over-segmentation of certain regions. This can be at least partially alleviated by post-processing using an agglomerate clustering algorithm.

Another disadvantage of the normalized cuts method is computation time. As mentioned earlier, a standard implementation using an Arnoldi/Lanczos eigensolver requires $O(N^{\frac{3}{2}})$ operations for each cut, making it very computationally expensive. Even when using an optimized multi-scale version of the algorithm, computation can take 20–30 seconds for a moderately sized image on a desktop computer. In some applications this complexity may be unacceptable.

This high computational complexity can be somewhat avoided by reducing the amount of data the algorithm has to process. A simple strategy is to subsample the image before segmentation, and resample the region mask after. This optimization will, however, result in loss of detail and diminish edge localization.

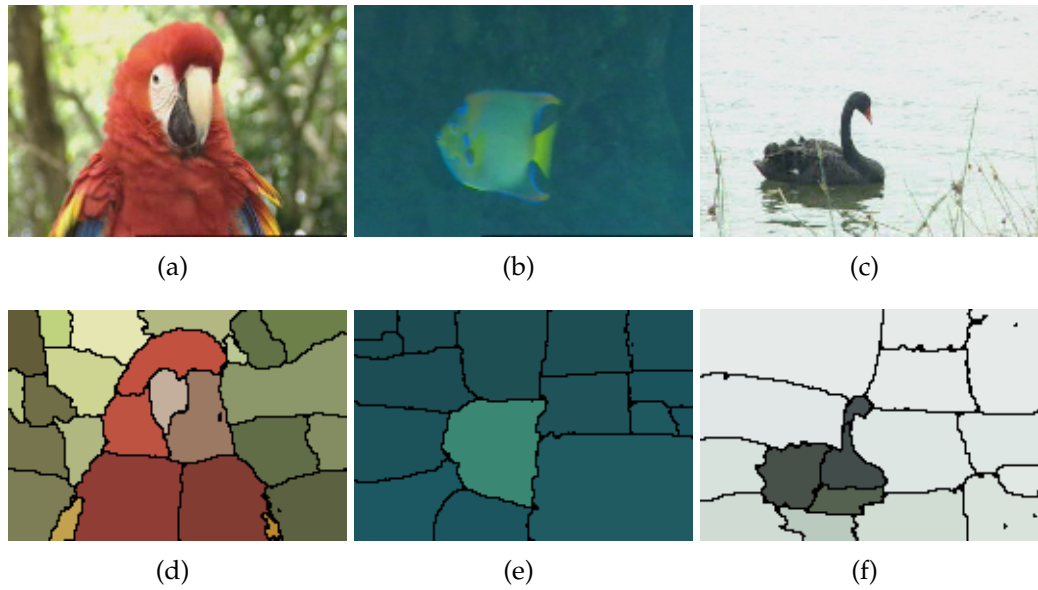


Figure 2.11: Example output of the normalized cuts algorithm

Figure 2.11 provides some sample images and the corresponding output of the normalized cuts algorithm; post-processing and agglomerate clustering were not performed.

2.4.4 Mean-shift Analysis for Image Segmentation

Unlike the previously discussed algorithms, which are all graph-based, the mean-shift algorithm [Comaniciu and Meer, 2002] is based upon determining local modes in the joint spatio-feature space of an image, and clustering nearby pixels to these modes. The technique is closely related to bilateral filtering [Tomasi and Manduchi, 1998], in that it is based on filtering simultaneously in the spatial and range domains. The method is general and robust; it has been adapted to several other problems including video surveillance and temporal image clustering.

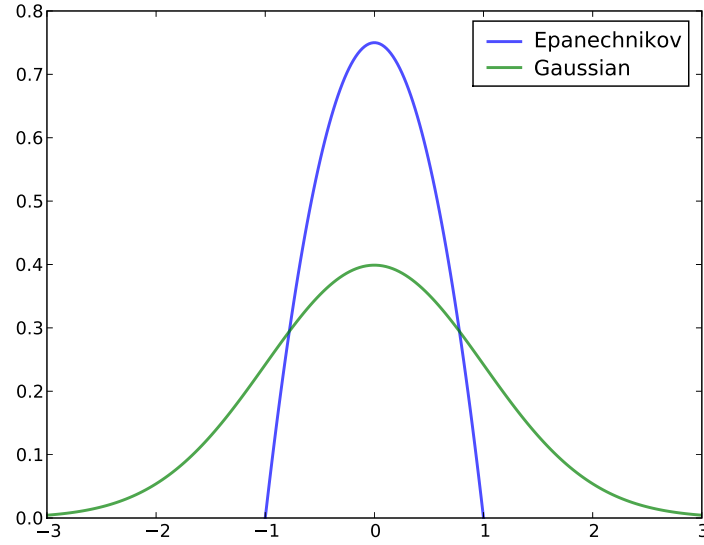


Figure 2.12: Two dimensional Gaussian and Epanechnikov kernels. The Gaussian kernel shown has zero mean and unit standard deviation.

Theory

The idea behind mean-shift image segmentation is quite simple. The image to be segmented is first processed to extract an d -dimensional feature vector describing each pixel. Each of these feature vectors is then visited in turn, and a d -dimensional kernel function is used to estimate the local density maxima at that point. The kernel is then shifted toward this maxima and the kernel function is used to re-evaluate the maxima at the new point in the feature space. This process repeats until a stationary point, the local feature space mode, is reached. The pixel that generated this mode is then clustered to the region defined by the mode. When the process completes, all pixels with the same local density mode in the feature space will belong to the same region in the segmentation.

If $\mathbf{x}_1, \dots, \mathbf{x}_n$ are vectors in a d -dimensional feature space, then the *multivariate kernel density estimator* gives an estimation of the density at a given point in the

space for a radial bandwidth parameter h , and is defined as:

$$\hat{f}(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^n K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right)$$

where $K(\mathbf{x})$ is some *kernel* function. In practice a Gaussian or Epanechnikov kernel is usually used (see Figure 2.12). If the kernel satisfies certain properties (see [Comaniciu and Meer, 2002]) and is differentiable, the density of the gradient at \mathbf{x} can be estimated as:

$$\nabla \hat{f}(\mathbf{x}) = \frac{1}{nh^{d+1}} \sum_{i=1}^n \nabla K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \quad (2.6)$$

In some cases, the kernel can be expressed as $K(\mathbf{x}) = ck(\|\mathbf{x}\|^2)$ where k is known as a profile function and c is a positive constant required to make $K(\mathbf{x})$ integrate to one. Both the Gaussian and Epanechnikov kernels can be expressed in this form. For example, the d -dimensional Gaussian kernel,

$$K_N(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}\|\mathbf{x}\|^2\right) \quad (2.7)$$

has the following kernel profile:

$$k_N(x) = \exp\left(-\frac{1}{2}x\right) \quad (2.8)$$

Using this kernel profile notation Equation (2.6) can be expressed as:

$$\nabla \hat{f}(\mathbf{x}) = \frac{2c}{nh^{d+2}} \sum_{i=1}^n (\mathbf{x} - \mathbf{x}_i) k'\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right) \quad (2.9)$$

Substituting $h(\mathbf{x}_i)$ for $k'(\|(\mathbf{x} - \mathbf{x}_i)/h\|^2)$ in Equation (2.9) produces

$$\nabla \hat{f}(\mathbf{x}) = \frac{2c}{nh^{d+2}} \sum_{i=1}^n (\mathbf{x} - \mathbf{x}_i) h(\mathbf{x}_i) \quad (2.10)$$

$$= \frac{2c}{nh^{d+2}} \left[\mathbf{x} \sum_{i=1}^n h(\mathbf{x}_i) - \sum_{i=1}^n \mathbf{x}_i h(\mathbf{x}_i) \right] \quad (2.11)$$

$$= \frac{2c}{nh^{d+2}} \sum_{i=1}^n h(\mathbf{x}_i) \left[\mathbf{x} - \frac{\sum_{i=1}^n \mathbf{x}_i h(\mathbf{x}_i)}{\sum_{i=1}^n h(\mathbf{x}_i)} \right] \quad (2.12)$$

The last bracketed term is called the mean-shift, and points in the direction of the local density maxima. If $S_h(\mathbf{x})$ is the set of points contained in a hypersphere with radius h centered on \mathbf{x} , substituting the derivative of the Epanechnikov kernel profile leads to the following mean-shift expression:

$$\mathbf{m}_E(\mathbf{x}) = \frac{1}{|S_h(\mathbf{x})|} \sum_{\mathbf{x}_i \in S_h(\mathbf{x})} \mathbf{x}_i - \mathbf{x} \quad (2.13)$$

while substituting the derivative of the normal kernel profile produces a weighted summation:

$$\mathbf{m}_N(\mathbf{x}) = \frac{\sum_{i=1}^n \mathbf{x}_i \exp\left(-\frac{1}{2} \left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2\right)}{\sum_{i=1}^n \exp\left(-\frac{1}{2} \left\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \right\|^2\right)} - \mathbf{x} \quad (2.14)$$

Algorithm

Before segmentation, each pixel in the image is associated with a feature vector \mathbf{x}_i in a joint spatial-feature space. That is, the feature vector \mathbf{x}_i is a vector composed from the spatial coordinates \mathbf{x}_i^s and a range (usually color) feature vector \mathbf{x}_i^r . Letting $\mathbf{m}(x)$ be the mean-shift for either the normal (2.14) or Epanechnikov (2.13) kernels, the segmentation algorithm proceeds as follows:

1. For each feature vector \mathbf{x}_i , perform the *mean-shift procedure*:

- (a) Set $\mathbf{y}_1 = \mathbf{x}_i$.

- (b) Repeat $\mathbf{y}_{i+1} = \mathbf{y}_i + \mathbf{m}(\mathbf{y}_i)$ while $|\mathbf{y}_{i+1} - \mathbf{y}_i| > \tau_1$.

- (c) Set \mathbf{x}_i to the converged value of \mathbf{y} .
2. Identify clusters of feature vectors by grouping all converged points that are closer than a prescribed threshold τ_2 .
3. Assign labels to clusters.

Implementation

Typically the spatial domain and the range domain are different in nature, so it is often desirable to employ separate bandwidth parameters $h = \{h_s, h_r\}$. When using the Epanechnikov kernel, this is not a problem: the hypersphere is simply constructed to contain only feature points that have a spatial distance less than h_s and range distance less than h_r . For the normal profile, we can define the kernel as a product of two separate kernels in each domain, and the mean-shift becomes:

$$\mathbf{m}_N(\mathbf{x}) = \frac{\sum_{i=1}^n \mathbf{x}_i \exp\left(-\frac{1}{2} \left\| \frac{\mathbf{x}^s - \mathbf{x}_i^s}{h_s} \right\|^2\right) \exp\left(-\frac{1}{2} \left\| \frac{\mathbf{x}^r - \mathbf{x}_i^r}{h_r} \right\|^2\right)}{\sum_{i=1}^n \exp\left(-\frac{1}{2} \left\| \frac{\mathbf{x}^s - \mathbf{x}_i^s}{h_s} \right\|^2\right) \exp\left(-\frac{1}{2} \left\| \frac{\mathbf{x}^r - \mathbf{x}_i^r}{h_r} \right\|^2\right)} - \mathbf{x}$$

In practice, summing over the entire domain for the normal kernel is inefficient, so the kernel is truncated in the spatial domain, allowing the summation to be carried out over a finite spatial window centered on the reference pixel.

There are several other optimizations that can be used to improve the efficiency of the mean-shift procedure; however, most of them involve approximations that may effect the quality of the result. Carreira-Perpiñán reviews several optimizations to the Gaussian kernel mean-shift algorithm and evaluates their consequences [Carreira-Perpiñán, 2006]. He concludes that, with careful parameter selection, optimization based on spatial discretization produces the most significant speedups (between $10\times$ and $100\times$, depending on the image and value of σ) and results in relatively low clustering error ($< 3\%$). If desired, the

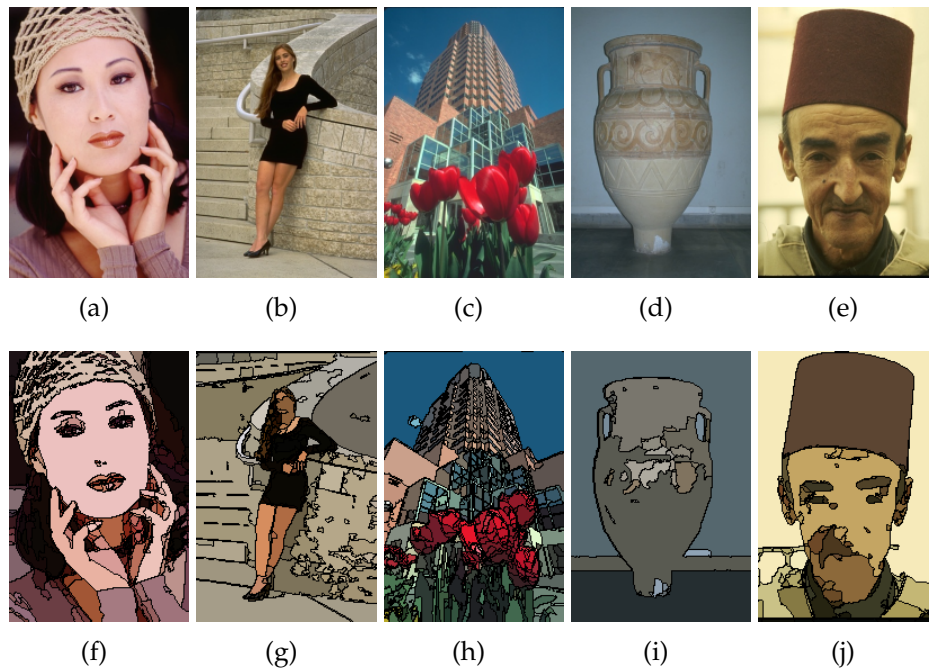


Figure 2.13: Examples of the mean-shift algorithm

clustering error can be further reduced at a small additional cost by varying the discretization parameter.

Analysis

The mean-shift procedure has several favorable characteristics. It is a robust, model-independent region-based segmentation and filtering operation, and can therefore be widely applied. It extends itself naturally to multiple, possibly heterogeneous, feature spaces. For segmentation operations this allows us to use a perceptually uniform color space (CIELAB or CIELUV), or to incorporate texture and discontinuity information. The algorithm can also be used in spatiotemporal feature spaces: it can be used for video segmentation. The mean-shift algorithm is quite general and has been applied to several diverse clustering tasks.

There are, however, some limitations of the mean-shift algorithm. The technique tends to lead to over-segmentation, especially when a small spatial kernel is

used. The use of a small spatial kernel is often desirable for performance reasons: it implies less features to be considered at each mean-shift step. Using even a moderately sized spatial kernel can have a significant performance penalty, as the number of feature points than need to be considered at each step increases proportional to πr^2 , where r is the kernel radius.

Step 2 in the algorithm is quite loosely defined; it involves clustering together feature points (modes) that are closer than a prescribed threshold. It may be useful to replace this step with a region adjacency graph like clustering of the mode points, or perhaps a top-down clustering using the normalized cuts algorithm.

We noted previously that the mean-shift algorithm can be used for video segmentation. This is not as straightforward as it may first appear; several considerations need to be addressed. The most important of these is to determine how exactly to incorporate the temporal dimension into the procedure. A first attempt might be to include it as a third spatial coordinate and cluster in 3-dimensional space. This approach neglects the different nature of the temporal and spatial domain; usually a different temporal and spatial resolution is desirable. This requires a prolate spheroid shaped kernel. Such a kernel not only increases algorithmic complexity, but also requires more computational effort and memory to find and buffer relevant feature points at each step. There has, nevertheless, been some success in segmenting video using the mean-shift algorithm [Wang et al., 2004]; although these algorithms tend to be quite slow.

Despite some limitations, the mean-shift algorithm has many favorable properties and has been applied to a variety of tasks. As an image segmentation algorithm, it is most useful when used as a low-level “first-step” in the overall segmentation process; indeed, this is the stated intention of the algorithm by its authors [Comaniciu and Meer, 2002].

Application-centric classification	
Interaction:	Automatic
Identification:	Region
Media:	Static/temporal
Generality:	Model independent
Algorithm-centric classification	
Perspective:	Homogeneity
Model:	Data driven
Scale:	Local

Table 2.4: Classification of the mean-shift algorithm

Figure 2.13 presents some example images and corresponding segmentations using the mean-shift algorithm. Table 2.4 classifies the algorithm according to our application-centric and algorithm-centric criteria.

2.4.5 Other Approaches

The survey papers by Cheng et al. [Cheng et al., 2001] and by Skarbek and Koschan [Skarbek and Koschan, 1994] provide an extensive overview of the challenges involved in color image segmentation. They also describe several specific image segmentation techniques. The report by Lucchese and Mitra describes various modern and traditional algorithms [Lucchese and Mitra, 2001].

Several traditional image segmentation techniques still enjoy popular use, due to their speed, simplicity, and inclusion in common machine vision and image processing toolkits. Many form the basis for more modern algorithms, and as such, merit a brief discussion here.

The Watershed transform [Vincent and Soille, 1991] is a grayscale morphology based image segmentation technique. It operates using a topological representation of the image gradient. The local minima of the gradient are selected as seed points, and a simulated “flooding” operation produces the image regions such

that the troughs of the topology form the contours of the segmentation. Watershed segmentation is included as part of the MATLABTM image processing toolbox.

Split and merge algorithms [Horowitz and Pavlidis, 1976] have been used as the basis for various modern algorithms. The original algorithm operates in two stages. The first stage is characterized by iteratively merging blocks of pixels in a tree structure until a stopping criteria has been reached. In the second stage, blocks are split if it is deemed appropriate, producing the final segmentation.

The recursive shortest spanning tree (RSST) algorithm, proposed by Morris et al. [Morris et al., 1986], is one of the earliest graph theoretic formulations of the image segmentation problem. In this method, the image is represented by a graph such that the nodes of the graph represent regions (initially pixels) and the links are functions of the dissimilarity between adjacent regions. The algorithm proceeds by recursively finding the shortest spanning tree of the graph, cutting the tree for the most costly links, followed by merging all nodes in the sub-tree and averaging the node values. In practice, this is usually accomplished by recursively merging adjacent nodes with minimal link cost. The RSST, Watershed, and split and merge algorithms can all be recast as region adjacency graph based algorithms.

The reader may wish to pursue other interesting algorithms not explored here. The ratio-cut algorithm [Wang and Siskind, 2003] is another graph-cut/spectral clustering based method, similar to the normalized cuts method we described. There are several image segmentation algorithms based on k-means clustering [Kanungo et al., 2002]: including the k-means with connectivity constraint algorithm, [Kompatsiaris and Strintzis, 1999] and the fuzzy c-means algorithm [Lim and Lee, 1990].

2.5 Interactive Segmentation

Automatic segmentation algorithms suffice for applications that require partitioning of an image into homogeneous regions. Other applications require high-level semantic objects. Automatic segmentation algorithms are, in general, unable to extract semantic objects without some high-level information about the scene. In some restricted domains this high-level information can be provided in the form of a prescribed model. For the more general application, like photo-editing, there is no general model that can be used; we need to obtain high-level knowledge about the scene in some other way.

Interactive segmentation algorithms provide the most obvious solution: the high-level scene knowledge is provided by the user. This high-level knowledge can be provided in many ways: dragging a slider to specify a threshold, drawing a rough object outline, or marking the inside and outside of the object with a mouse. The algorithms then use this knowledge to guide the segmentation, often providing feedback to the user and allowing them to iteratively improve the segmentation.

There are various different algorithms for performing interactive segmentation. Most fall into the following categories:

1. *Thresholding*. This is the simplest form of interactive segmentation. Segmentation by thresholding usually involves selecting a value that separates the object and background pixel classes. This simple form of segmentation is only effective if either all the pixels in the object region have a luminance value greater than the background, or the converse. There are several variations on this basic thresholding scheme: multi-band thresholding allows selecting a threshold for each color channel, adaptive thresholding uses a different threshold value for different parts of the image.

2. *Region Growing*. Region growing techniques are characterized as being initialized with one or more sets of seed pixels, then iteratively expanding these seed pixels to include neighboring pixels according to some predefined criteria, eventually forming regions. Usually two sets of seed pixels are used, one for the object and one for the background.
3. *Classifiers*. Interactive segmentation using classifiers begins by building a model of the known object and background pixels from the user interactions. This is followed by applying statistical or machine learning techniques to classify the remaining pixels in the image.
4. *Graph based*. Graph based interactive segmentation is similar to the region adjacency graph methods used for automatic segmentation in the previous section. A region adjacency graph is first built from the image pixels in the same way as described in Section 2.4.1. This graph is then used to partition the image by incorporating the image data and user interactions in some way.
5. *Deformable models*. Deformable models, also known as active contours or *snakes*, are a technique for delineating regions in an image by outlining the region using a closed curve near the real object boundary. This curve is then evolved toward the true object boundary using an iterative relaxation process. Active contours have been used extensively in medical image segmentation, where they have been shown to perform well.

In the remainder of this section we will investigate four specific algorithms for interactive segmentation. We focus on interactive segmentation techniques appropriate for object extraction from natural scenes. Specifically, we only discuss algorithms whose interactions can be modeled by pictorial input on an image grid [Olabarriaga and Smeulders, 2001]; we do not consider interactive segmentation

algorithms based on parameter tuning or other forms of interaction. This shall allow us evaluate and compare compatible algorithms in Chapter 5.

The first algorithm we discuss is a region growing algorithm: seeded region growing. Following this, we discuss two graph-based algorithms: the interactive graph cuts algorithm, and interactive segmentation using binary partition trees. The final algorithm we discuss is a classifier based method called simple interactive object extraction. It has been integrated into the popular open-source GIMP imaging tool. We do not investigate deformable models; they tend to perform better on medical images and do not lend themselves to iterative updates.

2.5.1 Seeded Region Growing

The seeded region growing algorithm was proposed by Adams and Bischof in [Adams and Bischof, 1994]. It is a simple and computationally inexpensive technique for interactive segmentation of images in which the relevant regions are characterized by connected pixels with similar color values.

Algorithm

The algorithm requires as input a set of seed points that have been grouped into n disjoint sets $S = \{A_j : 1 < j \leq n\}$, where n is the number of desired regions in the segmentation. Usually we select $n = 2$: one set of seeds denotes the object and the other set denotes the background.

At each step in the process, the algorithm chooses a single pixel in the image and appends it to one of the sets A_j according to the following procedure. Let $I = \{\mathbf{x}\}$ be the set of pixels in an image and $f(\mathbf{x})$ be a vector valued function that gives the color (or intensity) of the pixel \mathbf{x} . Also, define $\mathcal{N}(\mathbf{x})$ to be the set of pixels that are neighbors of \mathbf{x} (according to the usual four or eight connectivity constraint).

At each step, a set of candidate pixels T containing all pixels that are neighbors of pixels in S but not contained in S itself are identified. Formally:

$$T = \left\{ \mathbf{y} : \mathbf{y} \notin \bigcup A_j \wedge \mathcal{N}(\mathbf{y}) \cap \bigcup A_j \neq \emptyset \right\}$$

It is necessary to choose a single suitable pixel from the set of candidates $\mathbf{y} \in T$ to add to one of the seed sets $A_j \in S$. The simplest way of doing this is to choose the pixel that is nearest, in some way, to one of the seed sets. This requires a suitable distance function. Adams and Bischof suggest the distance function:

$$\delta(\mathbf{y}, A) = \|f(\mathbf{y}) - \text{mean}_{\mathbf{z} \in A}[f(\mathbf{z})]\|$$

A pixel $\mathbf{y}_i \in T$ is then selected to minimize $\delta(\mathbf{y}_i, A_j)$ for all j and appended to the set A_j . The process is repeated until all pixels in I are contained in one of the sets A_j .

Implementation

To efficiently realize the algorithm we need to be able to quickly find the nearest candidate pixel at each step. Adams and Bischof suggest storing the candidate pixels in a data structure they call a sequentially sorted list. We have found a heap data structure [Cormen et al., 2001] to be more efficient. Denoting this heap H the algorithm is then implemented as follows:

1. Compute set of neighboring pixels T . If $T = \emptyset$ finish.
2. Add pixels from T to H ordered by $\delta(\mathbf{y}, A)$.
3. Add first pixel \mathbf{p} from H to its corresponding seed set A_j .
4. Insert all neighbors of $\mathbf{y} \in \mathcal{N}(\mathbf{p})$ not already in H or S to H .
5. Update the region average for A_j .

Application-centric classification	
Interaction:	Semi-automatic
Identification:	Region/object
Media:	Static
Generality:	Model independent
Algorithm-centric classification	
Perspective:	Homogeneity
Model:	Data driven
Scale:	Local

Table 2.5: Classification of the seeded region growing algorithm

6. If H is not empty, return to step 3.

Note that the above algorithm implies that pixels which are already contained in H are not re-arranged to reflect the new region averages that may result from appending a pixel at the front of H to its corresponding set A_j . The authors claim that this leads to a negligible difference in results, but greatly enhanced execution speed.

Analysis

The seeded region growing algorithm has been successfully applied to a wide range of image segmentation problems, including: medical image segmentation [Olabarriaga and Smeulders, 2001], DNA microarray analysis [Yang et al., 2002], and has been used as a building block for more sophisticated segmentation algorithms such as [Fan et al., 2001]. Table 2.5 illustrates the properties of the algorithm. Note that although seeded region growing is a interactive algorithm, in some cases (e.g. [Yang et al., 2002]) it is possible to incorporate high-level knowledge of the scene to automatically select appropriate image seeds before applying the procedure. For example, if it is known in advance that the required

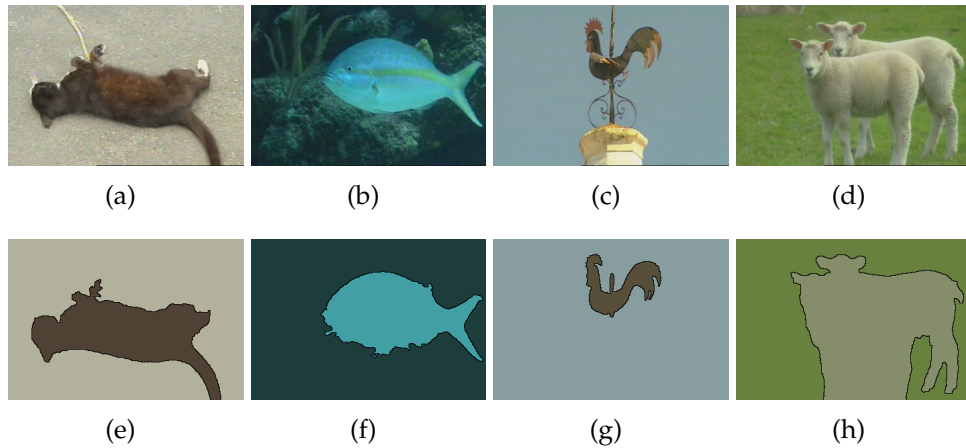


Figure 2.14: Examples output of the seeded region growing algorithm

foreground objects are brighter than the background, then local intensity maxima may be selected as foreground seeds, and minima as background seeds.

The original seeded region growing algorithm used only the grayscale values of the pixels to determine their distances from the seed sets, but it is straightforward to extend this to color. Figure 2.14 depicts several images and their corresponding segmentations obtained using our implementation of the seeded region growing algorithm, which uses the CIELUV color space. In fact, these images were segmented by automatic seed selection: using the local minima and maxima of a saliency map, computed using the Itti saliency model [Itti et al., 1998]. This strategy was successful for these particular images: they are quite simple and have a single salient object. In general we found that it is impossible to automatically and reliably select seeds for natural scenes using saliency maps.

There are several limitations of the algorithm. First, it is only reliable for extracting objects that are reasonably consistent in color. If the average color of the object and background are similar, the method is likely to perform poorly. The algorithm also has difficulty with textured images. Second, the algorithm is raster order dependent [Mehnert and Jackway, 1997]: it is sensitive to the order in which the pixels are processed. By simply altering the orientation of the image,

one may arrive at a different segmentation. [Mehnert and Jackway, 1997] propose several improvements that eliminate this deficiency, at the cost of a more complex algorithm.

2.5.2 Interactive Graph Cuts

The interactive graph cuts algorithm (IGC) is a graph-based interactive segmentation algorithm proposed by Boykov and Jolly in [Boykov and Jolly, 2001]. It formulates the interactive segmentation problem within a MAP-MRF framework [Greig et al., 1989], subsequently determining a globally optimal solution using a fast min-cut/max-flow algorithm. Due to the algorithm’s speed, stability, and strong mathematical foundation, it has become popular and several variants and extensions have been proposed. The “GrabCut” algorithm [Rother et al., 2004] and the “Lazy Snapping” algorithm [Li et al., 2004] are two such variants developed by Microsoft. We discuss the original version of the algorithm here.

Algorithm

The algorithm operates by minimizing a cost function that captures both the hard constraints provided by user interactions, and the soft constraints expressing the relationships between pixels in the spatial and range domains of an image. If $L = \{L_p \mid p \in \mathcal{P}\}$ is an object-background labeling of an image \mathcal{P} (i.e. a segmentation), the energy of the labeling can be expressed as the cost function:

$$E(L) = \sum_{p \in \mathcal{P}} D_p(L_p) + \sum_{(p,q) \in \mathcal{N}} V_{p,q}(L_p, L_q) \quad (2.15)$$

where $D_p()$ is a data penalty function, $V_{p,q}()$ is an interaction potential, and \mathcal{N} is the set of all pairs of neighboring pixels. The data penalty function represents a set of hard constraints that control which pixels belong to the object or background,

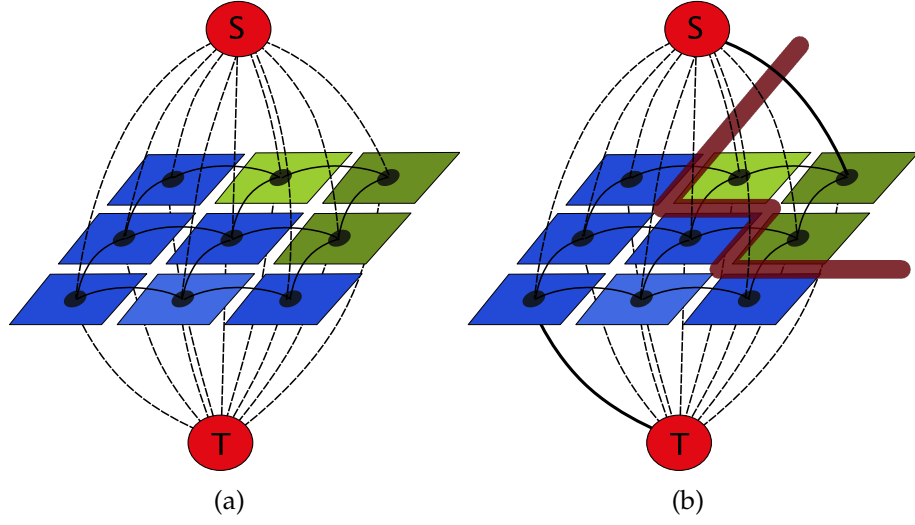


Figure 2.15: An illustration of the graph structure the IGC algorithm uses to minimize Equation (2.15). The nodes in the center are represent the image pixels in a simple 3×3 image. Each of these is connected to its neighbors in the same way as a region adjacency graph. The weights on these connections reflect the similarity between the pixels. Each node is also connected to the terminal nodes: S and T . The weights on these connections reflect the user interactions. In (b) two interactions have been added, illustrated by the solid lines connecting the corner pixels to S and T . The thick line demonstrates the min-cut.

and is derived from the user interactions. The interaction potential is used to encourage spatial coherence between similar neighboring pixels.

To minimize Equation (2.15), the image and user interactions are combined to create a weighted undirected graph. The graph is constructed by adding a node for each pixel in the image, then connecting each node to its neighbor with a weighted edge reflecting the similarity between the pixels (the interaction potential). This weight is usually chosen based on the color or intensity difference between the pixels. Boykov and Jolly suggest the following weighting function:

$$w_{pq} = \frac{1}{\text{dist}(p, q)} \exp \left(-\frac{(I_p - I_q)^2}{2\sigma^2} \right) \quad (2.16)$$

where $\text{dist}(p, q)$ is the spatial distance between nodes p and q , and I_p and I_q are the intensity values for the pixels represented by p and q .

Each node in the graph is also connected to two special terminal nodes using a weighted edge reflecting the user interactions (the data penalty function). These special nodes represent the object and background regions, and are denoted S and T . If a pixel p has been marked as *object* its connection to S is positively weighted with a value K , otherwise it is weighted zero. The weights to T are similarly selected. To ensure the pixels marked as object and background represent hard constraints, the value of K is chosen so that it is larger than the sum of all neighbor weights for any pixel:

$$K = 1 + \max_{p \in \mathcal{P}} \sum_{q \in \mathcal{N}(p)} w_{pq} \quad (2.17)$$

Equation (2.15) can be minimized by finding the min-cut of this graph. Figure 2.15 illustrates the graph and the min-cut for a simple 3×3 image.

Implementation

The min-cut of the graph can be found efficiently using the min-cut/max-flow algorithm described in [Boykov and Kolmogorov, 2004].

Analysis

A classification of the interactive graph cuts algorithm is shown in Table 2.6. Like the normalized cuts algorithm for automatic region segmentation, the interactive graph cuts algorithm optimizes a global criteria. Optimizing a specific function has obvious benefits for mathematically analyzing the algorithm, and in practice the global criteria leads to stable predictable behavior. Users prefer predictable algorithms, as we shall demonstrate in Chapter 5.

Application-centric classification	
Interaction:	Semi-automatic
Identification:	Object
Media:	Static
Generality:	Model independent
Algorithm-centric classification	
Perspective:	Homogeneity
Model:	Data driven
Scale:	Global

Table 2.6: Classification of the interactive graph cuts algorithm

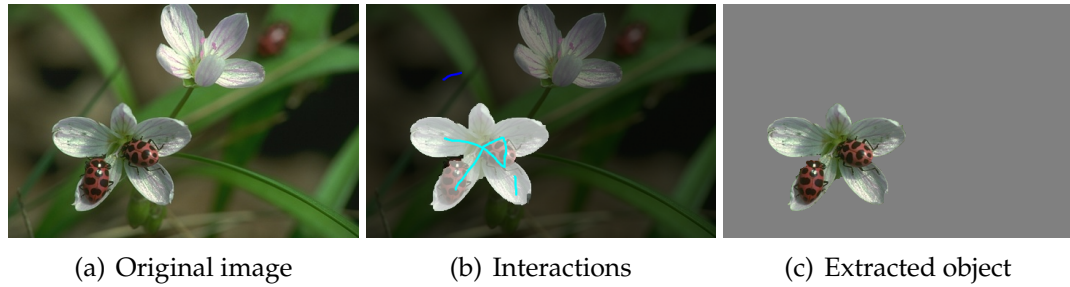


Figure 2.16: Extracting an object using the interactive graph cuts algorithm

The interaction potential in Equation (2.16) is, as Boykov and Jolly admit, ad-hoc. The interactive graph cuts algorithm is general enough to incorporate arbitrary features and similarity measures. Color features may improve performance here, as may texture features. Of course, we could modify the algorithm in many different ways; but we believe such speculation is useless without a formal way of comparing the results to the original algorithm to determine if, indeed, we have improved it.

Figure 2.16 shows an example of the interactive graph cuts algorithm being used to interactively extract an object. A comparative evaluation of the algorithm is presented in Chapter 5.

2.5.3 Interactive Segmentation using Binary Partition Trees

The binary partition tree algorithm is a graph-based interactive segmentation algorithm proposed by Salembier and Garrido in [Salembier and Garrido, 2000], and improved by Adamek in [Adamek, 2006]. The algorithm transforms a hierarchical region segmentation into an object-background segmentation by using the user interactions to split and merge regions in the tree. The algorithm can be adapted to use any automatic segmentation technique that can be tailored to produce hierarchical output in the form of a binary partition tree, in which the root node represents the entire image, and nodes lower down the tree represent regions at increasing levels of detail, with the leaf nodes being the individual image pixels. In [Salembier and Garrido, 2000] the authors used the region adjacency graph method discussed in Section 2.4.1 to create the initial hierarchical segmentation, whereas [Adamek, 2006] used an RSST [Morris et al., 1986] based algorithm. In this section we assume that a hierarchical region segmentation is available, and discuss the procedure to transform it to an object-background segmentation based on user interactions.

Algorithm

A hierarchical segmentation can be represented using a binary partition tree structure. In this tree each node represents a region, and the depth of the node in the tree indicates the level of detail of the region. The root node of the tree is the region containing the entire image, and the leaf nodes of the tree comprise the individual image pixels. Each non-leaf node has two child nodes, splitting its parent region into two sub-regions.

To transform the tree into an object-background segmentation, the algorithm proceeds as follows. In the first stage, the leaf nodes of the tree are assigned labels according to the pixels marked by the user as object and background. The

Application-centric classification	
Interaction:	Semi-automatic
Identification:	Object
Media:	Static
Generality:	Model independent
Algorithm-centric classification	
Perspective:	Homogeneity
Model:	Data driven
Scale:	Local*

Table 2.7: Classification of the binary partition tree algorithm

second stage involves propagating the labels upward toward the root of the tree. Each marked leaf node is propagated toward the root node, labeling each intermediate node with the same label, until a conflict occurs when a parent node has already been labeled differently by the current node’s sibling during a previous propagation stage. In this situation, the parent node is marked as conflicting and the algorithm proceeds to the next leaf node. This is repeated for every marked leaf in the tree. In the third stage of the algorithm, each non-conflicting labeled node is visited, and its label propagated to any unlabeled child nodes in the subtree.

At this stage in the algorithm, certain subtrees may yet remain unlabeled, being judged “too different” with respect to the regions defined by the user markup. The original technique for filling these unlabeled regions contains a flaw (see [Adamek, 2006]). As an alternative approach [Adamek, 2006] proposes labeling each unclassified region with the label of an adjacent but previously classified region. If there are several such regions, the one with the shortest distance is chosen. Adamek suggests using the Euclidean distance between the average colors of the regions in CIELUV space to compute this distance.

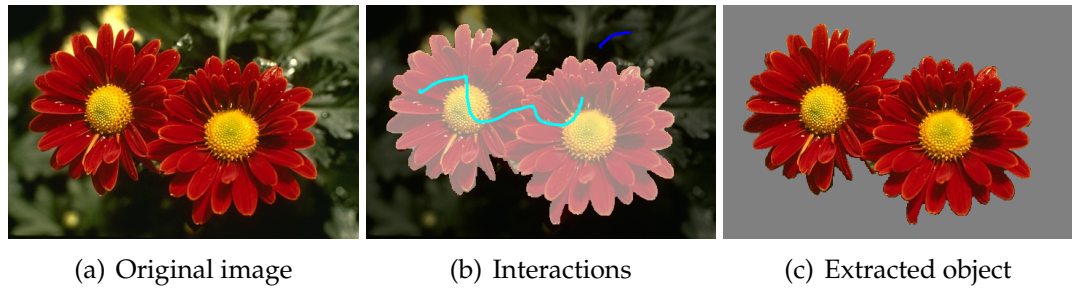


Figure 2.17: Extracting an object using the binary partition tree algorithm

Analysis

Table 2.7 classifies of the binary partition tree algorithm according to our criteria. In our classification we have denoted it as a local scale algorithm only because the implementations that we know of use local scale hierarchical segmentation algorithms as their basis. If, for example, the normalized cuts algorithm was used to create the hierarchical segmentation, then the overall algorithm could be considered a global scale algorithm. In practice, however, the normalized cuts algorithm is too computationally expensive for interactive segmentation.

The performance of the binary partition tree algorithm depends upon the performance of the hierarchical region segmentation algorithm used to implement it. In Chapter 5 we evaluate the implementation of the algorithm proposed in [Adamek, 2006]. Figure 2.17 gives an example of segmenting an image using the binary partition tree method.

2.5.4 Simple Interactive Object Extraction

Unlike the other interactive segmentation algorithms we have discussed, the simple interactive object extraction algorithm [Friedland et al., 2005] is a classifier based algorithm. The idea behind the algorithm is to use the pixels marked by the user to build a color model of the object and background regions, then classify

the pixels in the image as either object or background based on their distance from this model. The algorithm has recently been integrated into the popular open-source imaging program GIMP as the “Foreground Select Tool.”

Algorithm

The algorithm assumes a feature space that correlates well with human perception of color distances with respect to the Euclidean metric. As such, the first step in the method is to transform the image pixels to a perceptually uniform color space. Friedland et al. recommend the CIELAB space [Wyszecki and Stiles, 2000].

Once the image has been transformed into an appropriate color space, the next step is to generate a color signature [Rubner et al., 2000] for the known object and background pixels indicated by the user markup. A color signature is a compact description of the significant modes of a color distribution, represented as a set of cluster centers together with a weight denoting the size of the cluster.

To efficiently generate the color signatures, the modified version of the k-d tree optimization algorithm [Bentley, 1975], described in [Rubner et al., 2000] is used. Assuming known cluster sizes based on the perceived diversity on each color axis, the algorithm proceeds in two stages. In the first stage, given a single starting cluster containing the entire sample, the cluster is recursively partitioned into equal sized clusters until each cluster is within the prescribed cluster sizes. After this, the clusters are recombined by running the same k-d algorithm on the centroids found in the first stage. Clusters containing less than 1% of the pixels are then discarded. The final cluster centroids constitute the color signatures for the object and background regions.

Using the generated color signatures, the unknown image pixels are then classified as foreground or background according to the minimum distance to any mode in the foreground or background color signatures. The result is a confidence matrix, consisting of a value between zero and one, zero denoting background,

Application-centric classification	
Interaction:	Semi-automatic
Identification:	Object
Media:	Static
Generality:	Model independent
Algorithm-centric classification	
Perspective:	Homogeneity
Model:	Data driven
Scale:	Global

Table 2.8: Classification of the simple interactive object extraction algorithm

one denoting foreground. In the final stage of the algorithm, the confidence matrix is smoothed and regions disconnected from the largest object are removed.

Analysis

The algorithm has several advantages. It is simple and computationally inexpensive, allowing for a fast implementation. This is essential for an interactive algorithm. The reference implementation of the algorithm is written in Java and is responsive enough for most applications. The algorithm also uses a perceptually uniform color space; it is more likely to group pixels that are perceptually similar in color than an algorithm that only uses luminance, for instance.

The algorithm also has some limitations. It operates by building a color signature of the object and background regions from the known pixels, then labeling the unknown pixels based on their distances from these signatures. This strategy implies that the algorithm will only succeed for images in which the object and background regions have sufficiently different color signatures. When the object and background *do* have sufficiently different color signatures, the algorithm will succeed if enough pixels have been marked for it to recreate appropriate signatures from the user interactions.

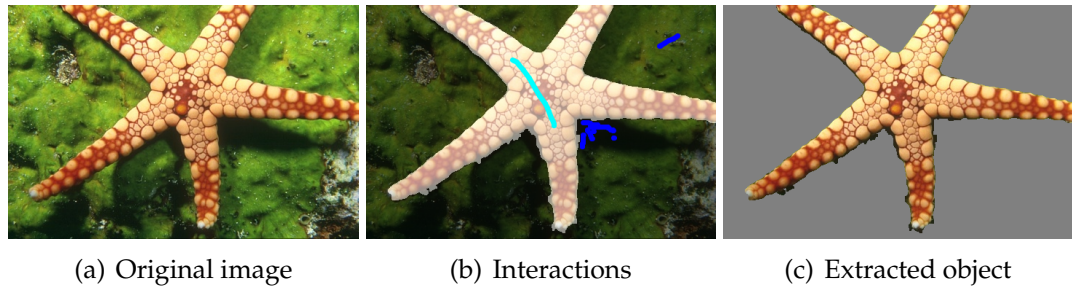


Figure 2.18: Extracting an object using the simple interactive object extraction algorithm

In practice, we have observed that if the object and background can be differentiated effectively using color signatures, then a few interactions are usually sufficient. Put another way, if the algorithm is indeed capable of separating the object and background for a particular image, it does not require many interactions. The corollary is that if the algorithm is unable to separate the object and background after the first few interactions, additional interactions will not improve the segmentation.

The above characteristic is interpreted by users as unresponsiveness and can be very frustrating (see Chapter 5). Incorporating the spatial locations of the marked pixels into the region signatures in some way could potentially resolve this deficiency.

A classification of the algorithm according to our criteria is shown in Table 2.8. Figure 2.18 demonstrates extracting an object with the simple interactive object extraction algorithm.

2.5.5 Other Approaches

The above discussion focused on the most popular methods used for interactive segmentation of natural images. There are some other well known algorithms for interactive segmentation that we have not discussed.

Algorithms based on active contours and other deformable models (for example: [Kass et al., 1988], and [Caselles et al., 1995]) are especially popular in medical image segmentation. We have found them to be less useful for natural image segmentation and photo-editing applications for two reasons. First, they are quite sensitive to initial parameterization. These parameters are, in general, harder to specify for natural images: the domain is broader. Second, they require interactions in the form of an object outline. This makes the segmentation more difficult to refine by adding more interactions.

Another class of algorithms popular in medical image segmentation are the “live-wire” algorithms [Falcão et al., 1998, Falcão et al., 2000]. These algorithms are designed to assist the user outlining an object by “snapping” the line segment currently being drawn to the object boundary. In essence, the user selects the end points of a boundary segment, and the algorithm finds the shortest path between these end points in a graph. The graph is constructed so that its nodes lie between the individual image pixels, and its edges are weighted using the image gradient. The shortest path connects the end points, and tends to lie along a gradient discontinuity. It is not necessary to create closed contours when using the live-wire algorithms, and as such, it may be more appropriate to consider them as assisted drawing methods, rather than interactive segmentation algorithms.

A few other methods for interactive segmentation of natural images have very recently been proposed. [Ning et al., 2009] propose an algorithm that uses the user interactions to merge regions in an initial region based segmentation. The algorithm is similar to the binary partition tree algorithm: it uses an automatic region segmentation and merges regions based on the user interactions. It differs in that it does require a hierarchical segmentation; an initial fine-grained segmentation is used instead. Ning et al. suggest using the mean-shift algorithm for this purpose (see Section 2.4.4). Mean-shift is ideally suited as it tends to over-segment objects. Also recently, [Protiere and Sapiro, 2007] propose a lin-

ear time algorithm for interactive segmentation of natural images that is based on adaptive weighted distances. Like the simple interactive object extraction algorithm, this is a classifier based algorithm. Unlike simple interactive object extraction, it uses adaptive texture features to model the object and background regions. Neither [Ning et al., 2009] nor [Protiere and Sapiro, 2007] evaluate their proposed approaches against the state-of-the-art.

2.6 Discussion

In this chapter we discussed and characterized eight state-of-the-art algorithms for image segmentation: four automatic algorithms, and four interactive algorithms. The algorithms were selected so as to be representative of the different classes of approaches in the literature. It is clear from the discussion that there are many different algorithms available, and that although there has been a lot of progress in image segmentation, the problem is far from solved.

We discussed the strengths and limitations of each of the selected algorithms. We also suggested some ways in which these algorithms could potentially be improved. There is potential for improving the efficiency of the normalized cuts algorithm by combining it with a bottom-up segmentation algorithm like mean-shift. The interactive graph cuts algorithm might be improved by integrating color and texture features. The simple interactive object extraction algorithm may be more responsive if the spatial location of the known pixels was integrated into the color signatures.

These enhancements may produce more effective algorithms. Then again, they may not. It is prudent, therefore, before implementing any such enhancements to have a reasonable way to compare segmentation results within the existing state-of-the-art. Indeed, we believe that it is key to progress. There are many algorithms for image segmentation; how are we to select the best one for a

particular application? New algorithms are continuously proposed; how are we to know if they actually constitute an improvement? Anecdotal examples are insufficient. What is needed is a way to formally evaluate and compare image segmentation algorithms. This is the subject of our next chapter.

Chapter 3

Segmentation Evaluation: A Review

3.1 Introduction

Segmentation evaluation research is concerned with the development of tools and techniques that allow us to measure and compare the performance of segmentation algorithms. What exactly is meant by *performance* ultimately depends upon the application. In one application, a segmentation algorithm might be considered to perform well if it closely mimics human perceptual grouping; in another, computational efficiency or stability may be more important. Segmentation evaluation techniques give us tools to measure and compare the characteristics of segmentation algorithms, and thereby gauge their performance.

It is clear from the previous chapter that there exist many different algorithms for image segmentation. It is important to be able to evaluate and compare these algorithms. Evaluation is important not only for application developers, who need to select the correct tool for the job, it is also important for researchers. Being able to accurately gauge the performance of an algorithm gives insight into what constitutes a good algorithm. It allows researchers to improve and justify new methods via formal comparison with existing methods.

Segmentation evaluation has recently been receiving more attention from researchers [Jiang et al., 2006, Ge et al., 2007, Zhang et al., 2008]. Nevertheless, it is still an emergent research area, and has received significantly less attention than image segmentation itself.

In the remainder of this chapter we review and characterize the current state-of-the-art in segmentation evaluation. Our first task is to develop a scheme to classify segmentation evaluation techniques; this scheme will allow us to characterize the existing evaluation techniques and identify areas that still need to be addressed. Following this, we review various published techniques for performing segmentation evaluation, and classify them according to our scheme. The chapter concludes with a discussion, where we identify the limitations of the state-of-the-art and the areas that need improvement.

3.2 Taxonomy

In this section we look at taxonomies for classifying segmentation evaluation techniques. A few such taxonomies have previously been proposed [Zhang et al., 2008, Jiang et al., 2006, Zhang, 1996]. Most of these classify algorithms based either on what is being evaluated, or on how the evaluation is performed. These taxonomies agree to a large extent conceptually, but differ in the terminology used.

Zhang [Zhang, 1996] classifies segmentation evaluation techniques into three groups: *analytical* methods, *empirical goodness* methods, and *empirical discrepancy* methods. By analytical methods, Zhang means methods that directly assess the quality of a segmentation algorithm by analyzing its principles and mathematical properties. Analytical evaluations are often given by the authors of a segmentation algorithm in the form of a proof that the algorithm optimizes some criteria. Analytical assessment is not, in general, based on properties of the output of

an algorithm, but on the mathematical properties of the algorithm itself. It is impossible to automate this kind of assessment.

The second group of methods identified by Zhang are the empirical methods. These are computed based on the output of a segmentation algorithm. Empirical discrepancy methods refer to techniques that compute a measure of agreement between the output of a segmentation algorithm and an existing reference segmentation. This reference segmentation is generally referred to as a *ground truth*. Empirical goodness methods do not require a ground truth; they measure an algorithm's performance by examining the output for certain properties that are assumed to be desirable.

Jiang et al. [Jiang et al., 2006] propose a different classification. They first divide segmentation evaluation methods into two categories: *theoretical evaluation*, and *experimental evaluation*. Theoretical evaluation methods generally correspond to the analytical methods in Zhang's classification, whereas Zhang's empirical methods fall under experimental evaluation. The experimental evaluation methods are subdivided into *task-based* and *feature-based* methods. Task-based evaluation refers to methods that evaluate image segmentation in the context of a particular application. Feature-based evaluation is further subdivided into *ground truth-based* and *non-ground truth-based* methods, which roughly correspond to Zhang's empirical discrepancy and empirical goodness methods.

Other authors have given similar taxonomies, again using different terminology. Yang et al. [Yang et al., 1995], Chabrier et al. [Chabrier et al., 2006], and Zhang et al. [Zhang et al., 2008] all use the term *supervised* for approaches that use ground truth, and *unsupervised* for those that do not. Correia and Pereira [Correia and Pereira, 2003] refer to the same methods as *standalone* and *relative*.

The existing classification schemes have some limitations. The scheme due to Jiang et al. assumes that task-based algorithms never use ground truth. The term "feature-based" is not well defined. The authors appear to be trying to categorize

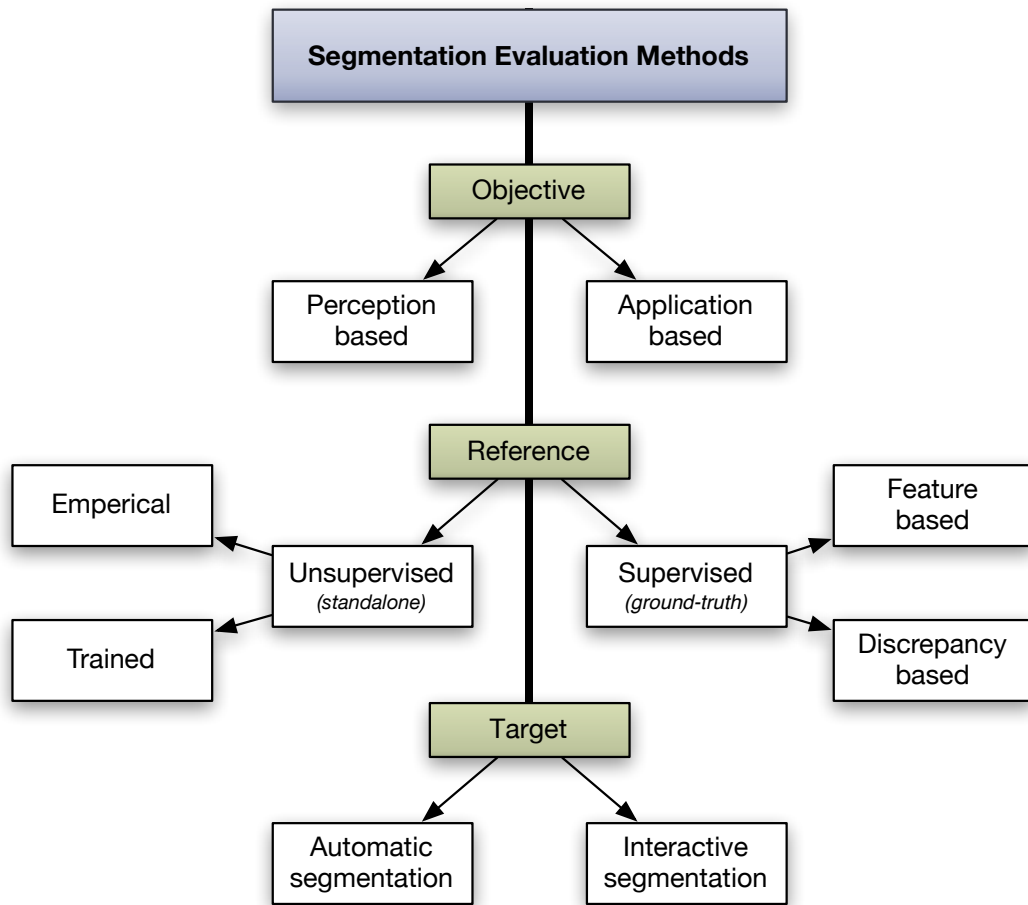


Figure 3.1: Taxonomy of segmentation evaluation algorithms

algorithms based on their objective. In this way, *task-based* algorithms evaluate a segmentation algorithm's suitability for a particular application. The objective of *feature-based* algorithms is presumably to evaluate how well a segmentation algorithm emulates human perceptual grouping, though this is not explicitly stated in their paper. The scheme of Zhang does not address the objective of a segmentation evaluation technique whatsoever. Finally, none of the existing taxonomies make a distinction between techniques designed to evaluate automatic segmentation algorithms, and those designed to evaluate interactive segmentation algorithms.

To unify the existing taxonomies and address their limitations, we propose the taxonomy shown in Figure 3.1. Under our new taxonomy, techniques for evaluating segmentation algorithms are classified under three headings: *objective*, *reference*, and *target*.

The term *objective* is used to specify what the segmentation evaluation technique aims to evaluate. Under this heading we can classify evaluation techniques as either *perception-based* or *application-based*. Perception-based techniques are designed to assess how well a segmentation algorithm approximates human perceptual grouping. Application-based techniques are designed to assess the performance and suitability of a segmentation algorithm in a particular application domain. These two categories are assumed to map to the feature-based and task-based categories in Jiang et al.’s scheme.

Our second heading, *reference*, characterizes the segmentation evaluation technique’s use of ground truth data. Under this heading, assessment strategies that explicitly reference one or more ground truth segmentations for each image used in the evaluation are classed as *supervised*. Assessment methods that do not require a ground truth are classed as *unsupervised*. Since we consider ground truth use under a different heading, the implicit assumption in Jiang et al.’s scheme, that application-based algorithms never make use of ground truth data, is dropped.

The above classification allows for unsupervised evaluation techniques to have undergone a training phase. Unsupervised evaluation techniques are therefore subdivided into *trained* and *empirical*. Trained evaluation techniques aim to automatically learn what properties make a segmentation result good from a set of training data. Purely empirical evaluation techniques specify explicitly how to determine a segmentation algorithm’s performance from its output; they do not require a training phase.

Supervised evaluation techniques are divided into *feature-based* and *discrepancy-based* methods. Discrepancy-based methods specify how to directly compute a

measure of disparity between the segmented image and its corresponding ground truth. Discrepancy-based methods are often defined using set theoretic operations over the regions in the machine segmentation and the ground truth. Feature-based methods compute some agglomerative features of the machine segmentation and compare these with similar features computed from the corresponding ground truth.

In our scheme, the empirical-goodness techniques described by Zhang correspond to the unsupervised empirical methods. The empirical-discrepancy techniques correspond to the supervised discrepancy methods.

The final heading in our scheme is *target*; it refers to the kind of segmentation algorithm the evaluation technique is designed to evaluate. We specify two types of target: automatic segmentation and interactive segmentation. Their distinction is important because techniques for evaluating automatic segmentation algorithms cannot be directly applied to evaluating interactive segmentation algorithms; interactive segmentation algorithms depend on human intervention, and accuracy must be measured over time. We discuss this in more detail in Chapter 5.

Our classification scheme is different from existing schemes in three important ways. First, we do not include analytical methods in our scheme: it is impossible to automatically evaluate segmentation algorithms analytically. Second, unsupervised methods are explicitly allowed a training phase. This implies an important difference, from an application perspective, between supervised and unsupervised methods: unsupervised techniques (even those that require training) can be used to automatically find the optimal parameterization of a segmentation algorithm for a particular image, whereas supervised techniques cannot.

The final difference is in considering the kind of segmentation algorithm the evaluation technique is designed to evaluate. To our knowledge, no existing taxonomy distinguishes between techniques designed to evaluate automatic

segmentation algorithms and techniques designed to evaluate interactive segmentation algorithms. This lack of distinction is understandable, however, since techniques for evaluating interactive segmentation algorithms had not yet been considered at the time these taxonomies were proposed.

3.3 Supervised Evaluation

This section examines three popular methods for supervised evaluation of automatic segmentation algorithms. Each of the examined techniques is discrepancy-based; we do not know of any feature-based methods for supervised evaluation. The described techniques are usually used to evaluate how well a segmentation algorithm imitates human perceptual grouping; they are used in conjunction with a ground truth dataset of human segmented images. When used in this way, all the algorithms we examine can be considered *perception-based*. By changing the dataset used, however, the same techniques could potentially be used to evaluate an algorithm for a particular application (i.e., be used for *application-based* evaluation).

The first technique we discuss is based on considering the image segmentation problem as a general clustering problem. Reinterpreting the evaluation problem from this perspective makes available a host of well established measures that were originally proposed for comparing pairs of clusterings. In this way, segmentation error can be gauged by measuring the error between the clustering represented by the ground truth and the clustering represented by the machine segmentation.

The next method we discuss is more specifically tailored toward evaluating perceptual grouping. The authors of this method observe that different people tend to produce segmentations that are consistent overall, although they may be at a different level of detail. To address this ambiguity, they propose two measures

to quantify the error between a pair of segmentations, while remaining tolerant of this kind of refinement.

The next measure we discuss is sensitive to refinement, and is therefore better at measuring the degree of under or over segmentation; measures that are allow refinement are, by definition, insensitive to such error. In some applications, error due to under or over segmentation can affect the performance of a system as a whole, and therefore need to be appropriately measured. For example, a system whose overall computational complexity is a function of the number of regions produced in the segmentation will clearly be slower if it uses a segmentation algorithm that is prone to over segmentation. On the other hand, an object recognition system may be able to handle a certain degree of over segmentation by combining regions at a later stage in the processing, but may not be able to handle two important objects being merged due to under segmentation. The evaluation measure should reflect, as well as possible, the requirements of the application; when the target application is sensitive to refinement error, it is prudent to use an evaluation measure that is responsive to such error.

The final subsection briefly outlines a few of the other supervised segmentation evaluation techniques that have been proposed, which are not discussed in this section, and provides references to the relevant literature.

3.3.1 Evaluation Metrics Based on Clustering

Jiang et al. [Jiang et al., 2005, Jiang et al., 2006] consider image segmentation to be a data clustering problem. This perspective opens the door to a host of established measures developed in the statistics and the machine learning communities. Jiang et al. propose a suite of measures, originally developed to compare data clusterings, for use in comparing different segmentations of the same image. They also introduce a new distance measure based on bipartite graph matching.

Theory

Clustering is defined as the process of partitioning of a set of objects $O = \{o_1, \dots, o_n\}$ into a set of k disjoint subsets $C = \{c_1, \dots, c_k\}$ called clusters. If the set of objects are pixels in an image, a clustering C may be interpreted as being analogous to a segmentation of O . Evaluating the performance of a segmentation algorithm against a ground truth can be thought of as comparing two separate clusterings $C_1 = \{c_1, \dots, c_k\}$ and $C_2 = \{c_1, \dots, c_l\}$ of the same scene. There exist several well known distance measures that can be used to compare two such clusterings.

One set of distance measures used in clustering is based upon counting the number of pairs of objects (o_i, o_j) that lie in the same or different clusters under C_1 and C_2 . Every pair of objects (o_i, o_j) falls into one of the following categories:

- $T_{11} = \{(o_i, o_j) : o_i, o_j \text{ are in the same cluster both in } C_1 \text{ and } C_2\}$
- $T_{10} = \{(o_i, o_j) : o_i, o_j \text{ are in the same cluster in } C_1 \text{ but not in } C_2\}$
- $T_{01} = \{(o_i, o_j) : o_i, o_j \text{ are in the same cluster in } C_2 \text{ but not in } C_1\}$
- $T_{00} = \{(o_i, o_j) : o_i, o_j \text{ are in different clusters in both } C_1 \text{ and } C_2\}$

Denoting $n_{xx} = |T_{xx}|$ the number of items in each of the above sets, clearly $n_{00} + n_{10} + n_{01} + n_{11} = n(n-1)/2$ holds, where n is the number of objects (pixels).

Using these sets, the following distance measures can then be defined:

$$\mathcal{R}(C_1, C_2) = 1 - \frac{n_{11} + n_{00}}{n(n-1)/2} \quad (3.1)$$

$$\mathcal{J}(C_1, C_2) = 1 - \frac{n_{11}}{n_{11} + n_{01} + n_{10}} \quad (3.2)$$

$$\mathcal{F}(C_1, C_2) = 1 - \sqrt{W_1(C_1, C_2)W_2(C_1, C_2)} \quad (3.3)$$

where

$$W_1(C_1, C_2) = \sum_{i=1}^k \frac{n_{11}}{|c_i|(|c_i| - 1)/2} \quad (3.4)$$

$$W_2(C_1, C_2) = \sum_{j=1}^l \frac{n_{11}}{|c_j|(|c_j| - 1)/2} \quad (3.5)$$

The three measures are known as the *Rand*, *Jaccard*, and *Fowlkes and Mallows* indices [Rand, 1971, Ben-Hur et al., 2002, Fowlkes and Mallows, 1983]. Each is a measure of segmentation error and lies in the range $[0, 1]$, where a value of zero indicates identical clusterings, and larger values indicate larger error.

Another way of measuring the similarity of two clusterings is by determining how much information is shared between them. One method of computing this is to employ a concept from information theory known as mutual information. Consider the clusterings C_1 and C_2 as random variables that can take on the discrete values $c_i \in C_1$ and $c_j \in C_2$. The probability that a given pixel $o_x \in c_i$ is given by the marginal distribution function $p(c_i)$, and the probability $o_x \in c_j$ is given by $p(c_j)$. The probability that a pixel is in both c_i and c_j is given by the joint distribution function $p(c_i, c_j)$. The mutual information for two clusterings is then given by:

$$\text{MI}(C_1, C_2) = \sum_{c_i \in C_1} \sum_{c_j \in C_2} p(c_i, c_j) \log \frac{p(c_i, c_j)}{p(c_i)p(c_j)} \quad (3.6)$$

To use mutual information as a performance indicator, two strategies have been proposed for its normalization. The first, known as normalized mutual information, is due to [Strehl et al., 2000], and is given by:

$$\mathcal{NMI}(C_1, C_2) = 1 - \frac{1}{\log(k \times l)} \text{MI}(C_1, C_2) \quad (3.7)$$

where k and l are the number of clusters in C_1 and C_2 . The second, due to Meila [Meila, 2003], is known as variation of information:

$$\mathcal{VI}(C_1, C_2) = H(C_1) + H(C_2) - 2 \text{MI}(C_1, C_2) \quad (3.8)$$

where $H(C)$ is the entropy of C , $H(C) = -\sum_{c \in C} p(c) \log p(c)$. The \mathcal{NMI} measure is in the range $[0, 1]$; the \mathcal{VI} is bounded by $\log n$ where n is the number of data points, and is also a metric.

Performance indicators can also be derived based on measuring the intersection of sets from different clusterings. Van Dongen [van Dongen, 2000] proposes the following index:

$$\mathcal{D}(C_1, C_2) = 2n - D_H(C_1 \Rightarrow C_2) - D_H(C_2 \Rightarrow C_1) \quad (3.9)$$

where $D_H(X \Rightarrow Y)$ is the Hamming distance. This measure is closely related to the Huang-Dom measure described in Section 3.3.3. In fact, the Van Dongen index is a simple linear transformation of the Huang-Dom (HDI) index, given by: $2n(1 - HDI)$.

Jiang et al. also propose a new index that uses bipartite graph matching to obtain the best match for each region. Given two segmentations $C_1 = \{c_i\}$, and $C_2 = \{c_j\}$, a weighted undirected bipartite graph $G = (V, E)$ can be constructed by adding an edge, weighted by $w_{ij} = |c_i \cap c_j|$, for all $w_{ij} \neq 0$. A *maximum weight bipartite graph match* is a sub-graph G' of G constructed by removing edges, such that each of the nodes c_i and c_j has at most one incident edge, and $\sum w_{ij}$ is maximized. Such a graph can be found using a bipartite graph matching algorithm. Given this graph, the new measure is defined as:

$$\mathcal{BGM}(C_1, C_2) = 1 - \frac{\sum w_{ij}}{n} \quad (3.10)$$

Implementation

The error measures outlined in the previous section can be efficiently computed using a matrix known as the *confusion*, or *association* matrix. Given two segmentations S_1 having n regions, and S_2 having m regions, then the confusion matrix M is an $n \times m$ matrix such that M_{ij} is equal to the area of intersection of region i from S_1 and region j from S_2 . Assuming that the regions in each segmentation are labeled consecutively, then M can be computed efficiently by iterating over each pixel in the segmentation masks, and incrementing M_{ij} using the corresponding region labels i and j .

For the *Rand*, *Jaccard*, and *Fowlkes and Mallows* indices, the numbers n_{xx} can be determined from a confusion matrix as follows:

$$n_{11} = \frac{1}{2} \left[\sum_{i=1}^k \sum_{j=i}^l M_{ij}^2 - n \right] \quad (3.11)$$

$$n_{10} = \frac{1}{2} \left[\sum_{i=1}^k |c_i|^2 - \sum_{i=1}^k \sum_{j=i}^l M_{ij}^2 \right] \quad (3.12)$$

$$n_{01} = \frac{1}{2} \left[\sum_{j=1}^l |c_j|^2 - \sum_{i=1}^k \sum_{j=i}^l M_{ij}^2 \right] \quad (3.13)$$

$$n_{00} = \frac{n(n-1)}{2} - n_{11} - n_{10} - n_{01} \quad (3.14)$$

When computing the \mathcal{BGM} measure, the weights w_{ij} are given directly by M_{ij} . The bipartite graph matching problem can be solved efficiently in polynomial time using the *Kuhn-Munkres* algorithm [Kuhn, 1955] (also known as the *Hungarian Method*).

As the Van Dongen index can be computed directly from the Huang-Dom measure (discussed later), only the mutual information based measures are considered here. All that is needed is the joint and marginal probability distribution

	\mathcal{R}	\mathcal{F}	\mathcal{J}	\mathcal{D}	\mathcal{BGM}	\mathcal{NMI}	\mathcal{VI}
E_{same} (Accurate)	0.117	0.197	0.317	0.123	0.215	0.772	1.114
E_{diff} (Inaccurate)	0.378	0.622	0.792	0.446	0.645	0.943	3.424

Table 3.1: Indicative values for accurate and inaccurate segmentation. E_{same} is the mean error found by comparing different segmentations of the same image. E_{diff} is the mean error found by comparing different segmentations of different images.

functions. Again, these can be computed directly from the confusion matrix:

$$p(c_i, c_j) = M_{ij}/n \quad (3.15)$$

$$p(c_i) = |c_i|/n \quad (3.16)$$

$$p(c_j) = |c_j|/n \quad (3.17)$$

Analysis

All of the measures discussed, except \mathcal{VI} , are in the range $[0, 1]$, and are defined as error measures, i.e., values closer to one are inferior. \mathcal{VI} is also an error measure, but is bounded by $\log n$. It is instructive to investigate typical values for accurate segmentation versus typical values for inaccurate segmentation. Jiang et al. present a table of these values that they computed experimentally. To find typical values for accurate segmentation, they compare multiple segmentations of the same image by different people using each of their error measures. To find typical values for inaccurate segmentation, multiple segmentations from different images are compared. Table 3.1 summarizes their findings.

The table indicates that mean error for accurate segmentation (pairs of segmentations of the same scene by different subjects) is lower than the mean error for inaccurate segmentation (pairs of segmentations of different scenes). Such a result suggests that the measures can indeed be used to distinguish between accurate

and inaccurate segmentation, at least in the average case, and may therefore be useful for evaluating image segmentation algorithms. To draw more definitive conclusions requires examining the distribution of the error values. We examine the distributions of several supervised segmentation evaluation measures in Chapter 4.

An important question to consider is what measures should be used for a particular evaluation task. This question is, of course, task-dependent, and measures should be selected that have characteristics favorable for a given application. In most cases, several aspects of a segmentation algorithm's performance must be considered, both for perception-based and application-based evaluation. Jiang et al. suggest a linear combination using a selection of the measures to form an overall performance indicator. There may, however, be significant correlation between some of the measures, and care should be taken to minimize redundancy, especially if the measures are being used by machine learning algorithms to find an optimal parameterization. An appropriate weighting scheme for the linear combination also warrants investigation.

All of the indices discussed are supervised discrepancy-based measures. They may be used for either application- or perception-based evaluation. Care should be taken when using the measures for evaluating perceptual grouping: they are general measures for comparing two clusterings; they have not been designed specifically to address what makes a segmentation "good" in a perceptual sense. Gestalt principles like proximity and closure are not directly assessed by these measures; a segmentation that appears inaccurate may score higher than expected.

3.3.2 Local and Global Consistency Error

Martin et al. propose two supervised discrepancy based performance measures, designed to measure how accurately a segmentation algorithm emulates human

perceptual grouping [Martin et al., 2001]. These error measures are called local consistency error and global consistency error. They are designed so that when comparing two different segmentations, if one is a refinement of the other, the error value should be very small, or even zero. By refinement, they mean that the segmentations are consistent, but one segmentation has a higher level of detail than the other. The justification is that although humans do not, in general, produce identical segmentations of the same scene, often these segmentations differ only in the level of granularity the person decided to represent; they do not imply a different perceptual organization of the scene. Martin et al. validate their proposed measures by showing that the error values found when comparing different human-generated segmentations of the same scene are lower, on average, than the error values found when comparing human and machine-generated segmentations.

Theory

The technique gives two such measures of segmentation error based on a definition of local refinement error. If $R(S, p_i)$ corresponds to the set of pixels (region) containing pixel p_i , then the local refinement error is defined as:

$$E(S_1, S_2, p_i) = \frac{|R(S_1, p_i) \setminus R(S_2, p_i)|}{|R(S_1, p_i)|} \quad (3.18)$$

Observe that Equation (3.18) is zero only when S_1 is a proper subset of S_2 at p_i , indicating no refinement error. This measure is not symmetric, so a simple summation of Equation (3.18) over all pixels is insufficient. To rectify this asymmetry, the global consistency error (GCE) and local (LCE) consistency error are defined as:

$$\text{GCE} = \frac{1}{A} \min \left\{ \sum_{\text{all pixels } p_i} E(S_1, S_2, p_i), \sum_{\text{all pixels } p_i} E(S_2, S_1, p_i) \right\} \quad (3.19)$$

$$\text{LCE} = \frac{1}{A} \sum_{\text{all pixels } p_i} \min\{E(S_1, S_2, p_i), E(S_1, S_2, p_i)\} \quad (3.20)$$

where A denotes the area of the segmentation masks, in pixels.

Both measures are in the range $[0, 1]$, where values closer to zero denote a better segmentation. Note that $\text{LCE} \leq \text{GCE}$ for any two segmentations, and that LCE is tolerant of mutual refinement, whereas GCE is not.

Implementation

Both error measures can be efficiently computed from the confusion matrix (see Section 3.3.1) as follows. Given a confusion matrix M , the areas of the regions R_i from S_1 , and R_j from S_2 are given by:

$$|R_i| = \sum_j M_{ij} \quad (3.21)$$

$$|R_j| = \sum_i M_{ij} \quad (3.22)$$

The confusion matrix M and the above region areas can be used to compute the local refinement error for every pixel (Equation (3.18)); determining the local and global consistency error from this is trivial.

Analysis

It is important to be able to interpret the local and global consistency errors if they are to be useful performance indicators. Ideally, we would like to know the range of values that imply a perceptually accurate segmentation, and the range of values that imply an inaccurate segmentation.

One way to determine the values that indicate accurate segmentation is to measure segmentation error between pairs of segmentations of the same scene created by different human participants (see Figure 3.2). The average of these values is an indicator of *best-case* performance values; if a segmentation algorithm achieves

this level of accuracy, it is effectively emulating human perceptual grouping, at least as far as we can measure with local and global consistency. Experimental evaluation by Martin et al. shows that the average error between pairs of segmentations of the same scene created by different people is 11% for global consistency error, and 7% for local consistency error. Ignoring degenerate cases (discussed at the end of this section), an automatic segmentation algorithm that can achieve values equal to or better than these can be considered perceptually accurate. Note that our own experiments found the slightly different values of 8% and 5%; Jiang et al. found similar values in [Jiang et al., 2006]. We further investigate the reason for the discrepancy in Chapter 4.

We would also like to know the range of values that imply inaccurate segmentation. A reasonable way to determine this range is to measure the average segmentation error between segmentations of different scenes, since different scenes, in general, imply different perceptual groupings. The average error between segmentations of different scenes gives us an indicator of *worst-case* performance values. We found average error values to be 37% for global consistency error, and 29% for local consistency error. Martin et al. found similar values: 39% and 30%. An automatic segmentation algorithm that gives error values equal to or greater than these must be considered perceptually inaccurate.

The above discussion begs the question: with respect to the local and global consistency error, how well do state-of-the-art segmentation algorithms approximate human perceptual grouping? We measured the mean error for several algorithms and found values between 16% and 36% for global consistency error, and between 11% and 27% for local consistency error. The figures demonstrate that segmentation algorithms do indeed perform better than random, but have yet to approach the 8% and 5% accuracies measured for humans. The results can also be interpreted as validation that the local and global consistency error

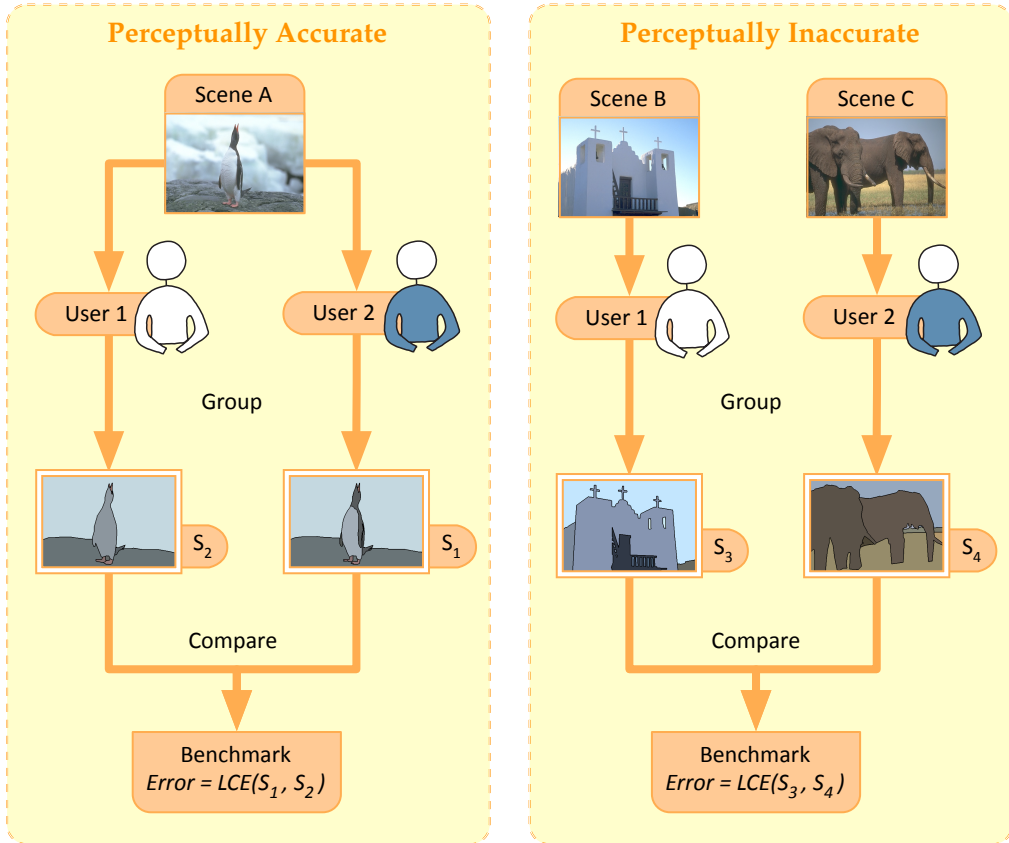


Figure 3.2: Measuring error-rates values for accurate and inaccurate segmentation. The left panel shows two segmentations of the same scene by different users. These are assumed to be mutually accurate. Comparing many such pairs gives a baseline for *accurate* segmentation. The right panel shows two segmentations of different scenes. These are assumed to be mutually inaccurate. Comparing many such pairs gives a baseline for *inaccurate* segmentation. Chapter 4 discusses this technique in more detail.

values are useful for measuring perceptual segmentation error. We discuss these experiments further in Chapter 4.

The local and global consistency error benchmarks should be interpreted with care. Both measures are tolerant of refinement, implying two degenerate cases:

1. a segmentation containing a region for each pixel is a refinement of every segmentation, and
2. every segmentation is a refinement of a segmentation containing a single region.

Both cases achieve zero error. As such, the error measures are suited only to comparing segmentations with a similar number of regions. Also note that local and global consistency error are often highly correlated. This correlation is especially significant if one intends to use these measures for automatic training or parameterization of a segmentation algorithm.

3.3.3 The Huang-Dom Evaluation Measure

The local and global consistency error measures are designed to be tolerant of refinement. Although this tolerance may be desirable in some circumstances, especially in the context of perceptual evaluation, sometimes we do not want to ignore refinement error. Huang and Dom [Huang and Dom, 1995] propose a performance indicator based on directional Hamming distance that is not tolerant of refinement; it is therefore more suitable for evaluation in applications when the degree of under or over-segmentation are important.

Theory

Huang and Dom's performance indicator, which we denote HD , is designed to gauge the accuracy of a segmentation against ground truth in terms of the number of regions, the region locations, and the region sizes. Let S and T be two segmentations of the same image, and $S = \{S_1, \dots, S_m\}$ and $T = \{T_1, \dots, T_n\}$ where S_i corresponds to the set of pixels in region i from segmentation S . We associate with each region S_i a region T_k such that $S_i \cap T_k$ is maximal. The Hamming distance between two segmentations is defined as:

$$D_H(T \Rightarrow S) = \sum_{S_i \in S} \sum_{T_j \neq T_k} |S_i \cap T_j| \quad (3.23)$$

which corresponds to the sum of areas of intersection for all non-maximally intersecting regions. To make Equation 3.23 symmetric and normalize it in the

range $[0, 1]$, we sum Hamming distances in both directions and divide by the twice the image area. The resulting performance measure is defined as:

$$HD = 1 - \frac{D_H(T \Rightarrow S) + D_H(S \Rightarrow T)}{2A} \quad (3.24)$$

Huang and Dom define their measure to be a performance indicator instead of an error measure: values closer to one denote a better segmentation. It is easily changed to an error measure by removing the subtraction from one in the above equation, giving the Huang-Dom index:

$$HDI = \frac{D_H(T \Rightarrow S) + D_H(S \Rightarrow T)}{2A} \quad (3.25)$$

Implementation

Similarly to the local and global consistency errors described in the last section, the Huang-Dom measure can be efficiently computed using the confusion matrix M . The directional Hamming distance $D_H(T \Rightarrow S)$ is obtained as follows:

$$D_H(T \Rightarrow S) = \sum_i \sum_{j \neq m(i)} M_{ij} \quad (3.26)$$

where $m(i)$ is the index of the largest value of row i in M , i.e.

$$m(i) = \arg \max_j M_{ij} \quad (3.27)$$

The Hamming distance in the other direction $D_H(S \Rightarrow T)$ can be similarly computed using M^T .

Analysis

When under or over-segmentation is a concern for an application, the Huang-Dom measure is an excellent choice for evaluation. It is a stable measure and, unlike the local and global consistency error measures, it does not suffer from degenerate cases. If refinement is to be tolerated, however, then the local and global consistency measures may be more appropriate. Arguably, using this measure in combination with one of the local or global consistency measures gives the most information; we take this approach in our evaluation experiment in Chapter 4.

In the previous section we experimentally investigated some properties of the local and global consistency error measures: the mean values that imply accurate and inaccurate segmentation, and the range of values achieved by state-of-the-art segmentation algorithms. We performed the same experimental investigation for the Huang-Dom measure. We found that mean error for pairs of segmentations of the same scene by different people is 12%; the mean error for comparing segmentations of different scenes is 40%. The average error achieved by state-of-the-art segmentation algorithms are between 23% and 37%. The experiments are described in more detail in Chapter 4.

3.3.4 Other Approaches

A number of other supervised segmentation evaluation techniques have been proposed. Mezaris et al. [Mezaris et al., 2003] propose a benchmark that combines region boundary accuracy and the degree of over-segmentation. Two separate measures, the first estimating boundary accuracy using weight functions, and the second gauging the level of over-segmentation, are additively combined to produce the final measure. However, the benchmark has some issues. Mezaris et al. do not attempt to normalize the final measure to a particular range, nor

do they justify the additive combination of the accuracy and over-segmentation measures. A linear combination would at least allow the contribution of these components to be weighted.

Correia and Pereira [Correia and Pereira, 2003, Correia and Pereira, 2006] describe a method for evaluating the quality of image or video segmentation based on object similarity and object relevance. Object similarity is computed using a variety of factors, including: shape fidelity (the number of misclassified pixels), geometric similarity (the size, position, elongation, and compactness of objects), internal edge similarity (the average difference of Sobel edges in the region), and statistical data similarity (the differences in brightness and “redness” values). In our classification scheme, their method is a hybrid evaluation technique: it combines discrepancy-based and feature-based measures. Again, the method has some problems. Their selection of measures, and how they are computed is not well justified; the “brightness and redness” criteria are particularly conspicuous in this regard. Furthermore, these values are combined to form the final metric using an apparently arbitrary weighting scheme.

The segmentation evaluation technique proposed by Usamentiaga et al. is different to the techniques we have discussed so far [Usamentiaga et al., 2006]. Instead of comparing a pair of segmentations by measuring, in some way, the degree of overlap between regions, they use the degree of overlap between the region boundaries. Usamentiaga et al. term this an “empirical edge discrepancy” technique, to set it apart from the more common region-based discrepancy methods. The difficulty faced when developing an edge discrepancy error measure is the inherent uncertainty in the edge positions: since the boundaries of a region are very narrow, the overlap between two segmentations may be low, even if the segmentations are perceptually very similar. Put another way, if a boundary pixel in the machine segmentation lies a single pixel away from a boundary pixel in the ground truth, then using classical methods we observe false positive and

a true negative. We would prefer if the penalty for a misclassified boundary pixel in the machine segmentation was in some way proportional to its distance to a boundary pixel in the ground truth. Usamentiaga et al. address this by generalizing some existing segmentation error measures using fuzzy set theory [Zadeh, 1965]. The result is a set of measures that are robust to small spatial errors in the boundary pixels. We take a similar approach when developing a boundary accuracy measure for interactive segmentation evaluation in Chapter 5.

For the interested reader, the survey paper by Zhang [Zhang, 1996] provides an excellent characterization and overview of several classical evaluation techniques. A recent special issue of the *EURASIP Journal on Applied Signal Processing* focuses on the evaluation of image processing techniques [Wirth et al., 2006], and contains papers on several of the techniques that we discussed.

3.4 Unsupervised Evaluation

In the previous section we discussed supervised measures for segmentation evaluation; in this section we consider unsupervised measures. Supervised evaluation measures operate by comparing the output of a segmentation with a reference ground truth. Unsupervised evaluation measures do not use a reference ground truth. The motivation for unsupervised evaluation is the hypothesis that there are properties of a segmentation that we can measure to judge segmentation quality, independent of how well the segmentation matches a specific ground truth. A simple justification for this hypothesis is that we (human beings, that is) can often intuitively infer some general properties of a segmentation algorithm by observing the output, even without knowing the image that generated it. For example, by observing that the output often has many regions, we may reason that the algorithm is prone to over-segmentation. Other intuitive conclusions may be formed by observing that the segmentations an algorithm produces are very

jagged, or very symmetrical, or that there are usually a few large regions and many small regions.

We are able to make such inferences by observing the results of the segmentation process, without knowing the content of the images that produced these results. This restriction is necessary for the justification we gave above: an innate perceptual grouping is unavoidable if we actually observe the image. It implies a kind-of mental ground truth, and it could therefore be argued that we are, in fact, performing supervised evaluation. The restriction is unnecessary when developing measures and algorithms for unsupervised segmentation evaluation; an evaluation measure having access to the generating image does not imply a ground truth.

The objective of an unsupervised evaluation technique is, therefore, to measure the quality of a segmentation given only the segmentation itself, and the corresponding generating image. Since we have less information, the task is harder. Unsupervised evaluation techniques do, however, have some distinct advantages.

First, since perceptual grouping is inherently hierarchical, for a scene of reasonable complexity, every ground truth is fundamentally ambiguous. Some supervised evaluation measures, such as the local and global consistency error measures discussed in Section 3.3.2, work around this using refinement tolerance. Unsupervised techniques remove the need for ground truth completely, instead focusing on general properties of the segmentation, as opposed to how well it matches a particular reference.

Second, and more importantly, the capacity to measure the quality of a segmentation without a reference implies an fundamental understanding of human perceptual grouping. When developing practical segmentation algorithms, one is constrained to find solutions in the context of what is computationally feasible. We have a tendency to approach the problem bottom-up; to focus on workable

solutions that use efficient algorithms; to focus effort on the means, rather than the ends. When developing an unsupervised evaluation measure, however, we are concerned with how we can assess segmentation quality. This shifts focus toward formalizing the objective of the image segmentation problem, and away from the details of how exactly it is to be implemented. It is, in essence, the core of the image segmentation problem: if we can specify exactly what criteria make a segmentation good, we can then turn our attention to finding specific algorithms to optimize this criteria.

Finally, most segmentation algorithms have one or more parameters that influence the output; unsupervised evaluation measures can be used to automatically select good values for these parameters. For instance, the statistical region merging algorithm discussed in Section 2.4.2 depends on a parameter Q , which controls the scale of the segmentation. If we can compute a measure of segmentation quality M without a ground truth reference, then we can automatically choose a good value of Q for a particular image by searching for the value of Q that optimizes M .

From our discussion it is clear that an unsupervised segmentation evaluation measure is a formal conjecture on the objective of image segmentation. In almost all applications this objective is extremely difficult to formalize. This is particularly true if the objective is perceptual grouping. It is unsurprising, therefore, that unsupervised segmentation evaluation measures have achieved limited success. In the remainder of this chapter we review some of the more successful measures.

3.4.1 Entropy Based Evaluation

Zhang et al. [Zhang et al., 2004] propose an unsupervised evaluation measure based on information theory. The method uses entropy to measure both the pixel uniformity within a region, and the complexity of the overall partitioning.

Their method is appropriate either for comparing different parameterizations of a segmentation algorithm, or for comparing the performance of different segmentation algorithms.

Theory

The function of any unsupervised evaluation measure is to take image I and a segmentation $S = \{R_1, \dots, R_n\}$ and produce a measure M that indicates how good the segmentation is. M is usually defined as a measure of segmentation error: values closer to zero indicate good segmentation and larger values indicate the opposite.

To produce a useful measure M it is necessary to formalize what makes a segmentation good. The objective of image segmentation is usually informally specified as partitioning an image into regions that are, in some sense, homogeneous. This definition is loose and problematic. In particular, the segmentation that has a single pixel for each region is perfectly homogeneous, regardless of the criteria. Most unsupervised evaluation measures therefore attempt to balance region homogeneity with either (1) the number of regions, or (2) a measure of the differences between adjacent regions.

Zhang et al. measure region homogeneity using the concept of entropy. Let $f(j)$ be a mapping from pixel $j \in I$ to a feature point describing the pixel at j , and let V_i be the set of all values that $f(k)$ can take on for all $k \in R_i$. If $N_i(x)$ is the number of pixels in region i that have the value x , the entropy of region i is given by:

$$H(R_i) = - \sum_{x \in V_i} \frac{N_i(x)}{|R_i|} \log \frac{N_i(x)}{|R_i|} \quad (3.28)$$

From this we define the expected region entropy for an image I as a weighted sum of the individual region entropies:

$$H_r(I) = \sum_{i=1}^n \frac{|R_i|}{|I|} H(R_i) \quad (3.29)$$

A region with a large number of equal feature points requires fewer bits to encode: it has lower entropy. The value of expected region entropy will be smaller for images that contain regions with many equal feature points. Consequently, if an image contains many small regions, it is more likely to have a lower expected region entropy; when each pixel is its own region, then expected entropy is zero. It is therefore necessary to adjust expected region entropy using some other measure to offset this over-segmentation bias.

While the expected region entropy decreases with the number of regions, the number of bits required to specify the region to which each pixel belongs increases. This is called the *layout entropy*, and is given by:

$$H_l(I) = - \sum_{i=1}^n \frac{|R_i|}{|I|} \log \frac{|R_i|}{|I|} \quad (3.30)$$

The layout entropy is biased toward under-segmentation whereas the expected region entropy is biased toward over-segmentation. Combining them can be used to balance the evaluation measure. Zhang et al. propose an additive combination. Their final entropy-based evaluation measure E is given by:

$$E = H_r(I) + H_l(I) \quad (3.31)$$

Analysis

In their paper, Zhang et al. compare their proposed benchmark to its predecessors, including the F measure by Liu and Yang [Liu and Yang, 1994], and the F' and Q

measures by Borsotti et al. [Borsotti et al., 1998]. All of these measures are based on balancing region homogeneity, determined using square color differences, with the area and number of regions. All are based entirely on empirical analysis, and, as demonstrated in [Zhang et al., 2004], all have strong bias. In contrast, Zhang et al.'s measure has a solid theoretical foundation. It is also shown to be less biased toward under or over-segmentation than the F , F' , and Q measures.

Zhang et al. continue their examination of unsupervised segmentation evaluation techniques in a later paper [Zhang et al., 2008]. In this work they compare a range of unsupervised evaluation techniques, investigating measure bias for under or over-segmentation, and evaluating how often each measure can differentiate between machine segmentation and human perceptual grouping. In the latter test, Zhang et al.'s measure significantly outperformed the other measures, correctly determining whether a segmentation was due to a human or a machine in 82.1% of tests.

In the same paper, Zhang et al. observe that when segmenting an image, people often extracted objects composed of several smaller regions because of the hierarchical nature of perceptual grouping. This often leads to a segmentation in which regions have several color modes. Because their measure is based on entropy, this is less problematic: a region composed of fewer colors requires less bits to encode and will score better.

The method implies an interesting view of segmentation as a method of simplification. In this context, simplification is to be understood in an information theoretic sense: reducing the number of bits required to represent a segmentation. This perspective agrees with the Gestalt principle of *Prägnanz*: human beings perceive reality in its simplest possible form. It is also in line with Attneave's [Attneave, 1954] interpretation of the Gestalt grouping principles: that they are used by the mind to eliminate the inherent redundancy in visual forms.

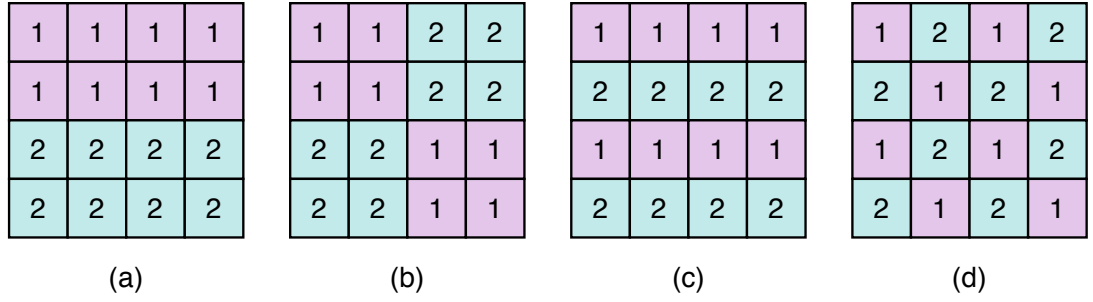


Figure 3.3: Figures (a), (b), (c), and (d) each contain two regions of eight pixels. Each has the same layout entropy.

Zhang et al.'s measure also has some limitations. First, the layout entropy component does not place any restrictions on spatial layout of the segmentation; every segmentation with regions of equal number and area will give the same value for layout entropy. Figure 3.3 illustrates this problem; intuitively we would think that Figure 3.3 (d) is more complex than Figure 3.3 (a). They both, however, have the same layout entropy.

The second disadvantage is that entropy treats feature points as nominal values. A luminance value of 200 is not considered any closer to a luminance value of 201 than it is to 10; it is simply another value that needs to be encoded. This has two ramifications. First, the measure depends on the number of luminance values in an image, rather than how many luminance values are perceptually significant. Second, the measure is difficult to extend to multichannel feature spaces or feature spaces with real valued components, since the number of possible values a pixel can assume increases dramatically. It is possible to handle such feature spaces by quantizing or binning the feature values in advance, though such quantization is likely to have a significant effect on the measure.

3.4.2 Visible Color Distance Based Evaluation

Chen and Wang propose an unsupervised evaluation measure based on perceived color distances. They define two measurements: intra-region visual error, and inter-region visual error [Chen and Wang, 2004].

Intra-region visual error measures the average homogeneity of each region in the image. It does this by counting the number of pixels whose color is perceptually different from the mean color of the region to which the pixel belongs. Whether two colors are perceptually different is determined by using Euclidean distance in CIELAB space. Inter-region visual error measures the average difference between neighboring regions in a segmentation. It is measured by counting the number of pixels on the boundary of each region that have an average region color perceptually indistinguishable from the average region color of an adjacent region. An additive combination of intra-region visual error and inter-region visual error produces their final measure.

Theory

Chen and Wang define intra-region visual error as follows. Let $f(x)$ be a function that gives the CIELAB color of pixel x in the image I . Further, let $\hat{f}(x)$ to be a function that gives the average CIELAB color of the region containing x in a segmentation S of I . The intra-region visual error for the segmentation S of I is defined as:

$$E_{\text{intra}} = \frac{1}{n} \sum_{x \in I} u(\|f(x) - \hat{f}(x)\| - t) \quad (3.32)$$

where n is the number of pixels in I , and $u(s)$ is a step function defined to be one if $s > 0$, and zero otherwise. The t value is a tolerance threshold for color differences; Chen and Wang suggest $t = 6$.

Chen and Wang define inter-region visual error as:

$$E_{\text{intra}} = \frac{1}{cn} \sum_{r_1 \in S} \sum_{r_2 \in S \setminus r_1} w_{r_1, r_2} u \left(t - \|\hat{f}(r_1) - \hat{f}(r_2)\| \right) \quad (3.33)$$

where $r \in S$ is a region in the segmentation S and $\hat{f}(r)$ gives the average color of region r . w_{r_1, r_2} is the length of the join between region r_1 and region r_2 . The value c is a normalization constant that Chen and Wang empirically set to $1/6$.

Analysis

Chen and Wang attempt to introduce some perceptual significance into their unsupervised evaluation measure by using visible color differences in a perceptually uniform color space. This effort is commendable; perceptual significance had been ignored by many previous measures.

The measures imply some questionable assumptions, however. First, the measures assume that a region has a single color mode. This assumption is probably not true; Zhang et al. observed that perceptual grouping often creates regions that have several color modes [Zhang et al., 2008]. Second, the measures assume that the average color of a region is perceptually significant, and that color differences can be applied between these averages in a perceptually meaningful way. Finally, the normalization constant in the inter-region visual error measure is conspicuously unjustified; the authors state they determined it empirically, but do not state how.

3.4.3 Other Approaches

A few other unsupervised segmentation evaluation techniques have been proposed; for the interested reader, the recent survey paper by Zhang et al. provides an excellent overview of the state-of-the-art [Zhang et al., 2008].

3.5 Discussion

Although many algorithms for image segmentation have been proposed, it is by no means a solved problem. To advance the state-of-the-art we must be able to determine if a new algorithm, or a modification of an existing one, does indeed constitute an improvement. Segmentation evaluation techniques help us to do this by providing a means of comparing segmentation algorithms with each other, and with human perceptual grouping.

An effective segmentation evaluation technique can have tremendous benefits. It can potentially give insight into what constitutes a good segmentation algorithm. It can allow researchers to add weight to an argument that a new segmentation algorithm is better than the existing technology. It can allow application developers to choose the best algorithm for their system.

Supervised segmentation evaluation methods operate by comparing a segmentation with an existing reference segmentation known as a ground truth. If the ground truth is created by a person, supervised evaluation can be used to evaluate how well an image segmentation algorithm approximates human perceptual grouping. The approach has limitations, however. Perceptual grouping is hierarchical; to evaluate a segmentation algorithm we must provide some tolerance to refinement. Unfortunately, introducing such tolerance often means we can no longer evaluate the level of under or over-segmentation.

A potentially better way to evaluate a machine segmentation against a perceptual grouping is to directly compare the grouping hierarchies. Most segmentation algorithms can be modified to produce a tree representing the different levels of detail in the segmentation. An interesting notion would be to compare this tree with a similar tree created by a person. A hierarchical dataset for human perceptual grouping has not yet been investigated, nor have methods for measuring similarity between grouping trees.

Unsupervised methods evaluate segmentation without using a ground truth. This is more difficult, but has many potential benefits. Since it does not use a reference segmentation, it avoids the previously discussed issue of ambiguous ground truth. It also means unsupervised evaluation can be used to automatically select a good parameterization for an algorithm. It shifts focus from implementation and algorithmic concerns onto the objective of segmentation; the capacity to measure the quality of a segmentation without a reference implies a fundamental understanding of human perceptual grouping.

The survey paper by Zhang et al. [Zhang et al., 2008] demonstrates that there is still much work to be done in defining useful unsupervised evaluation measures. In particular, it shows that most existing measures are incapable of reliably distinguishing between segmentations created by humans and those created by machines. The paper further shows that existing unsupervised measures often disagree with human judgement when comparing the performance of different machine segmentations. Some unsupervised measures do, however, appear to be useful for choosing a good parameterization of a segmentation algorithm; the Q measure by Borsotti et al. [Borsotti et al., 1998] is shown to be particularly effective. Recent efforts that combine unsupervised segmentation evaluation measures using machine learning also appear promising [Zhang et al., 2005, Zhang et al., 2006, Zhang et al., 2008].

In comparison to image segmentation, segmentation evaluation is in its incipience. Nevertheless, it has recently seen some significant advances. There are now several useful evaluation measures available for supervised segmentation evaluation, such as the measures described in: [Huang and Dom, 1995], [Martin et al., 2001], [Jiang et al., 2005], and [Jiang et al., 2006]. Measures for unsupervised evaluation have also been developed (such as: [Liu and Yang, 1994], [Borsotti et al., 1998], [Zhang et al., 2004], and [Chen and Wang, 2004]), and their limitations have been investigated [Zhang et al., 2008]. Perhaps more signifi-

cantly: datasets containing images and multiple ground truth segmentations of these images by different people have been made publicly available. The dataset due to Martin et al. [Martin et al., 2001] is ideal for supervised evaluation of region segmentation algorithms.

In spite of these advances, there remains considerable work to be done. Un-supervised measures need improvement; they cannot yet reliably distinguish human from machine segmentation, and many exhibit strong bias to under or over-segmentation. High quality datasets and measures for supervised evaluation have been developed, but they have not yet been widely applied for evaluating the state-of-the-art. Datasets and measures for evaluating interactive segmentation have not been investigated.

In the remainder of this thesis we will address some of these issues. The next chapter addresses evaluating well-known automatic segmentation algorithms using existing evaluation measures. Chapter 5 and 6 investigate interactive segmentation evaluation.

Chapter 4

Evaluating Automatic Segmentation

4.1 Introduction

In this chapter we use supervised segmentation evaluation measures to evaluate the performance of four popular automatic region segmentation algorithms. Our objective is to use existing tools and datasets to determine how well these algorithms approximate human perceptual grouping.

There are two reasons for carrying out this kind of evaluation. The first is to gain a better understanding of existing supervised segmentation evaluation measures. We would like to develop an understanding of the range of these measures, in particular, values that indicate accurate segmentation, and values that indicate inaccurate segmentation. We would also like to determine which, if any, of the existing supervised segmentation evaluation measures are most effective at evaluating a segmentation algorithm's ability to approximate perceptual grouping.

The second reason to run this kind of evaluation is to determine how accurately existing, but previously unevaluated, segmentation algorithms are at approximating human perceptual grouping. Ideally, we would like to create a ranked list of segmentation algorithms, ordered by performance. Knowing which

algorithms perform best not only equips us better to choose an appropriate one when developing a system, but also gives insight into which types of algorithms and strategies give the best results for perceptual grouping.

The remainder of this chapter is organized as follows. Section 4.2 briefly describes each of the algorithms that we selected for the evaluation. Section 4.3 discusses the various supervised evaluation measures that we choose, and Section 4.4 describes the ground truth dataset. Section 4.5 gives an overview of a software framework for image and video segmentation that we developed and used for the evaluation. Section 4.6 discusses the evaluation method, evaluation objectives, and statistical considerations. Section 4.7 describes the experiments and presents the results and implications of same. Finally, Section 4.8 discusses of our findings and outlines some recommendations.

4.2 Algorithms

We selected four algorithms for the experiment. The algorithms are representative of the state-of-the-art, yet had not previously been evaluated using supervised evaluation techniques. Further, each is based on one of the techniques described in Section 2.4. The following is a brief overview of the algorithms.

RSST with Syntactic Visual Features

The algorithm due to Adamek et al. [Adamek et al., 2005] is based on the recursive shortest spanning tree (RSST) segmentation algorithm [Morris et al., 1986]. Instead of grouping pixels based only on intensity, the algorithm first groups pixels using color differences in CIELUV space, then further groups these regions using syntactic visual features [Bennstrom and Casas, 2004]. Since it is RSST-based, it can be reformulated as a region adjacency graph algorithm (Section 2.4.1).

MRSST	RSST with syntactic visual features	[Adamek et al., 2005]
SRAG	Spatiotemporal region adjacency graphs	[Galmar and Huet, 2006]
MSHIFT	Optimized mean shift segmentation	[Bailer et al., 2005]
SRM	Statistical region merging	[Nock and Nielsen, 2004]

Table 4.1: Evaluated algorithms and their abbreviated names

The syntactic features represent geometric properties of the regions and their spatial configuration. They are used to model some of the more abstract principles in the Gestalt theory of perceptual grouping. Adamek et al.’s algorithm uses three syntactic features: *compactness*, *regularity*, and *inclusion*.

The algorithm begins by forming an initial over-segmentation using a merging predicate based on Euclidean distance in CIELUV space. Next, the algorithm switches to a color model designed to better handle larger regions. This color model allows for outliers, and handles gradients using the boundary melting approach [Sonka et al., 1998]. At this stage the syntactic visual features are also incorporated. These are designed to favor smooth, convex regions, similar to those found in natural scenes.

Spatiotemporal Region Adjacency Graphs

The spatiotemporal region adjacency graph algorithm, proposed by Galmar and Huet [Galmar and Huet, 2006], is another variant of the region adjacency graph algorithm. The algorithm includes several modifications of the basic region adjacency graph strategy, mostly targeted at producing temporally coherent regions when segmenting video. The interested reader is referred to [Galmar and Huet, 2006] for more details.

Optimized Mean-shift Segmentation

This algorithm, proposed by Bailer et al. [Bailer et al., 2005], is based on the mean-shift segmentation algorithm described in Section 2.4.4. It includes three specific modifications, intended to reduce the algorithm’s computational complexity and improve its temporal stability. We will only discuss one of these modifications; the others pertain to using the algorithm on video.

The first modification improves the performance of the algorithm by performing a moderate quantization of the CIELUV color space before segmentation. Bailer et al. state that this quantization does not significantly effect the quality of the segmentation, and results in a sparse feature space in which nearby colors have been grouped. The modified feature space effects a significant reduction in computational complexity, as less values are processed when computing the mean-shift vector.

The final algorithm we evaluate is the statistical region merging algorithm. This is exactly the algorithm described in Section 2.4.2, so we will not discuss it further here. Table 4.1 assigns more concise names to each of the algorithms. We use these abbreviations in the remainder of this chapter.

4.3 Measures

We selected six supervised evaluation measures for the evaluation; each was discussed in Section 3.3. The first two measures we chose are the local consistency error and global consistency error; both were designed specifically for this kind of evaluation. Neither measure is sensitive to refinement error. The next measure we selected is the Huang-Dom index, as it is more sensitive to this type of error, thus better at detecting under-segmentation and over-segmentation. The last three measures we selected evaluate cluster similarity: the Jaccard index, the Rand

LCE	Global consistency error	[Martin et al., 2001]
GCE	Local consistency error	[Martin et al., 2001]
HDI	Huang-Dom index	[Huang and Dom, 1995]
FMI	Fowlkes and Mallows index	[Fowlkes and Mallows, 1983]
JI	Jaccard index	[Ben-Hur et al., 2002]
RI	Rand index	[Rand, 1971]

Table 4.2: Evaluation measures and their abbreviated names

index, and the Fowlkes and Mallows index. All three are based on the notion of *counting pairs*, which we discussed in Section 3.3. As they are fundamentally quite similar, examining their statistical properties and the correlations between them may provide insight into their usefulness. Table 4.2 assigns each of the measures an abbreviated name.

4.4 Dataset

The next step is to select an appropriate dataset and ground truth for the evaluation. We used the Berkeley segmentation dataset [Martin et al., 2001]. The dataset comprises 300 images of natural scenes and multiple ground truth segmentations per image. The ground truth segmentations were generated by 28 human participants, each image being segmented by between four and nine different people, giving an average of 5.44 ground truths per image ($\sigma = 0.7439$, median = 5).

The images in the dataset are divided into two collections: *test* and *train*. The test collection consists of 100 images, and the train collection comprises the remaining 200. The train images are intended to be used for segmentation algorithms that require a training phase, or require parameter tuning. Since we do not perform any such training or parameter tuning, we use both collections for evaluation.

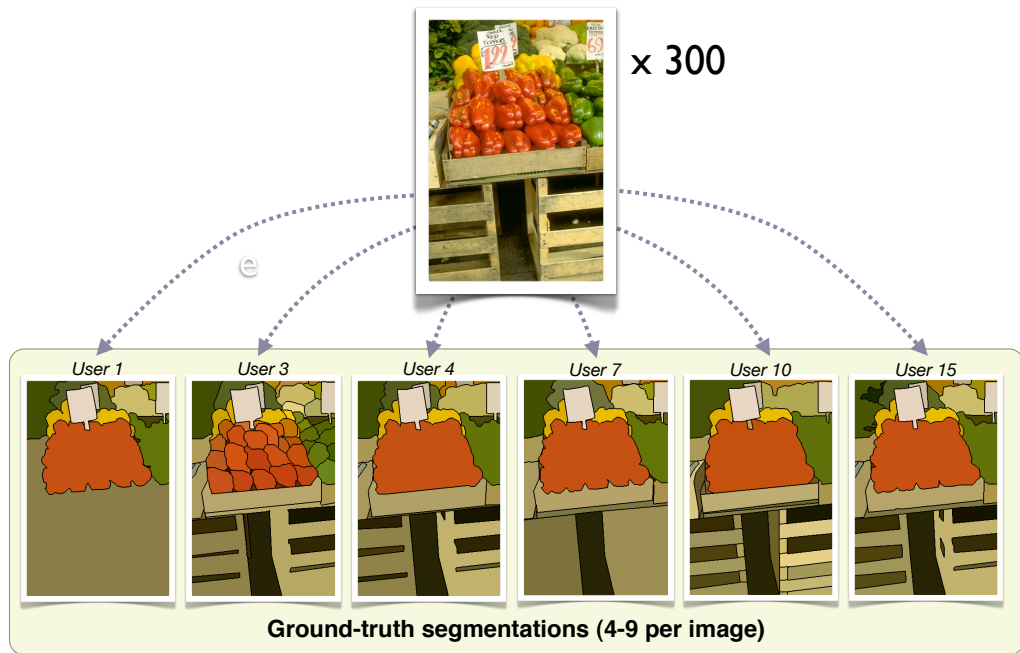


Figure 4.1: The Berkeley segmentation dataset.

The dataset contains a second set of ground truth based on grayscale versions of the images. All of the segmentation algorithms we evaluate use color information in some way; we therefore use only the color ground truth in our experiments.

Figure 4.1 shows an image from the Berkeley segmentation dataset, along with graphical representations of the corresponding ground truth segmentations. The dataset contains a total of 1633 full color segmentations; Figure 4.2 illustrates five more example images and their corresponding segmentations.

4.5 Software

The experiment requires us to segment each image in the dataset using all four of the algorithms being evaluated. Since we were also participating in TRECVID [Wilkins et al., 2007] at the time, and intended to use segmentation in our multime-

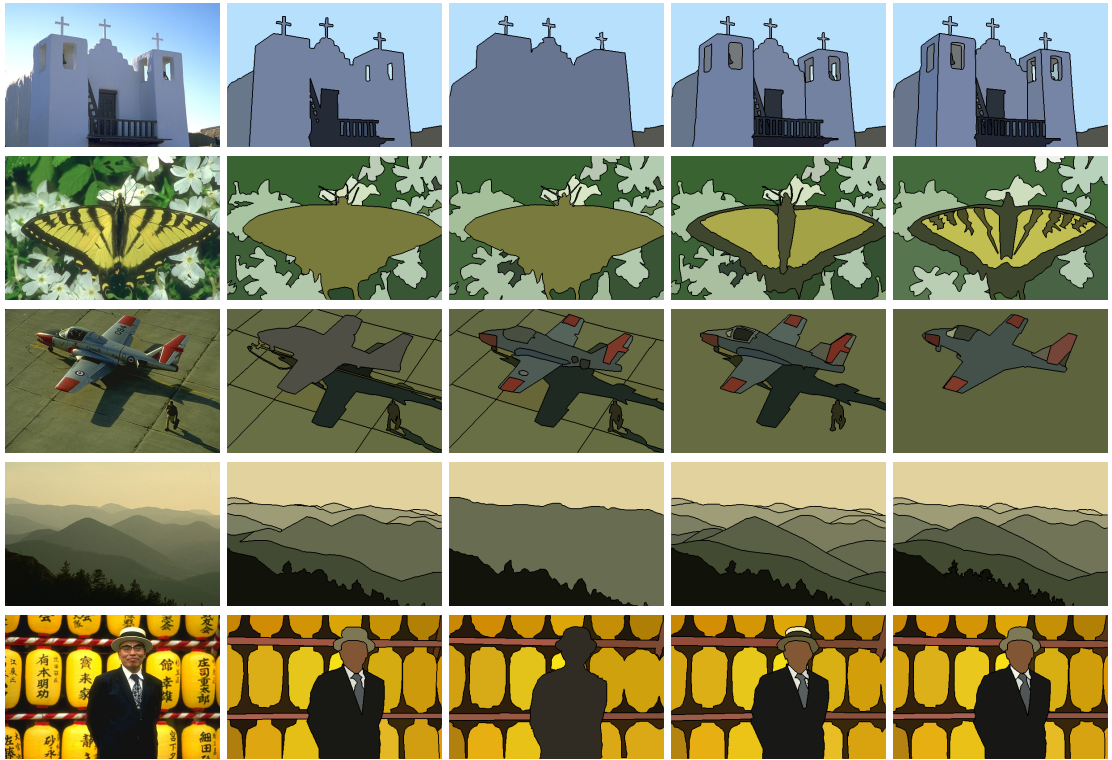


Figure 4.2: Sample images and segmentations from the Berkeley segmentation dataset

dia retrieval engine, it made sense for us to develop a tool that would allow us to more easily run large segmentation tasks, with multiple segmentation algorithms, and visualize the results.

We developed the *K-Space video region segmentation tool* to provide a unified interface for automatic segmentation of images and video sequences. Figure 4.3 shows a screenshot of the application’s graphical user interface. We now briefly review the features and architecture of the software.

4.5.1 Features

The following are the main features of the platform:

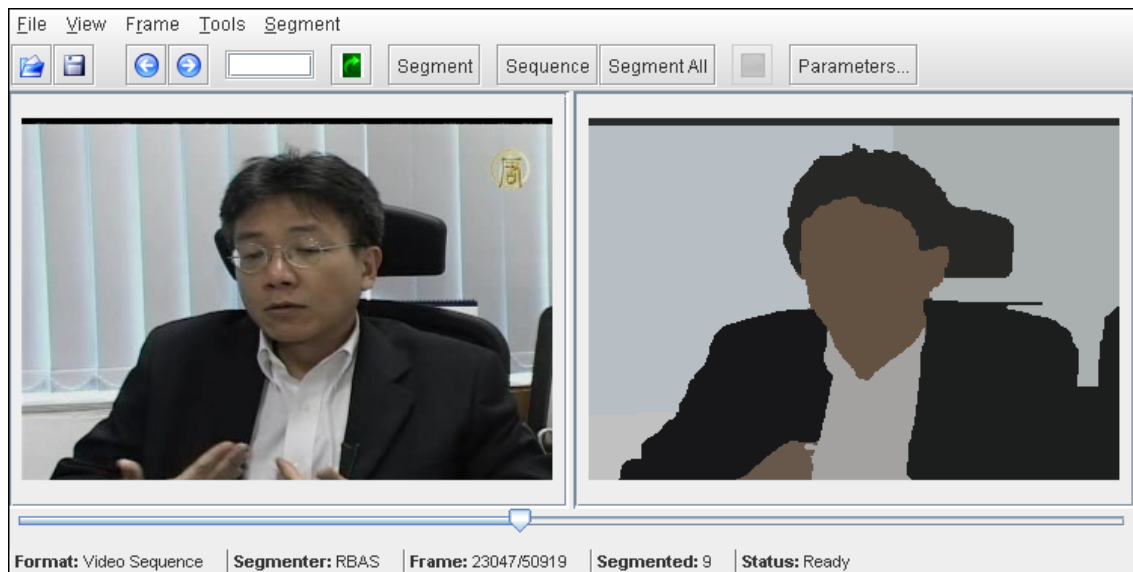


Figure 4.3: Screenshot of the K-Space video region segmentation tool.

Formats: The framework provides an interface for seek-able, frame accurate video decoding. The built in video decoder supports many video formats, including: MPEG-1, MPEG-2, and MPEG-4; Motion-JPEG; Quicktime; and WMV. Also included is an image decoder capable of decoding both individual images and sequences of video key-frames transparently. The image decoder supports a large range of image formats, including: JPEG, PNG, PNM, GIF, and BMP.

Region maps: The framework encodes segmentations as region maps, using an efficient, portable format, based on a subset of PNG. This allows segmentation of video sequences with minimal storage overhead.

User interface: The user interface provides: automatic decoder selection, concurrent browsing of video frames and segmented images, range segmentation, useful visualization methods, and a simple interface for selecting algorithms and their parameters.

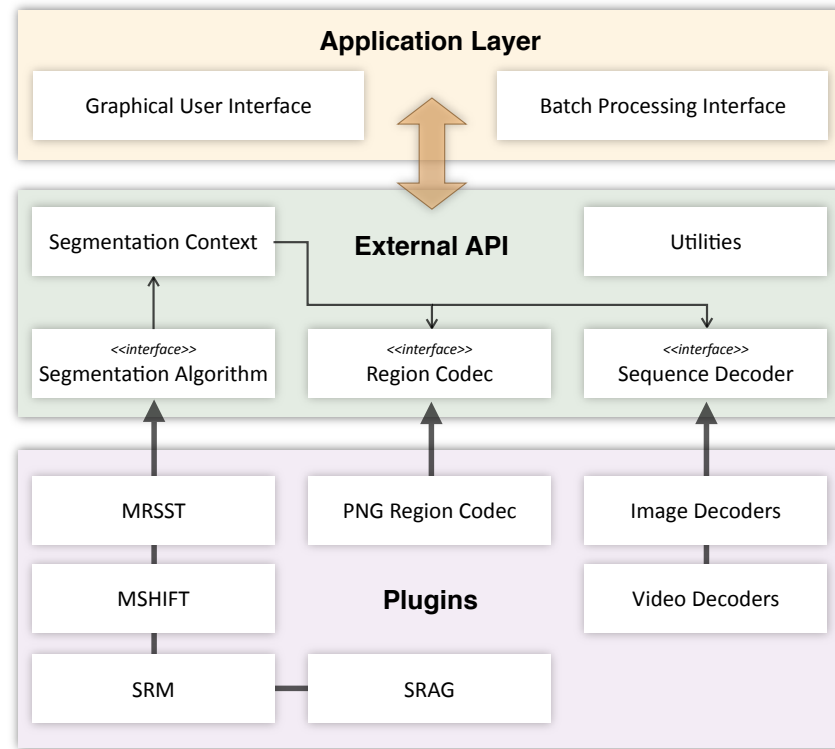


Figure 4.4: Software architecture model for the K-Space segmentation tool

Batch processing: The batch processing interface facilitates command line segmentation of large image or video collections. All the parameters that can be selected in the graphical user interface can be written to a parameter file.

4.5.2 Architecture

The framework is arranged into three layers. The application layer hosts the user interface, user preferences, batch processing interface, and integration logic. The encoding and decoding of images and video sequences and segmentation are all handled by the plugin layer. The application layer communicates with these plugins using the middle layer: the external API. Figure 4.4 illustrates the architecture.

Application: The application layer hosts the graphical user interface and the batch processing interface. The user interface provides a convenient way to segment images and video, select algorithm parameters, browse video frames, and visualize segmentation output. The batch interface is designed for off-line processing of larger datasets; the segmentation operations are completely configured using a parameter file. If desired, the output can be visualized using the user interface after processing.

Segmentation: To integrate additional segmentation algorithms, developers must implement a segmentation algorithm interface. The application uses this interface to configure and run each segmentation algorithm. At runtime, a configuration file is read to determine which segmentation algorithm implementations are available, and they are automatically added to the applications user interface.

The application carries out a segmentation of a video frame by frame. For each frame, the selected segmentation algorithm is passed a segmentation context object. This object contains all the information necessary to perform the segmentation: the current frame and index, the frame decoder object, the region map object, and an interface for retrieving previously segmented frames. If, instead of a video sequence, a static image is being segmented, this image appears to the segmentation algorithm as a video sequence consisting of a single frame. This design allows each segmentation to be a single operation, while also providing enough contextual information for segmentation algorithms that require previous segmentations or frames. It simplifies the integration of static image segmentation algorithms, but provides enough information for algorithms that operate in the temporal domain.

Decoders: A single interface is provided for both image and video decoders. This allows a segmentation algorithm to handle single images, multiple images,

and video sequences in the same way. The application bundles a comprehensive set of decoders; the plugin architecture ensures additional decoders can easily be added.

The built-in video decoder provides frame-accurate seeking and decoding of many video formats. It is based on the FFMPEG audio visual codec library, which supports most popular video formats. Unfortunately, frame-accurate video seeking and decoding from arbitrary frame indices is not natively supported by the FFMPEG library. This functionality is essential for video segmentation. As such, we added an additional layer to preprocess the video and index its frames before decoding. The built-in image decoder is based on the JAI Image IO library and also supports a comprehensive set of formats.

Segmentation Representation: We use the PNG format for encoding segmentations as region label maps. For segmentations containing fewer than 256 regions, we use the 8-bit grayscale format; for segmentations containing more than 256 regions, we use the 16-bit grayscale format. The region storage codec therefore supports a maximum of 2^{16} regions.

Experiments demonstrated a high compression rate for most region maps. A typical segmentation of 10 seconds of MPEG-1 video (resolution 352×240 , frame rate 29.97 fps) requires less than 500 KB of storage. Aside from high compression, the format has some additional advantages: it can be directly opened in most imaging applications, and easily visualized by stretching the contrast between the regions. There are also various freely available software libraries for decoding these images.

4.6 Method

This section will consider in more detail the research questions we aim to answer, and how we plan to answer them. The experiment has two objectives. The first objective is to examine the characteristics of the evaluation measures. The vital question to answer here is: which, if any, of the evaluation measures are useful for evaluating a segmentation algorithm's ability to emulate perceptual grouping? To answer this question, we must first determine base rates for each evaluation measure. Specifically, for each measure, we would like to estimate: (1) the base rate for accurate segmentation, and (2) the base rate for inaccurate segmentation. If both rates are reasonably stable (unimodal, with a small standard deviation), and the difference between them is statistically significant, then there is evidence to support the hypothesis that a measure can indeed differentiate between accurate and inaccurate segmentation.

Of course, we must define precisely what is meant by *accurate* and *inaccurate* to estimate these base rates. This is difficult; we avoid the problem by making some reasonable assumptions. First, we assume that humans always create perceptually accurate segmentations. This might not strictly be true, but the assumption is justified for two reasons: (1) we must expect the ground truth to be reasonably accurate, otherwise any form of supervised evaluation is spurious, and (2) averaging over multiple ground truths should suppress exceptional inaccuracies. Second, we assume that the images in the database are sufficiently different in content to result in different perceptual groupings. Again, this might not be true in every case, but we expect it is true on average.

Accepting the first assumption implies that two separate segmentations of the same scene created by different people are mutually accurate. Accepting the second implies that segmentations of different scenes are mutually inaccurate, regardless of whether they are created by the same person. Therefore, given a

large enough sample, we can estimate the base rate of accurate segmentation for a given measure by comparing pairs of segmentations of the same scene by different people. We can estimate the base rate for inaccurate segmentation by comparing pairs of segmentations of different scenes.

Another question we would like to address is: how correlated are the evaluation measures? If two measures are highly correlated, this suggests they measure similar things, and therefore using both may be redundant. Determining the redundancy among the measures is especially important if they are to be used for training: significant redundancies cause bias in some machine learning algorithms.

The second objective is to evaluate each of the four segmentation algorithms and rank them according to how well they match human perceptual grouping. This objective is dependent on the first: if none of the measures are demonstrated to be effective for evaluating how well a segmentation corresponds to perceptual grouping, then it makes little sense to use them for evaluation. Assuming useful measures are found, we would like to answer the following questions:

1. How accurate, on average, are each of the algorithms at approximating human perceptual grouping?
2. Are the algorithms significantly better than random? Are they significantly worse than humans?
3. How do the algorithms compare with one another? Do any of the algorithms significantly outperform the others?

To address these questions, we first need to segment each image in the dataset using all four segmentation algorithms, then compare each segmentation with all the available ground truth using the appropriate evaluation measures. For each machine segmentation there are two potential ways to measure accuracy.

The first assumes that a machine segmentation is accurate if it matches *any* segmentation performed by a human. From this viewpoint, we measure accuracy as follows. Let I be an image, M be a machine segmentation of this image, and G be the set of ground truth segmentations for I . Assume a measure $\mathcal{E}(M, G_i)$ that describes the error between M and $G_i \in G$. The accuracy of a segmentation is the best match with any corresponding ground truth segmentation, i.e., the match that gives the minimum error:

$$\mathcal{E}_{\text{best}}(M, G) = \min_{G_i \in G} \mathcal{E}(M, G_i) \quad (4.1)$$

The premise of the above measure appears reasonable, but there are some subtleties. Recall that we plan to determine the base rate for accurate segmentation by finding the mean error between pairs of segmentations of the same scene created by different subjects. Similarly, we shall determine the base rate for inaccurate segmentation as the mean error over a sample of segmentations of different scenes. Both base rates are therefore average error measures; comparing these with a minimum (i.e., Equation (4.1)) overestimates the comparative accuracy of a machine segmentation.

The second potential way to measure accuracy against multiple ground truths is to use the mean. The mean gives a more conservative estimate, and is directly comparable to the base rates. It has the added advantage of being more robust: adding additional ground truth for an image will not affect the estimate as dramatically as the minimum potentially can. The mean error for a machine segmentation is given by:

$$\mathcal{E}_{\text{mean}}(M, G) = \frac{1}{|G|} \sum_{G_i \in G} \mathcal{E}(M, G_i) \quad (4.2)$$

In the above we define accuracy in terms of error: small error denotes high accuracy. Since all our measures are in the range $[0, 1]$, we can convert an error measure to an accuracy measure, or an accuracy measure to an error measure, by subtracting it from one. For consistency, in the remainder of this chapter we use error measures.

4.7 Experiments

We carry out two separate experiments. The objective of the first experiment is to examine the statistical properties of the six selected evaluation measures, and to determine if they can differentiate between accurate and inaccurate segmentation. The objective of the second experiment is to use these measures to evaluate how accurately the four selected segmentation algorithms approximate perceptual grouping.

4.7.1 Experiment 1: Examining the Evaluation Measures

Our first task is to determine the base rates for accurate and inaccurate segmentation. According to the assumptions we outlined in the previous section, we can determine base rates for perceptually accurate segmentation by comparing pairs of segmentations of the same scene created by different subjects. Similarly, we can determine the base rates for inaccurate segmentation by comparing pairs of segmentations of different scenes created by either the same or different subjects.

The Berkeley segmentation dataset contains 3711 pairs of segmentations of the same scene created by different subjects. To guarantee maximally significant results, we determine the base error rate for accurate segmentation by evaluating each error measure over all of these pairs. We define the base error rate for

\mathcal{E}	min	max	median	mean	sd	skew	kurtosis
GCE	.00096	.42904	.05872	.07966	.06431	1.56270	2.62453
LCE	.00095	.24805	.04089	.04995	.03386	1.46581	2.73997
HDI	.00080	.42290	.10623	.12175	.08330	0.72115	-0.09587
FMI	.00081	.78288	.15360	.19557	.15855	0.88897	0.08252
JI	.00162	.95123	.27442	.31617	.22588	0.54253	-0.73274
RI	.00160	.75883	.06872	.11892	.12988	2.04665	4.46706

Table 4.3: Statistical properties for pairs of segmentations of the same scene.

\mathcal{E}	min	max	median	mean	sd	skew	kurtosis
GCE	.00782	.72880	.39262	.37548	.13624	-0.46685	-0.40064
LCE	.00393	.66759	.30660	.29506	.11325	-0.30871	-0.34907
HDI	.02826	.66046	.41059	.39953	.08546	-0.79052	1.27354
FMI	.05585	.82944	.60239	.58616	.11629	-0.93181	1.48339
JI	.10773	.96207	.76960	.75217	.10686	-1.57877	4.69401
RI	.09187	.92665	.35596	.38018	.14555	0.73450	0.14005

Table 4.4: Statistical properties for pairs of segmentations of different scenes.

accurate segmentation as the average over all pairs. Table 4.3 gives the computed means and other statistical properties of accurate segmentation for each measure.

The next step is to determine the base rates for inaccurate segmentation. We compute these using a random sample of pairs of segmentations of different scenes from the dataset. There are many more of these pairs than there are pairs of segmentations of the same scene. To simplify the statistics, and ensure no bias is introduced, we use a sample size equal to the sample size we used to determine the base rates for accurate segmentation. The base rate for inaccurate segmentation is then defined as the mean error over our random sample. Table 4.4 gives the computed means and other statistical properties for inaccurate segmentation.

Histogram plots for accurate and inaccurate segmentation are shown in Figure 4.5. The green bars correspond to pairs of segmentations of the same scene

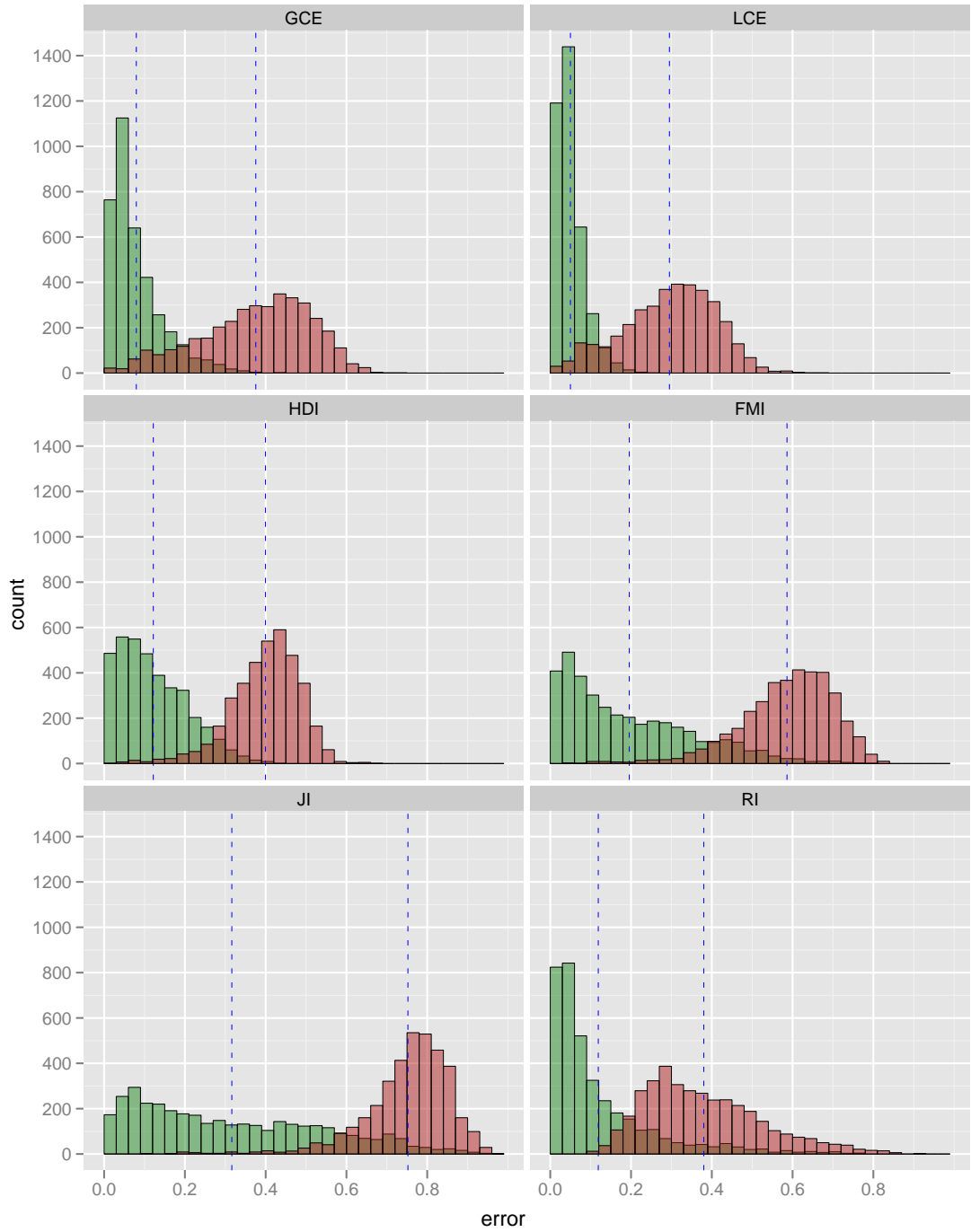


Figure 4.5: Histogram plots of accurate and inaccurate segmentation for the six evaluation measures. The green bars represent accurate segmentation: pairs of segmentations of the same scene by different subjects. The red bars represent inaccurate segmentation: pairs of segmentations of different scenes by the same or different subjects. The sample size for both accurate and inaccurate segmentation was 3711.

	accurate segmentation			inaccurate segmentation		
	μ_{low}	mean	μ_{high}	μ_{low}	mean	μ_{high}
GCE	0.0777	0.0797	0.0818	0.3699	0.3738	0.3775
LCE	0.0489	0.0500	0.0511	0.2908	0.2941	0.2972
HDI	0.1191	0.1217	0.1244	0.3971	0.3996	0.4019
FMI	0.1905	0.1956	0.2006	0.5831	0.5865	0.5897
JI	0.3089	0.3162	0.3233	0.7496	0.7527	0.7556
RI	0.1148	0.1189	0.1232	0.3770	0.3811	0.3852
LHE	0.0842	0.0859	0.0875	0.3442	0.3468	0.3494

Table 4.5: Confidence intervals for the mean error of accurate and inaccurate segmentation, computed using bootstrapping (BCa method, 5,000 samples, $p = 0.05$).

by different subjects (accurate segmentation); the red bars correspond to pairs of segmentations of different scenes (inaccurate segmentation). The dashed blue lines depict the mean error for accurate and inaccurate segmentation.

It is clear from the histograms in Figure 4.5, and the values in Table 4.3 and 4.4, that all the measures have distinct distributions for accurate segmentation versus inaccurate segmentation. We confirmed this using the Kolmogorov-Smirnov test, which rejects the null-hypothesis (i.e. the distributions are the same) in each case with high probability ($p < 10^{-12}$).

To determine if the mean values for accurate and inaccurate are significantly different, we need to estimate the confidence intervals. Unfortunately, the values of the error measures are not normally distributed, for either accurate or inaccurate segmentation. We tested this using the Shapiro-Wilks test, which rejects the null hypothesis with $p < 10^{-8}$ for all the measures: an unsurprising result given the skewness and kurtosis values. This non-normality implies we cannot use the standard parametric tests to determine the confidence intervals about the mean.

Bootstrapping [Chernick, 1999] provides a means of estimating confidence intervals for non-normal data. We estimated the mean confidence intervals for ac-

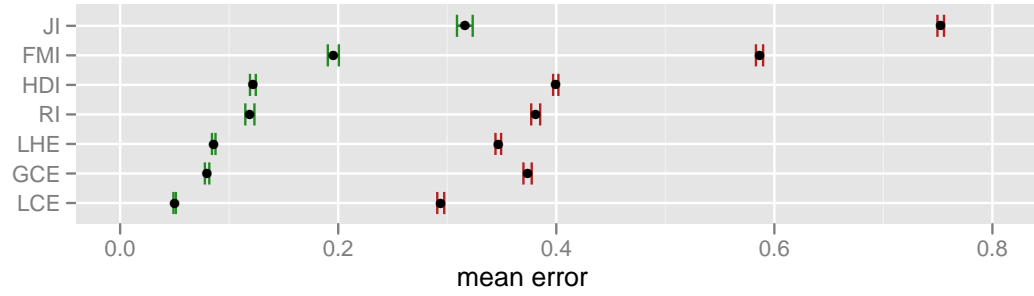


Figure 4.6: Means and confidence intervals for each of the measures. The red bars represent inaccurate segmentation; the green bars represent accurate segmentation.

accurate and inaccurate segmentation with 95% confidence using 5,000 replications and BCa intervals [Moore and McCabe, 2005] to adjust for bias and skewness. Table 4.5 shows the resulting confidence intervals. The intervals for accurate and inaccurate segmentation are non-overlapping for all of the measures tested (see Figure 4.6). We therefore conclude that the mean error given by each measure for accurate segmentation is significantly different from mean error for inaccurate, or random, segmentation. This answers the question as to whether any of the measures are useful for determining the perceptual accuracy of a segmentation: given enough samples, all of the measures are able to differentiate between a set of segmentations created by a human and a set of random segmentations.

We next investigate the inter-measure correlation. To compute the correlation among the measures we first concatenate the set of samples for accurate and inaccurate segmentation, and then compute the correlation between the concatenated samples. A non-parametric correlation statistic must be used as the measures are not normally distributed; we used Spearman’s rank correlation coefficient. Table 4.6 shows the inter-measure correlation (for clarity, we omit the lower triangular).

The correlation between many of the measures is very high, an indication that there is significant redundancy. High correlation is, however, expected;

	GCE	LCE	HDI	FMI	JI	RI
GCE	—	0.965	0.839	0.789	0.726	0.501
LCE	—	—	0.856	0.793	0.739	0.507
HDI	—	—	—	0.978	0.961	0.736
FMI	—	—	—	—	0.992	0.765
JI	—	—	—	—	—	0.789
RI	—	—	—	—	—	—

Table 4.6: Inter-measure correlation.

all the measures have the same purpose: to estimate accuracy. The correlation is particularly high between GCE and LCE, and among HDI, FMI, and JI. The correlation of 99.2% between FMI and JI is especially conspicuous; the scattergram in Figure 4.7 shows that the relationship between the measures can be modeled with reasonable accuracy by a quadratic polynomial. Further investigation reveals that we can rewrite the Fowlkes and Mallows index as:

$$\mathcal{F} = \sqrt{W_1(C_1, C_2)W_2(C_1, C_2)} \quad (4.3)$$

$$= \frac{n_{11}}{\sqrt{(n_{11} + n_{10})(n_{11} + n_{01})}} \quad (4.4)$$

since using Equation (3.11), (3.12), and (3.13) $W_1(C_1, C_2)$ and $W_2(C_1, C_2)$ can be rewritten as,

$$W_1(C_1, C_2) = \sum_{i=1}^k \frac{n_{11}}{n_i(n_i - 1)/2} = \frac{n_{11}}{\frac{1}{2}[\sum_{i=1}^k n_i^2 - n]} \quad (4.5)$$

$$= \frac{n_{11}}{n_{11} + n_{01}} \quad (4.6)$$

$$W_2(C_1, C_2) = \sum_{j=1}^l \frac{n_{11}}{n_j(n_j - 1)/2} = \frac{n_{11}}{\frac{1}{2}[\sum_{j=1}^l n_j^2 - n]} \quad (4.7)$$

$$= \frac{n_{11}}{n_{11} + n_{10}} \quad (4.8)$$

Comparing Equation (4.4) with the Jaccard index from Equation (3.2), we see that the indices have equal numerators, and that the denominator in both indices is a

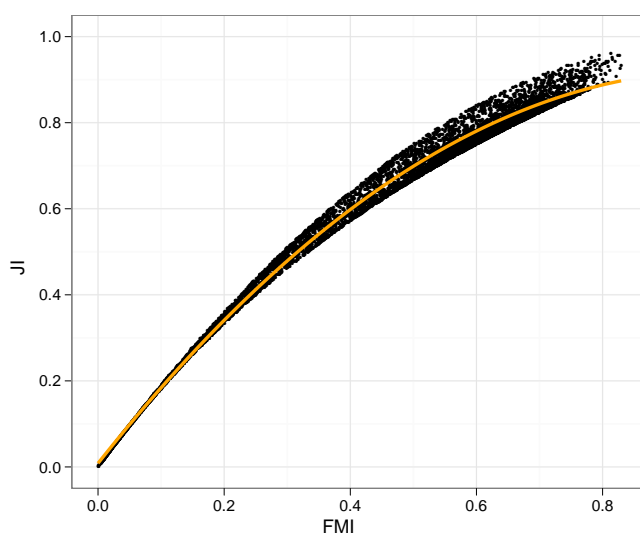


Figure 4.7: Scattergram showing the correlation between the Fowlkes-Mallows index and the Jaccard index. The orange line is the least squares quadratic fit ($y = -0.938x^2 + 1.859x + 0.008$).

combination of the same three terms: n_{11} , n_{10} , and n_{01} , which explains the high correlation observed.

Using all six measures for evaluating automatic segmentation is probably unnecessary. If possible, we would like to select one or two of the most effective measures. There are two reasons for this: the measures that exhibit high correlation give us very similar information, so using them all is redundant; the measures that exhibit low correlation disagree a lot, so are likely to be a source of contention in the analysis.

To select the most useful measures we must first define what makes a measure useful. Intuitively, we can reason that one measure is more useful than another if it is better at distinguishing between accurate and inaccurate segmentation. Following this reasoning, we could select the most useful measures to be those that have minimal intersection area between the probability distribution functions for accurate and inaccurate segmentation. Since these functions are not explicitly known, we would have to either (1) measure histogram overlap for ac-

curate and inaccurate segmentation from our sample sets, or (2) estimate the true distribution functions from our sample set and compute their intersection. The first approach is, unfortunately, sensitive to histogram quantization; the second approach involves approximations in both fitting the true distribution function and in computing the intersection.

Because of these issues, we take a different approach to determining the best measure. Any segmentation error measure \mathcal{E} can be turned into a binary classifier for segmentation accuracy by selecting an arbitrary threshold τ , and judging any segmentation with error $\mathcal{E} < \tau$ as accurate, and any with error $\mathcal{E} \geq \tau$ as inaccurate. The optimal τ for the sample set is the value that minimizes the error rate, which we define as the number of misclassifications. The number of misclassifications is equal to the number of false positives plus the number of false negatives. False positives are accurate segmentations that are classified as inaccurate; false negatives are inaccurate segmentations that are classified as accurate.

Since we penalize equally for false positives and false negatives, the error rate as defined above is equivalent to using a symmetrical zero-one loss function in Bayesian decision theory [Duda et al., 2001], and we seek the decision boundary that gives the minimum error rate for our sample. Because we selected the same sample size for accurate and inaccurate segmentation, we can measure the error rate directly for a given τ by adding the number of false positives to the number of false negatives—normalizing by sample sizes is unnecessary.

From our previous investigation we know the range of values for τ is bounded by the sample means. To determine the optimal τ we use a simple grid-search method: we test a large number (10,000) of equally spaced values between the sample means to find the one that gives the minimum error. Table 4.7 shows the resulting threshold values and the corresponding error for each of the measures. It is clear from the table that the measure with the lowest overall error is the HDI

	FP	%	FN	%	E	%	τ
GCE	250	6.74	427	11.51	677	9.12	0.194346
LCE	140	3.77	360	9.70	500	6.74	0.126437
HDI	192	5.17	267	7.19	459	6.18	0.275044
FMI	486	13.10	233	6.28	719	9.69	0.398542
JI	575	15.49	208	5.60	783	10.55	0.576078
RI	733	19.75	185	4.99	918	12.37	0.187306

Table 4.7: Accurate/Inaccurate thresholds discovered using grid-search. FP is the number of false positives; FN is the number of false negatives; E is the sum of FP and FN. The thresholds τ are found by searching for the value that minimizes the error E.

measure; the LCE measure gives the second lowest error. The measure with the highest overall error is RI.

Based on the table, we select the measures with the lowest overall error for the evaluation, namely the HDI and the LCE measures. Tables 4.3 and 4.4 show that the selected measures have the smallest variance out of the set: the sum of standard deviations is 0.14711 for LCE and 0.16876 for HDI. The LCE measure has the lowest variance of the set for accurate segmentation; the HDI measure has the lowest variance of the set of inaccurate segmentation. The HDI measure is the only measure that has no significant difference between variance for accurate and inaccurate segmentation: the ratio of variance (Fisher’s F) is 1.0527. It is also the measure that best balances false positives and false negatives in Table 4.7: $\frac{FP}{FN} = 0.7191$. The LCE measure has the narrowest confidence interval for accurate segmentation, and gives the least false positives. The measures are also complementary: HDI penalizes refinement error but does not suffer from degenerate cases, LCE tolerates refinement error but is susceptible to two degenerate cases (see Section 3.3.2).

The above discussion begs the question as to whether it is possible to combine the LCE and HDI measure to produce a measure that is better at differentiating

γ	FP	%	FN	%	E	%	τ
0.4735	105	2.83%	193	5.20%	298	4.02%	0.195153
0.5000	102	2.75%	202	5.44%	304	4.10%	0.191318

Table 4.8: Prediction error using a linear combination of the LCE and the HDI measures. The optimal weighting for our sample is $0.473 < \gamma < 0.474$; the difference in error for $\gamma = 0.5$ is negligible.

between accurate and inaccurate segmentation. We found that it is indeed possible to improve on the values in Table 4.7 using a linear combination of the measures. To keep the resulting measure in the range $[0, 1]$, we investigated linear combinations of the form:

$$M = \gamma M_1 + (1 - \gamma) M_2 \quad (4.9)$$

where M_1 and M_2 are the LCE and HDI measures, and $0 < \gamma < 1$.

We used grid-search to determine the γ that gives the lowest prediction error for our sample, and found that $0.473 < \gamma < 0.474$ is optimal, giving an error $E = 298$ (4.02%). This value is very close to the error of 304 (4.1%) attained at $\gamma = 0.5$, which corresponds to the simple average of LCE and HDI (see Table 4.8). The difference is potentially due to the variation in the data; parsimony dictates that we select the simplest model. We therefore propose a new measure that is a simple average of the LCE and the HDI measures. We denote this new measure LHE, which stands for combined LCE HDI error:

$$\text{LHE} = \frac{1}{2} (\text{LCE} + \text{HDI}) \quad (4.10)$$

The histogram plot in Figure 4.8 illustrates the distribution of this measure. Table 4.9 summarizes the properties of the three measures we use for evaluating automatic segmentation in the next section.

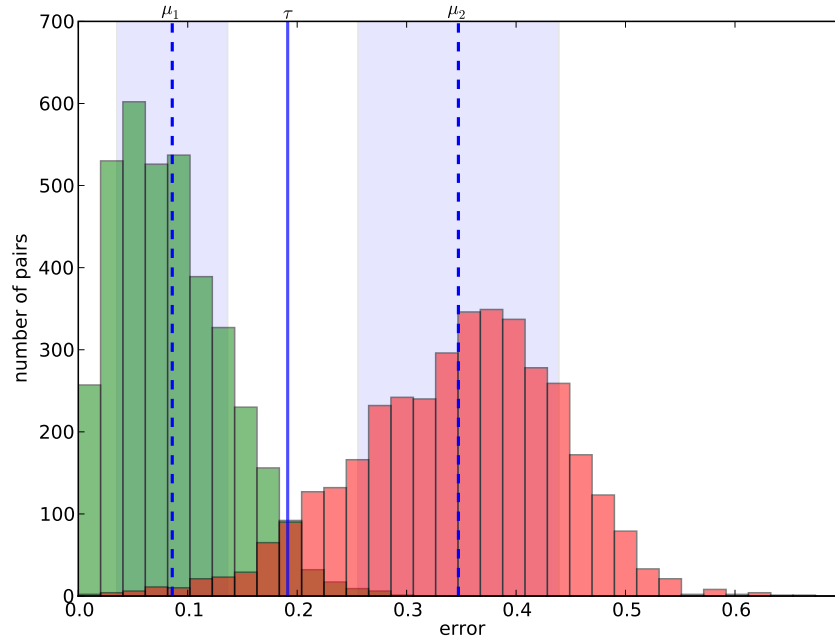


Figure 4.8: Histogram plots of accurate and inaccurate segmentation using the proposed LHE measure. The dashed blue lines are the means; the solid line in the center depicts the optimal classifier threshold τ .

Martin et al. also investigated the base rates for the GCE and LCE measures in their paper on the Berkeley segmentation dataset [Martin et al., 2001]. They found values for accurate segmentation that are slightly higher than the values we tested: they found the mean error for accurate segmentation to be 0.11 for GCE and 0.07 for LCE. We found the same values to be 0.08 (*c.i.* (0.0776, 0.0819), $p = 0.05$) for GCE, and 0.05 (*c.i.* (0.0487, 0.0511), $p = 0.05$). The apparent discrepancy is due to an important experimental difference. When establishing the mean error for accurate segmentation, Martin et al. compared pairs of segmentations that were created using grayscale versions of the images in the database; we used segmentations based on color images. In addition, Martin et al. used only a subset of the images in the dataset: at the time the dataset was incomplete.

The difference in mean error rates suggests that the use of color may improve the mutual consistency of segmentations created by different subjects. To check

	accurate segmentation				τ	inaccurate segmentation			
	sd	μ_{low}	mean	μ_{high}		μ_{low}	mean	μ_{high}	sd
LCE	.0339	.0487	.0500	.0511	.1264	.2908	.2941	.2973	.1133
HDI	.0833	.1190	.1217	.1244	.2750	.3971	.3996	.4020	.0855
LHE	.0506	.0841	.0859	.0875	.1913	.3442	.3468	.3495	.0918

Table 4.9: Properties of selected measures

this hypothesis, we ran the evaluation against all pairs of segmentations of the same scene by different subjects in the grayscale version of the database (3726 pairs). We found the mean error to be 0.083 (*c.i.* (0.0798, 0.0861), $p = 0.05$) for GCE, and 0.052 for LCE (*c.i.* (0.0492, 0.0626), $p = 0.05$). There is a small increase in mean error, but it is not significant. The Kolmogorov-Smirnov test also shows that the difference in the distribution of the error values is insignificant (the null hypothesis, that the distributions are the same, cannot be rejected: LCE: $p = 0.23$; GCE: $p = 0.071$; HDI: $p = 0.175$; LHE: $p = 0.203$). We therefore conclude that the mutual consistency for segmentations of the same scene by different subjects is not significantly affected when color is discarded.

The discrepancy between our findings and those in Martin et al. is likely due to the difference in sample size. Martin et al. state that they used 50 images with approximately 3 ground truth segmentations per image to establish the base rates for accurate and inaccurate segmentation. Assuming exactly three ground truth gives us $\binom{3}{2} \times 50 = 150$ pairs of segmentations of the same scene by different subjects, compared with the 3711 samples we used. Fewer samples generally produce wider confidence intervals that potentially explain the discrepancy. Unfortunately, we are unable to verify this as Martin et al. did not give confidence intervals in their paper. Martin et al.’s mean error rates for inaccurate segmentation do, however, agree with ours, and it is clear from the histograms in their paper that a larger sample size was used to establish the mean error for inaccurate segmentation.

4.7.2 Experiment 2: Evaluating the Algorithms

We now turn to evaluating the automatic segmentation algorithms. All of the images in the dataset were segmented by each of the four segmentation algorithms in turn, producing 300 machine segmentations per algorithm. We used the default algorithm parameters proposed by the original authors to perform the segmentations. Figure 4.9 gives some examples of the segmentations that the segmentation algorithms produced. Each of these segmentations was then compared with all of the corresponding ground truth segmentations using the evaluation measures we selected in the last section. There are an average of 5.443 ground truth segmentations per image, giving a total of 1633 comparisons per algorithm.

The first step in the analysis is to determine, for each of the four segmentation algorithms, the mean segmentation error corresponding to each image in the dataset (see Equation (4.2)). We can then compare the distribution of these error values with the distribution of the same values for accurate and inaccurate segmentation.

There is a subtle difference from the previous experiment in how we compare distributions. Previously, we compared the distribution of error values for every available pair of accurate segmentations with a random sample of the same number of pairs of inaccurate segmentation. This approach is appropriate in the context of the previous experiment: we are comparing the error distributions, not the mean error per image. When evaluating machine segmentation it is more appropriate to first determine average error for each image across the available ground truth for that image, then examine the distribution of these values. First determining the mean error for each image is more robust to outliers, and more sensitive to consensus, within the set of ground truth for that image.



Figure 4.9: Sample images and machine segmentations. The leftmost image is the original. The algorithms that were used to create the segmentations are, from left to right: MRSST, SRAG, SRM, and MSHIFT.

In effect, the above implies that a direct comparison of the pre-averaged error distributions for machine segmentation with the error distributions for accurate or inaccurate segmentation from the previous experiment is inappropriate. Instead, we first average across all ground truth segmentations of the same image to create a compatible distribution for accurate segmentation. The distribution for inaccurate segmentation we found in the last section was based on a uniform random sample of pairs of segmentations of different images. We can therefore create a compatible distribution of inaccurate segmentation in a similar way as for accurate segmentation; for each pair of segmentations from our sample containing a segmentation of image x , we average across all other pairs containing a segmentation of image x . The result is, for each image in the dataset, one mean error value that corresponds to accurate segmentation, and one mean error value that corresponds to inaccurate segmentation. These error values will be distributed slightly differently to those from the previous section, but the distributions are very similar overall, and are directly comparable with the error values for each segmentation algorithm.

Figure 4.10 graphically depicts the distribution of error for each of the segmentation algorithms evaluated. For each of the algorithms, the plots on the left depict the maximum, minimum, median, and inter-quartile ranges of each of the error measures. The plots on the right use jitter to depict the distribution of these same values. The error values from the four machine segmentation algorithms are centered in the plots; the leftmost and rightmost parts depict the distribution of error values for accurate and inaccurate segmentation. Table 4.10 describes the evaluation results in more detail.

Our first task is to determine whether any of the segmentation algorithms perform significantly better than random. Table 4.10 shows that all of the algorithms give a lower mean error, and a lower median error, than inaccurate segmenta-

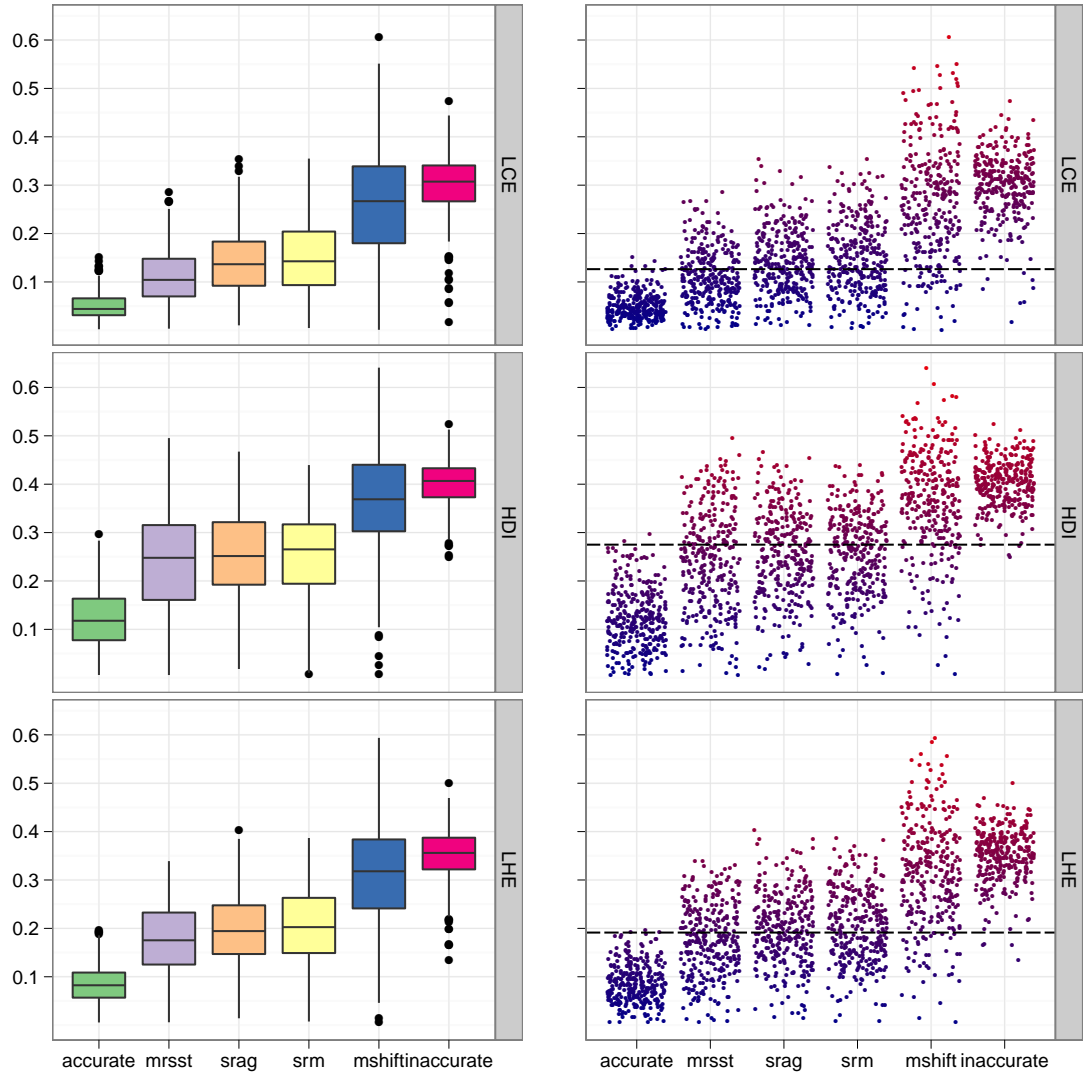


Figure 4.10: Boxplots (left) and jitter plots (right) of segmentation error for each algorithm. In the boxplots, the black line inside each of the boxes represents the median. The hinges depict the first and third quartiles. The lines and dots represent the range and outliers. The dashed black lines in the jitter plots are the prediction thresholds from the previous section.

		min	max	med	mean	sd	skew	kurt
LCE	accurate	0.0022	0.1505	0.0442	0.0501	0.0269	1.0049	0.9834
	MRSST	0.0034	0.2855	0.1043	0.1110	0.0576	0.4738	-0.1074
	SRAG	0.0103	0.3541	0.1366	0.1417	0.0658	0.4686	0.0347
	SRM	0.0048	0.3549	0.1427	0.1509	0.0742	0.3596	-0.4598
	MSHIFT	0.0011	0.6066	0.2669	0.2644	0.1186	0.1939	-0.3156
	inaccurate	0.0168	0.4740	0.3073	0.2983	0.0670	-0.9541	2.0760
HDI	accurate	0.0057	0.2973	0.1178	0.1226	0.0619	0.4061	-0.3059
	MRSST	0.0056	0.4953	0.2479	0.2431	0.1052	-0.0289	-0.6889
	SRAG	0.0181	0.4673	0.2515	0.2548	0.0913	-0.1663	-0.3203
	SRM	0.0085	0.4395	0.2653	0.2570	0.0858	-0.3173	-0.2349
	MSHIFT	0.0067	0.6409	0.3688	0.3627	0.1083	-0.5207	0.3856
	inaccurate	0.2488	0.5252	0.4067	0.4028	0.0472	-0.3673	0.2791
LHE	accurate	0.0055	0.1966	0.0824	0.0863	0.0404	0.4442	-0.2258
	MRSST	0.0059	0.3390	0.1754	0.1770	0.0751	-0.0453	-0.6544
	SRAG	0.0142	0.4039	0.1944	0.1983	0.0742	0.0675	-0.1184
	SRM	0.0075	0.3868	0.2025	0.2040	0.0755	0.0108	-0.4660
	MSHIFT	0.0065	0.5937	0.3180	0.3135	0.1089	-0.0755	-0.0988
	inaccurate	0.1355	0.4996	0.3560	0.3505	0.0552	-0.7523	1.3304

Table 4.10: Results of evaluating the algorithms using the three evaluation measures (*med*=median, *kurt*=kurtosis).

tion. To determine whether this difference is significant, we first establish the confidence intervals for each of the mean error rates. If the confidence intervals for two mean error rates do not overlap then the difference in the mean error rate is significant. More precisely, assuming that our dataset is representative of the segmentation problem (the population), if we establish the confidence intervals using the standard significance level of $p = 0.05$, then the probability that two means with non-overlapping confidence intervals are actually the same is $p < 0.025$. If, however, the confidence intervals *do* overlap, then further testing is required to establish if the difference in the means is significant.

As with the previous section, we use bootstrap resampling to establish confidence intervals for the means. We determine the confidence intervals using a significance of $p = 0.05$, 10,000 replications, and BCa intervals. Figure 4.11 depicts

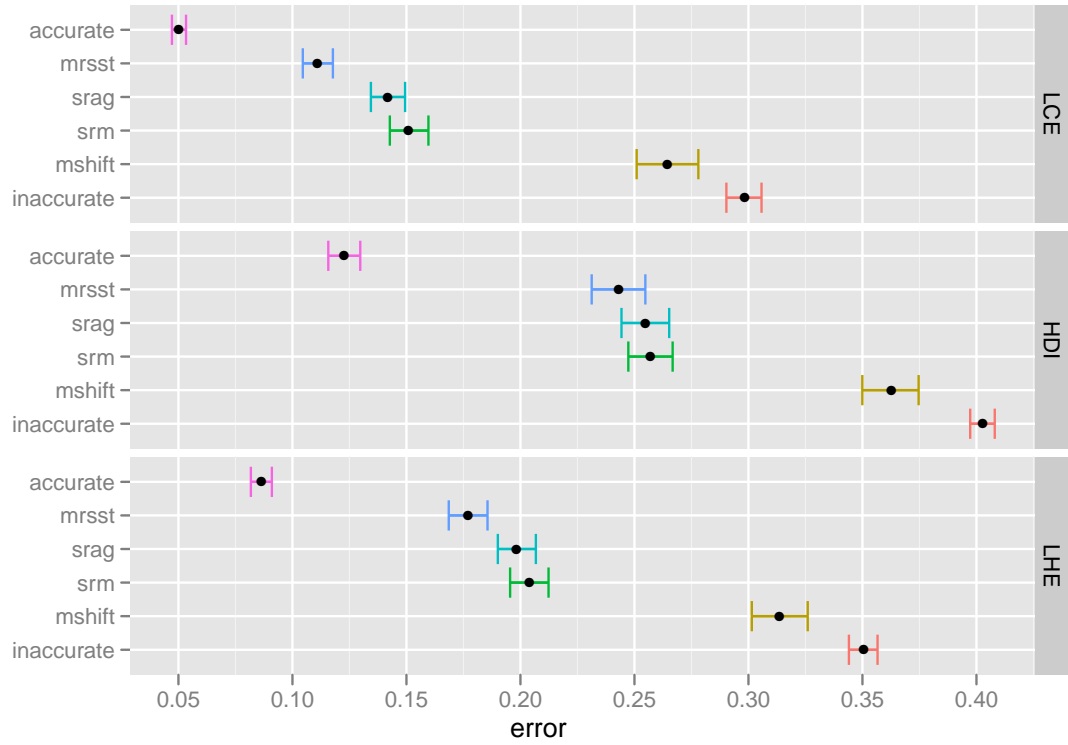


Figure 4.11: Mean error and confidence intervals for each of the algorithms. The black points represent the means and the bars represent the confidence intervals. The confidence intervals were established using bootstrapping (BCa method, replication = 10,000, $p = 0.05$).

the resulting confidence intervals. The following conclusions apply to all of the measures used:

- Segmentations from all four machine segmentation algorithms have significantly lower mean error than random.
- Segmentations from all four machine segmentation algorithms have significantly higher mean error than segmentations created by humans.
- Segmentations from the MRSST, SRAG, and SRM algorithms have significantly lower mean error than segmentations from the MSHIFT algorithm.

In addition, the LCE and the LHE measures show the mean error for MRSST to be significantly lower than the mean error for SRAG and SRM.

We next turn to examining the error means for algorithms with overlapping confidence intervals, namely: MRSST, SRAG, and SRM. To establish whether two means are significantly different, we use the two sample t-test. We use the paired version of test: mean error is determined on a per image basis using the same ground truth, so the samples are dependent and paired for each image. The t-test assumes that the data are normally distributed. The error measures for the three algorithms under consideration are approximately, but not exactly, normally distributed (see Figure 4.12). However, we have enough samples to assume the standard error of the mean *is* normally distributed, therefore the t-test is appropriate.

The t-test shows that the difference in mean error between SRAG and SRM for the HDI measure is not significant ($p = 0.6149$). The difference in mean error between MRSST and SRAG is significant ($p = 0.004$), as is the difference in mean error between MRSST and SRM ($p = 0.0025$). For the LCE measure the difference in means is significant among all three algorithms (in every case $p < 0.009$). For the LHE measure the difference is not significant between SRAG and SRM ($p = 0.1041$), but significant between MRSST and SRAG, and between MRSST and SRM ($p < 1.6 \times 10^{-12}$).

The only contention in the above is with the LCE measure: it indicates a significant difference in mean error between SRAG and SRM algorithms; the HDI and LCE measures indicate the difference is not significant. The difference in error is very small ($< 1\%$), and so we accept the consensus of the other two measures: that the two algorithms perform equally well on average. We therefore conclude that, with regard to mean error:

- The MRSST algorithm significantly outperforms the SRAG and SRM algorithms;
- There is no significant difference between the SRAG and SRM algorithms.

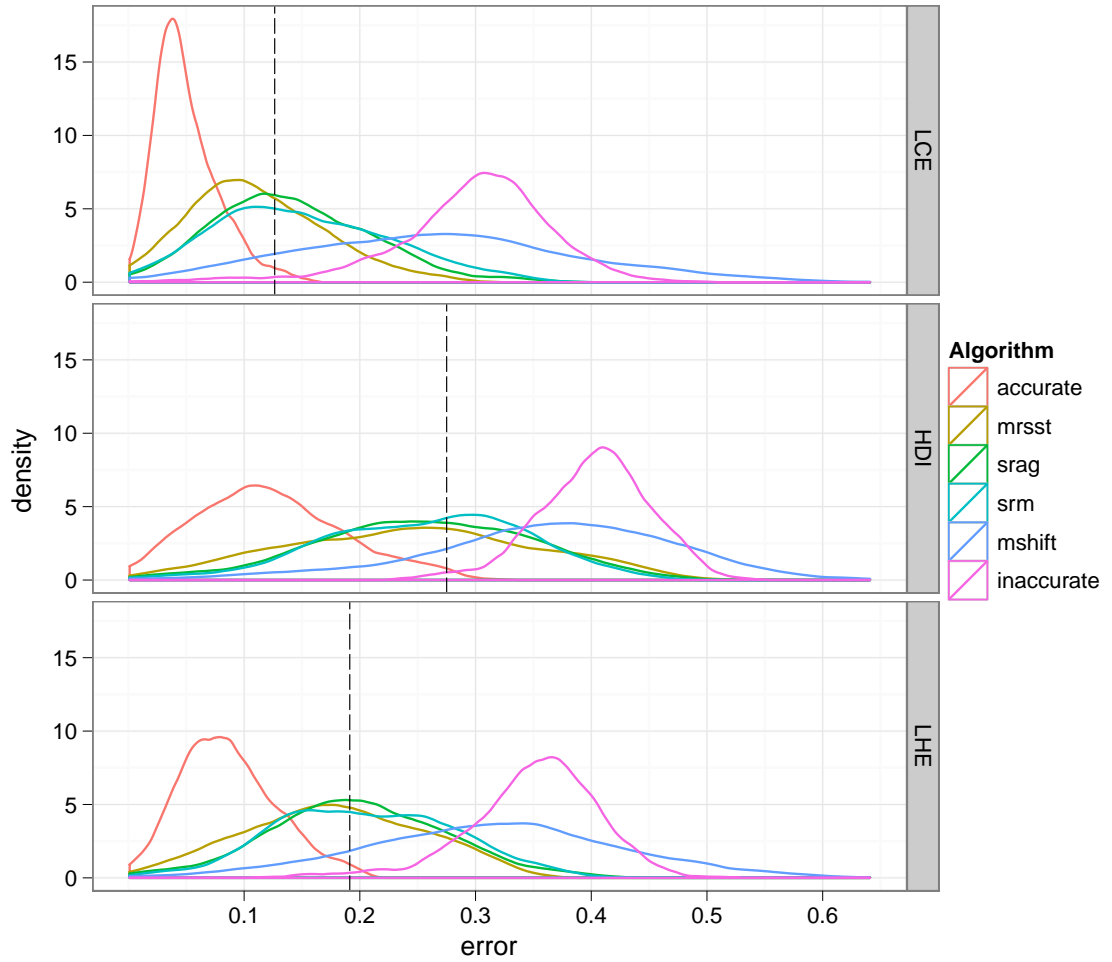


Figure 4.12: Estimated error distribution density for each algorithm. Density was estimated using the Epanechnikov kernel. The dashed black line is the prediction threshold from the previous section.

We conclude the analysis by ranking the algorithms according to the number of accurate segmentations they produce as judged by a simple binary classifier. We use the LHE measure and the corresponding prediction threshold we established in the previous experiment as the classifier. A segmentation is predicted to be accurate if the error when compared against a ground truth is below the prediction threshold. We perform two variants of the experiment. In the first, we measure each segmentation's error against all corresponding ground truth and average the result. The segmentation is judged to be accurate if this average is less

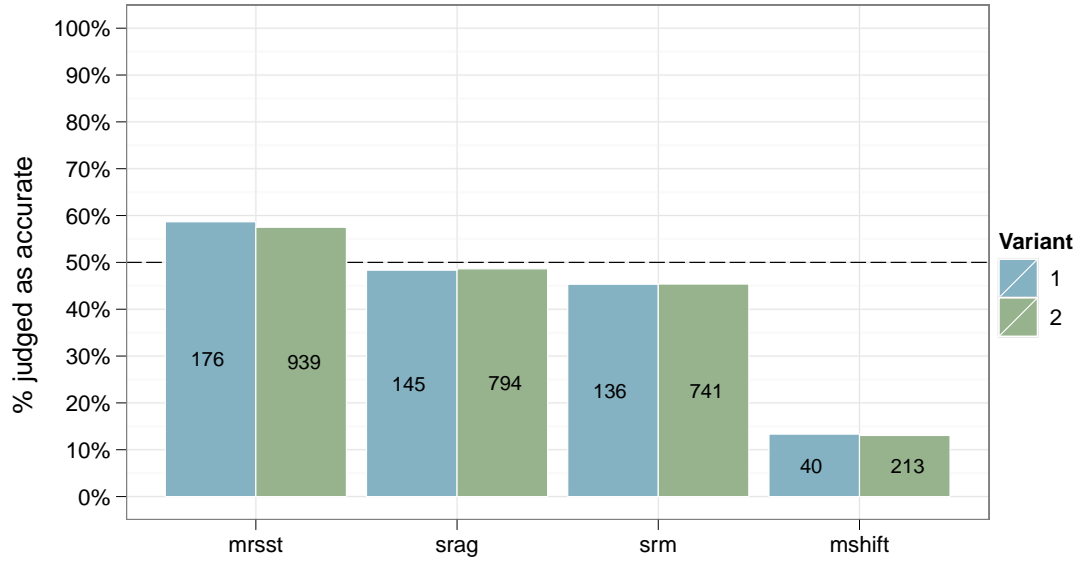


Figure 4.13: Prediction accuracy for each of the four segmentation algorithms. The values inside the bars are the total counts. For variant one, this value is the total number of images (out of 300); for variant two, it is the number of ground truth (out of 1633).

than the prediction threshold. The ranking is given by the number of machine segmentations deemed accurate.

In the second variant, each ground truth segmentation is compared against the corresponding machine segmentation. The ground truth is judged to be an accurate representation of the machine segmentation if the error is less than the prediction threshold. In other words, the first variant counts the number of machine segmentations whose *average* error over all corresponding ground truth is less than the prediction threshold; the second variant counts the total number of ground truth segmentations that, when compared to the machine segmentation, give an error below the prediction threshold.

Figure 4.13 shows the proportion of segmentations judged to be accurate for both variants. Ranking by either variant gives the same ordering: MRSST, SRAG, SRM, and MSHIFT. The ranked order from the classifier agrees also agrees with

the order that would be given if the algorithms were ranked by mean error, or median error.

Martin et al. evaluate the normalized cuts algorithm using the LCE measure in [Martin et al., 2001], and report that the mean error for the algorithm is 0.22. This indicates that the normalized cuts algorithm performs worse than the MRSST, SRAG, and SRM algorithms, but better than the MSHIFT algorithm. Note, however, that Martin et al. used grayscale images in their evaluation.

4.8 Discussion

In this chapter we investigated automatic segmentation evaluation by performing two separate, but complementary, experiments. In the first experiment we analyzed the characteristics of six supervised segmentation evaluation measures. For the analysis, we assumed that segmentations of the same scene by different subjects are mutually accurate, and that segmentations of different scenes are mutually inaccurate. These assumptions provided a means to estimate the distribution of the six measures for accurate and inaccurate segmentation. The distributions showed that, given enough samples, each of the measures can be used to distinguish between perceptually accurate segmentation and “random” segmentation. Our analysis also revealed that the measures are highly correlated; the Jaccard index and the Fowlkes and Mallows index exhibit particularly strong correlation. The high correlation indicates that using all of the measures for an evaluation is unnecessary; we therefore selected the two measures that were best able to distinguish between accurate and inaccurate segmentation: the local consistency error measure (LCE) and the Huang-Dom index (HDI). We then investigated if a linear combination of these measures (LHE) could produce a measure better able to distinguish accurate from inaccurate segmentation; we

found that a simple average of the two measures resulted in a new measure that is 2% better at distinguishing accurate from inaccurate segmentation.

The second experiment focused on using the selected measures to evaluate four modern segmentation algorithms. We used the measures to compare the segmentations created by each of the algorithms against a human generated ground truth. The analysis revealed that all of the algorithms produce segmentations that correspond better to the ground truth on average than do randomly selected segmentations: the algorithms perform significantly better than random. We also showed that, with regard to our measures, the mutual consistency among segmentations of the same scene by different human subjects is significantly higher on average than the consistency between human and machine segmentations; that is, all of the algorithms performed significantly worse than humans. The MRSST algorithm was shown to be significantly more accurate on average than the other algorithms; the MSHIFT algorithm was shown to be significantly less accurate. The SRAG and SRM algorithms demonstrated no significant difference in average accuracy. Ranking the algorithms by mean error, median error, or binary classifier accuracy all produced the same result: the best performing algorithm is MRSST, followed by SRAG, SRM, and MSHIFT. The rankings given by all three measures are consistent.

Based on these experiments we have several recommendations. First, for future segmentation evaluation tasks, it not necessary to use all three of the evaluation measures we used in Section 4.7.2: the experiment shows that the measures, in general, give very similar results. For general purpose segmentation evaluation tasks we recommend using the combined local consistency error-Huang-Dom index (LHE) measure: it gives the lowest prediction error when used as a classifier of segmentation accuracy, does not suffer from degenerate cases like the LCE measure, and provides moderate refinement tolerance. If, however, refinement tolerance is a priority, and degenerate cases are explicitly checked,

then we recommend using the LCE measure. If, on the other hand, the target application is particularly sensitive to over or under segmentation, we recommend using the HDI measure, as it penalizes refinement more severely.

Our second recommendation is for system designers that need to select an appropriate segmentation algorithm for a particular application. If the system requires a segmentation algorithm that closely approximates human perceptual grouping then, based on the experiments, we believe the MRSST algorithm to be a sound choice: it performed significantly better than the other algorithms in the experiment. This recommendation only applies to systems that handle images of which the Berkeley segmentation dataset is representative; our experiments assumed the dataset to be representative of the population: violating this assumption renders our conclusions inapplicable.

Chapter 5

Evaluating Interactive Segmentation

5.1 Introduction

In the previous chapter we focused on evaluating automatic segmentation algorithms; in this chapter we turn our attention to evaluating interactive segmentation algorithms. Our objective is to develop measures, tools, and techniques that allow us to effectively gauge the performance of scribble driven interactive segmentation algorithms.

At first glance, one might consider simply taking methods and measures from automatic segmentation evaluation and applying them to evaluating interactive segmentation. Closer inspection, however, reveals that automatic and interactive segmentation have quite different objectives. Automatic segmentation usually generates regions; methods for judging accuracy are complicated by issues like overlapping regions and refinement tolerance. Interactive segmentation usually generates semantic objects; compensating for overlapping regions or refinement tolerance is therefore unnecessary, and measures used to evaluate interactive segmentation should reflect this. Accuracy, on the other hand, is expected to be higher for interactive segmentation; measures for evaluating interactive segmentation require greater sensitivity. Automatic segmentation evaluation is primarily

concerned with judging how accurate a segmentation is. Interactive segmentation is concerned not only with accuracy, but also with efficiency: the time and effort necessary to achieve a particular level of accuracy in a segmentation. Furthermore, the interactions themselves may have a pronounced effect on the result, and so the sensitivity of the algorithm to these interactions is also of concern in an evaluation.

Interactive segmentation evaluation has received scant attention in the literature to date; almost all evaluation techniques and studies have focused exclusively on automatic segmentation. The few studies that have been undertaken focus on evaluating interactive segmentation in medical imaging (for example: [Mao et al., 1999]); however, to our knowledge there has been no research to date targeted specifically at evaluating general purpose interactive segmentation techniques.

In the remainder of this chapter we take the initial steps toward developing a complete system for supervised evaluation of interactive segmentation. We focus on four interactive segmentation algorithms suitable for extracting objects from natural scenes, carrying out the evaluation via a series of user experiments. The main contributions of this research are as follows: first, a software platform designed for hosting and evaluating different segmentation algorithms in a uniform environment. The platform includes four state-of-the-art interactive segmentation algorithms at present, and is available for public download from our website¹. Second, a ground truth dataset created specifically for evaluating interactive segmentation. The dataset comprises 100 objects from natural images with accompanying descriptions, and is also available on-line. Third, we propose and investigate two measures appropriate for evaluating interactive segmentation, including a new benchmark specifically designed to measure boundary accuracy against a ground truth. We compare the suggested measures with other measures that could potentially be used for interactive segmentation evaluation, and

¹<http://kspace.cdvp.dcu.ie/public/interactive-segmentation/>

demonstrate their relative effectiveness. Finally, we evaluate and compare four popular interactive segmentation algorithms using the proposed methodology, and thus demonstrate their performance and characteristics.

The remainder of the chapter is organized as follows. Section 5.2 discusses the four segmentation algorithms that we selected for the evaluation, and the reasons why we chose these particular algorithms. Section 5.3 outlines the objectives of the evaluation and the various considerations that need to be addressed. Having outlined these issues and objectives, we look at selecting a set of measures suitable for evaluating interactive segmentation, and formulate a new benchmark for measuring object boundary accuracy. Section 5.4 discusses the user experiments, including details of the participants involved and the software and tools used. In particular, this section describes the interactive segmentation tool developed to host the various algorithms, the dataset and ground truth used for the experiment, and the experiment setup and deployment strategy. Section 5.5 analyzes the results of the experiment, validates the selected evaluation measures, and demonstrates the relative performance of the four segmentation algorithms. Finally, Section 5.6 summarizes the work and presents our conclusions.

5.2 Algorithms

Different segmentation algorithms are often created with different application domains in mind, and are thus suited to different tasks. For example, some algorithms, such as active contours [Kass et al., 1988] and other similar approaches [Liang et al., 1999], are most effective at extracting regions of interest from medical images. Other algorithms, such as GrabCut [Rother et al., 2004], are designed for photo-editing applications and extracting objects from photographs of natural scenes. Due to the disparity of intended application, one cannot expect an algo-

Abbreviated Name	Algorithm
<i>SRG</i>	Seeded Region Growing
<i>IGC</i>	Interactive Graph Cuts
<i>SIOX</i>	Simple Interactive Object Extraction
<i>BPT</i>	Interactive Segmentation using Binary Partition Trees

Table 5.1: The evaluated algorithms and their abbreviated names

rithm designed for, say, biomedical image analysis to be equally effective when applied to a different domain, such as photo-manipulation.

Our evaluation focuses on interactive segmentation techniques appropriate for object extraction from photographs and natural scenes. Specifically, we only evaluate algorithms whose interactions can be modeled by pictorial input on an image grid [Olabarriaga and Smeulders, 2001]; we do not consider interactive segmentation algorithms based on parameter tuning or other forms of interaction. By narrowing our focus thus, we evaluate algorithms that are more directly comparable; the intention being a consistent and fair evaluation, albeit on a smaller subset of the available algorithms.

We chose four algorithms for the evaluation. The algorithms we selected provide good coverage of the various underlying algorithmic approaches used by current methods in the literature for object extraction from natural scenes. Table 5.1 lists the selected algorithms and assigns them abbreviated names that, for brevity, are used in the subsequent sections. Each of these algorithms was discussed in detail in Chapter 2. Note that we do not consider algorithms based on thresholding or deformable models, as the former cannot be adapted in a straightforward way to pictorial input, and the latter tends to perform better on medical images than on natural scenes.

5.3 Evaluation

In this section we discuss the methods and measures used for the evaluation. To effectively evaluate interactive segmentation, we need to consider three criteria [Olabarriaga and Smeulders, 2001]:

Accuracy: the degree to which the delineation of the object corresponds to the truth;

Efficiency: the amount of time or effort required to perform the segmentation; and

Repeatability: the extent to which the same result would be produced over different segmentation sessions when the user has the same intention.

This section is concerned with measuring accuracy; efficiency and repeatability are considered in Section 5.4 and 5.5. Nevertheless, it is important to note that for interactive segmentation the criteria are highly related. In particular, accuracy and efficiency are interdependent: given more time users can usually produce more accurate segmentations.

5.3.1 Human Factors

As we noted in the introduction, interactive segmentation is sufficiently different from automatic segmentation to warrant a distinct approach to its evaluation. The most important difference between automatic and interactive segmentation algorithms is, of course, that interactive segmentation algorithms require a human operator. The interactions provided by this operator usually have a pronounced affect on the resulting segmentation: good markup is usually needed to find a good segmentation. Clearly this is to be expected—if the interactions did not have such a profound affect on the result, they could be provided automatically, thus eliminating the need for human supervision.

The introduction of this human operator in the segmentation procedure requires several considerations. The nature of the image regions that human operators typically extract is different from those extracted by automatic methods. Humans typically desire more complex and meaningful semantic objects: a tree, a car, a person, or a face. Fully automatic algorithms, however, typically only parse images into regions of homogeneous color or texture, which may or may not correspond to a semantic object. Also, for many applications, such as photo-editing, people require very precise objects. For instance, if we wish to replace the background in an image, the segmented boundary of the object of interest needs to be highly accurate for the effect to be convincing. The required accuracy for this kind of application is higher than that usually required by applications that use automatic segmentation, such as multimedia indexing and retrieval.

The necessity for accurate semantic objects has direct consequences for evaluation. The accuracy requirement means that the measures we use to gauge performance must be sufficiently sensitive to any noticeable variation in object boundary precision. The need for semantic objects means unsupervised evaluation techniques [Zhang et al., 2008] and measures of empirical goodness are inappropriate: the features that characterize good semantic objects are decidedly more difficult to measure without ground truth than those that characterize good homogeneous regions.

5.3.2 Evaluation Measures

As unsupervised evaluation techniques are inappropriate for interactive segmentation, we will use supervised evaluation. This necessitates the creation of a ground truth dataset for the evaluation. Further details of the ground truth dataset we developed are discussed in Section 5.4.2, however, we will discuss one aspect in this section, as it is pertinent to the evaluation measures we develop.

The creation of a pixel accurate ground truth is, in general, impossible for natural images; alpha blending of pixels along the edges of objects make the true border position unattainable with absolute certainty. Our performance measures therefore need to balance the need for sensitivity to border variation with the inherent uncertainty in the boundary pixels of objects in the ground truth.

Aside from the imprecise nature of the object border pixels, it is also intuitively desirable for any measure we use to penalize a small imprecision near the object border less than, say, a large hole or missing piece of the object. Furthermore, as the objective of interactive segmentation is typically to extract some perceived object from a scene, our evaluation measures should reflect in some sense, the perceived accuracy of the segmentation i.e., there should be a correlation between measured accuracy and perceived accuracy.

It is also valuable to have an evaluation measure that is easy to interpret and compare. As such, it is desirable for any measure we use to be appropriately normalized in the interval $[0..1]$. For consistency, we define all employed measures as similarity functions (performance indicators): values closer to 1 indicate a better segmentation.

5.3.3 Boundary Accuracy

We now develop a means of measuring object boundary accuracy against a ground truth. Let $\mathbf{v} \in \mathbb{Z}^2$ be any pixel inside the ground truth object, and $G_O = \{\mathbf{v}\}$ be the set of all of these pixels. Similarly, define M_O to be the set of all pixels in the machine-segmented object. G_B and M_B denote the complements of these sets. Let \mathcal{N}_x be the standard set of 8-neighbors of any $\mathbf{x} \in \mathbb{Z}^2$. The internal border pixels for the ground truth object are defined as the set B_G , and for the

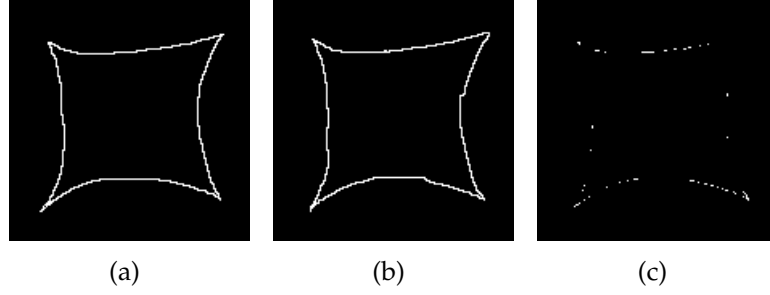


Figure 5.1: The internal border pixels of two similar objects (a), (b), and the pixels they have in common (c). The binary Jaccard accuracy measure \mathcal{A}_B is only 0.1. The fuzzy Jaccard measure for the same objects $\tilde{\mathcal{A}}_B$ is 0.85 when the bandwidth parameter $\sigma = 4$.

machine-segmentation, the set B_M , as follows:

$$B_G = \{\mathbf{x} : \mathbf{x} \in G_{\mathcal{O}} \wedge \mathcal{N}_{\mathbf{x}} \cap G_B \neq \emptyset\} \quad (5.1)$$

$$B_M = \{\mathbf{x} : \mathbf{x} \in M_{\mathcal{O}} \wedge \mathcal{N}_{\mathbf{x}} \cap M_B \neq \emptyset\} \quad (5.2)$$

Given the above definition of the border pixels, we could compute a measure of the accuracy of the border pixels as follows:

$$\mathcal{A}_B = \frac{|B_G \cap B_M|}{|B_G \cup B_M|} \quad (5.3)$$

Note that the value \mathcal{A}_B is equivalent to the Jaccard index [Ge et al., 2007]. Unfortunately, due to the previously discussed ambiguity in the positions of the boundary pixels in the ground truth, the value of \mathcal{A}_B will typically be excessively low. This is demonstrated in Figure 5.1. The object borders in 5.1(a) and 5.1(b) seem to be reasonably similar. Nevertheless, Figure 5.1(c) shows that the binary overlap between the pixels is quite small, resulting in a Jaccard index $\mathcal{A}_B = 0.1$. An additional problem is that small imprecisions near the object borders are penalized in equal measure to holes or missing pieces of the object.

To adapt the Jaccard index so that it is more appropriate for our purposes, we need to introduce some tolerance to error near the border pixels. A natural way of accomplishing this is to extend the definition of our sets of border pixels B_G and B_M using fuzzy-set theory [Zadeh, 1965] so as to capture the intrinsic uncertainty in the edge positions.

Of course, the degree of uncertainty, or tolerance, needs to be specified. Hence, it is necessary to introduce a parameter that quantifies the uncertainty, which we denote σ . Using this parameter, we propose to “fuzzify” the border pixel sets using the following Gaussian form:

$$\tilde{B}_G(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \hat{\mathbf{x}}\|^2}{2\sigma^2}\right) \quad (5.4)$$

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{y} \in B_G} \|\mathbf{x} - \mathbf{y}\| \quad (5.5)$$

The fuzzy set for the border of the machine segmentation \tilde{B}_M is similarly defined. The above definition effectively sets $\tilde{B}_G(\mathbf{x}) = 1$ for all $\mathbf{x} \in B_G$ with values decreasing with the Euclidean distance of \mathbf{x} from B_G at a rate controlled by the tolerance parameter σ . Moreover, the exponential function causes the value of $\tilde{B}_G(\mathbf{x})$ to approach zero for pixels that are a large distance from the border. This effect can be interpreted to mirror the saturation that has been observed in the human visual system: often it is easier for us to quantify small errors, but more difficult to quantify larger ones (for an example of this principle, see the discussion on color in 2.3.2). A representation of the function for different values of σ is shown in Figure 5.2.

Given the above fuzzy sets of border pixels \tilde{B}_G and \tilde{B}_M , we can reformulate the Jaccard index using fuzzy set theory as follows:

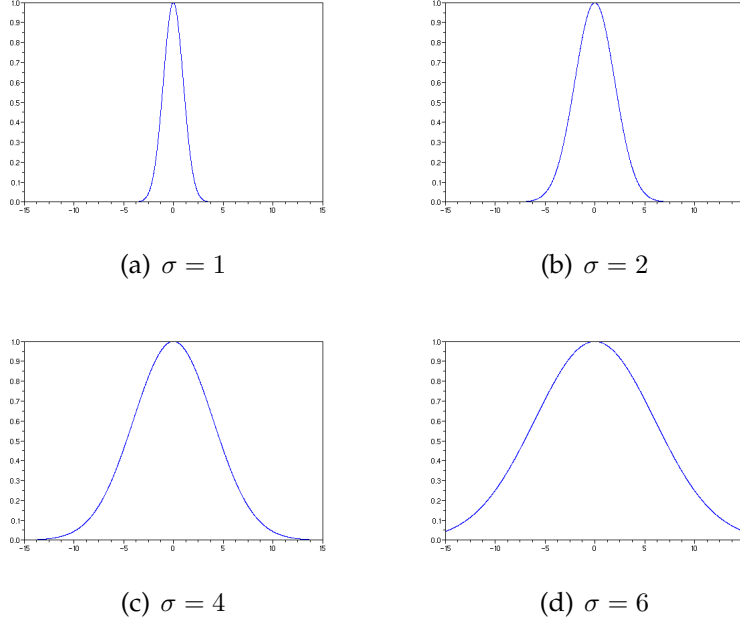


Figure 5.2: Representation of the fuzzy membership function for different tolerance parameters $\sigma = \{1, 2, 4, 6\}$

$$\tilde{\mathcal{A}}_B = \frac{\sum_{\mathbf{x}} \min(\tilde{B}_G(\mathbf{x}), \tilde{B}_M(\mathbf{x}))}{\sum_{\mathbf{x}} \max(\tilde{B}_G(\mathbf{x}), \tilde{B}_M(\mathbf{x}))} \quad (5.6)$$

The above formulation is already normalized in the desired range $[0..1]$, and takes the value 1 only for an exact match. Like the binary Jaccard index, the measure is symmetric, however, in contrast to the binary set formulation, close matches are now penalized proportional to the tolerance parameter σ . Also as σ approaches zero, $\tilde{\mathcal{A}}_B$ approaches the binary Jaccard index.

5.3.4 Object Accuracy

When considering the entire region accuracy, as opposed to the accuracy of the border, it is less important to “fuzzify” the evaluated sets. For regions, small inaccuracies around the border tend to be offset by larger overlapping areas, whereas for borders, the sets, even those very nearby spatially, may not strictly overlap at all. As such, we employ the previously described binary Jaccard

index to measure the object accuracy. This is consistent with our border accuracy measure, and also has the advantage of allowing the results presented herein to be directly compared with previous work in object-background based segmentation evaluation, such as [Ge et al., 2006, Ge et al., 2007]. The object accuracy measure is given by:

$$\mathcal{A}_O = \frac{|G_O \cap M_O|}{|G_O \cup M_O|} \quad (5.7)$$

5.3.5 Choosing Sigma

The fuzzy boundary accuracy measure requires appropriate selection of the tolerance parameter σ to regulate its sensitivity to error. The parameter should be chosen to reflect the degree of uncertainty of the object border pixels in the ground truth. If the parameter is too small, the measure becomes over-sensitive to inaccuracies in the ground truth, and will not reflect the perceived border accuracy. If the parameter is too large, the measure will not be sensitive enough to capture noticeable differences in precision.

For our experiments we chose a tolerance parameter of $\sigma = 4$. Using this value, pixels with a Euclidean distance less than 3 from the boundary are considered over 75% inside the boundary set, and pixels with a distance greater than 8 are less than 15% inside. The value was chosen empirically, based on a simple experiment. In the experiment, two different segmentations of the same object were chosen, one with a higher perceived accuracy than the other. For the two segmentations, the fuzzy boundary accuracy measure was computed using increasing values of sigma. From the resulting series, σ was chosen such that the difference between the computed values was consistent with the perceived difference in accuracy. The experiment was repeated fifteen times with the value 4 giving the most consistent result.

\mathcal{A}_O	Object accuracy (Jaccard index)
$\tilde{\mathcal{A}}_B$	Boundary accuracy (Fuzzy Jaccard index on border pixels)
\mathcal{A}_B	Binary boundary accuracy (Binary Jaccard index on border pixels)
$\mathcal{P}r$	Precision
$\mathcal{R}e$	Recall
$\mathcal{R}\mathcal{I}$	Rand Index

Table 5.2: Evaluation measures and their symbols

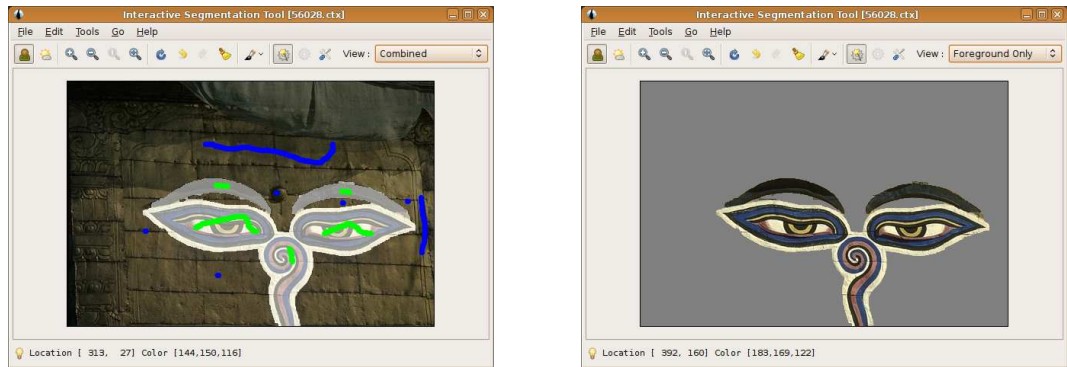
5.3.6 Other Measures

To validate the effectiveness of the selected measures, we also computed some other popular measures for comparison, including precision, recall, and the Rand index [Rand, 1971]. The computed evaluation measures and the corresponding symbols that will be used in the remainder of the text are shown in Table 5.2.

5.4 Experiment

In this section we discuss the evaluation experiment, detailing information about the participants involved, the software and ground truth used, and the experiment setup and deployment strategy. To create an effective experiment plan, we refer again to the three evaluation criteria from the beginning of Section 5.3: accuracy, efficiency, and repeatability; all three have implications for the experiment setup.

To effectively measure accuracy, the ground truth must be as precise as possible; errors in the ground truth directly affect the accuracy benchmarks. To effectively measure efficiency, changes to the segmentation need to be recorded as new refinements are added by the user over time. Accuracy and time are dependent; accuracy needs to be viewed as a function of time. Furthermore, it is prudent to prevent users spending *too* much time refining a segmentation. We consider this to be justified since the primary purpose of interactive segmentation is to provide an accurate segmentation faster than it would take to produce it by



(a) A combined view of the image, markup, and the segmented object. The segmentation mask is overlaid semi-transparently.

(b) A view displaying the segmented object only; the background region (gray) is suppressed.

Figure 5.3: Screenshots of the interactive segmentation tool (running on the Linux platform)

hand. To effectively measure repeatability, we need to ensure we have a sufficient number of participants; if we use enough participants to segment each image several times, then algorithms with good repeatability will benchmark higher on average than algorithms with poor repeatability.

5.4.1 Software

It is important to provide a single user interface with consistent capabilities for the experiment, allowing participants to segment the relevant objects in a uniform way using different algorithms. To this end, we developed a standalone scribble-based interactive segmentation application. The tool supports any segmentation technique that can be adapted to use a scribble driven interaction paradigm for providing iterative updates. All four algorithms from Section 5.2 are fully integrated. Figure 5.3 shows screenshots of the tool, demonstrating two of the six available *view modes*.

To extract an object from an image, users mark foreground pixels using the left mouse-button, and background pixels using the right mouse button, or by using the left-button while depressing the *Ctrl* key. As each interaction is provided,

the corresponding segmentation mask is updated. The segmentation can be visualized within the tool by switching between six different view modes. These view modes consists of: (1) a mode displaying the only the original image; (2) a mode showing the user markings; (3) a mode showing the current segmentation mask; (4) a mode showing the original image with the user markup and the segmentation mask transparently overlaid; (5) a mode showing the object borders; and (6) a mode showing the segmented object with the background elements removed.

The tool itself was developed as a general purpose application—we envisioned its utility would go beyond the experiment described in this chapter. To support the constraints of the experiment an *experiment mode* was included. In this mode the relevant algorithm is selected and locked automatically for the participant. The participant is shown an image and a short description of the object they are required to extract. When the participant clicks on *Start*, a timer begins a countdown, giving the user a finite period to extract the required object as best they can using the current algorithm. The tool stores each segmentation mask and a corresponding time-stamp as new refinements are added, forming a progressive collection of segmentations over time. When the user finishes, or the time elapses, the next image and object description are displayed. The process repeats until the experiment is completed. Figure 5.4 shows an example of the application in experiment mode.

In addition to the base functionality required for the experiment, we also considered it important in a realistic evaluation to provide features that are typically found in other modern graphics packages. As such, several other features were included, including zooming, undo/redo support, and altering the markup brush size. The application also supports exporting segmentations as HTML image maps, and can therefore be used to generate dynamic object-aware content for the

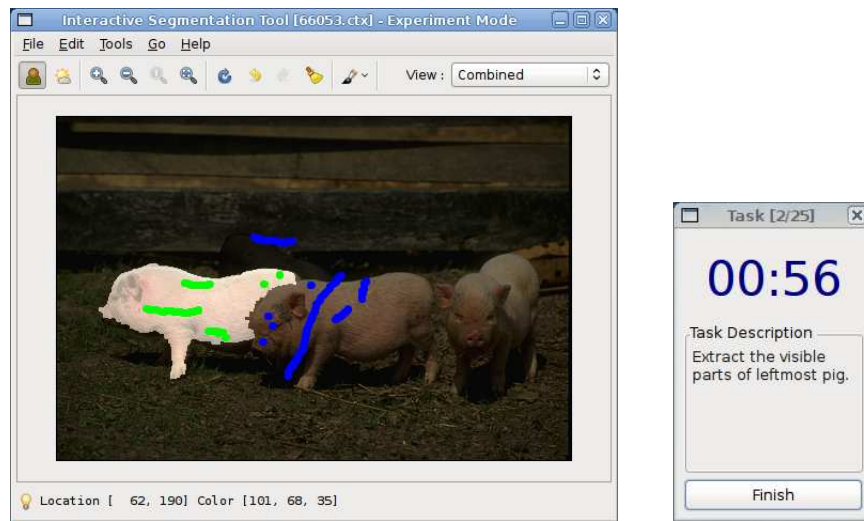


Figure 5.4: Screenshot of the interactive segmentation tool in experiment mode

web. This allows, for example, objects in images to be hyperlinked, and allows mouse-over effects to be applied to these objects.

The interactive segmentation tool, complete with the four algorithms evaluated in this chapter, is available for public download from our website². It is compatible with Linux, Windows, and Mac OS X.

5.4.2 Ground Truth

The images we used to compile the dataset for the experiments were taken from the publicly available Berkeley Segmentation Dataset [Martin et al., 2001]. The compiled dataset consists of 100 distinct objects selected from 96 of the 300 images in the Berkeley set. These images were chosen so that each image had one or more objects that could be unambiguously described to participants for extraction. Care was also taken to select images that were representative of a large variety of segmentation challenges, such as texture, camouflage, and various lighting conditions.

²<http://kspace.cdvp.dcu.ie/public/interactive-segmentation/>

To ensure the highest possible accuracy, the ground truth was created entirely by hand; no semi-automatic technique was used. This was also important to avoid potential bias to any algorithmic facet of the procedure used to create it. The object extraction was performed by marking pixels on the object border using a graphics tablet, and subsequently filling the object interior. The result is a series of binary masks, one for each object in the dataset, where zero valued pixels denote the background and non-zero valued pixels denote the object.

As noted in Section 5.3, creating a 100% pixel accurate ground truth is, in general, impossible, due to the ambiguity in the true positions of the border pixels. It is necessary, however, when creating a binary ground truth to decide which pixels belong to the object and which pixels belong to the background. To handle this ambiguity in the object border pixels, a simple heuristic was applied: retain pixels that appear to contain some of the objects color along the object border, and that do not appear to be image compression artifacts. This heuristic was chosen so that each pixel along the border would be, on average, half-inside and half-outside the true form of the foreground object.

Each object mask was annotated with a description of the object in the image to which it relates. The full ground truth dataset, including object masks and descriptions, is publicly available for download from our website. For reference, thumbnails and task descriptions for the complete dataset are also given in Appendix A.

5.4.3 Setup

A total of 20 volunteers participated in the experiment. Most of the participants were computer science or engineering graduates. Some of the participants were familiar with image processing and information retrieval techniques; however, none had any particular expertise in interactive segmentation. Each participant

Variant	Ground truth set			
A	S_1	S_2	S_3	S_4
B	S_2	S_3	S_4	S_1
C	S_3	S_4	S_1	S_2
D	S_4	S_1	S_2	S_3
Algorithm	A_1	A_2	A_3	A_4

Table 5.3: Experiment variants with ground truth set and algorithm assignments. Variant B uses ground truth set S_2 with algorithm A_1 , ground truth set S_3 with algorithm A_2 , S_4 with A_3 , and S_1 with A_4 .

was given a user guide and sufficient time to familiarize themselves and become proficient with the software that would be used for the experiment. Sample images were provided for training, but participants were not given access to the experiment dataset.

We considered it overly demanding to ask each participant to extract the entire set of 100 objects using all four segmentation algorithms. We therefore divided the ground truth randomly into four equally sized sets $\{S_1, S_2, S_3, S_4\}$ each containing 25 tasks. Each participant was given the task of segmenting the sets using a different algorithm for each set, resulting in a total of 100 tasks (as opposed to 400). Denoting the algorithms $\{A_1, A_2, A_3, A_4\}$, this gives four experiment variants, as shown in Table 5.3.

By distributing experiment variants to participants equally, we ensure that every image is segmented at least five times by each algorithm. Thus, we can minimize the affect of an individual’s markup skills and other human influenced variation by computing the average of the resulting benchmarks across segmentations of the same image with the same algorithm by different users. Repeatability is therefore implicitly evaluated: if a good segmentation is not repeatable by multiple users, the average evaluation measure will be lower.

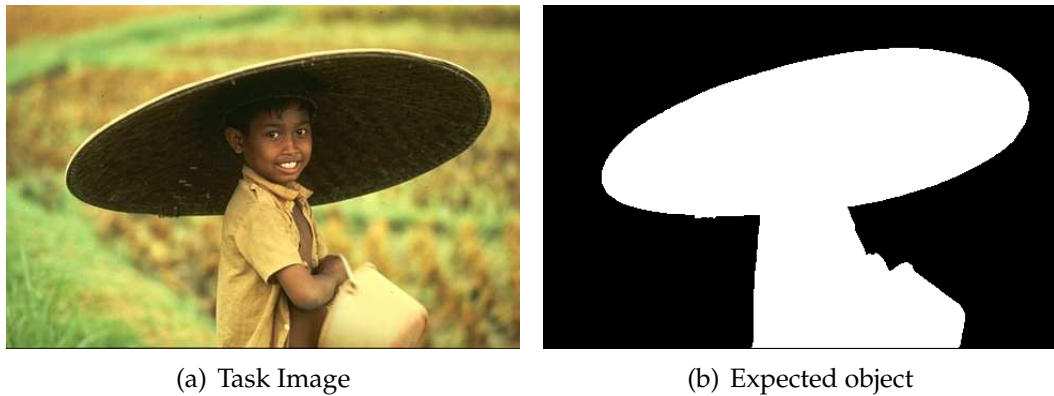


Figure 5.5: Sample task image and the expected object. The task description is *“Extract the person, hat, and bucket from the background.”*

The experiment proceeds as follows. Each task is presented to the user in the form of an image and task description. The image, of course, contains the relevant object, and the task description expresses as unambiguously as possible the part of the image to be extracted. Figure 5.5 presents a typical task, the corresponding description, and the expected object. Users are required to study the image and description and when ready, click on a *Start* button, and begin to extract the object as accurately as they can by marking areas of the image as foreground or background with the mouse. Since it is possible to achieve near perfect accuracy by manually segmenting an object (i.e. without the aid of interactive segmentation algorithms) when given an arbitrary amount of time, the usefulness of an interactive segmentation algorithm is in its ability to create a reasonably accurate segmentation in a significantly shorter time-span. For this reason, and to prevent some participants expending much more effort in improving their final segmentation than others, it is important to impose a reasonable time limit. We therefore restrict users to a maximum of 2 minutes per object. They may, however, proceed to the next task earlier if satisfied with their segmentation.

After the participant has finished extracting an object, they are asked to fill out a short questionnaire. The questionnaire was designed to coarsely assess, in subjective terms, how difficult the users found the segmentation, what they

considered to be the primary causes of any difficulties, and how accurate they perceived their final segmentation to be. Users are asked to rate how difficult they considered the task on a scale of 1 to 5, rate how accurate they considered their segmentation on a scale of 1 to 5, and to check a series of boxes indicating what they perceived to be the primary causes of any difficulty encountered. These checkboxes corresponded to low-level image features, such as color, texture, and object size.

When the entire set of 25 objects are extracted, participants are requested to rate the segmentation algorithm that they just used, again on a scale of 1 to 5. Once completed, the software automatically selects the next algorithm and participants proceed to extracting the next 25 objects. The experiment continues thus until all objects are extracted.

5.4.4 Deployment

The experiments were carried out by each user independently, and in their own time. Experiments took about 3 hours each to complete. Participants were permitted to take breaks between tasks: a continuous sitting was not required.

To ease deployment of the experiment, and efficiently collect the results, a deployment tool was created. When executed the tool prepares the user's system for the experiment as follows:

1. Information identifying the participant is collected.
2. The image and ground truth data files are automatically downloaded from a central server. These are placed in a known location on the participant's machine.
3. The deployment tool contacts a web-service running on the server, which assigns a particular experiment variant to the user. The web service maintains a database of participant-variant pairs, and assigns the variant using a

round-robin system, to ensure equal coverage of each of the four task sets with all corresponding algorithms.

4. Experiment files compatible with the segmentation tool are generated and the user is instructed to begin the experiment.

The deployment tool also displays the relevant questionnaire pages to the user at each stage of the experiment, and stores the answers. When the experiment is complete, all data generated by the segmentation tool and the deployment tool is automatically compressed and uploaded to the server for analysis.

5.5 Results and Analysis

All 20 participants completed the experiment in full, resulting in over 40,000 segmentation masks being collected for evaluation. In this section we present the results of the evaluation, and discuss their implications. To give a high-level idea of the accuracy and efficiency of the algorithms, we first describe the overall average accuracy (with respect to the measures discussed in Section 5.3) and the overall average time required to perform the segmentation with each algorithm. We then present average accuracy as a function of time, to attain a better understanding of the characteristics of each algorithm. Next we discuss perceived accuracy, as specified by participants in the questionnaires, and its significance. Finally, we investigate the correlation between the computed evaluation benchmarks and perceived accuracy.

5.5.1 Object and Border Accuracy

Using the object and boundary accuracy measures discussed in Section 5.3, for each algorithm evaluated we measured:

Algorithm	Boundary accuracy $\tilde{\mathcal{A}}_B$		Object accuracy \mathcal{A}_O	
	Best	Final	Best	Final
<i>BPT</i>	0.78	0.78	0.93	0.92
<i>IGC</i>	0.78	0.77	0.93	0.92
<i>SRG</i>	0.70	0.70	0.88	0.88
<i>SIOX</i>	0.64	0.64	0.85	0.85

Table 5.4: Overall average boundary accuracy and object accuracy.

- The average final segmentation accuracy: the object and boundary accuracy measured when the participant was finished the segmentation or the allocated time elapsed, averaged over all objects from the same segmentation algorithm.
- The average best segmentation accuracy: the best object and boundary accuracy achieved per object, averaged over all objects from the same segmentation algorithm.

The resulting values are shown in Table 5.4. It is clear from the table that the best performing algorithms, in terms of measured accuracy, are the BPT and IGC algorithms, which perform equally well on average. The SIOX algorithm is the poorest; this is perhaps due to the difficulty, noted by some participants in the questionnaires, of producing any reasonably accurate segmentation for some images in the dataset.

In addition to accuracy, it is also critical to measure time when evaluating interactive segmentation: given enough time, arbitrary precision can be achieved manually. Table 5.5 shows, for each algorithm, the average time required until a user attains their best object and boundary accuracy for an image, and the average total time spent per image. From this, we can see that users spent the least amount of time with the BPT algorithm, and the most with the SRG algorithm.

Algorithm	Best $\tilde{\mathcal{A}}_B$	Best \mathcal{A}_O	Final/total
<i>BPT</i>	59.76	59.09	64.25
<i>IGC</i>	62.93	62.53	66.43
<i>SIOX</i>	69.88	68.90	73.08
<i>SRG</i>	80.77	80.73	85.32

Table 5.5: Average time required for users to achieve their best accuracies and average total time used to complete a task (seconds).

The times given in Table 5.5 are, however, likely achieved at varying accuracies for each individual algorithm. Thus, the table only gives an overview of the typical time required to achieve the best possible result with each algorithm. Gauging accuracy over time gives a more complete picture of each algorithm’s performance.

Time Series

We now consider measuring how accuracy varies over time for each segmentation algorithm. Figure 5.6 shows a scatterplot of the raw boundary accuracy measurements and the times they were recorded for a single participant. The measurements are grouped by algorithm; the color of the points on the plot represents the image being segmented. Clearly there is a considerable amount of data. The nature of the experiment means that measurements are taken at different times. Also, because people are allowed to finish the segmentation before the allotted time, the time series for each task may have different durations.

Having each segmentation task produce compatible time series data greatly simplifies analysis. Our first task, therefore, is to process the raw experiment data to produce time series measurements that are (1) equally spaced, and (2) equal in duration. To do this, we extend all time series to the full duration (120 seconds) by duplicating the final accuracy measurement. The extended time series is then

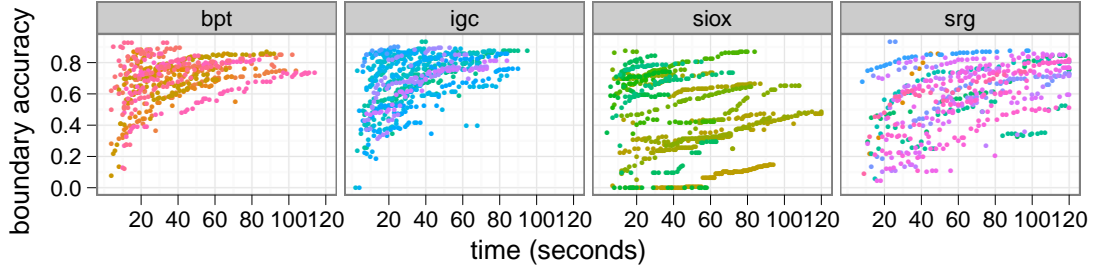


Figure 5.6: Scatterplot of accuracy measurements against time. The plot shows the raw boundary accuracy measurements for a single participant.

resampled at regularly spaced intervals. We chose to resample every two seconds; this rate halves the number of samples with negligible information loss.

The above procedure results in time series measurements from each task that are evenly spaced and have equal duration. We can therefore determine, at each point in time, the mean accuracy per object, per user, or per algorithm by averaging across the remaining dimensions. For example, to find the mean time series for segmenting object o using algorithm a , we average across each participant that segmented o using a . Figure 5.7 shows the resulting mean time series for 40 (of 100) objects in the dataset. The time series shown is for boundary accuracy. Each panel represents a different object; each line represents a different segmentation algorithm.

The *overall* mean accuracy time series for each algorithm can be similarly computed: by finding the average of all time series over the user and object dimensions. Since time and precision are dependent, this provides one of the most useful illustration of an algorithms performance. The result for both object accuracy and boundary accuracy is shown in Figure 5.8.

Several observations can be drawn from the figure. On average, the BPT and IGC algorithms consistently outperform the SIOX and SRG algorithms for both

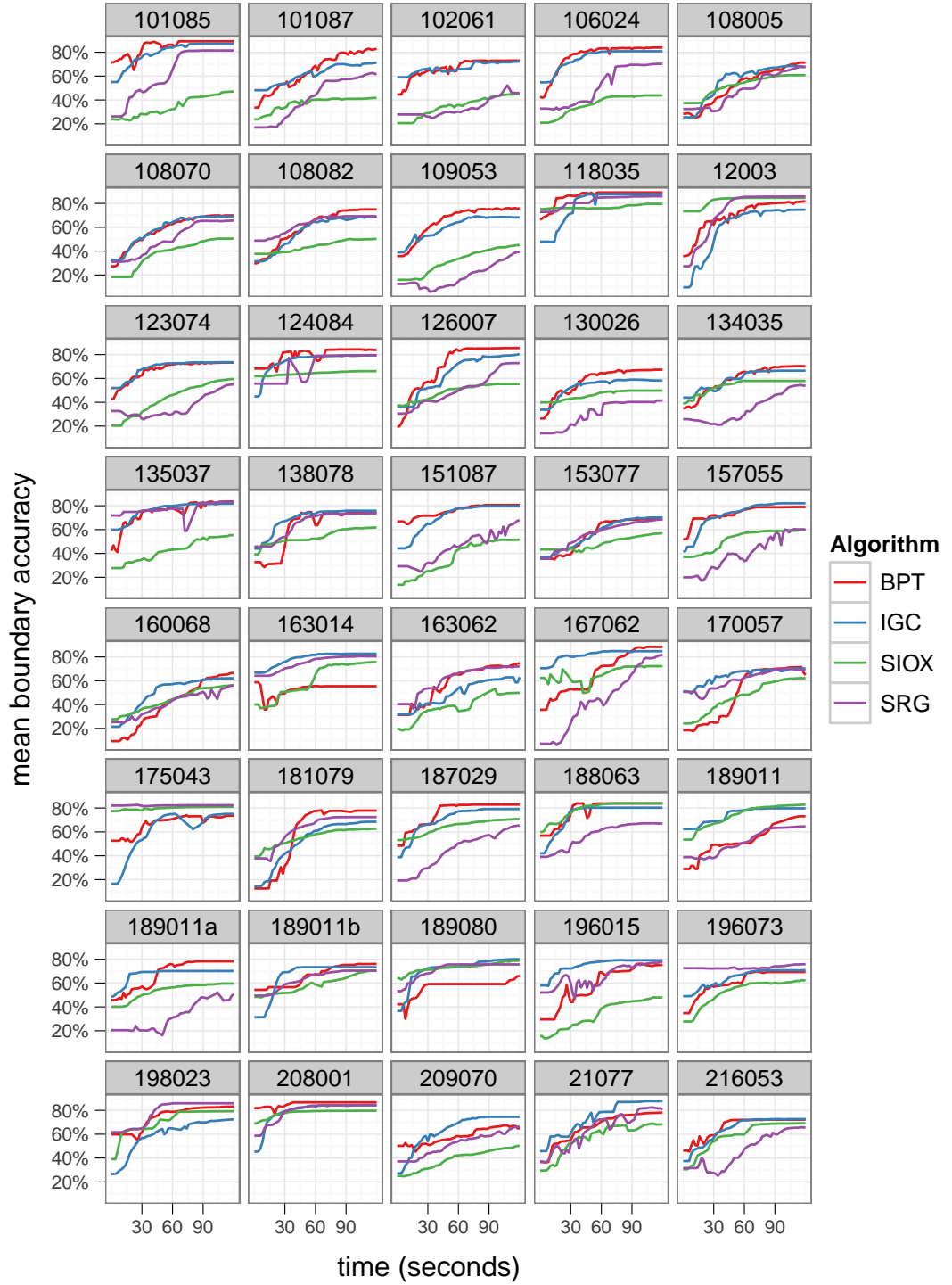


Figure 5.7: Mean boundary accuracy time series per image. The time series values are first extended to the full range, and the spacing is equalized using linear interpolation. The resulting series are then averaged across users. Each panel in the figure represents a segmentation task, and is labeled with the task identifier. The corresponding task definitions are given in Appendix A.

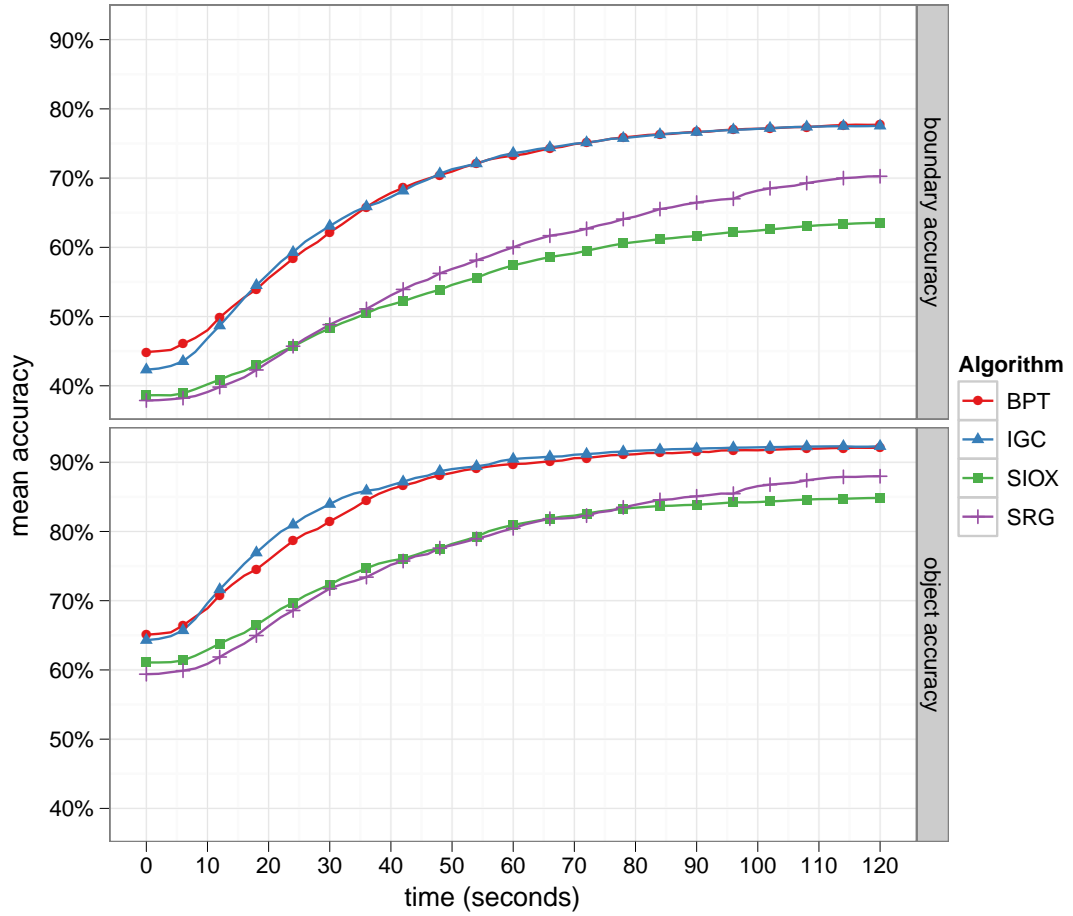


Figure 5.8: Mean accuracy time series for both object and boundary accuracy.

measures. The BPT and IGC algorithms have comparable performance throughout; after approximately 50 seconds the difference in average precision between the two is negligible. The SIOX and SRG algorithms also have comparable performance. The SIOX algorithm performs marginally better than SRG early on, but is surpassed by SRG after about a minute. This SIOX algorithm has the flattest time series curve, implying it is one of the least responsive algorithms and tends to inhibit iterative improvement.

The mean time series is a concise and effective way of visualizing the performance of an interactive segmentation algorithm. Although averaging across all

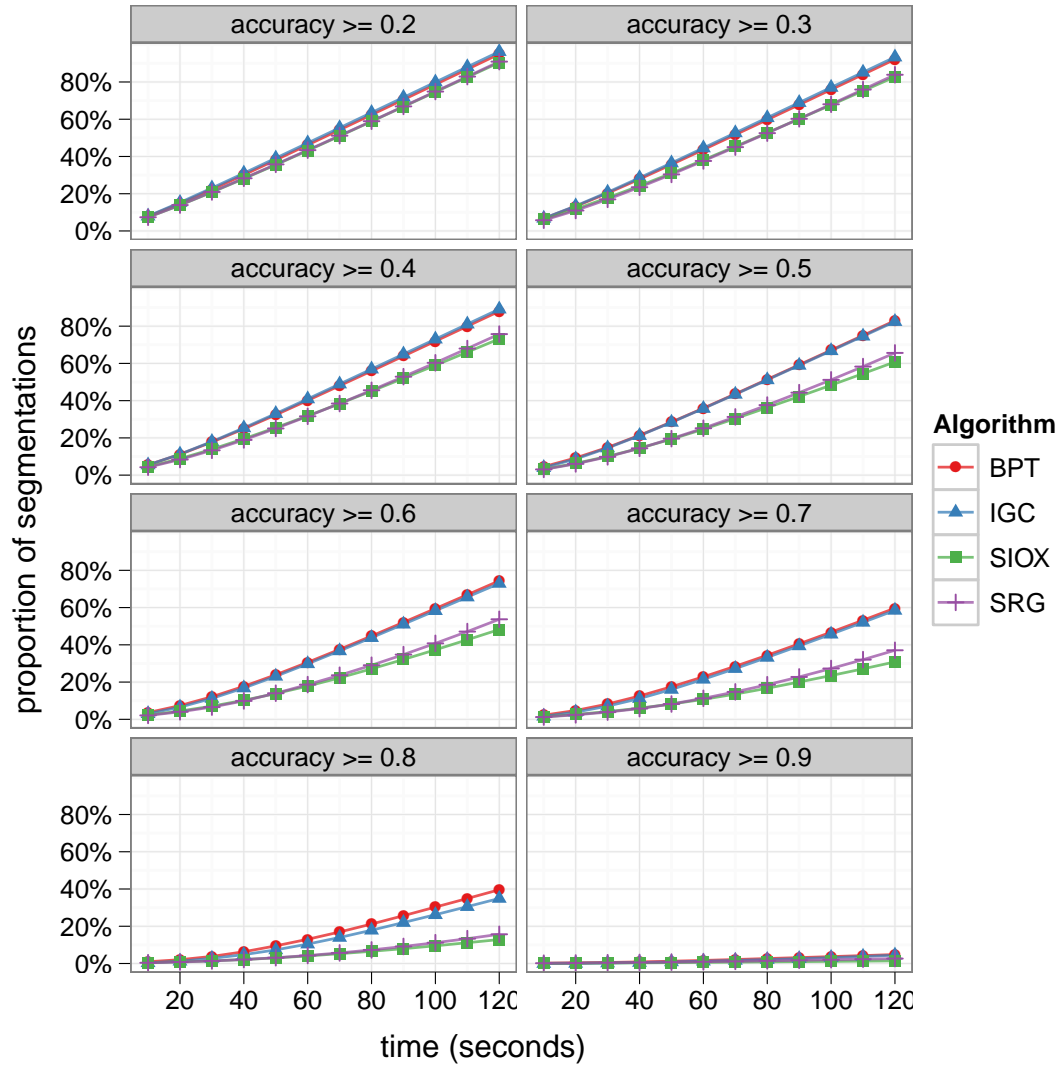


Figure 5.9: Proportion of segmentations attaining at least the given boundary accuracy, plotted over time. Each panel represents a different minimum accuracy. The y-values are the proportion of the collection segmented to at least this accuracy; the x-values represent the corresponding elapsed time.

participants and images results in a robust time series, it discards a lot of information. Plotting the mean time series for each image (Figure 5.7), on the other hand, gives substantially more information, but the result is convoluted and difficult to interpret. Our final analysis retains more information than the overall mean time series, but affords simpler interpretation. The plots in Figures 5.9 and 5.10 show the proportion of segmentations that exceed varying degrees of accuracy

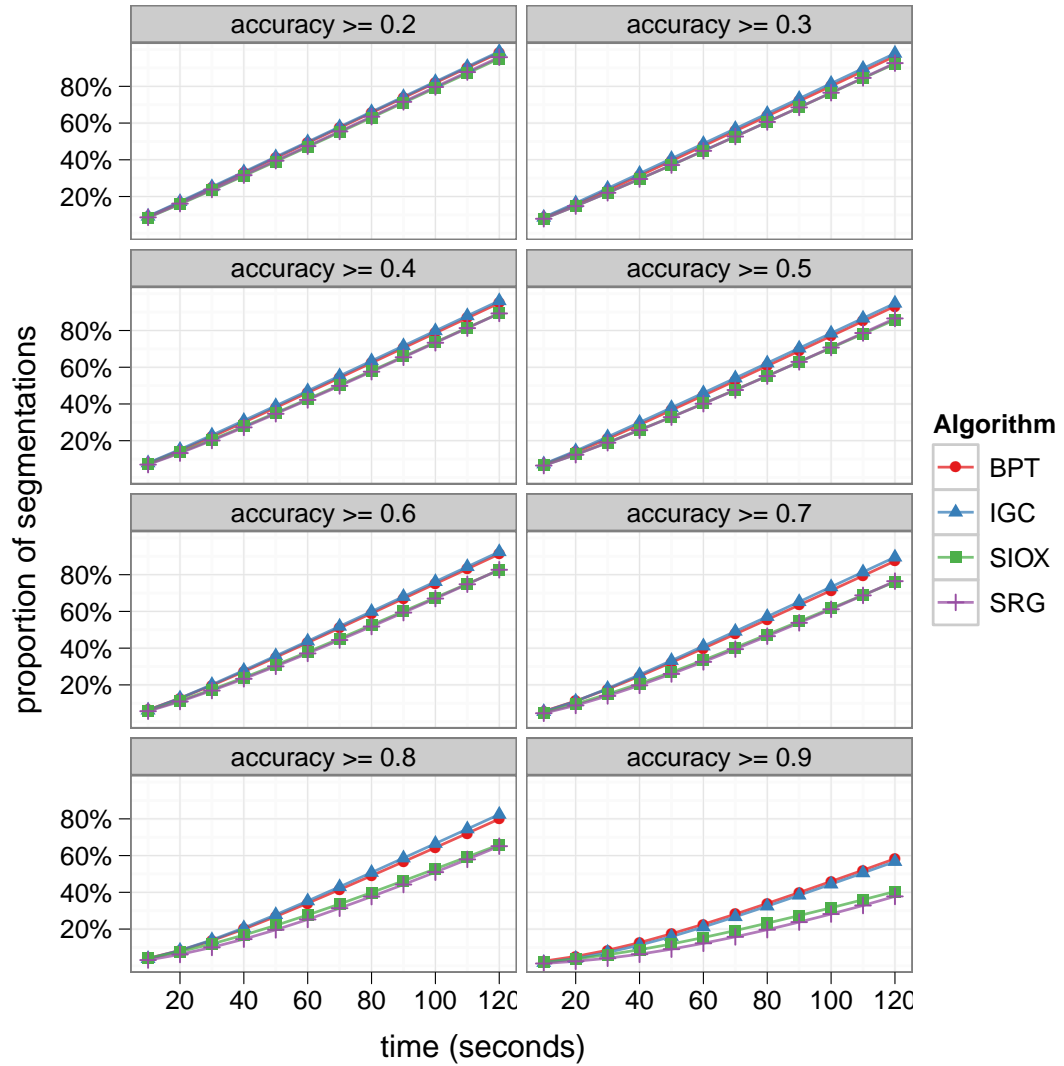


Figure 5.10: Proportion of segmentations attaining at least the given object accuracy, plotted over time. Each panel represents a different minimum accuracy. The y-values are the proportion of the collection segmented to at least this accuracy; the x-values represent the corresponding elapsed time.

over time. The plots are generated by measuring the number of segmentations that have an accuracy exceeding α at time τ for discrete values of α and τ . The result is a three-dimensional plane, illustrated in slices in Figures 5.9 and 5.10. The figures essentially afford the same conclusions as Figure 5.8: that overall BPT and IGC are comparable and outperform SRG and SIOX.

It is also worth noting that the two accuracy measures are well correlated. The Pearson correlation coefficient for the two measures, computed over all recorded measurements, is 0.834. The measures also demonstrate high rank correlation: Spearman's ρ coefficient over all recorded measurements is 0.823.

5.5.2 Perceived Accuracy

To measure perceived accuracy, participants were asked to rank how accurate they perceived their final segmentation on a scale of 1 to 5: 5 meaning highly accurate, and 1 meaning highly inaccurate. We also asked users to rank the performance of each algorithm on a scale of 1 to 5, again higher ranks indicating better performance. Figure 5.11 uses histograms to illustrate the distribution of user responses. The mean perceived accuracy and the mean performance ranks are shown in Figure 5.12. Clearly, participants felt that, on average, the BPT and IGC algorithms produced significantly more accurate final segmentations than SRG and SIOX. On average, participants also perceived that the BPT and IGC algorithms out-perform the SRG and SIOX algorithms. The results agree with the average measured accuracies from Table 5.4.

There is no significant difference between the mean perceived performance of the BPT and IGC algorithms. The histograms in Figure 5.11, however, clearly show that most users assigned a higher performance rank to the IGC than the BPT algorithm, despite them having comparable performance for the time period shown in Figure 5.8. Potential reasons for this are explored in the next section.

5.5.3 User Feedback

In addition to asking participants to rank the accuracy of their final segmentations in the questionnaires, we also asked participants to comment on each of the

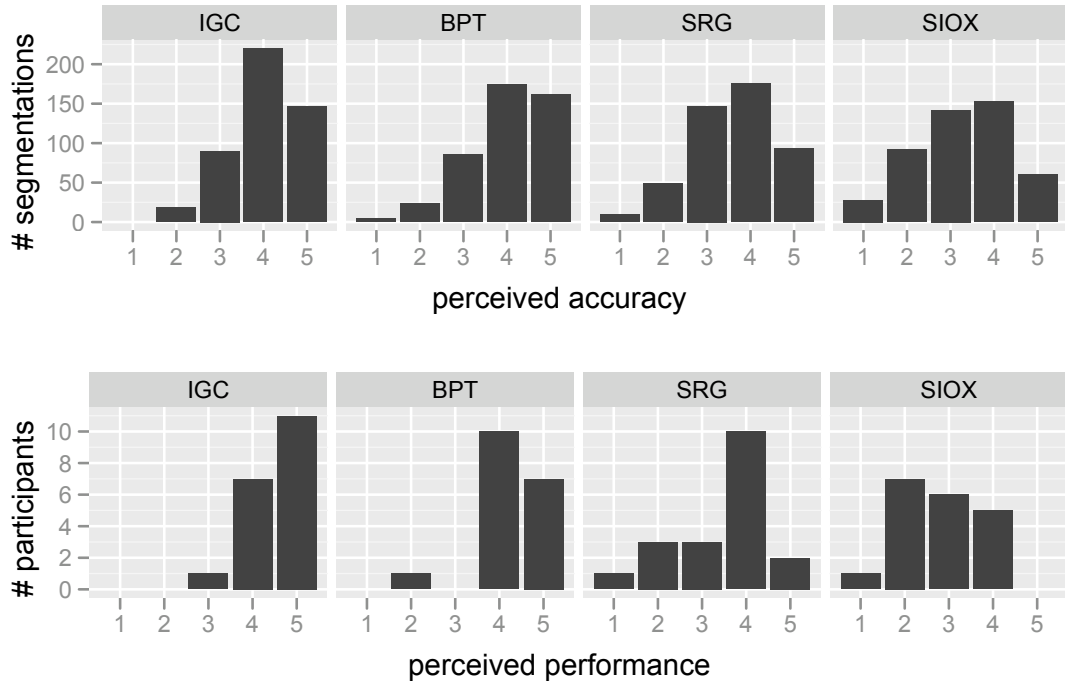


Figure 5.11: Histogram plots of segmentation accuracy (per image), and algorithm performance (per algorithm) as judged by the participants.

evaluated algorithms. This included asking participants: which algorithm they preferred, what they believed were the strengths and weaknesses of each, and if they had any other general remarks or comments. Figure 5.13 shows the distribution of votes given by users for their preferred algorithm.

From Figure 5.11 and Figure 5.13, it is clear that most users preferred the IGC algorithm, despite their comparable performance in terms of their time accuracy profiles (Figure 5.8). Analysis of the user comments revealed an interesting explanation for the discrepancy—the algorithm’s *behavioral predictability*. The IGC algorithm tends to behave more conservatively than BPT: additional interactions tend to produce small predictable changes, whereas larger more unpredictable changes can sometimes occur with BPT. This gives the BPT algorithm the potential to improve its segmentation faster than IGC, but may also induce the perception of erratic behavior.

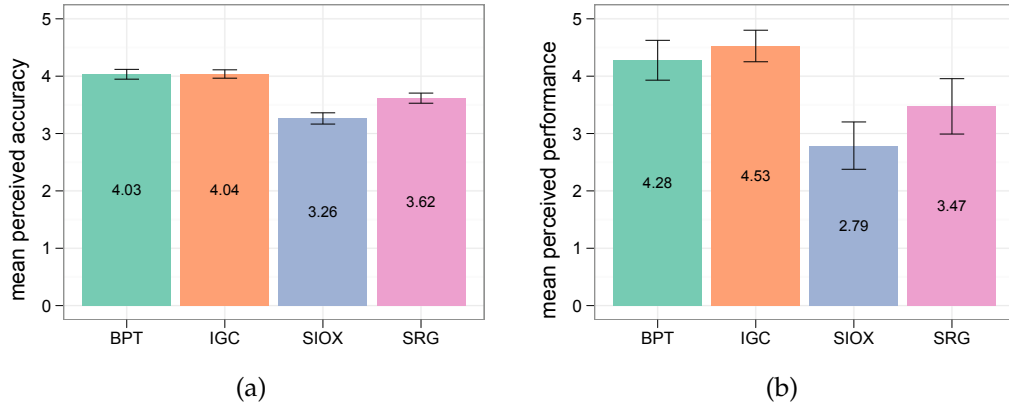


Figure 5.12: Mean results from user feedback. (a) shows the mean segmentation accuracy as judged by participants; (b) shows mean algorithm performance as judged by participants. The error bars represent the confidence intervals assuming a normal distribution for $p = 0.05$ ($\pm 1.96 \cdot SE_{\bar{x}}$).

From the comments it was clear that participants strongly preferred more conservative algorithms. For the IGC algorithm, users remarked that the algorithm “reacted well to local changes, without causing too much global deformation.” They liked that “small localized scribbles only have a local effect.” Conversely, users disliked algorithms in which small additions to the markup could cause large differences to the segmentation. Commenting on the SRG algorithm one user complained that “adding one scribble can completely change the segmentation.” This apparently erratic behavior was also noted by participants with regard to the BPT and SIOX algorithm, and is likely the reason why more participants preferred the IGC algorithm to the BPT algorithm.

Another issue commonly indicated as important in the feedback was *algorithm responsiveness*. Participants, in general, disliked algorithms that made it difficult for them to refine their segmentation. This was the most common reason that users cited for disliking the SIOX algorithm: although it was sometimes “very quick to capture initial object,” “if it doesn’t find the correct boundary in the beginning, then it is simply impossible to refine.” The comments revealed that



Figure 5.13: Preferred algorithm as voted by users

many users become quickly frustrated with algorithms that make it difficult to add iterative refinements to their segmentation. This is further reinforced by comparing the time spent on each task with the rankings given: users prefer using algorithms that require longer to segment an object, but allow iterative improvements (SRG), than using algorithms that make a better initial guess, but make improvement more difficult (SIOX).

Similar observations have also been made when evaluating other interactive systems. Koenemann and Belkin [Koenemann and Belkin, 1996] showed that users perform better when using information retrieval systems if they understand the underlying relevance-feedback mechanism. They also point out that users subjectively preferred more transparent systems. This is related to behavioral predictability—systems that are easier to understand are easier to predict. As a design principle for creating semi-automatic annotation interfaces, Suh and Bederson [Suh and Bederson, 2007] propose that users should be in control at all times, and that systems should not hamper a user’s freedom to make manual annotations. This proposition is supported by the comments made by our users when the algorithms provided inadequate response to their attempted refinements.

In the feedback, participants not only identified properties of interactive segmentation algorithms that they felt were important, but also identified specific image features that appeared to cause difficulties. Two image features in particular were recognized as a source of difficulty for all of the algorithms evaluated: *texture* and *object-detail*. Users commented that the algorithms were often “confused by texture,” and had “difficulty with very fine details.” The problems with texture are expected: none of the algorithms explicitly use texture features. The problems with segmenting edge detail are often related to contour smoothing performed by algorithms to prevent jagged object boundaries, which tend to be visually disturbing. Interestingly, one participant recognized this link between object detail and boundary jaggedness, suggesting that “there could be a boundary smoothness tool to control the jaggedness of a region.”

5.5.4 Validation

To demonstrate the benefits of the proposed measures we compare our suggested benchmarks, and several other popular measures, with perceived accuracy as indicated by the participants. For the comparison, we computed two measures of correlation between measured accuracy and perceived accuracy, specifically: the (Pearson product-moment) correlation coefficient, and Kendall’s tau [Kendall, 1938] rank correlation coefficient. Kendall’s tau is a measure of the strength of association of cross tabulations, and has values in the interval $[-1, 1]$, where 1 indicates perfect agreement and -1 perfect disagreement.

Instead of computing the correlation coefficients directly against all the perceived and measured accuracies for all final segmentations, we first average the values for the each segmentation with the same algorithm, and of the same object, for different users. This pre-averaging helps to mitigate outliers, and is moti-

	Correlation Coefficient	Kendall's τ
$\tilde{\mathcal{A}}_B$	0.679	0.494
\mathcal{A}_O	0.669	0.516
\mathcal{A}_B (binary)	0.606	0.445
\mathcal{P}_r	0.564	0.448
\mathcal{R}_e	0.469	0.382
\mathcal{R}_I	0.375	0.350

Table 5.6: Correlation of measured and perceived accuracy

vated by participants expressing that they had either made some errors in the questionnaires, or had misread some of the task descriptions.

The resulting correlation values are shown in Table 5.6. The values show that the suggested object and boundary accuracy measures are more closely correlated with human perception than are the other tested measures, with boundary accuracy $\tilde{\mathcal{A}}_B$ having a higher correlation coefficient and object accuracy \mathcal{A}_O having a higher value of Kendall's tau. Furthermore, the proposed fuzzy version of boundary accuracy is also better correlated with perceived accuracy than the binary case \mathcal{A}_B for both coefficients.

5.6 Conclusion

In this chapter we presented a comparative evaluation of four interactive segmentation techniques. This evaluation was carried out in the form of a user experiment in which 20 participants were asked to segment objects using different interactive segmentation algorithms. To support the experiment, we developed a consistent user interface for hosting scribble driven interactive segmentation algorithms, that also supports the most important features of other image editing tools. We selected a set of 100 objects from a publicly available dataset, containing a good cross-section of segmentation challenges. These images were then manually segmented, and annotated with unambiguous descriptions of the desired

objects. The interactive segmentation tool, complete with the four algorithms that were evaluated, and the ground truth dataset are available from our website.

We selected two measures for evaluation: the Jaccard index to measure object accuracy, and a new fuzzy Jaccard index to evaluate boundary accuracy. Object segmentation masks were stored after each participant performed a new interaction, and the accuracy benchmarks were computed against each stored mask. The resulting plots of average accuracy over time demonstrated that the two most effective techniques were the interactive graph cuts algorithm and the binary partition tree algorithm.

In addition to measuring accuracy against a ground truth, participants were asked to rank the accuracy of each final segmentation. The results of this ranking were shown to correspond well with the average measured accuracy. Furthermore, the correlation between perceived accuracy and measured accuracy was shown to be higher for the proposed measures than for other commonly used measures, including precision, recall, and the Rand index.

Chapter 6

Automating Interactive Segmentation Evaluation

6.1 Introduction

In the previous chapter we focused on supervised evaluation of interactive segmentation by means of user experiments. In practice, carrying out user experiments every time a new algorithm, or new variation of an algorithm, is to be evaluated can be prohibitively time consuming and labor intensive. In this chapter we aim to develop a method for supervised evaluation of interactive segmentation algorithms that eliminates the need for user experiments.

Automating the evaluation of interactive segmentation involves replacing the human operator with an algorithmic process designed to emulate the behavior of an operator as closely as possible. To achieve this, we propose driving the interactive segmentation by automatically deriving the user interactions from the current segmentation error and ground truth data. In this chapter we explore four strategies for deriving these interactions. The first of these strategies is deterministic, needing only to be run once to obtain a rough evaluation. The remaining three are probabilistic: they aim to more realistically approximate an

actual user. We evaluate the four interactive segmentation algorithms from the previous chapter using each of these strategies, and compare the results with the user experiments.

The remainder of this chapter is organized as follows. Section 6.2 discusses the objective of automating the evaluation. We first outline some general considerations, then develop four different strategies for automating the user interactions, beginning with the simplest strategy and iteratively developing more complex ones. Section 6.3 discusses the parameters of the experiment and outlines the software and tools that we developed for automated evaluation. Section 6.4 analyzes the results of the experiment and compares them with the results of the user experiments from the previous chapter. Section 6.5 presents our conclusions and outlines some recommendations for using the proposed system.

6.2 Automation

The previous chapter described evaluating interactive segmentation by means of user experiments. In these experiments, the participant is required to provide seed pixels for the object and background regions by marking the image with the mouse. The segmentation algorithm builds an initial segmentation using these seed pixels as priors, and provides feedback to the participant. The participant may then iteratively refine this segmentation by marking additional pixels until either a satisfactory segmentation is obtained, or the allotted time expires. This kind of experiment, while invaluable for establishing the usability of an algorithm, is often prohibitively difficult and time-consuming, especially considering that it needs to be repeated each time a new algorithm is evaluated. The idea behind automating the evaluation is to devise an algorithmic process that can simulate, in some reasonable way, all the actions that are usually performed by the human operator, thereby eliminating the need for user experiments.

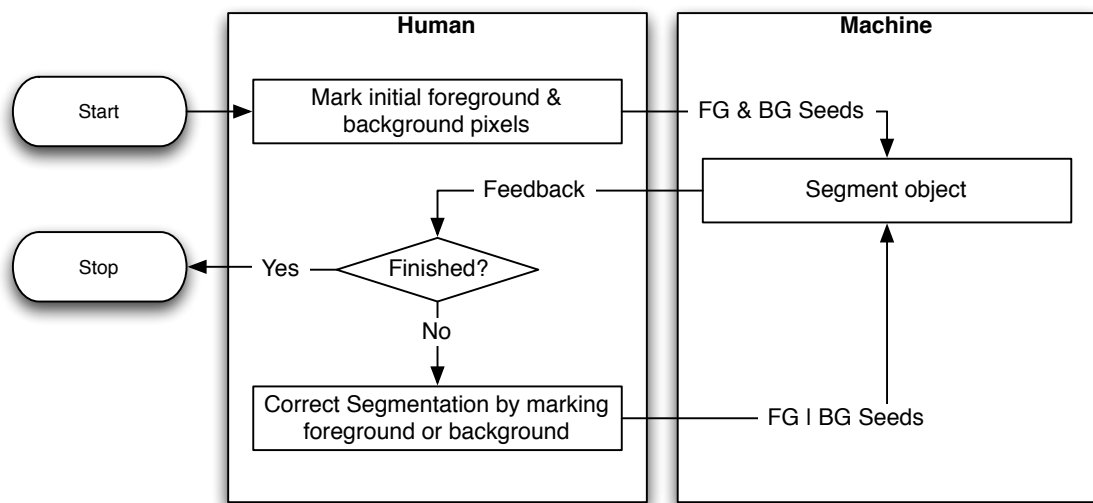


Figure 6.1: Flow chart of user activity when performing a segmentation task

When first considering this problem, one might initially reason that since we have already performed the user experiments, we can simply record each interaction performed by the participants and automate the procedure by replaying these interactions. Inspection reveals this reasoning to be flawed: although the first set of foreground and background seeds supplied by the user could indeed be used by an automation algorithm, all subsequent interactions are reactive. That is, at each step, the user is attempting to correct the current segmentation error. This error depends on the algorithm being evaluated, and on the interactions from previous steps.

Automating the evaluation requires us to identify each decision that is made by the user, so that it can be replaced by an automated action. Figure 6.1 depicts a high-level view of the flow of activity for a user during a segmentation task. From this we can identify the following user responsibilities:

1. Identify the object to be extracted from the task description;
2. Select initial foreground and background seed pixels;

3. Correct errors in the segmentation by selecting additional foreground or background seed pixels; and
4. Decide after each interaction if the segmentation is satisfactory.

Identifying the object to be extracted from the task description is, undoubtedly, the most difficult of these tasks to automate. It is, however, possible to bypass this step: we are performing a supervised evaluation, so the object to be extracted is coded exactly in the ground truth. Step 2 can be accomplished by selecting, in some way, the initial seed pixels from the object and background regions in the ground truth. After an initial segmentation has been found, we can find the mislabeled pixels by comparing this initial segmentation against the ground truth. This gives us the means to automate Step 3: by selecting the additional object or background seed pixels from the set of mislabeled pixels.

The final step is to decide if the segmentation is satisfactory, and if so, terminate the process. A straightforward criterion is to declare a segmentation to be satisfactory if and only if it exactly matches the ground truth. In our experiments, however, we observed that this strategy often results in a great deal of time toward the end of the segmentation process being spent correcting insignificant errors along the boundary of the object. Since, as noted in the previous chapter, the true boundary of an object is inherently ill-defined, and since human operators do not normally notice such slight error, much less spend time correcting it, we recommend terminating the segmentation if the only remaining error pixels lie on the inner or outer boundary of the object.

We have not yet specified how to select the seed pixels in the initialization stage or correction stage. Furthermore, we have made some important assumptions in the above discussion that warrant further consideration. We address both these issues shortly. First, let us outline the general automation algorithm by

drawing on the above discussion. Recall the notation we created in Section 5.3.2 for important sets of pixels in the machine segmentation and ground truth:

- G_O : the set of all pixels inside the ground truth object;
- G_B : the set of all pixels outside the ground truth object;
- M_O : the set of all pixels inside the machine segmented object;
- M_B : the set of all pixels outside the machine segmented object;
- B_G : the set of internal border pixels for the ground truth object;
- B_M : the set of internal border pixels for the ground truth background region.

Note that the set of internal border pixels for the background region equals the set of external border pixels for the object region, and similarly, the set of internal border pixels for the object region equals the set of external border pixels for the background region.

In addition to the above definitions, we denote the initialization seeds, the update seeds, and the current segmentation error as follows:

- I_O and I_B are the sets of object and background seeds used by the automation algorithm to initialize the segmentation;
- U_O and U_B are the sets of object and background seeds used to update (refine) the segmentation;
- E_O and E_B are the sets of object and background pixels that have been misclassified by the segmentation algorithm (the error).

Using this notation, our proposed general automation algorithm proceeds as follows:

1. *Initialize*: Select the initial object seed points I_O , and the initial background seed points I_B , such that $I_O \subseteq G_O$ and $I_B \subseteq G_B$. Mark these points as object and background and update the segmentation.
2. *Compute error*: Determine the set of misclassified object pixels E_O , and the set of misclassified background pixels E_B , as:

$$E_O = M_B \cap G_O \quad (6.1)$$

$$E_B = M_O \cap G_B \quad (6.2)$$

3. *Check for termination*: If the sets of misclassified pixels above contain only pixels from the object's internal or external border, terminate the algorithm. More formally, the algorithm terminates if:

$$E_O \subseteq B_G \wedge E_B \subseteq B_M$$

Note that the above holds when both E_O and E_B are the empty set.

4. *Correct*: Update the segmentation by selecting either additional object seeds $U_O \subseteq E_O$, or additional background seeds $U_B \subseteq E_B$, and return to Step 2.

There are two important, and related, implications of the above algorithm that need to be addressed. First, if the interactive segmentation algorithm being evaluated is not “well-behaved” the algorithm may never halt. By well-behaved, we mean that if we mark a pixel as object, the algorithm will always classify it as object, and if we mark it as background, the algorithm will always classify it as background. We believe that it is justified to assume such behavior, as we explain shortly. It is straightforward to modify an existing segmentation algorithm to conform to this behavior by post-processing the output.

The second implication is that since the automation algorithm only ever chooses object seeds that are inside the ground truth object, and background seeds that are outside the ground truth object, it is impossible for our automation algorithm to ever make a mistake—i.e., to incorrectly mark an object pixel as background or vice versa. Our experiments have shown that users, in general, are not so diligent.

However, we believe that both these assumptions are sensible design decisions. In our user experiments, participants universally agreed that they preferred algorithms with predictable behavior. So although it is possible for an interactive segmentation algorithm to compensate for inaccurate interactions, we do not necessarily believe that they should: such compensation is a direct violation of the user’s instructions, however imprecise. This kind of behavior may be helpful in a few cases, but more often than not, is an endless source of frustration to users [Spolsky, 2001]. The desire for predictable behavior justifies the requirement that algorithms must be well-behaved; well-behaved algorithms are, by definition, incapable of compensating for user errors.

What remains is to decide a suitable way of selecting the initialization seeds I_O and I_B , and the update seeds U_O and U_B . We now explore four strategies for their selection.

6.2.1 Strategy 1

We begin by investigating a very simple deterministic strategy for choosing the initialization and update seed pixels. We do so to set up a baseline approach against which we can compare our more sophisticated strategies, which attempt to more closely approximate real user interactions.

The basis of this strategy is the observation that users tend to begin extracting objects by marking as foreground some pixels in the middle of the object, and

marking as background some pixels well outside the object. They then proceed to refine the initial segmentation by marking pixels that lie inside large areas of misclassified pixels. To emulate this behavior, strategy 1 initializes the segmentation by selecting pixels that are near the center of the ground truth object as object seeds, and selecting pixels that are distant from the ground truth object as background seeds. Similarly, to update the segmentation, the strategy chooses pixels that are farthest from the correctly classified pixels as the update seeds.

Let $D(\mathbf{x}, R)$ be the minimum distance from a pixel \mathbf{x} to any pixel in the set R :

$$D(\mathbf{x}, R) = \min_{\mathbf{y} \in R} \|\mathbf{x} - \mathbf{y}\| \quad (6.3)$$

and let the $Z(Q, R)$ be the set of all points in Q that are maximally distant to their nearest points in R :

$$Z(Q, R) = \arg \max_{\mathbf{x} \in Q} D(\mathbf{x}, R) \quad (6.4)$$

We choose the initial seed points I_O and I_B as:

$$I_O = \{\mathbf{x} : \mathbf{x} \in \text{Br}(\mathbf{y}, G_B), \mathbf{y} \in Z(G_O, G_B)\} \quad (6.5)$$

$$I_B = \{\mathbf{x} : \mathbf{x} \in \text{Br}(\mathbf{y}, G_O), \mathbf{y} \in Z(G_B, G_O)\} \quad (6.6)$$

where $\text{Br}(\mathbf{y}, R)$ is a brush function that returns all pixels within a fixed radius of \mathbf{y} :

$$\text{Br}(\mathbf{y}, R) = \{\mathbf{x} : \|\mathbf{x} - \mathbf{y}\| \leq r(\mathbf{y}, R)\} \quad (6.7)$$

The brush radius is given by the $r(\mathbf{y}, R)$ function. It is chosen proportional to the minimum distance from the center seed points \mathbf{y} to the object boundary, within the constraints of the interactive segmentation tool. Note that all such points $\mathbf{y} \in Z(G_O, G_B)$ are equidistant from the background, and similarly all points

$\mathbf{y} \in Z(G_B, G_O)$ are equidistance from the object. The interactive segmentation tool has a maximum brush radius of 20 pixels, so we define our brush radius function as:

$$r(\mathbf{y}, R) = \min \left(\frac{1}{2} D(\mathbf{y}, R), 20 \right) \quad (6.8)$$

Our reasoning here is simple: users tend to use larger brush sizes to correct larger errors, and smaller brush sizes to correct minor details; they cannot set the brush to a size larger than 20 because our tool does not support it.

To update the segmentation we follow a similar strategy, this time taking our update seeds U_O and U_B from the sets of misclassified object and background pixels E_O and E_B :

$$U_O = \{\mathbf{x} : \mathbf{x} \in \text{Br}(\mathbf{y}, E_O^C), \mathbf{y} \in Z(E_O, E_O^C)\} \quad (6.9)$$

$$U_B = \{\mathbf{x} : \mathbf{x} \in \text{Br}(\mathbf{y}, E_B^C), \mathbf{y} \in Z(E_B, E_B^C)\} \quad (6.10)$$

where E^C denotes the complement of the set E . In the interactive segmentation tool, the segmentation is updated after each interaction. It is impossible for users to simultaneously mark object and background pixels in a single interaction. We therefore update the segmentation using only one of the above sets: U_O or U_B . We select the set of update pixels U to be the set that has larger minimum distance between its center and the external border of the set of misclassified pixels it is drawn from:

$$U = \begin{cases} U_O & \max_{\mathbf{y} \in U_O} D(\mathbf{y}, E_O^C) > \max_{\mathbf{z} \in U_B} D(\mathbf{z}, E_B^C) \\ U_B & \text{otherwise} \end{cases} \quad (6.11)$$

Figure 6.2 shows the first few steps of this strategy when evaluating the IGC algorithm. As can be seen, the strategy begins by placing one or more circular

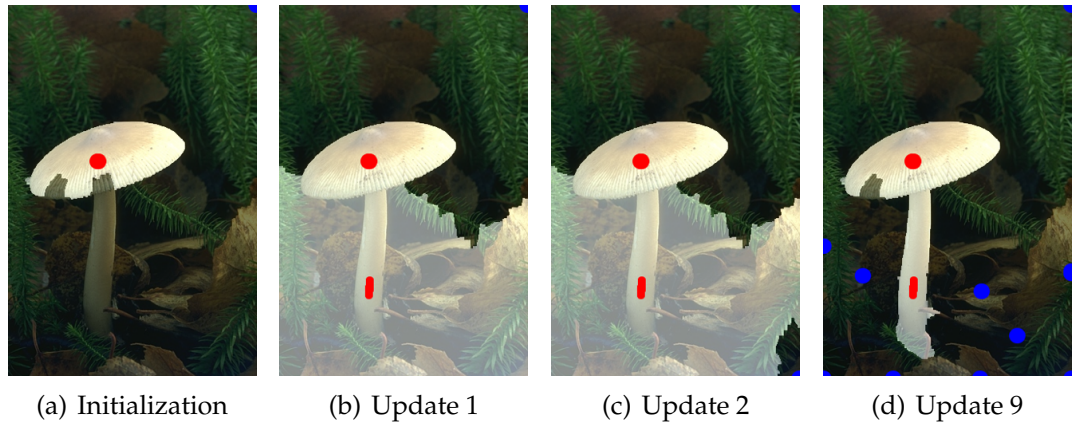


Figure 6.2: An illustration of the first few the steps of automation strategy 1. The segmentation algorithm being evaluated is IGC. Each update adds one or more circular blobs of seed pixels to correct errors in either the object or background regions.

blobs of seed pixels in the center of the object to be extracted, and one or more blobs of seed pixels well outside the object. Each update then corrects errors in the segmentation by placing seed pixels in the center of the largest regions of misclassified pixels.

The strategy can be efficiently implemented using a fast 2D Euclidean distance transform algorithm. Our implementation uses the linear time algorithm proposed by Meijster et al. [Meijster et al., 2000], which was demonstrated in [Fabbri et al., 2008] to be one of the fastest 2D Euclidean distance transform algorithms available. This allows us to evaluate Eq. (6.3) for all values of x in a region in linear time, and therefore initialization and each update also run in linear time.

This strategy has some advantages; it is relatively quick to compute, and since it is deterministic, it will produce the same sets of seed points given the same segmentation algorithm and ground truth each time it is run (provided, of course, that the segmentation algorithm is also deterministic). It therefore needs to be run only once for each algorithm being evaluated.

There are, however, two distinct disadvantages of this approach. First, since it is deterministic, it does not evaluate repeatability. The strategy gives no indication

of how robust the algorithm being evaluated is to small variations in markup: it always produces the same markup given the same algorithm and input. This behavior is in direct contrast to real users, who are unlikely to produce the same sets of markings when extracting a given object. Second, the strategy produces pixel blobs instead of lines and curves, similar to a user repeatedly clicking the mouse on the misclassified regions each time they wish to correct the segmentation. Although this is a perfectly valid way of extracting objects, observation indicates that users prefer to draw lines or curves.

We address both of these issues as we investigate more complex automation strategies. Strategy 1 is a useful baseline against which we can compare more sophisticated approaches.

6.2.2 Strategy 2

Recall that choosing our initialization and update seeds requires selecting a set of seed pixels S from a set of candidate pixels C . For strategy 1, we selected the set of pixels from C that were maximally distant from their nearest neighbors in C^c , then expanded our selection using a brush function. We do the same for strategy 2, except this time we select pixels from C non-deterministically. We propose selecting pixels from C such that the probability of selecting $x \in C$ is proportional to the spatial distance from x to C^c . This way we are more likely to select pixels that are nearer to the center of the object on initialization, and nearer to the center of groups of misclassified pixels on update.

To achieve this, we need to define a discrete probability distribution for selecting pixels from C based on their spatial distances to their nearest neighbors in C^c . Such a distribution could be defined in various ways; for simplicity, we opted to design our distribution using the sum normalized distances. This gives the

following discrete probability mass function:

$$\Pr[X = \mathbf{x}] = \frac{D(\mathbf{x}, C^c)}{\sum_{\mathbf{y} \in C} D(\mathbf{y}, C^c)} \quad (6.12)$$

The above mass function can be used to select seeds with the desired probabilities using the inversion method [Hörmann et al., 2004] as follows:

1. First, compute a discrete estimation of the cumulative distribution function for Eq. (6.12) as:

$$F(X) = \sum_{Y \leq X} \Pr[Y] \quad (6.13)$$

2. Generate a random number $u \in (0, 1)$ from the standard uniform distribution.
3. Find the smallest value \mathbf{x}_i such that $F(\mathbf{x}_i) \geq u$. Binary search on F can be used to find \mathbf{x}_i in $O(\log n)$ time.

The brush function is then applied as before to expand the selection. Since we now have a non-deterministic method of selecting the initialization and update pixels, we can evaluate repeatability by using multiple runs of the method, simulating multiple users.

6.2.3 Strategy 3

We noted previously that users tend to draw lines and curves to mark up objects, rather than simply pointing and clicking. To make the evaluation more realistic, we would prefer if our automation strategy provided similar interactions.

Our goal here is to select a sequence of seed points $P = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_n)$ from our candidate points C , such that each seed point is a neighbor of a previously

selected point:

$$\mathbf{x}_i \in P \implies \mathbf{x}_i \in C \quad (6.14)$$

$$\mathbf{x}_i \in P \wedge i > 1 \implies \mathbf{x}_{i-1} \in \mathcal{N}(\mathbf{x}_i) \quad (6.15)$$

where $\mathcal{N}(\mathbf{x}_i)$ is the set of 8-neighbors of \mathbf{x}_i .

There are typically many sequences P which satisfy the above predicates. Most of these, however, are not usually what we would (intuitively) consider realistic for a user to draw when marking up objects. Our experiments suggest that users tend to draw smooth, simple curves; we would ideally like to devise a strategy that emulates this behavior.

The path $P = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_n)$, has a start point \mathbf{x}_1 , and an end point \mathbf{x}_n . A logical way to create such a path is to select the start and end point using the strategy outlined in Section 6.2.2, and find a sequence of adjacent pixels joining the start and end points. The simplest and shortest path of pixels joining \mathbf{x}_1 and \mathbf{x}_n is a straight line rasterized on the pixel grid. Unfortunately, since the required objects are not always convex, a straight line is not guaranteed to fall entirely within the region we are marking. Figure 6.3 illustrates this issue. Figure 6.3(a) shows a synthetic non-convex object; 6.3(b) illustrates its distance transform. Figure 6.3(c) shows the maxima of the distance transform. These are also the two most likely points to be selected as endpoints by the method described in Section 6.2.2. Figure 6.3(d) shows that the line joining these two points lies outside the object. The ideal solution we would like to approximate is shown in Figure 6.3(e).

Our first attempt at approximating the kind of lines and curves generated by human operators uses the shortest spatial path on the image grid between points \mathbf{x}_1 and \mathbf{x}_n that lies completely inside the candidate region. To compute this path,

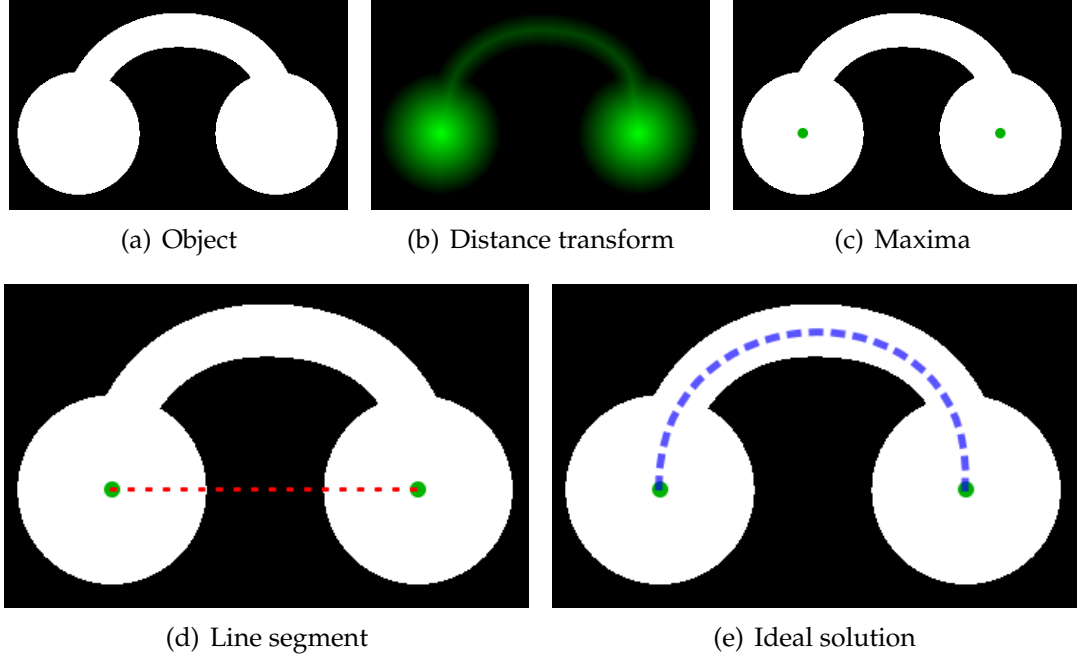


Figure 6.3: Extracting a non-convex object with straight line segments: (a) is a synthetic, non-convex object, designed so that it has two distinct maxima that cannot be joined using a straight line. (b) is the distance transform of (a). (c) is the maxima of the distance transform overlaid on the object. (d) shows a dashed line segment joining the maxima points. (e) is the ideal solution we would like to approximate.

we construct a graph $G = (V, E)$ from the candidate pixels C , such that each pixel $\mathbf{x} \in C$ is a vertex in the graph, and each vertex is connected to all of its eight neighbors also in C . That is:

$$G = (V, E) \tag{6.16}$$

$$V = \{\mathbf{x} : \mathbf{x} \in C\} \tag{6.17}$$

$$E = \{(\mathbf{x}, \mathbf{y}) : \mathbf{x} \in C, \mathbf{y} \in \mathcal{N}_{\mathbf{x}} \cap C\} \tag{6.18}$$

Each edge in E is then given a weight equal to the spatial distance between the vertices it joins. Since we are operating on an 8-connected graph, these weights are equal to 1 for horizontal and vertical edges, and $\sqrt{2}$ for diagonal edges. Having constructed G , we can now use Dijkstra's algorithm [Cormen et al., 2001] to find

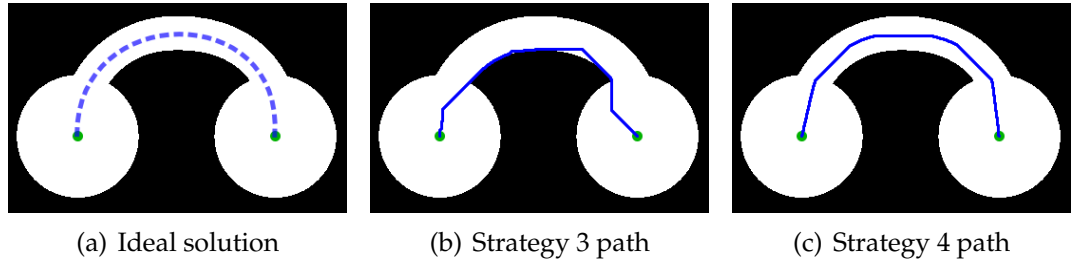


Figure 6.4: Paths found between the maxima points of Figure 6.3 using automation strategies 3 and 4.

the shortest path between x_1 and x_n . Figure 6.4(b) shows the path found using this approach on the example in Figure 6.3.

The algorithm, as specified thus far, will fail if x_1 and x_n lie in regions that are spatially disjoint. We must, therefore, avoid this situation and only choose our endpoints so that there always exists a path between them. The final strategy 3 algorithm is as follows:

1. Construct the candidate graph G .
2. Select an initial point x_1 using the non-deterministic strategy in Section 6.2.2.
3. Determine which other vertices in G are connected to x_1 using Dijkstra's algorithm. This also gives us the shortest path from x_1 to every other connected vertex in G .
4. Remove x_1 and all vertices not connected to x_1 in G from the candidate pixels C .
5. Select a second point x_n , again using the non-deterministic strategy in Section 6.2.2.
6. Set P equal to the shortest path between x_1 and x_n as found in step 3.

The set of seed pixels P' is then chosen by expanding the path P using a brush function similar to that in Equation (6.7). In this instance, the brush radius is

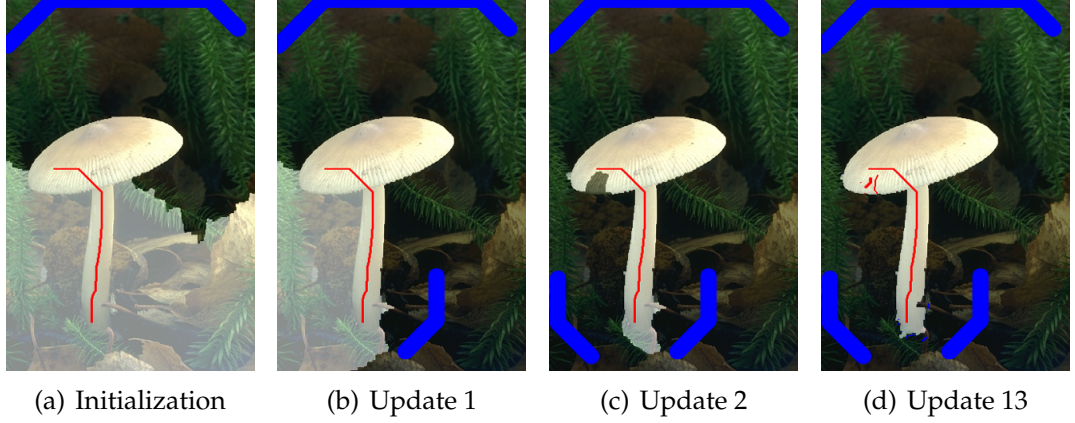


Figure 6.5: An illustration of the first few the steps of automation strategy 4. The segmentation algorithm being evaluated is IGC.

chosen relative to the minimum distance from any pixel on the path P to one of the non-candidate pixels C^c as follows:

$$P' = Br'(P, C^c) \quad (6.19)$$

$$Br'(P, R) = \{\mathbf{y} : \|\mathbf{x} - \mathbf{y}\| \leq r'(P, R), \mathbf{x} \in P\} \quad (6.20)$$

$$r'(P, R) = \min_{\mathbf{x} \in P} r(\mathbf{x}, R) \quad (6.21)$$

6.2.4 Strategy 4

The paths found by strategy 3 are the shortest possible. They will, therefore, often yield paths that pass very close to the boundary of the candidate region (see Figure 6.4(b)). We would prefer to generate paths that stay closer to the center of the region, as in Figure 6.4(a).

To achieve this, we propose adjusting the weights on the candidate graph G , so that paths that move toward the center of the object are preferred over the shortest possible path. Previously we set the weight for the edge between vertex

\mathbf{x} and \mathbf{y} equal to the spatial distance between them:

$$w_{\mathbf{x} \rightarrow \mathbf{y}} = \|\mathbf{x} - \mathbf{y}\|$$

Modulating the above by the exponent of the normalized distance from \mathbf{y} to the boundary introduces a preference to move toward the center of the object. Our modified distance function is:

$$w'_{\mathbf{x} \rightarrow \mathbf{y}} = \|\mathbf{x} - \mathbf{y}\| \exp \left(\frac{D(\mathbf{y}, C^c)}{\max_{\mathbf{z} \in C} D(\mathbf{z}, C^c)} \right)$$

which, for our previous example, yields the path in Figure 6.4(c). The path is again expanded using the brush function to form the set of seed pixels. Figure 6.5 shows a more realistic example of this strategy in action.

6.3 Evaluation

To run the automated evaluation we developed the automator tool, shown in Figure 6.6. The tool allows configuration of all aspects of the evaluation, including: the algorithm being evaluated, the automation strategy, the input and ground truth files, and the evaluation measures to use. When the evaluation is run, the tool processes each input image and corresponding ground truth with the selected automation strategy. After each automation step is taken, the segmentation is updated, and accuracy is computed against the ground truth using the selected accuracy measures. For the non-deterministic automation strategies, the process is repeated the desired number of times.

We set an upper limit of 100 steps for each automation strategy. This limit is imposed not only to ensure that the automation strategies terminate in a reasonable amount of time, it is also important during the analysis of the results, as we

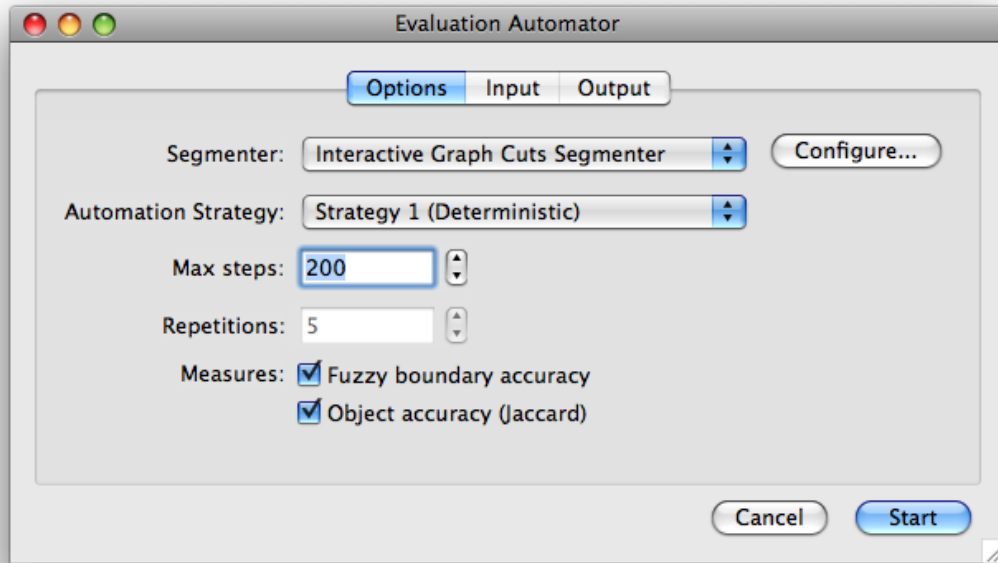


Figure 6.6: The configuration window for our automator tool.

shall see in the next section. To effectively evaluate repeatability, evaluation using the non-deterministic strategies needs to be repeated several times; we used five repetitions for our experiments.

The time required to run the automated experiments depends on the automation strategy used, on the algorithm being evaluated, on the number of images in the dataset, and on the number of repetitions when using a non-deterministic automation strategy. Strategy 1 is the least computationally intensive automation strategy, and strategy 4 is the most intensive. To give an idea of the typical time required to run an automated evaluation, Table 6.1 shows the approximate run-times of two automation tasks on our server machine (3 GHz Intel Xeon CPU, Linux kernel 2.6.20 x86_64).

Strategy	Algorithm	Repetitions	Max Steps	Objects	Time
1	IGC	1	100	100	25 min
4	IGC	5	100	100	2 hr 30 min

Table 6.1: Indicative runtimes of the automation strategies.

6.4 Analysis

The objective of our analysis is twofold. First, since there is a large amount of data generated by the experiments, we need to develop a effective means to reduce and interpret this data. We can then apply this to each evaluation strategy to investigate the characteristics of the evaluated algorithms. Second, we want to compare each of the evaluation strategies, and determine which best approximates full user experiments.

Every ground truth object evaluated results in a time series of object accuracy and boundary accuracy values. Figure 6.7 shows two such time series from our experiments. Each interactive segmentation algorithm evaluated produces 100 of these time series (one for each ground truth object). This is increased to 500 for the non-deterministic strategies, since the evaluation is run five times. To allow us to more easily interpret these data, we aggregate it in two ways.

The first way is to examine the average accuracy over all images as a function of time. We refer to this as the time-accuracy profile curve. The aim is to determine how, on average, accuracy varies over time for each evaluated algorithm. To compute time-accuracy profile curves, we first need to expand the time series data to the maximum number of steps. This is done by duplicating the final accuracy measured (i.e. the accuracy when the automation terminates) for each subsequent step up to the maximum. Once the time series data have been expanded to equal length, they are sampled at regular intervals using a fixed sampling window. These samples are then averaged to produce the time-accuracy profiles.

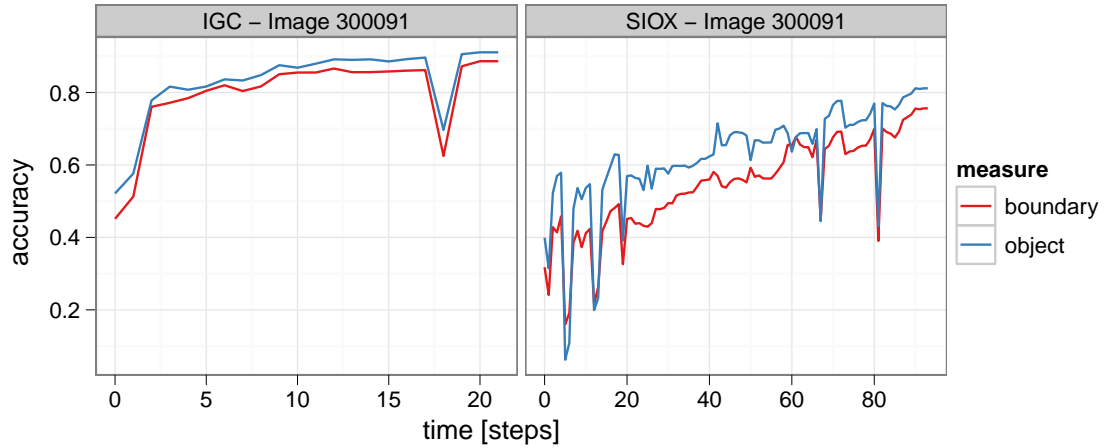


Figure 6.7: Sample time series data for individual images. The time series were created by evaluating the IGC algorithm (left) and the SIOX algorithm (right) with automation strategy 1 against the same ground truth image.

The second way we aggregate the data is by computing scalar features of the individual time series, and averaging these features across the collected data. We use two scalar features. The first is *final accuracy*, defined as the accuracy measured when the automation terminates; it gives an indication of the accuracy that can be achieved in a reasonable amount of time using a given segmentation algorithm. In Figure 6.7 this is simply the last value on the right hand side of the curves.

The second feature we compute is termed *integrated accuracy*. Consider again the time series in Figure 6.7. The time series on the left clearly indicates better performance than the time series on the right: it achieves a higher accuracy in less steps. Observe that algorithms that are performing well tend to produce curves that increase quickly at first and then gradually level-off (similar to a cumulative exponential distribution); algorithms that are performing poorly tend to have more gradual, “choppy” curves. One useful way to reduce these curves to a scalar value is to examine the area under the curves.

If we expand each time series data so that they are of equal length (in the same way as when computing the time-accuracy profile curves) then the area under the time series curve is a good indication of the overall performance of an algorithm. A large area relative to another indicates that one segmentation algorithm maintains a higher average accuracy the other, usually as a result of achieving an accurate segmentation faster and subsequently maintaining, or slowly improving upon, this segmentation. Furthermore, this area is bounded by the area of the unit height rectangle that is the same width as the expanded time series, and can therefore be easily normalized to the range $[0, 1]$.

It is possible to approximate the area under a time series by summation when the data points for the time series are unit spaced. The data points collected during an automated evaluation experiment *are* unit spaced: accuracy is measured after each step in the automation process. The area beneath this time series can, therefore, be approximated by the sum of all data points in the series.

We also need to calculate the integrated accuracy feature for the user experiments so that we can determine how well they correlate with the automated experiments. This again necessitates determining the area beneath the time series curves. However, the data points from the user experiments are non-unit spaced: accuracy is measured at different points in real time, i.e., every time the user refines the segmentation by marking additional pixels as object or background. To approximate the area under these time series using summation, linear resampling can be used to coerce the series to one that *is* unit spaced. It is also possible to approximate the area under non-unit spaced points using the trapezoid rule for numeric integration [Atkinson, 1989].

Denoting $\mathcal{A}(a)$ the area under the expanded time series $a = (a_1, a_2, \dots, a_k)$, as computed using one of the above procedures, we define the integrated accuracy feature as this area $\mathcal{A}(a)$ normalized by the area of the minimal unit height rectangle that encloses these points. For our experiments, we resample each time

series so that all data points are unit spaced, then approximate the area under the curves using the summation method. In this case, the integrated accuracy feature $\mathcal{I}(a)$ is equivalent to the area under the expanded time series normalized by the number of data points (i.e., it is the mean of the expanded time series):

$$\mathcal{I}(a) = \frac{1}{k} \sum_{i=1}^k a_i \quad (6.22)$$

Once the final accuracy and integrated accuracy features for each image have been computed, they can be averaged across all objects in the dataset to obtain an indication of the overall performance of the evaluated algorithm. Furthermore, they can be used to compare the output of the automation strategies with the user experiments. Section 6.4.2 uses these scalar features to investigate how well the four automation strategies approximate a real user.

6.4.1 Experiments

We evaluated each interactive segmentation algorithm using all four evaluation strategies. When computing the time-accuracy profiles and the aggregate features, each time series is expanded to the maximum number of steps by duplicating the final accuracy value. In addition, the time series values for each of the non-deterministic automation strategies (strategies 2, 3, and 4) is averaged across all the repetitions. Figure 6.8 shows the time-accuracy profiles for each of the four automation strategies. Figures 6.9 and 6.10 show the average final accuracy and integrated accuracy features.

It is difficult to formally compare the time-accuracy profiles from the user experiments (Figure 5.8) with those from the automated experiments. This is due to the difficulty in aligning the time series—such alignment would require

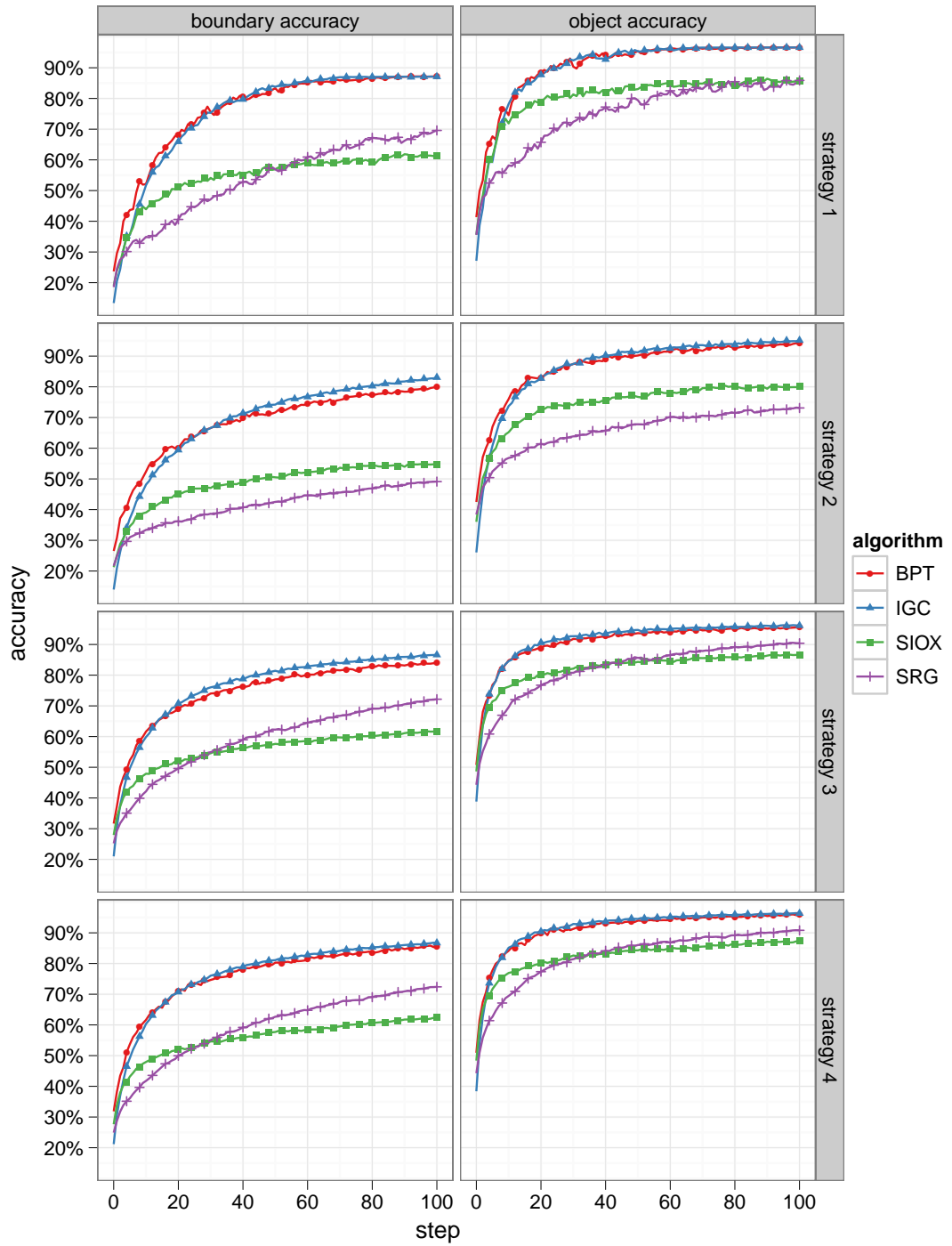


Figure 6.8: Mean accuracy over time for each of the evaluation strategies.

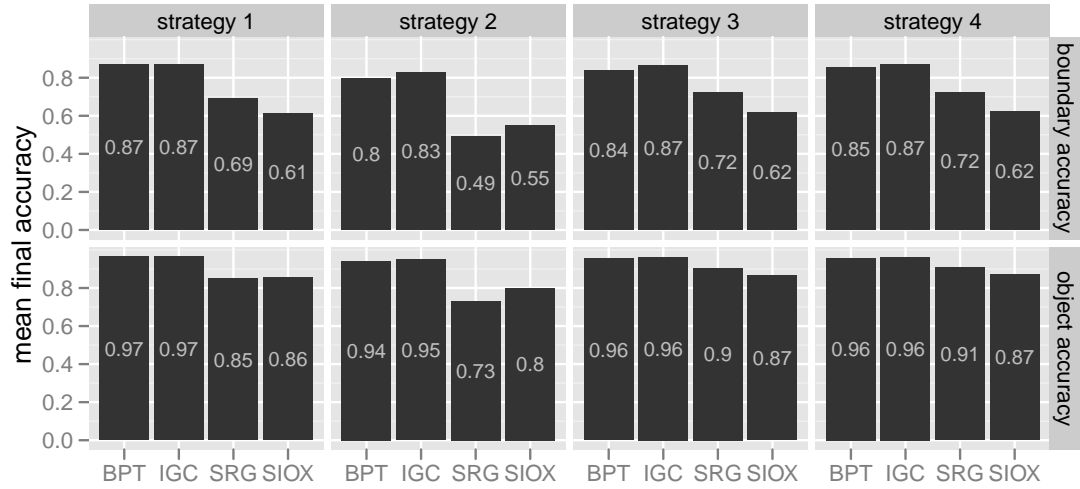


Figure 6.9: Mean final accuracy after 100 steps for each of the evaluation strategies

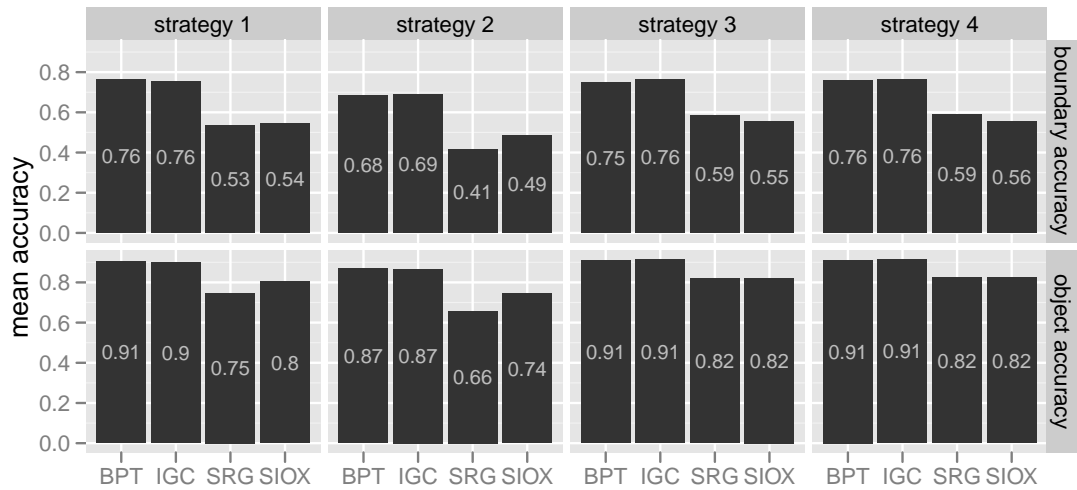


Figure 6.10: Mean integrated accuracy for 100 steps for each of the evaluation strategies

equating effort spent by an automated algorithm with effort spent by human user in a meaningful way. The y-axis on the time-accuracy profiles is designed to quantify this effort. For the user experiments, the actual elapsed time, in seconds, is a meaningful indicator of effort. For the automated experiments, however, the number of steps is more appropriate, as each step invokes the same

procedure. To compute a direct correlation between the time-accuracy profiles for the user experiments and the same profiles for the automated experiments necessitates aligning the data in some way. It is, however, possible to compute a more meaningful rank correlation using the aggregate features. Section 6.4.2 examines the correlation between aggregate features.

Despite the difficulty in performing a formal comparison between the time-accuracy profile curves in the user experiment and the time-accuracy profile curves for each of the automation strategies, a visual comparison is informative. Visually comparing the profile curves from strategy 1 with the profile curves for the user experiment indicate that strategy 1 does indeed give similar results to the user experiments. Furthermore, we can draw similar conclusions from the profile curves from strategy 1 as we did from the user experiments. The strategy 1 profile curves again indicate that the BPT and IGC algorithms are comparable, both demonstrating the best overall performance. The SIOX algorithm initially performs better the SRG algorithm. After about 60 steps the performance of the SIOX and SRG algorithms are comparable. The SRG algorithm surpasses the SIOX algorithm in terms of boundary accuracy after about 60 steps, again indicating that the SRG algorithm is more receptive to iterative refinement.

Visual comparison shows that strategy 2 is less effective than strategy 1 at approximating the results of the user experiments. In particular, the time-accuracy profile for strategy 2 suggests that the SIOX algorithm consistently outperforms that SRG algorithm, a conclusion not supported by the user experiments. Strategy 3 and 4 rectify this, giving the most satisfactory visual correspondence with the profile curves from the user experiments. The same conclusions can be drawn from these profile curves as were found in the user experiments.

Figure 6.9 shows the final accuracy values for each of the strategies. The results from strategy 1, 3, and 4 largely agree. The final accuracy values for the BPT and IGC algorithms are comparable. The BPT and IGC algorithms give higher

final accuracy than the SIOX and SRG algorithms. For strategy 3 and 4 the SRG algorithm gives higher final accuracy than the SIOX algorithm. Again, strategy 2 gives different results, showing the SIOX algorithm to have higher final accuracy than SRG. The integrated accuracy features in Figure 6.10 give similar rankings to the final accuracy features.

6.4.2 Correlation and Validation

The objective of the automation strategies is to simulate the interactions that a user would produce when given a particular segmentation task. Suppose a user is tasked with extracting two separate objects from different images using a particular segmentation algorithm. Usually, one of these objects will be more difficult to extract than the other; extracting it will require more time and more interactions. If an automation strategy is effective at emulating user behavior, then we would expect the same objects to be proportionately difficult for the automation strategy.

If a user finds object x more difficult to extract than object y , this will be reflected in the time-accuracy series produced during the segmentation. Therefore, if an automation strategy is effective, we expect there to be a correlation between the time-accuracy series produced by the user and the time-accuracy series produced by the automation strategy.

We discussed in the previous section the difficulty in performing a direct correlation between the time-accuracy series from the user experiments and the automated experiments. We can, however, examine the correlation between the aggregate time series features: the integrated accuracy and final accuracy values. These features can be interpreted as being indicative of the difficulty of a segmentation. High integrated accuracy and final accuracy values indicate that segmenting a particular image is easier, low values indicate that it is more difficult.

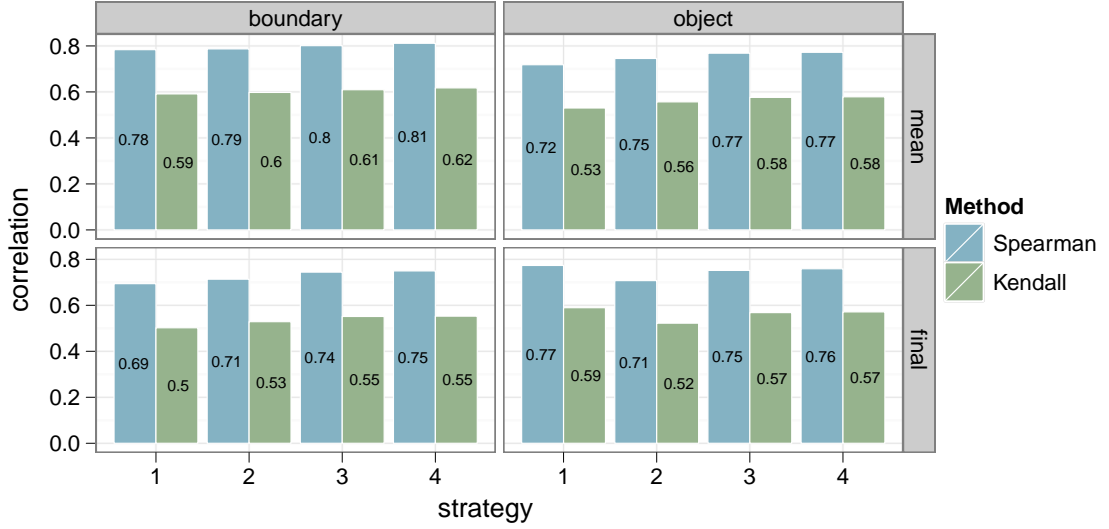


Figure 6.11: Rank correlation between the user experiments and each of the automation strategies.

Therefore, the rank correlation over all images between the features produced by an automation strategy and the user experiments should be relatively high if the automation strategy is successfully emulating user interactions.

To perform the comparison, we proceed as follows. First, we compute the integrated score and final accuracy features for each time series generated from the user experiments. When several users have segmented the same image with the same segmentation algorithm, we average the features across the different users. This gives us two integrated score values (one for boundary accuracy and one for object accuracy) and two final accuracy values for each image and algorithm evaluated. We follow a similar procedure for each automation strategy, this time averaging over runs for the non-deterministic strategies.

For each algorithm and image pair, the result is a set of four feature values for each of the automation strategies: mean final boundary accuracy, mean final object accuracy, mean integrated boundary accuracy, and mean integrated object accuracy. From this we calculate the rank correlation between the automation

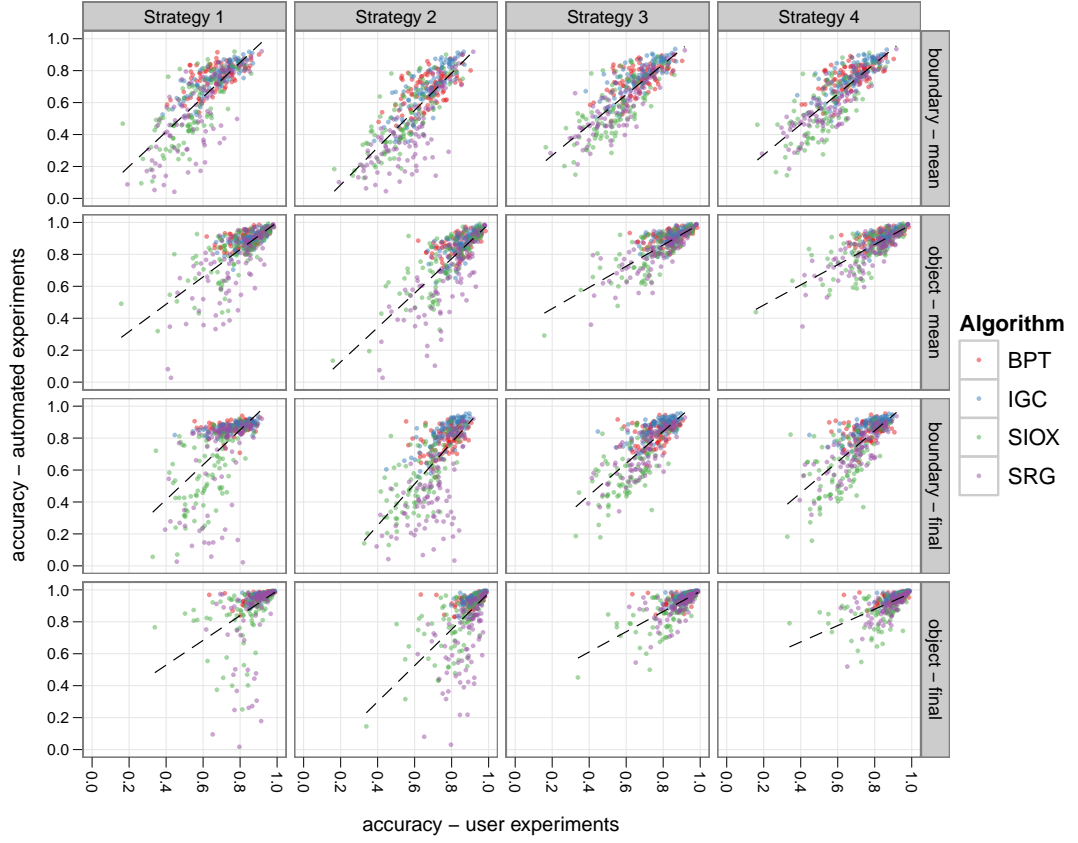




















Figure 6.12: Accuracy features for the user experiments plotted against the same features from the automated experiments. The panels depict the automation strategies from left to right, and the accuracy features from top to bottom.

strategies and the user experiments, for the two integrated score values and the two final accuracy values. We computed two rank correlation coefficients: Spearman's ρ , and Kendall's τ , shown in Figure 6.11. Both Spearman's ρ and Kendall's τ coefficients range from -1 to 1, where 1 indicates perfect correlation, -1 indicates perfect negative correlation and 0 indicates no correlation.

Figure 6.11 demonstrates that there is a high rank correlation between final accuracy and integrated score from the user experiments and the automated experiments, indicating that the time-accuracy series from the user experiments are similar to those of the automated experiments. Specifically, if we rank seg-

1 - 35008 (0.876)	2 - 12003 (0.847)	3 - 188063 (0.840)	4 - 388016 (0.830)	5 - 189011 (0.829)
				
6 - 293029 (0.816)	7 - 175043 (0.810)	8 - 208001 (0.797)	9 - 118035 (0.796)	10 - 198023 (0.792)
				





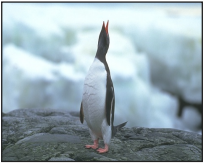





(a) User experiments

1 - 118035 (0.915)	2 - 293029 (0.894)	3 - 35008 (0.890)	4 - 310007 (0.879)	5 - 25098 (0.876)
				
6 - 12003 (0.873)	7 - 175043 (0.871)	8 - 188063 (0.869)	9 - 86016 (0.859)	10 - 388016 (0.852)
				








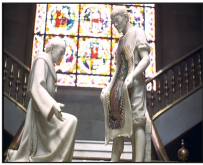


(b) Automated experiments (strategy 4)

Figure 6.13: Top ten “easiest” images to segment with the SIOX algorithm as judged using final boundary accuracy. These are the ten images from the dataset that gave the highest final accuracy values. The images are ranked left to right, top to bottom. The panel strips show the image tags and final boundary accuracy values.

mentation task difficulty based on the average final accuracy and integrated score measures, then the resulting segmentation task ranking from the user experiments exhibits high correlation with those from the automated experiments: they agree on how difficult a segmentation task is. This is a good indication that the automation strategies are, indeed, approximating user behavior in some useful way. The correlation values are very similar for each of the strategies. Strategy 4 gives

1 - 302008 (0.328)	2 - 271031 (0.345)	3 - 101087 (0.417)	4 - 253027 (0.436)	5 - 106024 (0.438)
				
6 - 102061 (0.449)	7 - 42078 (0.451)	8 - 109053 (0.452)	9 - 41033 (0.460)	10 - 101085 (0.471)
				

(a) User experiments

1 - 41033 (0.157)	2 - 302008 (0.183)	3 - 109053 (0.272)	4 - 108070 (0.323)	5 - 209070 (0.351)
				
6 - 42078 (0.355)	7 - 42012 (0.365)	8 - 24077 (0.368)	9 - 92059 (0.404)	10 - 101085 (0.406)
				

(b) Automated experiments (strategy 4)

Figure 6.14: Top ten “most difficult” images to segment with the SIOX algorithm as judged using final boundary accuracy. The images are ranked left to right, top to bottom. The panel strips show the image tags and final boundary accuracy values.

the best overall correlation with the user experiments for integrated boundary accuracy, integrated object accuracy, and final boundary accuracy.

Figure 6.12 shows a scatterplot of the accuracy features from the user experiments plotted against the same features from the automated experiments. Systematic deviations from the regression line (dashed black) indicate disagreement between the accuracy values produced by the user experiments and those

produced by the automated experiments. The figure clearly shows there are fewer such deviations for strategies 3 and 4 than there are for strategies 1 and 2: further evidence that the more complex strategies better approximate real users.

As a final illustrative example, Figure 6.13 shows the top ten “easiest” images to segment using the SIOX algorithm, as judged using the final boundary feature. Figure 6.13(a) shows the top images from the user experiments, and Figure 6.13(b) shows the top images from strategy 4 of the automated experiments. The groups share seven out of ten images, suggesting that the user experiments and the automated strategies agree on which images in the dataset are the easiest to segment using this particular segmentation algorithm. Similarly, Figure 6.14 shows the top ten most difficult objects to extract using the SIOX algorithm. In this case the sets share only five of the ten images; however, there is clearly strong visual similarity between the two sets.

6.5 Conclusion

When introducing a new interactive segmentation algorithm it is important to be able to compare its performance with the state-of-the-art. In the previous chapter we developed a set of benchmarks and software for supervised evaluation of interactive segmentation using user experiments. Carrying out these user experiments is, however, a time consuming and labor intensive exercise, often prohibitively so.

This chapter focused on eliminating the need for user experiments. To this end, we investigated four strategies for automating the evaluation of interactive segmentation algorithms. The objective of these strategies is to simulate interactions that would normally be provided by a human operator using the ground truth and current segmentation error. The first of these strategies is a simple, deterministic strategy: it always produces the same set of interactions

given the same segmentation algorithm and input. The remaining three strategies are non-deterministic, and therefore also allow evaluating the repeatability of an algorithm. Strategies 3 and 4 produce lines and curves instead of simple point interactions, aiming to more closely approximate the kinds of interactions usually produced by humans.

The experiments demonstrated that the results of the automated experiments are very similar to those of the user experiments. Evaluating the four segmentation algorithms using strategies 1, 3, and 4 all produced similar conclusions about the evaluated algorithms, and these conclusions agreed with the previously conducted user experiments. Validation using the rank correlation of aggregate features between the automation strategies and the user experiments indicated that strategy 4 is the most effective at approximating real user input.

Based on this analysis, we recommend using automation strategy 4 for practical experiments; of the four strategies, strategy 4 produced the time-accuracy profile curve that had the closest visual correspondence with the profile curves from the user experiments, and it produced the aggregate features that had the highest rank correlation with the user experiments. We recommend using a maximum of 100 steps or less; overall accuracy does not vary much after this point, and using too many steps can result in a rather unrealistic evaluation, since users rarely spend a lot of time correcting minor errors near the boundary of objects. Averaging over five repetitions of the experiment should be sufficient to evaluate repeatability.

If user experiments are feasible, then they are certainly the most effective way to evaluate interactive segmentation: it is difficult to evaluate any interactive system without getting feedback from real users. The automation strategies presented in this chapter are perhaps most useful when used as a preliminary step in an evaluation process. They allow algorithm developers to experiment with different variants of an algorithm to determine which is the most effective, without

having to re-conduct an entire set of user experiments each time. Automated evaluation also provides a means for researchers to determine if a particular approach to interactive segmentation appears to have practical merit, if it requires further consideration, if it needs modification, or if it should be abandoned, before expensive user-experiments are undertaken.

Of course, if they are feasible, user experiments should be performed for the final evaluation of an algorithm when comparing it against the state-of-the-art. In their absence, however, an automated evaluation, even if it is imperfect, will be more informative than no evaluation whatsoever. The system described in this chapter enables researchers to perform a useful and informative evaluation of their algorithms, even when full user experiments are impracticable.

Chapter 7

Conclusion

This thesis has focused on advancing image segmentation research by developing robust means for evaluating image segmentation algorithms. We reviewed the literature in image segmentation and segmentation evaluation, classified existing techniques, and identified areas that suggested further research. We applied existing evaluation measures to examine their properties and evaluate several segmentation algorithms, and developed new techniques to address gaps in the current research. This conclusion summarizes the research, outlines the key contributions, suggests potential directions for future research, and notes our related publications to date.

The following is a summary of the research that has been described in this thesis. Chapter 1 discussed the purpose and definition of image segmentation, described the motivations for the thesis, and outlined its structure. Chapter 2 and 3 reviewed the literature in image segmentation and segmentation evaluation.

Chapter 2 described the state-of-the-art in image segmentation algorithms. We began by creating a taxonomy of the different approaches to image segmentation. Our taxonomy classified segmentation algorithms using two perspectives: the application-centric perspective, and the algorithm-centric perspective. These perspectives comprised several facets: from the application-centric perspective

we classified algorithms by interaction, identification, media, and generality; from the algorithm-centric perspective we classified algorithms by perspective, model, and scale. We then discussed the grouping cues upon which image segmentation algorithms are either explicitly or implicitly based. We then proceeded to describe in detail four specific algorithms for automatic image segmentation: region adjacency graphs, statistical region merging, normalized cuts, and mean shift segmentation; and four specific algorithms for interactive image segmentation: seeded region growing, interactive graph cuts, interactive segmentation using binary partition trees, and simple interactive object extraction.

Chapter 3 reviewed the state-of-the-art in evaluating image segmentation algorithms. Again, we began our discussion by creating a taxonomy of segmentation evaluation techniques; the objective was to create a taxonomy that encompassed other taxonomies in the literature, and addressed their limitations. Our taxonomy classified segmentation evaluation techniques using three facets: objective, reference, and target. Objective referred to what the technique aims to evaluate; reference referred to an evaluation technique's use of ground truth; target referred to the type of algorithm that the algorithm is intended to evaluate. We then discussed several specific methods for segmentation evaluation, both supervised and unsupervised.

Chapter 4 focused on an experiment that we carried out for evaluating automatic segmentation algorithms. The objective of the experiment was twofold. Our first aim was to determine the characteristics of several supervised segmentation evaluation measures, and in particular, to establish which of these methods is the most effective for evaluating an image segmentation algorithm's performance at emulating human perceptual grouping. Our second objective was to use the best of these measures to evaluate four different image segmentation algorithms, and to determine how well the algorithms approximate perceptual grouping.

Chapter 5 focused on evaluating interactive segmentation algorithms. Since interactive segmentation evaluation had not yet been addressed in the literature, it was necessary to develop new techniques. We began by selecting four interactive segmentation algorithms for evaluation. Following this, we considered what the objectives of the evaluation were, and what we were required to evaluate. We then proposed two measures for the evaluation: one to measure object accuracy, and another to measure boundary accuracy. The object accuracy measure is simply the Jaccard index applied to the segmented object pixels; the advantage being that the values it gives can be directly compared with the existing literature (for example, [Ge et al., 2007]). The boundary accuracy measure is a fuzzy measure designed to gauge the accuracy of the object contour. After discussing these measures, we outlined the experiment itself, including: the software used, the ground truth, and the experiment setup and deployment. We then analyzed the results of the experiment and discussed their implications. We validated our proposed measures against perceived accuracy as gauged using questionnaires, and demonstrated that they correlate better with perceived accuracy than other benchmarks. We found that the best performing algorithms were the binary partition tree algorithm and the interactive graph cuts algorithm. Our experiment also suggested that people prefer predictable algorithms, even if this means the algorithm converges on the correct segmentation more slowly.

The methodology developed in Chapter 5 for evaluating interactive segmentation requires user experiments. While this is natural (and some might argue, necessary) for evaluating an interactive process, the time and effort required is often prohibitively costly. Chapter 6 addressed this by investigating whether user interactions can be satisfactorily simulated using an algorithmic process. We investigated four different strategies for simulating the user interactions and showed that the best of these produces results very similar to the results found by the user experiments in Chapter 5.

The key contributions of the work are: (1) a review of the state-of-the-art in image segmentation and image segmentation evaluation; (2) an investigation into the properties of existing automatic segmentation evaluation techniques; (3) an evaluation of four well-known algorithms using existing automatic segmentation evaluation techniques; (4) a complete framework for evaluating interactive segmentation algorithms by means of user experiments, including software, benchmarks, and ground truth; (5) an evaluation of four popular interactive segmentation algorithms using this framework; (6) a method for evaluating interactive segmentation algorithms that does not require user experiments.

There are several potential directions for future work based on what we have discussed. The review in Chapter 2 suggests several potential enhancements for existing segmentation algorithms, and our evaluation in Chapter 4 gives a baseline against which new algorithms can be compared. A straightforward direction for future research could, therefore, be to implement some of the proposed enhancements and evaluate them using the measures from Chapter 4 to determine if performance has been significantly enhanced as a result. Another similar direction for future research could be to use the technique and measures described in Chapters 5 and 6 to evaluate possible enhancements to existing interactive segmentation algorithms. This direction may give answers to many interesting research questions, for example: whether the introduction of color or texture features into the interactive graph cuts algorithm improves performance, or whether introducing a spatial bias into the simple interactive object extraction algorithm improves performance. Finally, the strategies we investigated for evaluating interactive segmentation in Chapter 6 are based on empirical observation and are, therefore, somewhat ad-hoc; another future research direction could be to investigate new ways to simulate the user interactions.

Parts of the research in this thesis have been published elsewhere. In particular, the following publications relate to the work described herein. The image

and video segmentation software framework that we used for the experiments in Chapter 4 was described in a publication for the SAMT conference in 2006 [McGuinness et al., 2006]. The preliminary investigation that formed the basis of Chapter 4 was published at the VIE conference in 2007 [McGuinness et al., 2007]. A paper at the 2008 SAMT conference describes the interactive segmentation software that we used in Chapter 5, and a web-based version of the automatic segmentation software that we used in Chapter 4 [McGuinness and O'Connor, 2008]. The interactive segmentation evaluation framework and the experiments described in Chapter 5 were published in the Pattern Recognition Journal in 2009 [McGuinness and O'Connor, 2009].

Appendix A

Evaluation Dataset

The table on the following pages shows the full dataset that we developed for evaluating interactive segmentation. The table contains the task descriptions, thumbnails of the images, and thumbnails of the object masks. The object masks were created manually using a graphics tablet and the GNU image manipulation program (GIMP). The full dataset comprises 100 tasks, and can be downloaded from the interactive segmentation website: <http://kspace.cdvdp.dcu.ie/public/interactive-segmentation>.

The images in the dataset were selected from the Berkeley segmentation dataset. The full Berkeley set is available from: <http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds>. The task numbers shown in the table correspond to the image identifiers used in the Berkeley set. The aspect ratio of the images has been altered for presentation purposes.







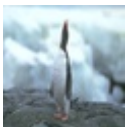

















Image	Object	Task description
		Task #101085: Extract the statue on the left hand side of the image.
		Task #101087: Extract the person. Include all the clothing and decorative wear. If possible, also include the long object the person is holding.
		Task #102061: Extract the building, roof and roof structures. Do not include the walls on the left and right hand sides of the building. Do not include any water or trees.
		Task #106024: Extract the penguin. Include feet and tail.
		Task #108005: Extract the tiger.
		Task #108070: Extract the tiger. The overlapping foreground twig can be ignored.
		Task #108082: Extract the tiger.
		Task #109053: Extract the wolf.
		Task #118035: Extract the entire building. Background cables etc. can be ignored. Remember to exclude the sky area beneath the bell, if possible.
		Task #12003: Extract the starfish.
		Task #123074: Extract the mouse. Include the tail.
		Task #124084: Extract both flowers (red and yellow).







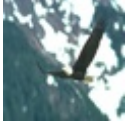





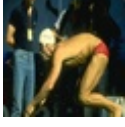



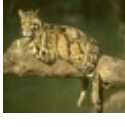





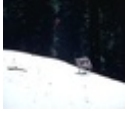
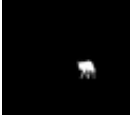
Image	Object	Task description
		Task #126007: Extract the building. Try to include the spire structures on the rooftops, if possible.
		Task #130026: Extract the crocodile.
		Task #134035: Extract the leopard.
		Task #135037: Extract the eagle. Include the wing feathers.
		Task #138078: Extract the boat. Include the rope. Do not include the oar.
		Task #151087: Extract the man in the center with the baseball bat. Include the bat.
		Task #153077: Extract the swimmer.
		Task #157055: Extract the man on the left of the image. Include the glass that he is holding.
		Task #160068: Extract the jaguar. Include feet and tail. Do not include the branch.
		Task #163014: Extract the upper bird (black and yellow). Include the birds feet.
		Task #163062: Extract the bird (in flight). Include wings, beak, tail and feet.
		Task #167062: Extract the wolf.










Image	Object	Task description
		Task #170057: Extract the visible parts of the soldier on the right hand side of the image. Do not include the gun or strap.
		Task #175043: Extract the green snake. Do not include what the snake is eating.
		Task #181079: Extract the woman from the image. Include her hair.
		Task #187029: Extract the child from the image.
		Task #188063: Extract the visible parts of the tent.
		Task #189011: Extract the person, hat and bucket from the background.
		Task #189011a: Extract the person and bucket from the background. Do not include the hat.
		Task #189011b: Extract the hat from the image. Do not include the person.
		Task #189080: Extract the person. Include the hat, head and shoulders.
		Task #196015: Extract the bird, including feet and tail, from the image.
		Task #196073: Extract the snake from the image.





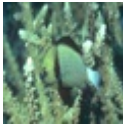
















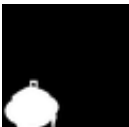
Image	Object	Task description
		Task #198023: Extract the face and hair of the woman from the image. Include the visible part of her neck. Do not include any of her clothing or hands.
		Task #208001: Extract the visible parts of the mushroom from the image. Avoid occluding foreground objects.
		Task #209070: Extract the fish.
		Task #21077: Extract the white car. Include the driver, wheels and all other visible parts of the car. Try not to include any of the road or the red car
		Task #216053: Extract the woman in blue from the image. Include her umbrella and the bag she is holding.
		Task #216066: Extract the visible parts of the stone sign from the image.
		Task #227092: Extract the urn and handles.
		Task #229036: Extract both persons and the objects they are holding (drum, batons etc.) from the background.
		Task #23080: Extract the man painting. Include all visible parts of the man. Include the paint brush. Do not include the paint can.
		Task #239007: Extract both the girl and the black object on the bottom left from the background.
		Task #239007a: Extract the black object on the lower left from the image.









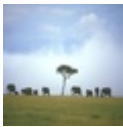


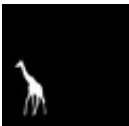


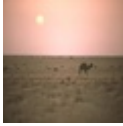
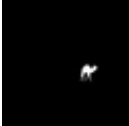




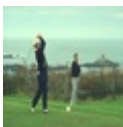
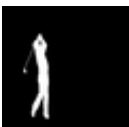
Image	Object	Task description
		Task #239007b: Extract the girl from the image.
		Task #24077: Extract the statue on the left hand side of the image.
		Task #25098: Extract the sign that says "Sweet Red Peppers - \$1.99". Include the wooden handle on the sign.
		Task #253027: Extract the rightmost zebra from the image. Include visible parts of it's legs and tail.
		Task #253036: Extract visible parts of the tree in the center of the image from the background. Try to include as much of the tree and as little of the sky as possible.
		Task #253055: Extract the leftmost giraffe from the image.
		Task #268002: Extract the visible parts of the bird from the image. Include feet and tail.
		Task #271031: Extract the camel from the background.
		Task #271035: Extract the person from the image. Include visible parts of the hands head and torso. Do not include the items the person is carrying.
		Task #285079: Extract the visible parts of the fireman from the image. Include his helmet. Do not include the object he is holding. The helmet visor can be ignored.
		Task #286092: Extract the golfer on the left from the image. Include his golf club, if possible.







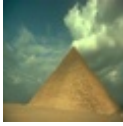












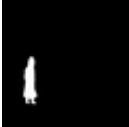


Image	Object	Task description
		Task #291000: Extract the horse from the image. Include the object in the horses mouth. Try and avoid the overlapping fence in the foreground, if possible.
		Task #293029: Extract just the man's hat. Do not include the rest of the man or any or any other objects.
		Task #296059: Extract all visible parts of both elephants from the image.
		Task #299091: Extract the pyramid from the image. Do not include any of the sand or sky.
		Task #300091: Extract the visible parts of the surfer from the image. Include his surf board.
		Task #302008: Extract the man's face, ears, hair and neck from the image. Do not include his shirt or shoulders.
		Task #304034: Extract visible parts of the black panther from the image. Try to avoid any significant occluding objects, such as twigs and leaves.
		Task #304074: Extract the ram.
		Task #3096: Extract the airplane. Include the visible propellers, if possible.
		Task #310007: Extract the person in red on the left from the background.
		Task #35008: Extract the white flower in the upper part of the image. Do not include the stem.



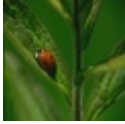

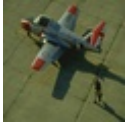







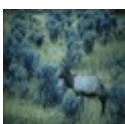
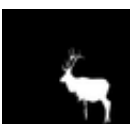
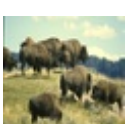



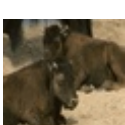

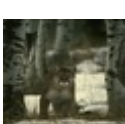
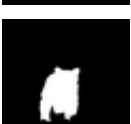
Image	Object	Task description
		Task #35010: Extract the butterfly. Include the legs and antenna if possible.
		Task #35058: Extract the ladybird (ladybug).
		Task #37073: Extract the airplane. Include the front wheel. Do not include the ground, shadow, or person.
		Task #372047: Extract the guard from the image. Include the rifle and sword.
		Task #376020: Extract the man in orange from the image. Avoid including the yellow item on the left.
		Task #376043: Extract the visible parts of the man from the image. Include his helmet, helmet-straps and boots. Avoid the grass and background wall.
		Task #38082: Extract the deer. Include the antlers, if possible. Try to avoid including as much of the obstructing bushes etc. as possible.
		Task #38092: Extract the central bison (at position [285,165])
		Task #388016: Extract the woman from the image. Include her hair.
		Task #41033: Extract the calf on the left hand side of the image.
		Task #42012: Extract as much as is visible of the cougar from the image.

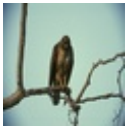




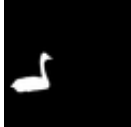






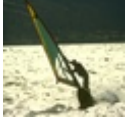






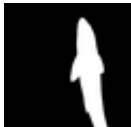




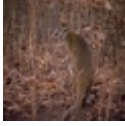



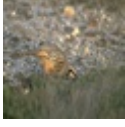













Image	Object	Task description
		Task #42049: Extract the eagle. Include the feet, wings and and tail. Try to avoid including any of the branch.
		Task #42078: Extract the wolf.
		Task #43070: Extract the leftmost swan.
		Task #43074: Extract as much of the pheasant as is visible. Small pieces of overlapping grass can be ignored.
		Task #56028: Include the painted Nepalese symbol (Buddist wisdom eyes) from the image.
		Task #58060: Extract the corn from the bag in the image. Do not include the bag itself.
		Task #62096: Extract the windsurfer, including board and sail, from the image. The transparent portion of the sail, and it's handles should also be included.
		Task #65019: Extract the full person from the image. Include his hat.
		Task #65074: Extract the woman and musical instrument from the image. Include all visible parts of the woman and instrument. Do not include the sheet music in front of her.
		Task #65132: Extract the topmost fish on the center-right of the image.
		Task #66053: Extract the visible parts of leftmost pig.

Image	Object	Task description
		Task #69020: Extract the kangaroo. Include the legs and tail.
		Task #69040: Extract the kangaroo. Include all visible parts of the kangaroo. Try to avoid the occluding branches.
		Task #78004: Extract the white boat.
		Task #8023: Extract the bird from the image. Include all visible parts of the bird.
		Task #85048: Extract the person. Include the hat and hammer. Avoid any obscuring slabs and the block of wood between the hammer and the man's leg.
		Task #86016: Extract the circular area in the center of the picture from the rest of the image.
		Task #89072: Extract the man from the image. Include the camera and helmet. Do not include the sign that the man is holding on the left hand side.
		Task #90076: Extract the boy from the image. Include what he is holding.
		Task #92059: Extract the boat from the image. Include the oars.
		Task #97033: Extract the house structure on the left hand side of the image. Include the roof. Omit the shelter area in the center and any areas obscured by foreground snow.

Bibliography

- [Adamek, 2006] Adamek, T. (2006). *Using Contour Information and Segmentation for Object Registration, Modeling and Retrieval*. PhD thesis, Dublin City University, Dublin 9, Ireland.
- [Adamek and O'Connor, 2006] Adamek, T. and O'Connor, N. (2006). Interactive object contour extraction for shape modeling. In *1st International Workshop on Shapes and Semantics*, pages 31–39.
- [Adamek et al., 2005] Adamek, T., O'Connor, N., and Murphy, N. (2005). Region-based segmentation of images using syntactic visual features. In *6th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*.
- [Adams and Bischof, 1994] Adams, R. and Bischof, L. (1994). Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):641–647.
- [Adelson, 1995] Adelson, E. H. (1995). The checkers shadow illusion. http://web.mit.edu/persci/people/adelson/checkersshadow_illusion.html.
- [Athanasiadis et al., 2006] Athanasiadis, T., Avrithis, Y., and Kollias, S. (2006). A semantic region growing approach in image segmentation and annotation. In *Proceedings of 1st International Workshop on Semantic Web Annotations for Multimedia (SWAMM)*.

- [Athanasiadis et al., 2005] Athanasiadis, T., Tzouvaras, V., Petridis, V., Precioso, F., Avrithis, Y., and Kompatsiaris, Y. (2005). Using a multimedia ontology infrastructure for semantic annotation of multimedia content. In *5th International Workshop on Knowledge Markup and Semantic Annotation*.
- [Atkinson, 1989] Atkinson, K. E. (1989). *An Introduction to Numerical Analysis*. Wiley, 2nd edition.
- [Attneave, 1954] Attneave, F. (1954). Some informational aspects of visual perception. *Psychological Review*, 61(3):183–93.
- [Bailer et al., 2005] Bailer, W., Schallauer, P., Haraldsson, H. B., and Rehatschek, H. (2005). Optimized mean shift algorithm for color segmentation in image sequences. *Proceedings of SPIE*, 5685:522–529.
- [Ben-Hur et al., 2002] Ben-Hur, A., Elisseeff, A., and Guyon, I. (2002). A stability based method for discovering structure in clustered data. *Pacific Symposium on Biocomputing*, 7:6–17.
- [Bennstrom and Casas, 2004] Bennstrom, C. F. and Casas, J. R. (2004). Binary-partition-tree creation using a quasi-inclusion criterion. In *Proceedings of the Eighth International Conference on Information Visualization (IV)*.
- [Bentley, 1975] Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517.
- [Borsotti et al., 1998] Borsotti, M., Campadelli, P., and Schettini, R. (1998). Quantitative evaluation of color image segmentation results. *Pattern Recognition Letters*, 19(8):741–747.
- [Boykov and Jolly, 2001] Boykov, Y. and Jolly, M.-P. (2001). Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *International Conference on Computer Vision*, pages 105–112.

- [Boykov and Kolmogorov, 2004] Boykov, Y. and Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137.
- [Brice and Fennema, 1970] Brice, C. and Fennema, C. (1970). Scene analysis using regions. *Artificial Intelligence*, 1:205–226.
- [Canny, 1986] Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 8(6):679–698.
- [Carreira-Perpinán, 2006] Carreira-Perpinán, M. A. (2006). Acceleration strategies for gaussian mean-shift image segmentation. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 1160–1167.
- [Carson et al., 2002] Carson, C., Belongie, S., Greenspan, H., and Malik, J. (2002). Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1026–1038.
- [Caselles et al., 1995] Caselles, V., Kimmel, R., and Sapiro, G. (1995). Geodesic active contours. *Proceedings of the Fifth International Conference on Computer Vision*, pages 694–699.
- [Chabrier et al., 2006] Chabrier, S., Emile, B., Rosenberger, C., and Laurent, H. (2006). Unsupervised performance evaluation of image segmentation. *EURASIP Journal on Applied Signal Processing*, 2006.
- [Chang et al., 2002] Chang, D., Dooley, L., and Tuovinen, J. E. (2002). Gestalt theory in visual screen design: a new look at an old subject. In *Proceedings of the*

7th world conference on computers in education conference on Computers in education: Australian topics (CRPIT), volume 8, pages 5–12.

- [Chen and Wang, 2004] Chen, H.-C. and Wang, S.-J. (2004). The use of visible color difference in the quantitative evaluation of color image segmentation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 593–6.
- [Cheng et al., 2001] Cheng, H. D., Jiang, X. H., Sun, Y., and Wang, J. (2001). Color image segmentation: advances and prospects. *Pattern Recognition*, 34(12):2259–228.
- [Chernick, 1999] Chernick, M. R. (1999). *Bootstrap methods: a practitioner's guide*. Wiley.
- [Christoudias et al., 2002] Christoudias, C. M., Georgescu, B., and Meer, P. (2002). Synergism in low level vision. In *Proceedings of the 16th International Conference on Pattern Recognition (ICPR)*, pages 150–155, Washington, DC, USA.
- [Clausi and Jernigan, 2000] Clausi, D. A. and Jernigan, M. E. (2000). Designing gabor filters for optimal texture separability. *Pattern Recognition*, 33(11):1835–1849.
- [Comaniciu and Meer, 2002] Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619.
- [Cormen et al., 2001] Cormen, T. H., Clifford, S., Leiserson, C. E., and Rivest, R. L. (2001). *Introduction to Algorithms*. MIT Press, 2nd edition.
- [Correia and Pereira, 2003] Correia, P. and Pereira, F. (2003). Objective evaluation of video segmentation quality. *IEEE Transactions on Image Processing*, 12(2):186–200.

- [Correia and Pereira, 2006] Correia, P. and Pereira, F. (2006). Video object relevance metrics for overall segmentation quality evaluation. *EURASIP Journal on Applied Signal Processing*, 2006.
- [da F. Costa and Cesar, 2001] da F. Costa, L. and Cesar, R. M. (2001). *Shape Analysis and Classification*. CRC Press.
- [Duda and Hart, 1972] Duda, R. O. and Hart, P. E. (1972). Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15.
- [Duda et al., 2001] Duda, R. O., Hart, P. E., and Stork, D. E. (2001). *Pattern Classification*. Wiley, 2nd edition.
- [Ebner, 2007] Ebner, M. (2007). *Color constancy*. Wiley.
- [Fabbri et al., 2008] Fabbri, R., Costa, L. D. F., Torelli, J. C., and Bruno, O. M. (2008). 2D Euclidean distance transform algorithms: a comparative survey. *ACM Computing Surveys*, 40(1):1–44.
- [Falcão et al., 2000] Falcão, A. X., Udupa, J. K., and Miyazawa, F. (2000). An ultra-fast user-steered image segmentation paradigm: live wire on the fly. *IEEE Transactions on Medical Imaging*, 19(1):55 – 62.
- [Falcão et al., 1998] Falcão, A. X., Udupa, J. K., Samarasekera, S., Sharma, S., Hirsch, B. E., and de A. Lotufo, R. (1998). User-steered image segmentation paradigms: live wire and live lane. *Graphical Models and Image Processing*, 60(4):233–260.
- [Fan et al., 2001] Fan, J., Yau, D., Elmagarmid, A., and Aref, W. (2001). Automatic image segmentation by integrating color-edge extraction and seeded region growing. *IEEE Transactions on Image Processing*, 10(10):1454–1466.

- [Farag and Delp, 1995] Farag, A. A. and Delp, E. J. (1995). Edge linking by sequential search. *Pattern Recognition*, 28(5):611–633.
- [Fowlkes and Mallows, 1983] Fowlkes, E. and Mallows, C. (1983). A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569.
- [Friedland et al., 2005] Friedland, G., Jantz, K., and Rojas, R. (2005). SIOX: simple interactive object extraction in still images. In *Proceedings of the 7th IEEE International Symposium on Multimedia*, pages 253–260.
- [Fu and Mui, 1981] Fu, K. and Mui, J. (1981). A survey on image segmentation. *Pattern Recognition*, 13(1):3–16.
- [Galmar and Huet, 2006] Galmar, E. and Huet, B. (2006). Graph-based spatio-temporal region extraction. In *3rd International Conference on Image Analysis and Recognition (ICIAR)*, Portugal.
- [Garrido et al., 1998] Garrido, L., Salembier, P., and Garcia, D. (1998). Extensive operators in partition lattices for image sequence analysis. *Signal Processing*, 66(2):157–180.
- [Ge et al., 2006] Ge, F., Wang, S., and Liu, T. (2006). Image-segmentation evaluation from the perspective of salient object extraction. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1146–1153.
- [Ge et al., 2007] Ge, F., Wang, S., and Liu, T. (2007). New benchmark for image segmentation evaluation. *Journal of Electronic Imaging*, 16(3):033011.
- [Greig et al., 1989] Greig, D., Porteous, B., and Seheult, A. (1989). Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, 51(2):271–279.

- [Grossberg and Raizada, 2000] Grossberg, S. and Raizada, R. (2000). Contrast-sensitive perceptual grouping and object-based attention in the laminar circuits of primary visual cortex. *Vision Research*, 40(10-12):1413–1432.
- [Hörmann et al., 2004] Hörmann, W., Leydold, J., and Derflinger, G. (2004). *Automatic Nonuniform Random Variate Generation*. Springer.
- [Horowitz and Pavlidis, 1974] Horowitz, S. L. and Pavlidis, T. (1974). Picture segmentation by a directed split and merge procedure. In *Proceedings of the 2nd International Joint Conference on Pattern Recognition*.
- [Horowitz and Pavlidis, 1976] Horowitz, S. L. and Pavlidis, T. (1976). Picture segmentation by a tree traversal algorithm. *Journal of the ACM*, 23(2):368–388.
- [Huang and Dom, 1995] Huang, Q. and Dom, B. (1995). Quantitative methods of evaluating image segmentation. In *International Conference on Image Processing (ICIP)*, pages 53–56.
- [Itti et al., 1998] Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259.
- [Jiang et al., 2005] Jiang, X., Marti, C., Irniger, C., and Bunke, H. (2005). Image segmentation evaluation by techniques of comparing clusterings. In *International Conference on Image Analysis and Processing (ICIAP)*, Cagliari, Italy.
- [Jiang et al., 2006] Jiang, X., Marti, C., Irniger, C., and Bunke, H. (2006). Distance measures for image segmentation evaluation. *EURASIP Journal on Applied Signal Processing*, 2006.
- [Johnson and Fairchild, 2003] Johnson, G. M. and Fairchild, M. D. (2003). A top down description of S-CIELAB and CIEDE2000. *Color Research & Application*, 28(6):425–435.

- [Kanungo et al., 2002] Kanungo, T., Mount, D., Netanyahu, N., Piatko, C., Silverman, R., and Wu, A. (2002). An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892.
- [Kass et al., 1988] Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: active contour models. *International Journal of Computer Vision*, 1(4):321–331.
- [Kendall, 1938] Kendall, M. (1938). A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.
- [Kikinis et al., 1996] Kikinis, R., Shenton, M., Iosifescu, D., McCarley, R., Saiviroonporn, P., Hokama, H., Robatino, A., Metcalf, D., Wible, C., Portas, C., Donnino, R., and Jolesz, F. (1996). A digital brain atlas for surgical planning, model-driven segmentation, and teaching. *IEEE Transactions on Visualization and Computer Graphics*, 2(3):232 – 241.
- [Koenemann and Belkin, 1996] Koenemann, J. and Belkin, N. J. (1996). A case for interaction: A study of interactive information retrieval behavior and effectiveness. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 205–212.
- [Koffka, 1935] Koffka, K. (1935). *Principles of Gestalt Psychology*. Harcourt, New York.
- [Kompatsiaris and Strintzis, 1999] Kompatsiaris, I. and Strintzis, M. G. (1999). Spatiotemporal segmentation and tracking of objects for visualization of video-conference image sequences. In *IEEE International Conference on Multimedia Computing and Systems (ICMCS)*, volume 01, page 9709.
- [Kuhn, 1955] Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

- [Lee et al., 1998] Lee, T., Mumford, D., Romero, R., and Lamme, V. (1998). The role of the primary visual cortex in higher level vision. *Vision Research*, 38(15/16):2429–2454.
- [Li et al., 2004] Li, Y., Sun, J., Tang, C.-K., and Shum, H.-Y. (2004). Lazy snapping. *ACM Transactions on Graphics*, 23(3):303–308.
- [Liang et al., 1999] Liang, J., McInerney, T., and Terzopoulos, D. (1999). Interactive medical image segmentation with united snakes. *Medical Image Computing and Computer-Assisted Intervention*, 1679:116–127.
- [Lim and Lee, 1990] Lim, Y. and Lee, S. (1990). On the color image segmentation algorithm based on the thresholding and the fuzzy c-means techniques. *Pattern Recognition*, 23(9):935–952.
- [Liu and Yang, 1994] Liu, J. and Yang, Y. (1994). Multiresolution color image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(7):689–700.
- [Lucchese and Mitra, 2001] Lucchese, L. and Mitra, S. (2001). Color image segmentation: a state-of-the-art survey. In *Proceedings of the Indian National Science Academy (INSA-A)*, volume 67-2, pages 207–222.
- [Mao et al., 1999] Mao, F., Gill, J. D., and Fenster, A. (1999). Technique for evaluation of semiautomatic segmentation methods. In *Proceedings of the SPIE*, volume 3661, pages 1027–1036.
- [Marr and Hildreth, 1980] Marr, D. and Hildreth, E. (1980). Theory of edge detection. In *Proceedings of the Royal Society of London. Series B, Biological Sciences*, volume 207, pages 187–217.
- [Martin et al., 2001] Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001). A database of human segmented natural images and its application to evaluating

- segmentation algorithms and measuring ecological statistics. In *Proceedings of the 8th International Conference Computer Vision*, volume 2, pages 416–423.
- [Materka and Strzelecki, 1998] Materka, A. and Strzelecki, M. (1998). Texture analysis methods - a review. Technical report, Technical University of Lodz, Institute of Electronics, Poland.
- [McGuinness et al., 2006] McGuinness, K., Keenan, G., Adamek, T., and O'Connor, N. E. (2006). A framework for integrating and evaluating automatic region-based segmentation algorithms. In *Proceedings of The First International Conference on Semantics And Digital Media Technology (SAMT)*.
- [McGuinness et al., 2007] McGuinness, K., Keenan, G., Adamek, T., and O'Connor, N. E. (2007). Image segmentation evaluation using an integrated region based segmentation framework. In *Proceedings of the 4th IET Visual Information Engineering Conference (VIE)*, Royal Statistical Society, London, UK.
- [McGuinness and O'Connor, 2008] McGuinness, K. and O'Connor, N. E. (2008). The K-Space segmentation tool set. In *Proceedings of The Third International Conference on Semantics And Digital Media Technology (SAMT)*.
- [McGuinness and O'Connor, 2009] McGuinness, K. and O'Connor, N. E. (2009). A comparative evaluation of interactive segmentation algorithms. *Pattern Recognition*.
- [Mehnert and Jackway, 1997] Mehnert, A. and Jackway, P. (1997). An improved seeded region growing algorithm. *Pattern Recognition Letters*, 18(10):1065–1071.
- [Meijster et al., 2000] Meijster, Roerdink, J., and Hesselink, W. (2000). A general algorithm for computing distance transforms in linear time. In *Proceedings of the 5th International Symposium on Mathematical Morphology and its Applications to Image and Signal Processing*, volume 18, pages 331–340. Springer.

- [Meila, 2003] Meila, M. (2003). Comparing clusterings by the variation of information. In *Proceedings of the 16th Annual Conference on Computational Learning Theory and 7th Workshop on Kernel Machines (COLT/Kernel)*, pages 173–187.
- [Mezaris et al., 2003] Mezaris, V., Kompatsiaris, I., and Strintzis, M. G. (2003). Still image objective segmentation evaluation using ground truth. In *Proceedings Fifth COST 276 Workshop on Information and Knowledge Management for Integrated Media Communication*, pages 9–14.
- [Moore and McCabe, 2005] Moore, D. S. and McCabe, G. P. (2005). *Introduction to the Practice of Statistics*. W. H. Freeman, 5th edition.
- [Moore and Fitz, 1993] Moore, P. and Fitz, C. (1993). Gestalt theory and instructional design. *Journal of technical writing and communication*, 23(2):137–57.
- [Morris et al., 1986] Morris, O., Lee, M., and Constantinides, A. (1986). Graph theory for image analysis: an approach based on the shortest spanning tree. In *IEE Proceedings F. Communications, Radar and Signal Processing*, pages 146–152.
- [Mullet and Sano, 1995] Mullet, K. and Sano, D. (1995). *Designing Visual Interfaces: Communication Oriented Techniques*. Prentice Hall.
- [Ning et al., 2009] Ning, J., Zhang, L., Zhang, D., and Wu, C. (2009). Interactive image segmentation by maximal similarity based region merging. *Pattern Recognition*.
- [Nock and Nielsen, 2004] Nock, R. and Nielsen, F. (2004). Statistical region merging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1452–1458.
- [Ohta and Robertson, 2005] Ohta, N. and Robertson, A. R. (2005). *Colorimetry: Fundamentals and Applications*. WileyBlackwell.

- [Ojala and Pietikainen, 1999] Ojala, T. and Pietikainen, M. (1999). Unsupervised texture segmentation using feature distributions. *Pattern Recognition*, 32(3):477–486.
- [Olabarriaga and Smeulders, 2001] Olabarriaga, S. D. and Smeulders, A. W. M. (2001). Interaction in the segmentation of medical images: a survey. *Medical image analysis*, 5(2):127–42.
- [Papadopoulos et al., 2006] Papadopoulos, G. T., Mylonas, P., Mezaris, V., Avrithis, Y., and Kompatsiaris, I. (2006). Knowledge-assisted image analysis based on context and spatial optimization. *International Journal on Semantic Web & Information Systems*, 2(3):17–36.
- [Petrrou and Sevilla, 2006] Petrrou, M. and Sevilla, P. G. (2006). *Image Processing: Dealing with Texture*. Wiley.
- [Pham et al., 2000] Pham, D., Xu, C., and Prince, J. (2000). A survey of current methods in medical image segmentation. *Annual Review of Biomedical Engineering*, 2:315–337.
- [Pietikäinen, 2000] Pietikäinen, M. K., editor (2000). *Texture Analysis in Machine Vision*. World Scientific.
- [Protiere and Sapiro, 2007] Protiere, A. and Sapiro, G. (2007). Interactive image segmentation via adaptive weighted distances. *IEEE Transactions on Image Processing*, 16(4):1046–1057.
- [Rand, 1971] Rand, W. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.
- [Randen and Husoy, 1999] Randen, T. and Husoy, J. (1999). Filtering for texture classification: a comparative study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):291–310.

- [Rother et al., 2004] Rother, C., Kolmogorov, V., and Blake, A. (2004). Grabcut. *ACM Transactions on Graphics*, 23(3):309–314.
- [Rubner et al., 2000] Rubner, Y., Tomasi, C., and Guibas, L. (2000). The earth mover’s distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121.
- [Saad, 1992] Saad, Y. (1992). *Numerical Methods for Large Eigenvalue Problems*. Algorithms and Architectures for Advanced Scientific Computing. Manchester University Press.
- [Sabera’b et al., 1997] Sabera’b, E., Tekalpa, A., and Bozdagi, G. (1997). Fusion of color and edge information for improved segmentation and edge linking. *Image and Vision Computing*, 15:769–780.
- [Salembier and Garrido, 2000] Salembier, P. and Garrido, L. (2000). Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *IEEE Transactions on Image Processing*, 9(4):561–576.
- [Schanda, 2007] Schanda, J. (2007). *Colorimetry: Understanding the CIE System*. WileyBlackwell.
- [Sezgin and Sankur, 2004] Sezgin, M. and Sankur, B. (2004). Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–168.
- [Sharon et al., 2000] Sharon, E., Brandt, A., and Basri, R. (2000). Fast multiscale image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 70–77.
- [Shevell, 2003a] Shevell, S. K. (2003a). *The science of color*, pages 202–203. Elsevier.
- [Shevell, 2003b] Shevell, S. K. (2003b). *The science of color*. Elsevier.

- [Shi and Malik, 2000] Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905.
- [Skarbek and Koschan, 1994] Skarbek, W. and Koschan, A. (1994). Color image segmentation: a survey. Technischer Bericht 94-32, Technical University of Berlin.
- [Smith and Brady, 1997] Smith, S. and Brady, J. (1997). SUSAN—a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78.
- [Sobel and Feldman, 1968] Sobel, I. and Feldman, G. (1968). A 3×3 isotropic gradient operator for image processing. Presentation for Stanford Artificial Project.
- [Sonka et al., 1998] Sonka, M., Hlavac, V., and Boyle, R. (1998). *Image Processing, Analysis, and Machine Vision*. PWS, 2nd edition.
- [Sorensen, 1996] Sorensen, D. C. (1996). Implicitly restarted Arnold/Lanczos methods for large scale Eigenvalue calculations. Technical Report TR-96-40, Institute for Computer Applications in Science and Engineering.
- [Spolsky, 2001] Spolsky, J. (2001). *User interface design for programmers*. Apress, Berkely, CA, USA.
- [Strehl et al., 2000] Strehl, A., Ghosh, J., and Mooney, R. (2000). Impact of similarity measures on web-page clustering. In *Proceedings of the 17th National Conference on Artificial Intelligence: Workshop of Artificial Intelligence for Web Search (AAAI)*, pages 58–64.

- [Suh and Bederson, 2007] Suh, B. and Bederson, B. (2007). Semi-automatic photo annotation strategies using event based clustering and clothing based person recognition. *Interacting with Computers*, 19(4):524–544.
- [Tomasi and Manduchi, 1998] Tomasi, C. and Manduchi, R. (1998). Bilateral filtering for gray and color images. In *Proceedings of the 6th International Conference on Computer Vision (ICCV)*, pages 839–846.
- [Usamentiaga et al., 2006] Usamentiaga, R., García, D. F., López, C., and González, D. (2006). A method for assessment of segmentation success considering uncertainty in the edge positions. *EURASIP Journal on Applied Signal Processing*, 2006.
- [van Dongen, 2000] van Dongen, S. (2000). Performance criteria for graph clustering and markov cluster experiments. Technical Report INS-R0012, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, The Netherlands.
- [Vincent and Soille, 1991] Vincent, L. and Soille, P. (1991). Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598.
- [Wang et al., 2004] Wang, J., Xu, Y., Shum, H., and Cohen, M. (2004). Video tooning. *ACM Transactions on Graphics*, 23(3):574–583.
- [Wang and Siskind, 2003] Wang, S. and Siskind, J. (2003). Image segmentation with ratio cut. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(6):675–690.
- [Wertheimer, 1997] Wertheimer, M. (1997). Laws of organization in perceptual forms. *A Sourcebook of Gestalt Psychology*, pages 71–88. Gestalt Journal Press.
- [Wilkins et al., 2007] Wilkins, P., Adamek, T., Byrne, D., Jones, G., Lee, H., Keenan, G., Guinness, K. M., O'Connor, N., Smeaton, A. F., Amin, A., Obrenovic,

- Z., Benmokhtar, R., Galmar, E., Huet, B., Essid, S., Landais, R., Vallet, F., Papadopoulos, G. T., Vrochidis, S., Mezaris, V., Kompatsiaris, I., Spyrou, E., Avrithis, Y., Morzinger, R., Schallauer, P., Bailer, W., Piatrik, T., Chandramouli, K., Izquierdo, E., Haller, M., Goldmann, L., Samour, A., Cobet, A., Sikora, T., and Praks., P. (2007). K-Space at TRECVID 2007. In *TRECVID 2007 - Text REtrieval Conference TRECVID Workshop*.
- [Wirth et al., 2006] Wirth, M., Frascini, M., Masek, M., and Bruynooghe, M., editors (2006). *Special Issue on Performance Evaluation in Image Processing*. EURASIP Journal on Applied Signal Processing. Hindawi Publishing Corporation.
- [Wu and Leahy, 1993] Wu, Z. and Leahy, R. (1993). An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1001–1113.
- [Wyszecki and Stiles, 2000] Wyszecki, G. and Stiles, W. S. (2000). *Color science: concepts and methods, quantitative data, and formulae*. Wiley, 2nd edition.
- [Yang et al., 1995] Yang, L., Albregtsen, F., Lønnestad, T., and Grøttum, P. (1995). A supervised approach to the evaluation of image segmentation methods. In *Proceedings of the 6th International Conference on Computer Analysis of Images and Patterns*, pages 759–765.
- [Yang et al., 1997] Yang, M., Lee, J., Lien, C., and Huang, C. (1997). Hough transform modified by line connectivity and line thickness. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(8):905–910.
- [Yang et al., 2002] Yang, Y. H., Buckley, M. J., Dudoit, S., and Speed, T. P. (2002). Comparison of methods for image analysis on cDNA microarray data. *Journal of Computational & Graphical Statistics*, 11(1):108–136.

- [Zadeh, 1965] Zadeh, L. (1965). Fuzzy sets and systems. *Information and Control*, 8(3):338–353.
- [Zhang et al., 2006] Zhang, H., Cholleti, S., Goldman, S., and Fritts, J. (2006). Meta-evaluation of image segmentation using machine learning. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1138–1145.
- [Zhang et al., 2008] Zhang, H., Fritts, J., and Goldman, S. (2008). Image segmentation evaluation: a survey of unsupervised methods. *Computer Vision and Image Understanding*, 110(2):260–280.
- [Zhang et al., 2004] Zhang, H., Fritts, J. E., and Goldman, S. A. (2004). An entropy-based objective evaluation method for image segmentation. In *Proceedings of IS&T/SPIE’s 16th Annual Symposium on Electronic Imaging Conference on Storage and Retrieval Methods and Applications for Multimedia*.
- [Zhang et al., 2005] Zhang, H., Fritts, J. E., and Goldman, S. A. (2005). A co-evaluation framework for improving segmentation evaluation. In *Proceedings of SPIE Defense and Security, Signal Processing, Sensor Fusion, and Target Recognition, XIV*.
- [Zhang, 1996] Zhang, Y. J. (1996). A survey on evaluation methods for image segmentation. *Pattern Recognition*, 29(8):1335–1346.

List of Figures

2.1	Application-centric classification of segmentation algorithms	13
2.2	Algorithm-centric classification of segmentation algorithms	18
2.3	The Gestalt law of closure	22
2.4	The Gestalt law of Prägnanz	23
2.5	Illustration of information loss when converting to grayscale	25
2.6	The saturation effect for large color differences.	26
2.7	The gray square optical illusion.	27
2.8	A 4-connected region adjacency graph.	31
2.9	Merging nodes in a region adjacency graph algorithm	33
2.10	Examples of the statistical region merging algorithm	40
2.11	Example output of the normalized cuts algorithm	47
2.12	Two dimensional Gaussian and Epanechnikov kernels.	48
2.13	Examples of the mean-shift algorithm	52
2.14	Examples output of the seeded region growing algorithm	61
2.15	An illustration of the graph structure used by the IGC algorithm. .	63
2.16	Extracting an object using the interactive graph cuts algorithm . . .	65
2.17	Extracting an object using the binary partition tree algorithm	68
2.18	Extracting an object with simple interactive object extraction. . . .	71
3.1	Taxonomy of segmentation evaluation algorithms	78

3.2	Measuring baseline error-rates for accurate and inaccurate segmentation.	92
3.3	Spatial configuration of regions and layout entropy.	103
4.1	The Berkeley segmentation dataset.	114
4.2	Sample images and segmentations from the Berkeley segmentation dataset	115
4.3	Screenshot of the K-Space video region segmentation tool.	116
4.4	Software architecture model for the K-Space segmentation tool . . .	117
4.5	Histogram plots of accurate and inaccurate segmentation for the six evaluation measures.	125
4.6	Means and confidence intervals for each of the measures.	127
4.7	Scattergram showing the correlation between the Fowlkes-Mallows index and the Jaccard index.	129
4.8	Histogram plot of accurate and inaccurate segmentation using the LHE measure.	133
4.9	Sample images and machine segmentations	136
4.10	Box and jitter plots of segmentation error for each algorithm. . . .	138
4.11	Mean error and confidence intervals for each of the algorithms. . .	140
4.12	Estimated error distribution density for each algorithm.	142
4.13	Prediction accuracy for each of the four segmentation algorithms. .	143
5.1	The internal border pixels of two similar objects.	154
5.2	The fuzzy membership function for different tolerance parameters.	156
5.3	Screenshots of the interactive segmentation tool	159
5.4	Screenshot of the interactive segmentation tool in experiment mode	161
5.5	Sample task image and the expected object.	164
5.6	Scatterplot of accuracy measurements against time.	169
5.7	Mean boundary accuracy time series per image.	170

5.8	Mean accuracy time series.	171
5.9	Proportion of segmentations attaining at least the given boundary accuracy, plotted over time.	172
5.10	Proportion of segmentations attaining at least the given object accuracy, plotted over time.	173
5.11	Histogram plots of perceived accuracy and performance.	175
5.12	Results of user feedback.	176
5.13	Preferred algorithm as voted by users	177
6.1	Flow chart of user activity when performing a segmentation task .	183
6.2	An illustration of the first few the steps of automation strategy 1. .	190
6.3	Extracting a non-convex object with straight line segments.	194
6.4	Paths found between the maxima points of Figure 6.3 using au- tomation strategies 3 and 4.	195
6.5	An illustration of the first few the steps of automation strategy 4. .	196
6.6	The configuration window for our automator tool.	198
6.7	Sample time series data for individual images	200
6.8	Mean accuracy over time for each of the evaluation strategies. . . .	203
6.9	Mean final accuracy after 100 steps for each of the evaluation strate- gies	204
6.10	Mean integrated accuracy for 100 steps for each of the evaluation strategies	204
6.11	Rank correlation between the user experiments and each of the automation strategies.	207
6.12	Accuracy features for the user experiments plotted against accuracy features from the automated experiments.	208
6.13	Top ten easiest images to segment with the SIOX algorithm	209
6.14	Top ten most difficult images to segment with the SIOX algorithm .	210

List of Tables

2.1	Classification of the region adjacency graph algorithm	36
2.2	Classification of the statistical region merging algorithm	41
2.3	Classification of the normalized cuts algorithm	46
2.4	Classification of the mean-shift algorithm	54
2.5	Classification of the seeded region growing algorithm	60
2.6	Classification of the interactive graph cuts algorithm	65
2.7	Classification of the binary partition tree algorithm	67
2.8	Classification of the simple interactive object extraction algorithm .	70
3.1	Indicative values for accurate and inaccurate segmentation.	87
4.1	Evaluated algorithms and their abbreviated names	111
4.2	Evaluation measures and their abbreviated names	113
4.3	Statistical properties for pairs of segmentations of the same scene. .	124
4.4	Statistical properties for pairs of segmentations of different scenes. .	124
4.5	Confidence intervals for the mean error of accurate and inaccurate segmentation.	126
4.6	Inter-measure correlation.	128
4.7	Accurate/Inaccurate thresholds discovered using grid-search. . . .	131
4.8	Prediction error using a linear combination of the LCE and the HDI measures.	132
4.9	Properties of selected measures	134

4.10	Results of evaluating the algorithms using the three evaluation measures.	139
5.1	The evaluated algorithms and their abbreviated names	150
5.2	Evaluation measures and their symbols	158
5.3	Experiment variants with ground truth set and algorithm assignments.	163
5.4	Overall average boundary accuracy and object accuracy.	167
5.5	Average time required for users to achieve their best accuracies and average total time used to complete a task (seconds).	168
5.6	Correlation of measured and perceived accuracy	179
6.1	Indicative runtimes of the automation strategies	199