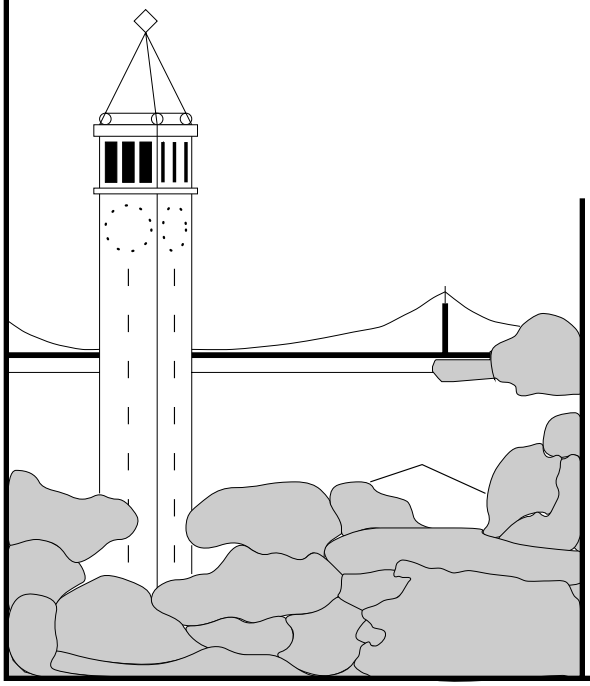
An Empirical Approach to Grouping and Segmentation

David R. Martin



Report No. UCB/CSD-3-1268

August 2003

Computer Science Division (EECS)
University of California
Berkeley, California 94720

An Empirical Approach to Grouping and Segmentation

by

David Royal Martin

B.S.E. (Princeton University) 1996
M.S. (University of California, Berkeley) 1998

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Jitendra Malik, Co-Chair
Professor David Patterson, Co-Chair
Professor Stephen Palmer

Fall 2002

The dissertation of David Royal Martin is approved:

Co-Chair Date

Co-Chair Date

Date

University of California, Berkeley

Fall 2002

An Empirical Approach to Grouping and Segmentation

by

David Royal Martin

Abstract

An Empirical Approach to Grouping and Segmentation

by

David Royal Martin

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Jitendra Malik, Co-Chair
Professor David Patterson, Co-Chair

This thesis presents a novel dataset of 12,000 segmentations of 1,000 natural images by 30 human subjects. The subjects marked the locations of objects in the images, providing ground truth data for learning grouping cues and benchmarking grouping algorithms. We feel that the data-driven approach is critical for two reasons: (1) the data reflects "ecological statistics" that the human visual system has evolved to exploit, and (2) innovations in computational vision should be evaluated quantitatively.

We develop a battery of segmentation comparison measures that we use both to validate the consistency of the human data and to provide approaches for evaluating grouping algorithms. In conjunction with the segmentation dataset, the various measures provide "micro-benchmarks" for boundary detection algorithms and pixel affinity functions, as well a benchmark for complete segmentation algorithms. Using these performance measures, we can systematically improve grouping algorithms with the human ground truth as our goal.

Starting at the lowest level, we present local boundary models based on brightness, color, and texture cues, where the cues are individually optimized with respect to the dataset and then combined in a statistically optimal manner with classifiers. The resulting detector is shown to significantly outperform prior state-of-the-art algorithms. Next, we learn from data how to combine the boundary model with patch-based features in a pixel affinity model to settle long-standing debates in computer vision with empirical results: (1) brightness boundaries are more informative than patches, and vice versa for color; (2) texture boundaries and patches are the two most powerful cues; (3) proximity is not a useful cue for grouping, it is simply a result of the process; and (4) both boundary-based and region-based approaches provide significant independent information for grouping.

Co-Chair Date

Co-Chair Date

Contents

List of Figures

Chapter 1

Introduction

The generic goal of computer vision is to extract semantic information from digital images. The difficulty, of course, lies in the word *semantic* and the human-centric nature of this information. For lack of any other reasonable approach, we study and model the six billion working vision systems presently in circulation. Lacking the tools for direct observation, we rely on the indirect observations of psychophysicists, the introspection of the Gestaltists, and, all too often, our own intuition.

The legacy of the Gestalt school is a variety of *ceteris paribus*[1] rules for perceptual grouping. For example, if all other factors remain constant, common motion is a grouping cue. The psychology community has diligently tested these and other rules for perceptual organization with good effect. Better understanding of human vision does lead to improved machine vision systems. However, the controlled experiments required for scientific deduction limit the complexity of stimuli that one can use with the typical psychophysical approach. Even if the Gestalt rules for grouping could be rigorously prioritized, and their applicability to various stimuli extensively catalogued, it is not clear that an effective recipe for grouping would emerge.

This conjecture arises from two concerns. First, there is the staggering complexity gap between optical images of the natural world and the stimuli used in the majority of psychophysical experiments. This gap will presumably close over time, though more slowly than at the exponential rate of progress to which computer scientists have grown accustomed. There is an opportunity to harness the growing power of computers to explore the space of perceptual organization rules. For example, if a machine were able to show that to operate in the natural world it must give priority to convexity over symmetry for figure-ground separation, then does this not settle the question for computer vision? The second concern is whether the statistics of natural images are relevant to human perception. If these *ecological statistics* are absent from psychophysical stimuli, as they often are, then

[1] Literally, "other things being equal".

experimental results may be compromised.

Vision is an intractable inverse problem, where the proximal stimulus of photons on the retina cannot be mathematically inverted to reveal the structure of the distal stimulus that caused the photons to enter the eye. Even so, the human visual system is shockingly robust. The problem appears so difficult that we must assume that the human visual system takes advantage of any useful information available to it. Helmholtz conjectured that sensory input is interpreted as the most likely configuration of objects in the environment. In addition to high-level knowledge about the world, any relevant statistical regularities in the stimulus will clearly affect the likelihood. It has proven difficult to incorporate high-level information into machine vision systems. We have, however, grown adept at incorporating probabilistic information.

This leads to the topic of this dissertation. How can we exploit the statistics of natural images to improve the quality of perceptual organization by machines? For example, a brightness discontinuity in an image can be caused by the boundary between objects. However, brightness discontinuities have many other causes — surface markings, shadows, and texture — all of which are frequent in the natural world. The statistical value of brightness discontinuities for finding object boundaries in natural scenes can be judged only by measuring the fraction of brightness edges that suit that purpose.

The human visual system is an equal party to the environment in the ecological statistics we seek. We must, therefore, study the world as it is seen by human beings. To this end, we have compiled a large dataset of natural images which have been segmented by human subjects. Each of 1000 images has been segmented by approximately 10 different subjects. Given an image, each subject was instructed to divide it into pieces that represent objects or pieces of objects. As a first step in perceptual organization, we aim to recover object boundaries. It is presently inconceivable that a machine vision system could replicate the human segmentations, because the subjects used any number of high-level cues to perform their grouping. Our immediate objective is to determine how closely we can approximate human-level performance on this task by exploiting statistical regularities in the dataset without incorporating high-level object-specific knowledge.

More low-level information is available than one might suspect. In computer vision's sister domain of computational linguistics, one can achieve 90% accurate part-of-speech tagging simply by using the most likely tag for each word. The best probabilistic parsers currently approach 90% accuracy on the problem of labeling grammatical structure without using the semantics of the words. [2] These successes, as well as the more colloquial success of Internet search engines, have lent weight to a new 90-10 rule: statistical approaches can provide 90% solutions to seemingly complex semantic problems. It is time that we had statistics of these sorts for computer vision tasks.

[2] Performance numbers are taken from [41].

This dissertation is distinguished by its two most central themes. First, we use a large dataset of complex, natural images. We have made no effort to simplify the image set, since we believe that the unbiased statistics of the images must be preserved. Second, we take a scientific approach to building a machine vision system. Since human-level vision is our goal, it seems most sensible to study humans. We spent one year collecting data from human subjects so that we could subsequently analyze patterns in the data, build models, and measure the accuracy of the models in relation to the performance of the human subjects. Although this thesis focuses on relatively low-level issues such as boundary detection and pixel affinity, it is meant as a first step in the pursuit of an approach to perceptual organization that takes full advantage of ecological statistics from the start.

The organization of this dissertation is as follows. Chapter 2 presents the segmentation dataset and the details of its construction. Chapter 3 presents several methods for measuring the consistency of a segmentation with respect to the ground truth dataset. Chapter 4 presents a local boundary model that was learned from the human data. Chapter 5 builds on the boundary model by learning a pixel affinity function from the dataset. I conclude in Chapter 6.

Chapter 2

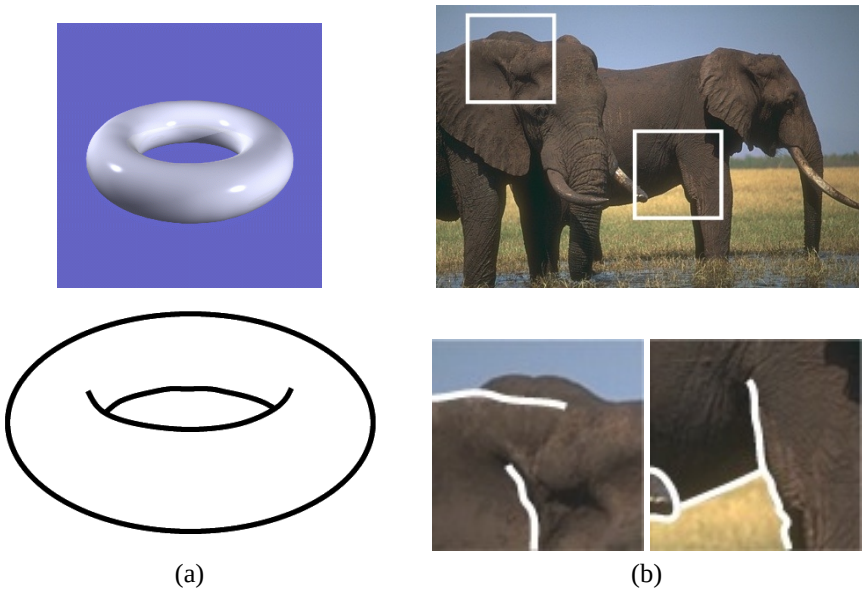
A Dataset of Human Segmented Natural Images

If we wish to create computer systems that decompose images in a manner similar to human beings, then we should study the manner in which humans decompose images. In order to do this, we must decide what it means to "decompose" an image. Images of the natural world are generally composed of physically disjoint objects whose juxtaposition in a scene leads to an image consisting of spatially coherent groups of pixels. Each group of pixels originates from a single object, and may or may not be visually coherent also. Let us call the pixel groups *segments*, and the process of dividing an image into segments *segmentation*.

This high-level definition of segmentation is quite different from the definition typically used in the computer vision literature. The term *segmentation* is most often used to refer to the low-level process of grouping pixels into visually coherent groups. The argument for such a process is generally one of efficiency. Pixels are an inconvenient image representation due to their large number. Segments defined by uniform photometric properties are essentially super-pixels that reduce the number of primitive objects an algorithm must consider. In addition, one can define new features on a pixel group, such as shape and texture, for which pixels are simply too small.

However, the low-level definition of segmentation is imprecise because it depends on the current state-of-the-art in defining uniformity. Uniformity based on photometric properties is limiting and arbitrary. The only stable definition of segmentation is also the statement of the ultimate goal: finding regions of semantic coherence. To ensure the longevity and more general applicability of a segmentation dataset, we therefore adopt the high-level notion of segmentation based on objects as seen by generic humans rather than on features as seen by vision scientists.

As a practical issue in constructing a dataset, we need a data representation. However, we wish



(a) (b)

Figure 2.1: Open Contours. The open-contour objection to region-based segmentation. (a) An oblique view of a torus produces open contours that result from self-occlusion. These open contours reveal information about the geometry of the object that one cannot obtain from the contours of closed regions. (b) An example of this phenomenon in our dataset of natural images. The non-convex elephants produce similar open contours where the ear both occludes and attaches to the head, and where the leg occludes and attaches to the body. These contours provide valuable clues about the geometry of the elephant, but the contours are not needed to separate the two elephants from each other or from the background. The open-contour phenomenon is restricted to occur within an object; our focus is on the boundaries between different objects.

(a) (b)

Figure 2.2: Hallucinated Boundaries. (a) An image with deep shadows that obscure the boundary of the shoulder and hair. (b) The boundaries that were "hallucinated" by four of our subjects. Note that the hallucinated boundaries agree more than one might expect, suggesting that the subjects possess strong prior knowledge about the shape of people.

to remain as agnostic as possible on the widely debated issue of boundary-based versus region-based structures. Note that a decomposition of an image into regions necessarily provides the boundaries one would need to achieve the same segmentation. Thus, we can always treat a segmentation as a boundary map. That every segmentation is a boundary map suggests that a boundary-based representation is more general, and indeed this is the case. In a region-based representation, we cannot represent structures in the scene that are not closed. Such structures may arise from at least two sources: objects with internal boundaries arising from a folded topology, such as a donut or elephant (see Figure 2.1), and objects that appear to blend into each other because of shading (see Figure 2.2).

Fortunately, neither situation concerns us since both result from a low-level perspective. Folded objects may have internal open boundaries that are useful features for recognition, but it is the entire object we wish to partition from the image and recognize, not the view-dependent topological features. As for severe shading gradients that cause the boundary of an object to disappear, it is still the case that the object has a boundary in the scene. That the boundary is not visible does not mean that it should not be represented. With this dataset, we aim for "gold-standard" image segmentations that may contain high-level information. Our goal is, after all, to integrate high-level information into vision algorithms!

Thus, we settle on a region-based representation because the region-based representation is equivalent to a boundary-based representation when the segments correspond to whole objects in

the real world. To be precise, we define our region-based representation as follows: *a segmentation is a partition of the pixels of an image into disjoint sets.* The sets need not be contiguous in the image plane.

One significant image structure that cannot be represented with the abstraction that each pixel belongs to a single object is translucence. The obvious examples of translucent objects are ones involving materials such as glass or plastic. However, our dataset has few of these synthetic translucent objects. Water is the predominant translucent material in the natural world. However, water is often rendered largely opaque by surface ripples. In the few images we have of still water, the water is more often transparent or perfectly reflective than translucent. We attempted to balance these issues against the desire to keep the data representation as simple as possible.

Additional concerns arise from shadows and reflections. It is not clear how we should treat these effects in a segmentation, both because they are not objects proper and because they introduce a translucent layer. We expected our subjects to ignore shadows and reflections, and so we did not wish to unduly complicate the segmentation representation by adding layers. In addition, we did not wish to draw the subjects' attention to the concerns of computer vision researchers by explicit instructions about such complexities, and therefore bias the experiment.

We settled on the simple one-segment-per-pixel data representation and laid out a set of high-level goals for the dataset:

1. *Natural Images.* We are interested in natural images because of their ecological statistics [2, 5, 26, 27, 31, 33, 61, 62, 71]. A large enough dataset should permit us to extract the statistics of natural images that the human visual system likely exploits. To this end, I selected 1000 representative images from the Corel image library, excluding synthetic and abstract images, but not excluding man-made structures (see Section 2.1.3 for the image selection details). The Corel image dataset is commonly used in computer vision because it is readily available and contains a wide range of subject matter.

2. *Integrity.* Many segmentation datasets used in computer vision are small, contain simple images, and were created by computer vision researchers. These deficits are understandable considering that more rigorous methods using a large set of natural images and naïve subjects are time consuming. We strove to break this tradition, however, by using 1000 Corel images that contain a wide selection of subject matter. In the interest of ensuring that the data is biased as little as possible by our immediate research agenda, we collected data from subjects not trained in computer vision. In addition, we collected multiple segmentations of each image from different subjects and validated the data by ensuring that the data is consistent across subjects.

3. *Longevity.* By collecting high-level image segmentations, we hope to ensure that the data remains useful for many years. Matching these "gold-standard" segmentations computationally represents a challenging goal that will not be achieved in the near future.

4. *Wide Applicability.* Because of the effort involved in creating a large dataset of this type, it should be useful for many tasks. In this thesis, I use the dataset to learn and evaluate both local boundary models and pixel affinity models. Presumably, others will find additional uses for the data. It was important to keep the data representation simple in order to facilitate its use in unexpected applications.

5. *Public Availability.* As a service to the community, we have made the dataset and the tools used to collect it from human subjects available for research and educational use. This is in contrast with the only other large segmentation dataset we know of (the Sowerby dataset [31, 25]) which has not been made public. The dataset and tools can be downloaded from the Internet [11].

This outline of this chapter is as follows. Section 2.1 describes in detail the procedure we followed in constructing the dataset. Section 2.2 demonstrates the integrity of the data by showing that different human subjects are consistent with one another. Section 2.3 discusses the lessons we learned in running the experiment. Section 2.4 presents future work. I conclude in Section 2.5.

2.1 Dataset Construction

2.1.1 Experimental Setup

Subjects were solicited through the U.C. Berkeley work-study program, which draws undergraduate students from all disciplines. They were paid at a rate set by the work-study program, which was approximately $12 per hour. Figure 2.3 shows some summary information about our 30 subjects. The gender split was nearly equal, and few subjects were computer science students. To our knowledge, none had any training in computer vision.

After an interview to assess commitment and to ensure that the subject had normal vision with any necessary correction, I spent one to two hours with each subject individually to help them learn to use the segmentation tool. Most subjects had little trouble mastering the procedure in one session, but a few required additional one-on-one sessions. Nearly all subjects were able to produce accurate segmentations on their first attempt. Any subsequent training was geared toward increasing the speed at which subjects could segment images while retaining a high degree of accuracy in the boundary localization.

	ID	Sex	Age	Art	Segs	Start Date	End Date	Min/ Seg	Total Hours
1	1102	M	19	N	82	Feb 28	Apr 13	9.2	12.6
2	1103	M	19	N	509	Feb 28	Aug 13	9.9	84.4
3	1104	F	20	Y	54	Mar 06	Apr 30	19.7	17.7
4	1105	M	22	N	978	Mar 05	Oct 05	7.6	123.4
5	1106	F	19	N	84	Apr 10	May 08	15.4	21.6
6	1107	F	19	N	920	Feb 27	May 12	4.8	73.8
7	1108	M	22	N	1000	Mar 07	Oct 05	4.0	66.7
8	1109	F	20	Y	1020	Mar 09	Jun 06	4.4	74.5
9	1110	M	21	N	46	Mar 09	May 19	12.5	9.6
10	1111	F	23	N	13	Mar 13	Apr 12	8.5	1.8
11	1112	F	20	N	344	Mar 13	Oct 05	8.7	50.0
12	1113	F	20	Y	308	Mar 14	Aug 14	8.7	44.6
13	1114	M	20	N	313	Mar 20	Sep 06	8.1	42.4
14	1115	M	20	Y	953	Apr 02	Aug 28	6.7	106.7
15	1116	M	20	N	358	Jul 15	Sep 29	7.5	44.9
16	1117	M	19	Y	207	Jul 05	Aug 22	8.9	30.8
17	1118	F	23	Y	14	Jun 12	Jul 25	19.3	4.5
18	1119	F	22	N	290	Jul 09	Sep 26	8.7	42.0
19	1121	M	20	N	432	Jun 13	Aug 31	6.2	44.6
20	1122	M	20	N	249	Jun 13	Aug 21	6.2	25.8
21	1123	F	19	N	1020	Jul 10	Oct 05	6.9	116.8
22	1124	M	21	Y	771	Jul 18	Oct 05	12.5	160.6
23	1125	M	22	N	2	Jul 18	Jul 18	21.1	0.7
24	1126	M	20	N	166	Jul 18	Sep 26	11.0	30.4
25	1127	F	19	Y	248	Jul 18	Oct 02	10.0	41.3
26	1128	F	21	N	50	Jul 19	Aug 08	9.1	7.6
27	1129	M	20	N	104	Jul 19	Aug 26	22.2	38.5
28	1130	M	21	Y	678	Jul 19	Sep 20	5.6	63.4
29	1131	M	20	N	1	Jul 19	Jul 19	13.4	0.2
30	1132	F	20	N	381	Jul 24	Sep 20	12.0	76.2
Totals					11,595	28 Feb	05 Oct	7.5	1458

Figure 2.3: Table of Human Subjects. The gender split is 17-13 male-female. All subjects had normal 20/20 vision with any necessary correction. Nine subjects stated on the questionnaire that they had artistic training. The last two columns show the average minutes spent on a segmentation, and the total number of hours the subject worked. Data collection lasted about 7 months in the year 2001, resulting in 11,595 segmentations of 1020 Corel images.

We provided subjects with the following instructions, which were intentionally brief and non-technical:

> *You will be presented a photographic image. Divide the image into some number of segments, where the segments represent "things" or "parts of things" in the scene. The number of segments is up to you, as it depends on the image. Something between 2 and 30 is likely to be appropriate. It is important that all of the segments have approximately equal importance.*

These directions were posted in the lab in clear view of the machines that the subjects used. The last clause was a reluctant addition to the directions after observing that some subjects occasionally focused on a single object (often a human face) to the exclusion of other objects. If a subject was confused by the directions, they were additionally told to pay equal attention to all the objects in the scene and not to fixate on a single object. Whenever one subject was confused about the task, all subjects were sent the additional instructions over email.

We did not use any unusual equipment in the experiment. The subjects work stations were standard PCs running Microsoft Windows and Sun's Java implementation. The monitors were 21" Sony Trinitrons, and the pointing device a standard mouse.

Subjects set their own hours but were limited by work-study to no more than 19 hours per week. I required a minimum of 5 hours per week. Data collection ran for approximately 7 months. Very few subjects worked for the entire duration, either because of attrition at academic term boundaries or because they segmented all 1000 images. To be sure we met our goal of 10,000 segmentations, I recruited new subjects continuously throughout the duration of the experiment.

The experimental protocol was approved by the U.C. Berkeley Committee for the Protection of Human Subjects. The data produced by the subjects was anonymized in accordance with the Committee's regulations.

2.1.2 The Segmentation Tool

In order to collect segmentations efficiently, we required a simple tool that would allow a user to divide the pixels of an image into disjoint groups, or segments. We considered the following options:

Adobe Photoshop Photoshop has several selection tools, including the *magic lasso*, that would allow a user to mark boundaries. However, an explicit decomposition of an image into many segments is not a paradigm supported by Photoshop. The subjects could use the selection tools either to mark boundaries with a unique color, or to cut them into separate images.

Pen and Paper We considered printing the images on paper, and then having subjects draw the boundaries with a pen either on the paper itself or on an overlayed transparency. We could then scan the boundary drawings.

Special Hardware Wacom manufactures pen-based tablets, as well as pen-sensitive flat-panel LCDs that lay flat on a desk. One could present an image on such a device, and use the pen to mark boundaries.

Custom Solution We considered creating a custom Java application tailored specifically to the task at hand.

The non-custom solutions all share the problem of how to automatically convert boundary maps to segmentations. The task would be confounded by having to deal with open contours, as there is no way to enforce the closure of contours in Photoshop or on paper. In addition, any algorithmic solution to this problem would introduce some sort of bias into the data.

A few weeks spent building a custom Java application seemed well worth the time investment. If the user were able to construct segmentations directly, then no further processing would be required. Two issues remained. First, there was the question of whether to use standard or special-purpose pointing devices. The Wacom LCD tablet seems ideal for the task, but at the time, the units were expensive, so they would become a bottleneck in the data collection process. Qualitative tests indicated that non-LCD tablets did not provide any convenience over using a mouse. The pen-based pointing paradigm seems natural for the task, but hand tremors were a problem. When holding a pen, any hand tremor is translated directly into pointer motion. In contrast, one can push a mouse around the surface of a desk with far more accuracy and steadiness. Image contours are often long and smooth, and it seemed that only a trained artist could draw such contours with any accuracy using a pen.

The second issue was whether or not to use algorithmic assistance in marking boundaries, as in Photoshop's magic lasso tool. There are many variational snake-type methods [45] that would be appropriate for the task of localizing boundaries. We were apprehensive, however, about using this type of aid, for fear of introducing bias into the dataset. For example, variational boundary localization methods have roughness penalty terms that prohibit sharp turns in the boundary. Far worse is the problem of texture. Snakes climb some field gradient in the image plane. The gradient is often defined in terms of image brightness, and so has problems with texture. A texture-savvy snake would require us to define a field that reflects both brightness as well as texture gradients. The combination of brightness and texture cues is an open research issue and indeed is one of our primary research objectives!

Thus, we decided to build a custom segmentation tool. This tool would enable the user to directly divide an image into segments, obviating the need for any further data processing. The segmentation task would be manual, without any algorithmic aids. One should be able to make the task efficient given complete control over the application.

In the following sections, I will present the segmentation tool and the editing paradigm it supports. We arrived at our solution after several versions. A few weeks of use by our research group ironed out some of the major problems. A trial run of the entire dataset construction procedure on a graduate computer vision class provided much valuable feedback on the interface.

Editing Paradigm

When presented with a new image, the subject begins with a single segment containing all the pixels. The segmentation then evolves by a sequence of splitting, merging, and editing steps, described below. All editing operations are governed by a single editing paradigm – that of moving pixels from one segment to another. Figure 2.4 shows the operation. The figure shows the tool's editing window containing two segments, one in each pane. Any pixels selected in one pane will move to the other. All editing operations are performed in this manner.

Creating a new segment. New segments are created by splitting some existing segment into two pieces. See Figure 2.4(a,d). The subject first moves a single segment into the editing window. The left pane contains the segment to be split, and the right pane is empty. By moving pixels from the non-empty pane into the empty pane, the subject creates a new segment.

Deleting a segment. A segment can be deleted by merging it into some other segment. See Figure 2.4(e). First, the subject moves the two segments into the editing window. By moving all of the pixels of the right segment into the left pane, the subject has effectively deleted the right segment. Alternately, the left segment could be deleted by moving all of its pixels into the right pane.

Editing a boundary. The location of a boundary can be easily modified by moving pixels from one side to the other. See Figure 2.4(b,c).

The most common operation is that of creating new segments. The other operations are provided "for free" by the interface, since the pixel exchange editing paradigm supports them. We intended the paradigm to be flexible so as not to force subjects to segment an image in any particular manner. For example, both top-down and bottom-up segmentation styles are naturally supported. A subject favoring top-down division can recursively split segments from larger to smaller pieces. A subject favoring a bottom-up approach could repeatedly carve the leaf segments from the original image

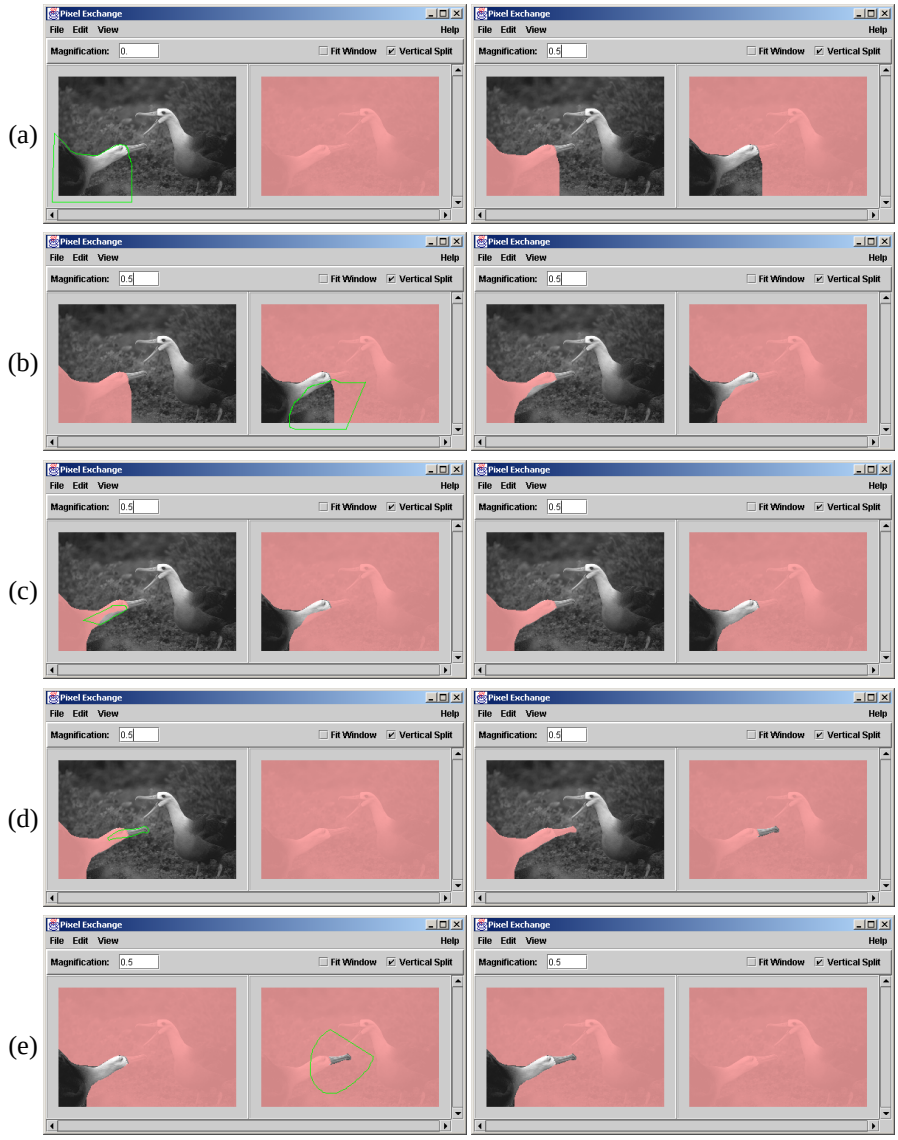


Figure 2.4: Segmentation Tool Editing Paradigm. The editing window of the segmentation tool (shown here 10 times) supports a single flexible mechanism: moving pixels from one side to the other. Pixels selected in one panel (by encircling them with a closed contour) simply move to the other panel. (a) We begin with all pixels in a single segment. The user moves pixels to the empty right panel to create a new segment. (b) The user moves the extraneous background pixels back, but makes a mistake on the neck contour. (c) The neck contour is cleaned up. (d) The user has selected the background segment and carved out a beak segment. There are now 3 segments total. (e) The user merges the beak and left bird segments, leaving 2 segments total. This is not a realistic editing session, but is meant to be illustrative. The image is drawn as a faded backdrop in each panel to orient the user. Note that the selection contour can contain freehand sections (by dragging the mouse) and straight-line sections (by clicking). The contour is closed with a right-click, at which time the pixels are moved. Straight-line sections are particularly useful when segmenting man-made structures. Low-curvature contours are also more efficiently traced in linear pieces rather than freehand.

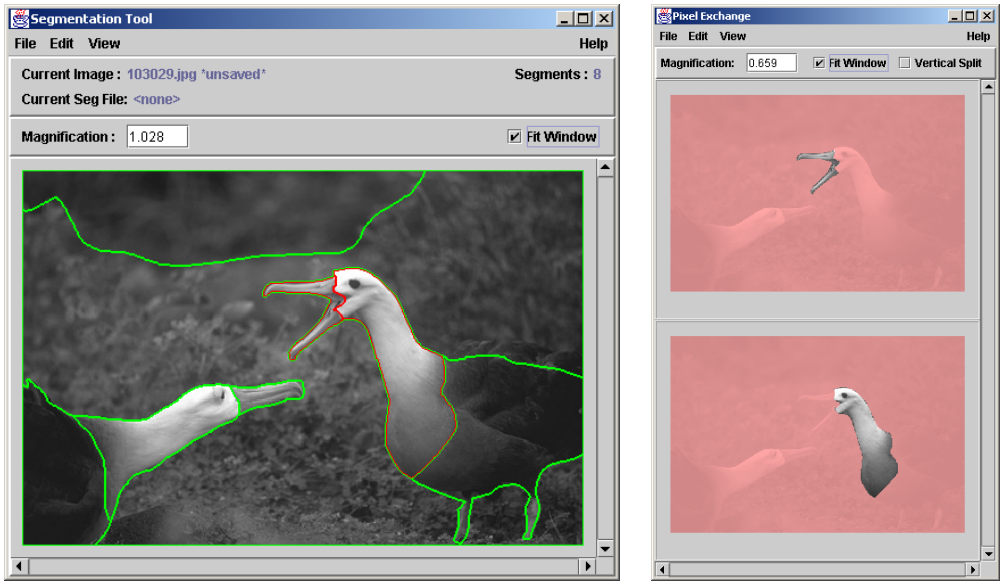


Figure 2.5: Segmentation Tool Windows. The main window (left) shows the current state of the segmentation. There are 8 segments, outlined in green. The two segments outlined in red are selected, and so are shown in the editing window (right).

segment. In addition, the tool does not constrain the user to define a new segment in one step. Figure 2.4(a-c) shows how a new segment can be carved out of an existing segment in multiple steps. Again, there is no new operation involved; the editing paradigm provides it for free.

Note that there is no constraint that segments be contiguous. Subjects are free to create discontinuous segments, which is often natural when one object is split up into several segments by occlusions.

Windows

The tool consists of two windows: the main window and the editing window (see Figure 2.5). The main window shows the current state of the segmentation as neon green contours overlayed on the image. Both the display of the image and the contours can be toggled.

The user selects segments by clicking on them in the main window. Any selected segment has a red contour to distinguish it from the unselected segments with green contours. From 0 to 2 segments may be selected at any moment. Selected segments appear in the editing window. A plain click in the main window will place the newly selected segment in the editing window with the other panel empty. The segment can then be split, if that is the user's intention. If not, another segment can be placed alongside the first in the editing window by selecting a second segment in the main window

with a shift-click. The two segments may then be merged or otherwise edited, as described above.

The editing window is not modal. Any changes made to the segments in the editing window are immediately reflected in the main window. Modal interfaces tended to confuse users.

Undo Support

Though the editing interface is rich enough to easily compensate for any conceivable editing mistake, our testors universally requested undo support. The power and simplicity of a good undo model cannot be overstated. We therefore implemented two levels of undo support: fine and coarse. All undoable operations can be redone.

The fine level undo operates within the editing window during the process of selecting pixels. As described in Figure 2.4, pixels are selected by encircling them with a contour drawn with the mouse. The contour is a polygon defined by a sequence of points laid down by the user. By dragging the mouse, the user lays down points at a rapid rate to provide the illusion of a smooth contour. Single mouse clicks lay down single points, so that piecewise linear contours are easy to construct. The fine level of undo permits the user to remove the most recent point of the contour. The contour can be undone back to the first point of the contour.

Completing a contour in the editing window, which causes pixels to move from one pane to the other, defines the undoable unit for the coarse level of undo. At this point, the fine level undo/redo operations are no longer available. Undo will undo the entire pixel-move operation. The segmentation is evolved by a sequence of such operations, and these can be undone and redone without limit.

Pan and Zoom

Both windows support flexible pan and zoom. Since the contours drawn by the user to select pixels in the editing window conceptually fall between pixels, we do not smooth the zoomed image in order to keep the task clear. With a 2x to 3x zoom, virtually any contour can be drawn at pixel accuracy with modest practice. Note that the two panes in the editing window are linked to the same scrollbars, so that they pan in unison.

Discussion

After a few hours of practice, subjects were able to segment an image to single-pixel accuracy in 5 to 30 minutes, depending on the complexity of the image and the subject's ability. Our fastest subjects were consistently 2-4 times faster than our slowest subjects. Had we been more resource limited in the project, we could have retained only the more efficient subjects because it was generally obvious

after 5 hours of work how fast the subject would be. In all, considering the tedium of the task, we were greatly pleased at the efficiency they achieved.

One of the greatest motivations for us to build a custom tool in Java was that it can be run from virtually any computer over the Internet. Some subjects were approved to work at home, which allowed us to support 30 subjects with a lab of only four PCs. We had hoped to recruit more subjects over the Internet, perhaps providing motivation with a reward or prize. However, the task of segmenting an image is a bit too time-consuming and the learning curve too long for this option. Our time was better spent training a smaller number of dedicated, paid subjects.

The issue of discontinuous segments was a problem for many of the subjects. Few seemed to understand the utility of the concept, and discontinuous segments were not used consistently by any subject. These consistency problems are probably a deficit of the tool itself, as it is difficult to distinguish a discontinuous segment of two pieces from two separate segments in the main window. (In the editing window it is very clear, since each panel always contains zero or one segments.) When using the segmentation data we recommend breaking up all discontinuous segments into their connected components, which is what we do for all our experiments.

2.1.3 Image Selection and Presentation

I selected 1020 images from 30,000 images in the Corel image dataset. The images were chosen to be a representative sampling of the wide variety of images in the Corel dataset, with synthetic and abstract images removed. We deemed synthetic images, such as images of marble texture, and abstract photographs, such as images of the reflections of neon signs in wet sidewalks, to be inappropriate for our segmentation task, since those images do not contain distinct objects. The majority of the images are of animals in natural scenes, but there are also many images of people, man-made structures, and urban scenes. Figure 2.6 shows a representative sample of the image set.

The segmentation tool was set up as a client in conjunction with an image server. Subjects would log into the system using their unique subject ID. Each time the subject would request a new image to segment, the tool would contact the server to request a task. The server would choose a random image while imposing certain constraints on the choice:

1. The subject should not see any image more than once.
2. Each image should be segmented by multiple subjects, but as few images as possible should be segmented by exactly the same group of subjects.
3. No image should be segmented more than N times.
4. Images that have been segmented by other subjects were given priority, so that we obtain a stream of “completed” images as time progresses.

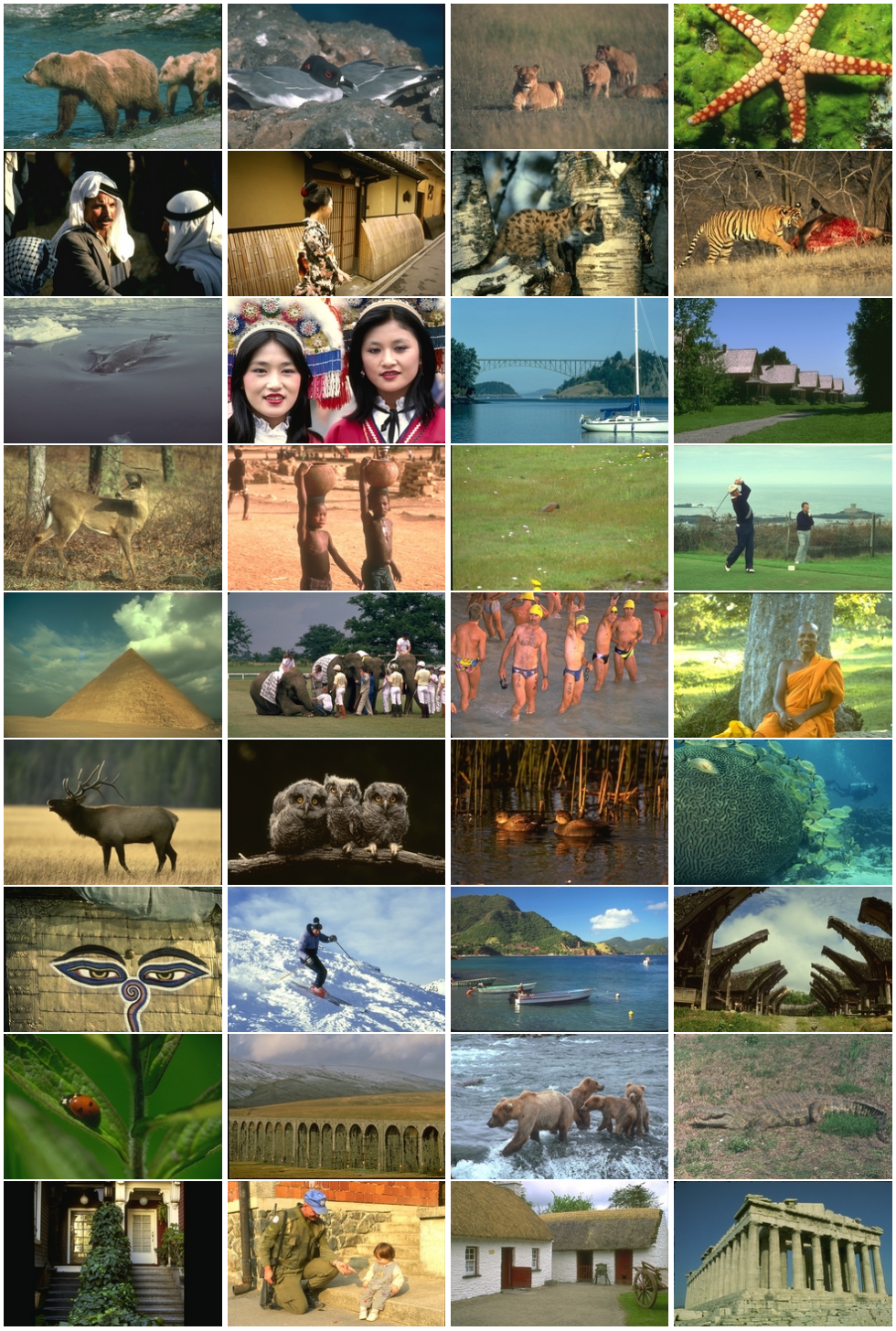
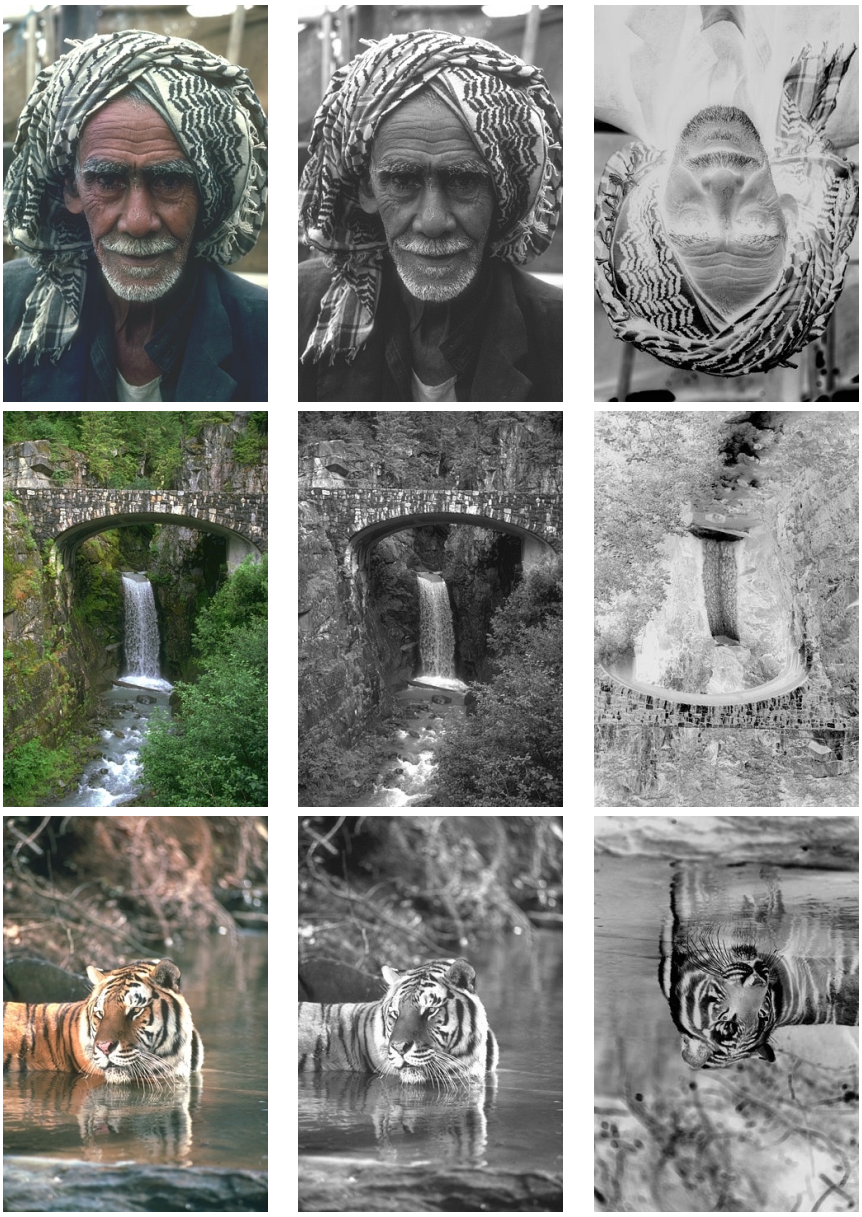


Figure 2.6: Example Images. 36 randomly selected images from the dataset. The images are of complex, natural scenes with a wide variety of subject matter.



(a) (b) (c)

Figure 2.7: Image Presentation. Examples of the three ways images were presented to subjects. (a) Color, (b) Grayscale, (c) Inverted and negated grayscale ("invneg"). The dataset contains 5555 color segmentations of all 1020 images, 5554 grayscale segmentations of the 1020 images, and 486 inverted-negated segmentations of 102 images. We may determine what role color had in the subjects' task by comparing the color and grayscale segmentations. The inverted-negated transformation is intended to disrupt high-level cues involving shadows, reflections, perspective, and familiar configuration, without changing the low-level statistics of the images.

The images were placed in a canonical random order. For each image request, the server would choose randomly from the next W images that satisfied the constraints, where the window size W was set to 100. The client-server architecture was ideal because we could vary the constraints over time to match the throughput achieved by our subjects. Completed segmentations were also sent to the server, facilitating automatic backup of the data.

In addition to picking the image for a subject, the server would choose a presentation mode. Figure 2.7 shows the three ways in which images were presented to the subjects for segmentation: full color, grayscale, and inverted-negated grayscale. The subjects could not change or choose the presentation mode of the image. The server would choose it randomly, ensuring that each image was segmented the appropriate number of times in each presentation mode. The server was configured to have each image segmented not more than five times in color and grayscale, and to have each tenth image segmented not more than four times inverted-negated.

2.1.4 Data Structures

The advantage of the custom tool was that it produced an explicit partition of the pixels of an image into disjoint sets. The segmentations were stored in this manner, as a map of pixels to segment numbers. In addition to the segmentation data, each segmentation file contains the subject's ID, a timestamp marking the creation time of the segmentation, and the presentation mode.

2.1.5 The Dataset

In all, after 8 months of data collection from 30 subjects, we have 11,595 segmentations of 1,020 images. There are 5,555 color, 5,554 grayscale, and 486 inverted-negated segmentations. Every image is segmented a minimum of 5 times in color and 5 times in grayscale. Every tenth image is segmented $\sim$4 times inverted-negated. Figures 2.8 and 2.9 show example segmentations.

2.2 Dataset Integrity

As Figures 2.8 and 2.9 shows, the segmentations produced by different humans for a given image are not identical. But, are they consistent? The integrity, and indeed the utility, of the dataset depends on a positive answer to this question.

2.2.1 Human Variability

Ignoring for the moment differences that arise from boundary localization errors, two subjects may segment an image differently for any of several reasons:

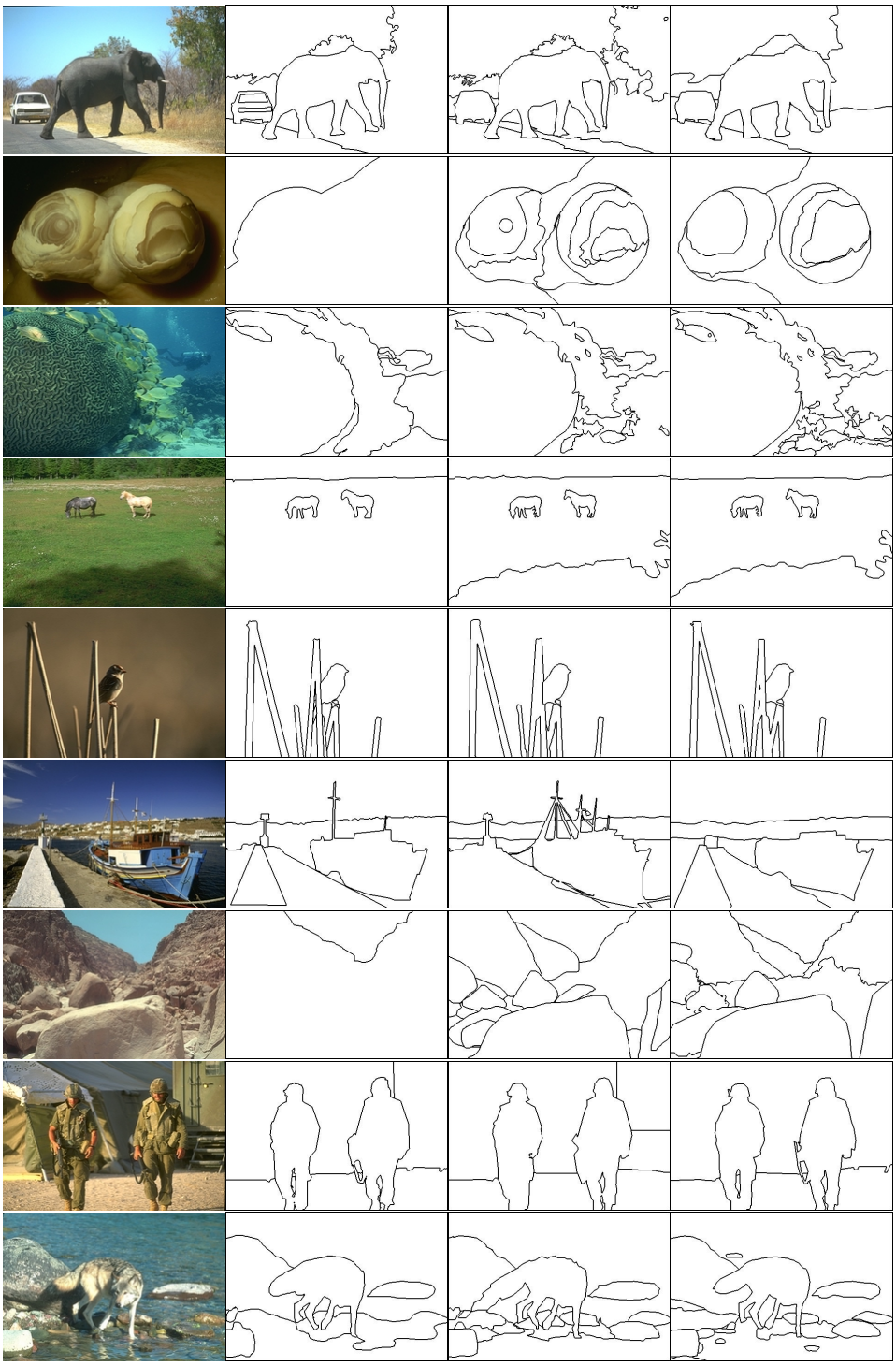


Figure 2.8: Example Human Segmentations I. Segmentations randomly selected from the dataset. Each row contains three segmentations of the image in the first column by three different subjects.

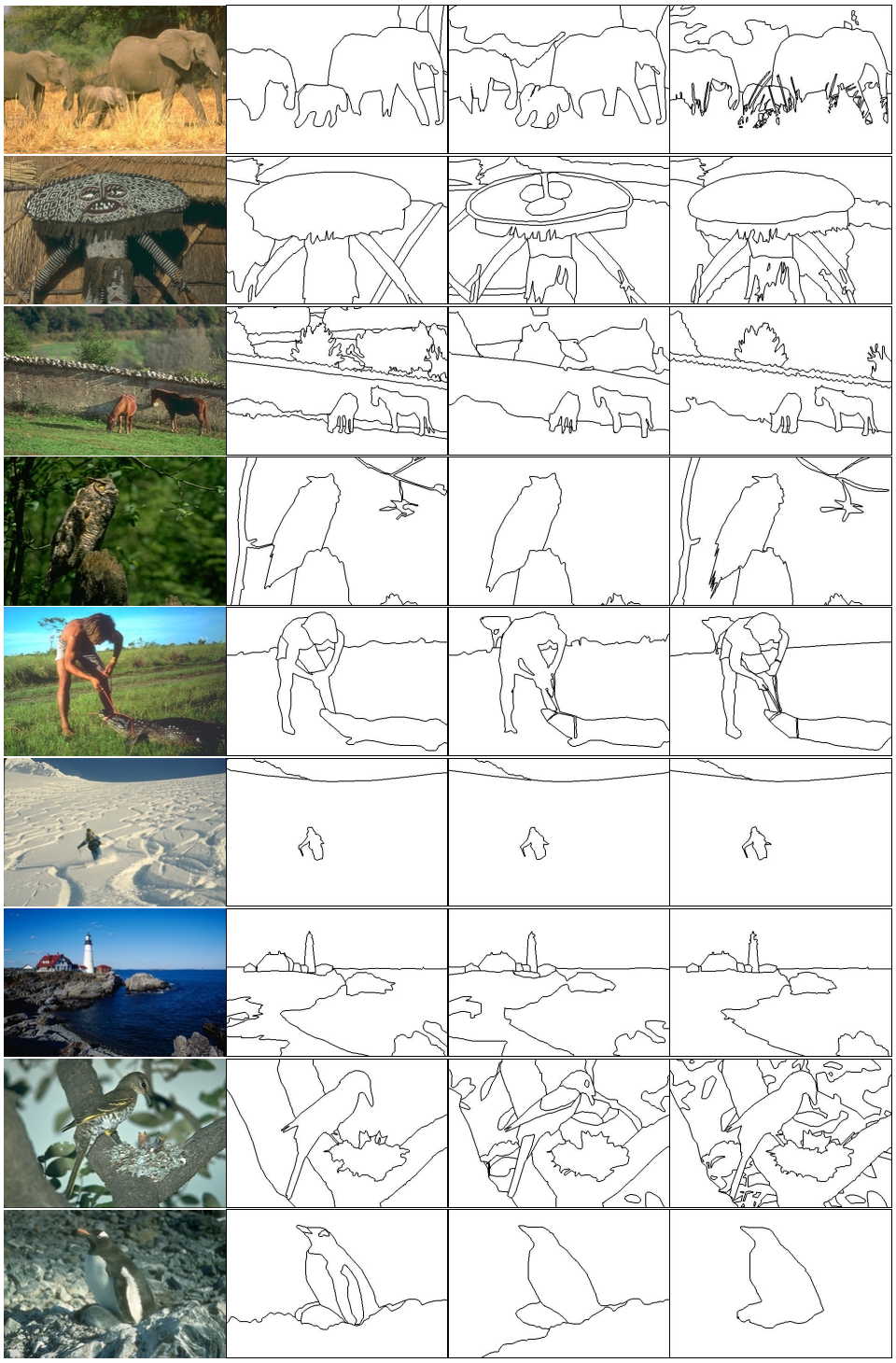


Figure 2.9: Example Human Segmentations II. Segmentations randomly selected from the dataset. Each row contains three segmentations of the image in the first column by three different subjects.

Different percepts. If two subjects perceive the same scene in two different ways, then they may see different objects and produce different segmentations.

Attention. Subjects may attend to different parts of the scene to different degrees, and may therefore over-segment (in a relative sense) the objects of focus, and under-segment the objects to which they did not attend.

Refinement. Two subjects may segment an image identically in all regards, except that one subject may divide objects into smaller pieces than the other subject did. In other words, subjects may segment at different granularities.

The last two effects produce variation between segmentations, but not inconsistencies. We would like to show that the variation in the segmentation dataset is predominantly caused by these two effects, and that the effect of different percepts is non-existent, or at least minimal.

Our working hypothesis is that all subjects share the same percept, and each subject's segmentation is a sample from that percept. Under the assumptions discussed previously — namely, ignoring translucence and shadows — we can model any perception of a scene as a tree. Consider Figure 2.10. The root of the tree represents the entire scene or image, and each pixel is represented by a leaf node. The internal nodes and their connections provide a complete hierarchical decomposition of the image into object groups, objects, and object parts. This hierarchy is akin to Palmer's whole-part structure [50] as a model for the perceptual organization of a scene. We will call this tree the *percept tree*.

The segmentation that a subject produces is a flat decomposition of the image into pieces. This segmentation must represent a cut through the subject's percept tree. Figure 2.11 shows an image from the dataset, and three subjects' segmentations of the image. The segmentations are far from identical, but do not seem contradictory. The reason is that each segmentation (and hopefully the reader's percept also!) is consistent with a single percept tree, and all the differences can be explained by selective refinement. Note that two segmentations may be refinements of each other in different regions of the image. We term this *mutual refinement*.

Our hypothesis is that there exists a single percept tree that can explain all subjects' segmentations of an image. Why do we not suppose that all subjects share the same percept tree? Although the flat decomposition of an image into objects and object parts seems relatively unambiguous, the grouping of objects at the top levels of the tree is quite subjective. Objects may be grouped by any number of valid criteria, such as physical appearance, function, scene interaction, or depth ordering. Fortunately, permutations of the tree that occur above the level of the cut represented by a subject's segmentation cannot change the segmentation. Since we instructed the subjects to segment below the object level, different subjects' interpretations of how to group objects should not be observable

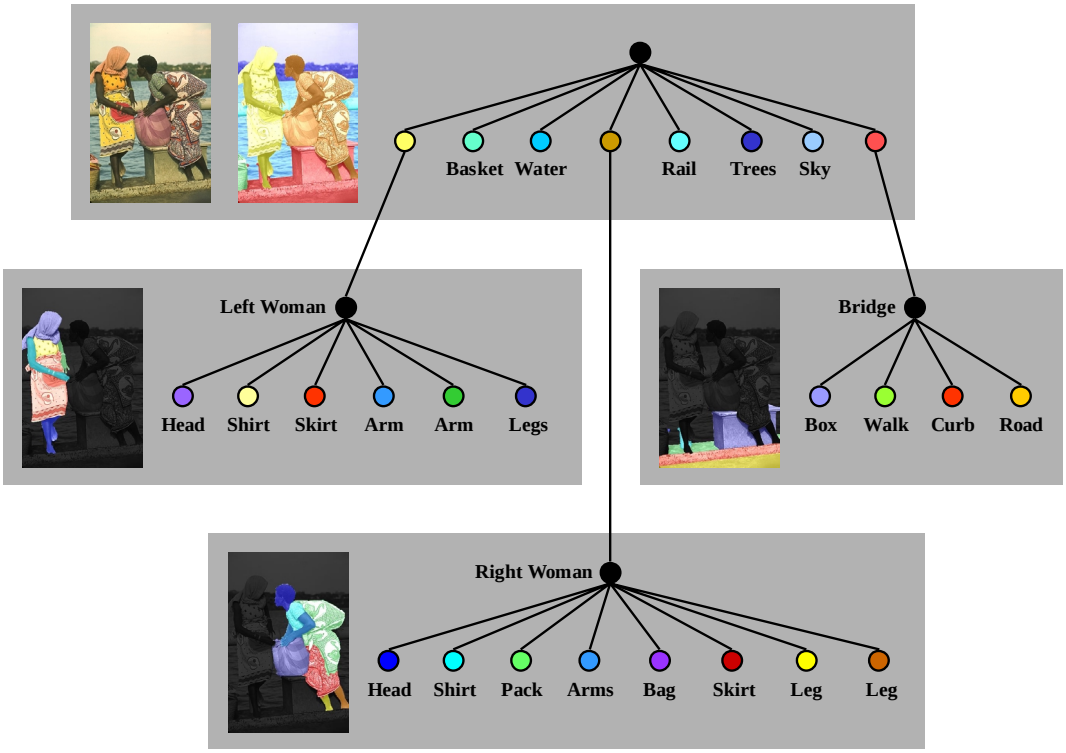


Figure 2.10: Percept Tree. The *percept tree* depicts the hierarchical decomposition of a scene that is presumably present in its percept. The root of the tree represents the entire scene, or image. Each image pixel is a leaf in the tree, although the figure shows only the top three levels. The tree represents the hierarchical grouping of pixels into object parts, object parts into objects, and objects into groups.

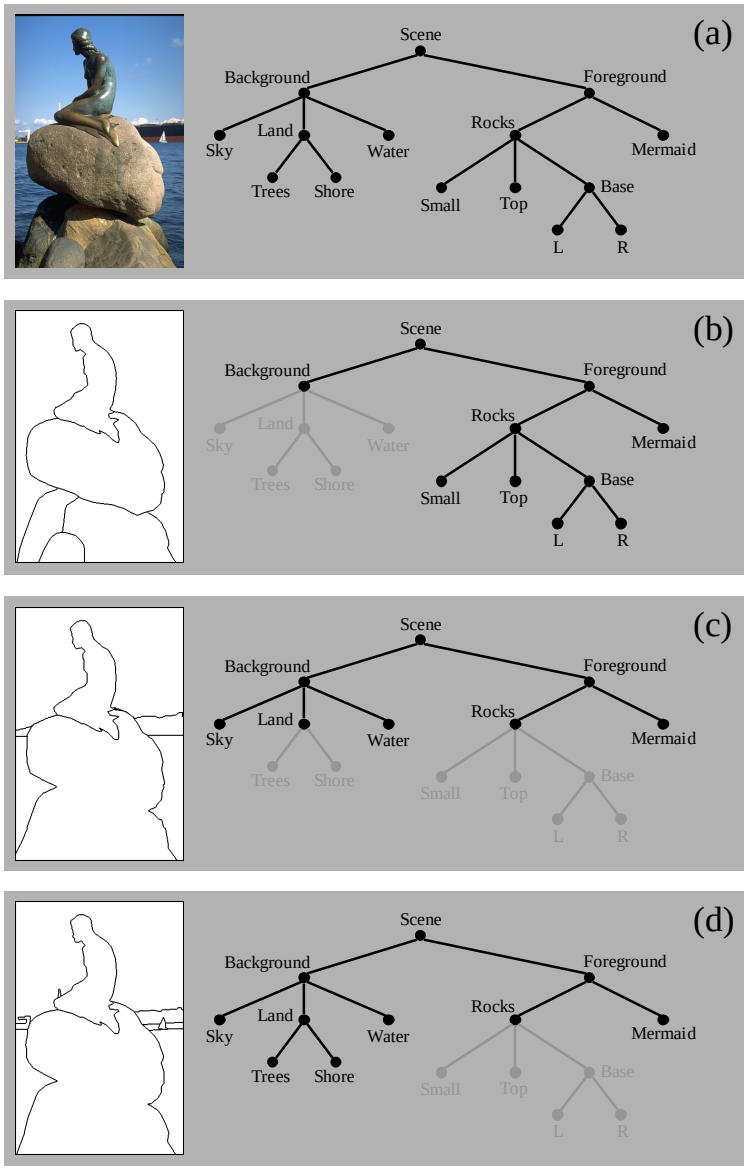


Figure 2.11: Examples of Refinement. Refinement characterizes most of the variation in the dataset. (a) An image from the dataset along with one possible percept tree. (b)-(d) Three subjects' segmentations of the image with the subset of the percept tree that the segmentation represents. Although these three segmentations are superficially different, they do not seem inconsistent with each other. This is because they are all readily explained by the same percept tree. All of the objects are segmented consistently; it is simply the degree to which each subject further segments, or *refines*, each object that changes. Note that *mutual refinement* is common, when two segmentations are refinements of each other in different regions of the image. Figure 2.12 shows an example of segmentations that do not share a percept tree.

Figure 2.12: Example of Inconsistent Segmentations. This figure shows two additional segmentations of the image used in Figure 2.11. In this case, the two segmentations are not consistent with one another because there is no single hierarchical decomposition of the scene that can explain both. The inconsistency is manifested as overlapping, rather than nested, segments. The right subject's shadow segment spans two rock segments in the left subject's segmentation.

in the dataset.

2.2.2 Human Segmentation Consistency

It is often stated that segmentation is an ill-posed problem. We will show that this is *not* the case by showing that segmentations of the same image by different subjects are consistent with each other. If two segmentations are consistent, then they are each cuts through some percept tree. In this case, if we pick a pixel in the image and consider the segment in each segmentation that contains that pixel, then the two segments must have a subset relationship. If one segment does not contain the other, then the there can be no common percept tree, and the segmentations are inconsistent. We can formulate a segmentation error measure based on these ideas that tolerates refinement but not overlapping regions.

First we define a quantity $E(S_1, S_2, p)$ called the *local refinement error*, which measures the degree to which two segmentations S_1 and S_2 agree at pixel p. Let $R(S, p)$ be the set of pixels in segmentation S which are in the same segment as pixel p. Where $|\cdot|$ denotes cardinality and $\cdot \backslash \cdot$ set difference,

$$E(S_1, S_2, p_i) = \frac{|R(S_1, p_i) \backslash R(S_2, p_i)|}{|R(S_1, p_i)|} \tag{2.1}$$

Note that this quantity is not symmetric: it is defined in two directions at each pixel. $E(S_1, S_2, p_i)$ is 0 when S_1 is a perfect refinement of S_2, and 1 when S_1 is disjoint from S_2.[1] We will consider two ways to combine the pixelwise errors into an overall segmentation error measure, depending on what type of refinement we would like to permit.

Local Consistency Error permits refinement in different directions in different parts of the image:

$$\text{LCE}(S_1, S_2) = \frac{1}{n} \sum_i \min \left\{ E(S_1, S_2, p_i), E(S_2, S_1, p_i) \right\} \tag{2.2}$$

Figure 2.13 depicts the computation of LCE for a pair of segmentations. The error measure penalizes both boundary localization error and refinement error in a soft manner.

Global Consistency Error (GCE) forces all local refinements to be in the same direction, i.e. from one segmentation to the other:

$$\text{GCE}(S_1, S_2) = \frac{1}{n} \min \left\{ \sum_i E(S_1, S_2, p_i), \sum_i E(S_2, S_1, p_i) \right\} \tag{2.3}$$

Note that GCE is stricter than LCE, so GCE>LCE. Since mutual refinement is common in the dataset, and since mutual refinement does not produce the perception of inconsistency, we favor the LCE measure over the GCE measure. It is worth noting that no qualitative result changes when we use GCE instead of LCE.

In addition to being tolerant to refinement, a segmentation error measure should also have the following properties: (1) Independence from the coarseness of pixelation, (2) Robustness to noise along region boundaries, and (3) Tolerance to different segment counts between the two segmentations. Both measures clearly satisfy (1). They also satisfy (2), since the error measures pool the local refinement error over the interior of regions. Property (3), however, is a potential problem. Degenerate segmentations that have either 1 region for the entire image or 1 region per pixel will have a zero LCE when compared to any other segmentation. Although these extreme cases will be a concern when we later attempt to use these error measures to compare machine segmentation to the human segmentations, they are not a problem for validating the dataset. At this point, we care only that the human segmentations are consistent with each other. Excessive refinement does not represent an inconsistency and is uncommon in the dataset anyway.

The top panel of Figure 2.14 shows the distribution of LCE over the dataset for both same-image and different-image pairs, comparing pairs of color segmentations. If the measure is meaningful and the data consistent, then we expect the same-image and different-image distributions to be separable. We see that the distributions for same-image pairs is as desired: it is unimodal and peaked at zero

[1] But they must share at least 1 pixel.

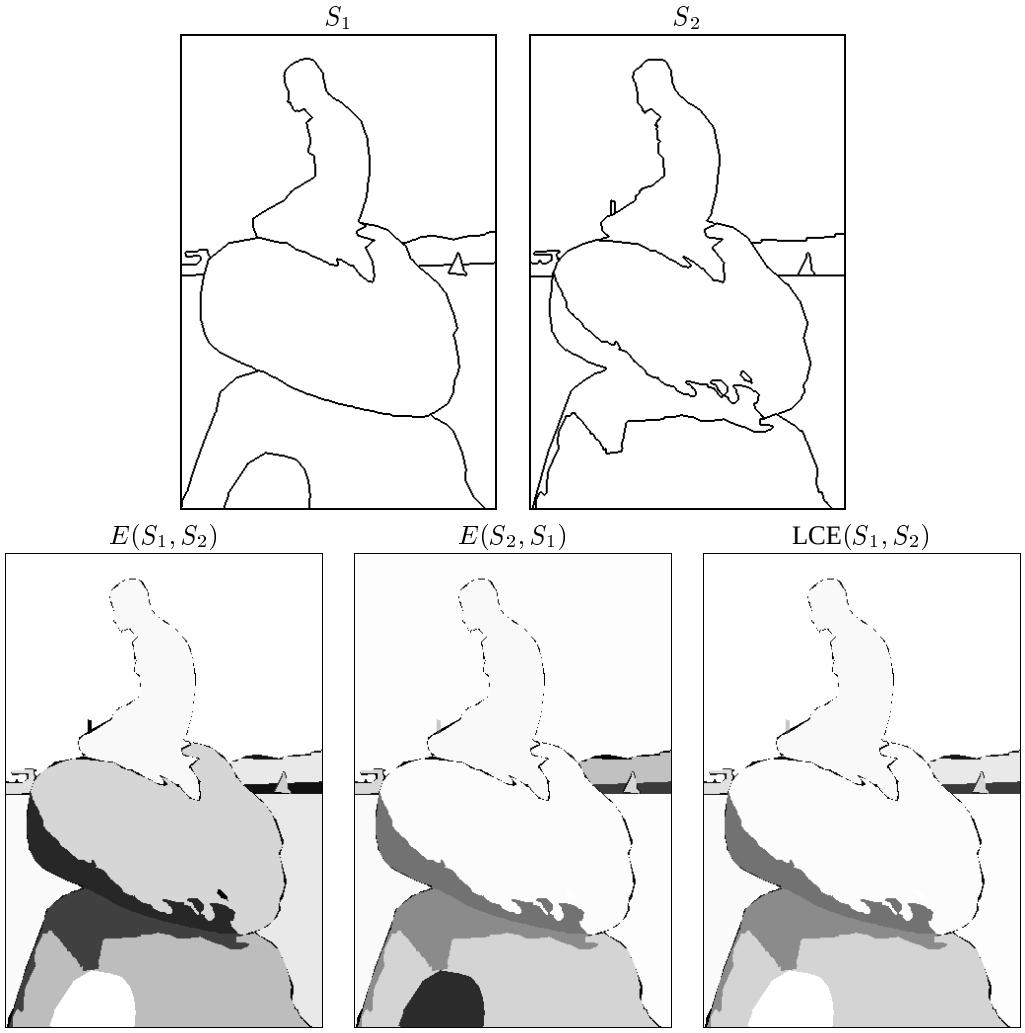


Figure 2.13: LCE Computation Example. Given two segmentations S_1 and S_2, we first compute the local refinement error images $E(S_1, S_2)$ and $E(S_2, S_1)$, which measure the degree to which one segmentation is a refinement of the other segmentation at each pixel. The pixelwise min of these two images yields the LCE image, and the average pixel value of the LCE image gives the LCE measure for these images, in this case 0.082. Pixel values are zero (white) in areas of perfect refinement, and 1 (black) in areas of inconsistent overlap.

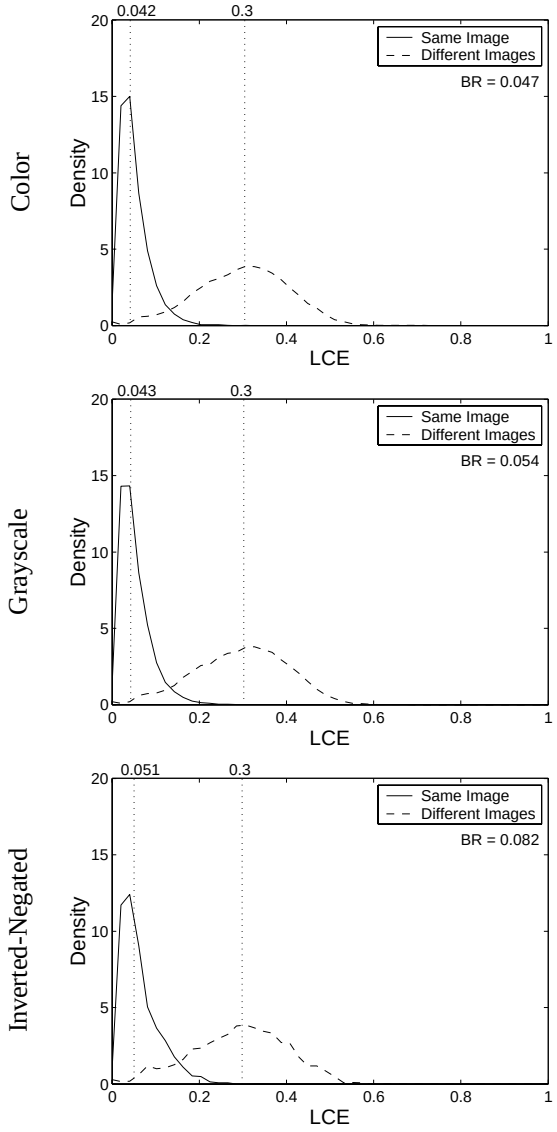


Figure 2.14: Distribution of LCE. The distribution of local consistency error (see Equation 2.2) for pairs of human segmentations. The solid lines show the distributions for segmentations of the same image; the dashed lines show the distributions for segmentations of different images. The fraction of overlap is shown below the legend as Bayes Risk (BR). The labeled vertical dotted lines mark the median values for each distribution. The first graph shows color segmentations compared to color segmentations; the second graph shows grayscale segmentations compared to color segmentations; the third graph shows inverted-negated segmentations compared to color segmentations. The human segmentations show a great deal of consistency, with the same-image distributions peaked at zero error. The grayscale and color segmentations are nearly identical, while the inverted-negated segmentations show significantly higher error.

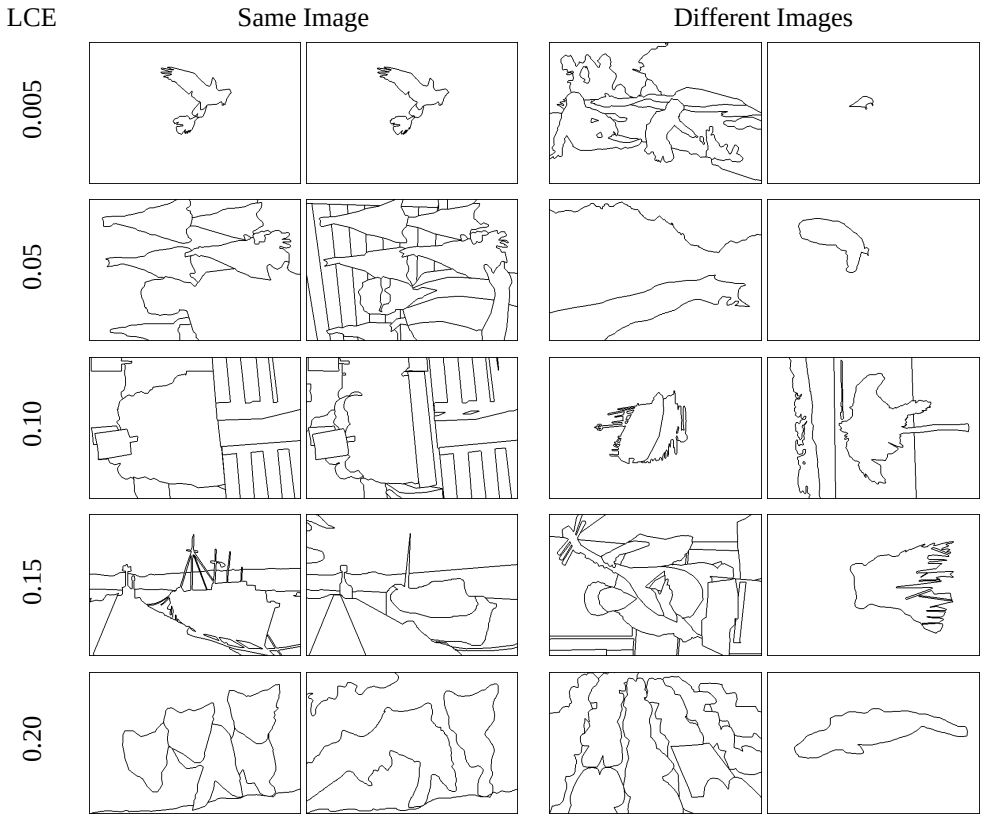


Figure 2.15: Example Pairs at Various LCE. Some examples of segmentation pairs at five values of LCE. In the same-image column, we see more agreement at lower LCE values. The different-image pairs do not seem to show the same trend, because of the problem LCE has with degenerate segmentations. The rightmost segmentation in the first row will compare favorably with nearly any other segmentation because all the segments are very small; any other segmentation is a good refinement of it. The same problem is present for the different-image pairs at LCE=0.05. The different-image pairs at LCE=0.1 show a fortuitous alignment of the figure in the center of the image.

with a light tail, with a median value of 0.042. The different-image pairs have larger error on average (0.3), so that the two populations are separable with a Bayes Risk of 4.7%. It is a bit disappointing, however, that there is a significant density of different-image pairs below the Bayes threshold. What do these pairs look like? Figure 2.15 shows both same-image and different-image pairs at a range of LCE values. The figure confirms that lower LCE corresponds to greater consistency, and that the different-image pairs with low LCE correspond to somewhat degenerate segmentations.

In addition to comparing pairs of color segmentations, Figure 2.14 shows the result of comparing grayscale to color segmentations (middle panel) and inverted-negated to color segmentations (lower panel). The progression from color to grayscale to invneg represents a monotonic decrease in information: first, we remove color, and then we remove some high-level cues relating to perspective and familiar configuration. We would like to know what effect these presentation modes had on the segmentation task. Though there is a miniscule increase in error 0.042 to 0.043 from color to grayscale, the error increases significantly to 0.051 for the inverted-negated segmentations. This supports our intuition that removing color should not affect the subject's task of segmenting the image. In contrast, the inverted-negated transformation was successful at removing some high-level information. Further study is needed to determine the nature of the lost information.

2.3 Discussion

We learned a few lessons in this endeavor that may be useful to those who might build similar datasets in the future.

2.3.1 Dataset Parameters

The first set of issues relates to the basic parameters of the dataset. Did we use enough images? Did we have enough segmentations per image? What was the utility of the grayscale, color, and inverted-negative modes of image presentation?

There is no doubt that in the future, a dataset of 1000 images will seem small. However, at this time, we have found 1000 images to be sufficient. This judgment is based more on the practical limits of current computing power than on the information content of the dataset. Most segmentation or boundary detection algorithms take minutes or hours of CPU time per image. Evaluation of these algorithms requires many executions per image. Even with the U.C. Berkeley Millennium cluster at our disposal – a PC cluster containing 250 CPUs with a total of 150GB of memory and an aggregate 125 GHz – we are pressed to use all of the images in our benchmarking work. This issue will disappear over time as computers continue their exponential speedup, and so the more interesting question relates to the content of the data. Do the 1000 Corel images capture enough of the variation

in the world?

The general answer to this question is surely not, but it depends on the task. We have been using the dataset primarily to explore issues in low-level vision having to do with boundaries and texture. The 1000 481x321 pixel images provide 150M pixels total. If we are concerned with the statistics of local image patches, the dataset appears to be large enough. However, the more high-level the task, the smaller the dataset will seem. Our future experience with this data will provide guidelines for how much data one needs for higher-level studies.

As for the number of segmentations per image, again the answer depends on the task. The work in subsequent chapters requires only that the dataset contain statistical trends that distinguish boundary pixels from non-boundary pixels. Even one segmentation per image would provide this information. If one requires a canonical segmentation for an image that does not miss any objects, then our dataset will be sufficient only for the images that are not too complex. When using the data, it is important to remember that the subjects' segmentations provide a sampling from the full segmentation tree, and so the segmentations should be interpreted in a probabilistic framework. It would certainly be interesting to attempt to construct a more strictly structured dataset that contained less inter-subject variation.

What about the different presentation modes? It appears that color played very little role in the task performed by the subjects. In retrospect, this is not terribly surprising. The subjects viewed each image for many minutes, during which time they were able to construct a complete mental model of the scene. There is no evidence that color is required for this type of task. Color would likely affect more short-term tasks related to the first object of focus, or more complex tasks such as judging the ripeness of fruit, but color should not greatly affect the basic perception of the structure and spatial arrangement of objects.

The inverted-negated segmentations are potentially more interesting. The intention of this transformation was to leave low-level image statistics unchanged, and to disrupt various mid-level and high-level cues related to the processing of shadows, the perception of depth, and the special treatment of familiar objects and configurations. Some images are difficult to decipher when inverted and negated, but most are not. One is generally able to mentally undo spatial inversion. Many objects, such as leaves and rocks, have no preferential orientation to start with, so inverting the image has little effect on them. Negating pixel values, on the other hand, is far more disruptive, since human perception is not based simply on local luminance differences. For example, shadows are dark and often not consciously perceived, while shadows in an inverted image are white and become difficult to ignore. It is possible, then, that shadows are more often marked in the inverted-negative segmentations. Qualitatively, this claim appears to be true, but we do not have any quantitative measures to prove it. In general, the effects of the inverted-negative transformation are difficult to analyze

because our ability to quantify mid-level and high-level vision concepts is so limited. However, we have found the inverted-negated segmentations to be both quantitatively different (at the low-level) and qualitatively different than the color and grayscale segmentations.

2.3.2 Task Description

We made every attempt to use naïve subjects untrained in computer vision to get unbiased data. In addition, the subjects' instructions were intentionally high-level to keep the task rooted in lay terms. Despite these precautions, subjects still asked questions we did not want to answer.

Shadows and reflections were a popular issue. Should they be segmented? Are shadows objects? Some subjects were content to ignore shadows completely. Others would seem to fixate on the shadows and treat them as first-class objects. Perhaps artistic training played a role? One wonders how often shadows are perceived by truly naïve subjects. Reflections are less common in our dataset, but the same issues arose. When pressed, we suggested that shadows and reflections be segmented if they seemed "visually important".

Translucence was a more frequent phenomenon than expected, arising in the form of sparse vegetation. Objects partially occluded by sparse vegetation are often easily perceived. The amodal completion performed by the human visual system seems particularly strong in this situation, where occlusions are many, but the gaps short. The perception is one of translucent vegetation, though strictly speaking the vegetation is not translucent. When the fine structure of vegetation falls below the image resolution, pixels cannot be divided between the vegetation and the occluded object. A layered model would be appropriate in this situation.

Human bodies were another interesting issue. Most of our subjects would systematically under-segment (from our perspective!) human figures. We would often get a single segment for each human in the image, even when the person's different pieces of clothing were the most distinct regions in the image. There is no inconsistency here, but other objects were not treated with such reverence. It is not surprising that human figures would receive special treatment by our perceptual systems, and the nature of this special treatment warrants further study. One possible explanation is that human figures can be segmented using two rather different hierarchies: one based on body parts and one based on clothing. The whole figure is the common ancestor of two conflicting perceptual hierarchies, and our subjects were uncomfortable either choosing one over the other or mixing the two. We explicitly instructed subjects that if they were to divide a human or animal into pieces, their segmentation should be based on visual appearance and not body parts.

Faces seemed to receive more scrutiny by our subjects than any other object class. Some subjects would routinely mark the eyes and mouth (and sometimes eyeballs and teeth) of any face in the image — human or animal — even when those regions were very small. I reminded subjects with

this tendency of the instructions to ensure that all segments should be of "approximately equal importance".

The final area in which subjects seemed to disagree was in the segmentation of background regions. The Corel images are not random images. They are "interesting" photographs, often with central figural objects at the camera's focal depth. Backgrounds are consequently often a bit out of focus. Some subjects would segment only the figural objects, while some would also segment the background. We encouraged subjects to attend to all areas of the image.

2.3.3 Subject Training

One of the motivations for writing the segmentation tool in Java was that it could be run from any computer connected to the Internet, and some subjects were permitted to work from home once they had demonstrated sufficient aptitude. We had plans to recruit subjects on the Internet, enticing people to segment images for us with prizes, eye candy, or simply the knowledge that they were advancing human understanding. This latter motivation actually worked for the NASA Clickworkers project [54], where subjects annotated craters on the surface of Mars.

Unfortunately, our segmentation task proved too complex (and boring) to rely on charity. The training proved more involved than we expected, and so we valued our trained subjects highly. However, the staggering scale of the Internet offers a source of human subjects provided the task is fast and requires little training. More focused projects than ours might be able to leverage this resource.

2.4 Future Work

The inverted-negated protocol is simple to use in an experiment, but we have not found satisfactory use for the data yet. One reason, perhaps, is that our data representation is too simple, so that we cannot identify the high-level patterns in the data that the inversion-negation is designed to disrupt. For this reason, and because each is independently worthy of study, future datasets should address some of the following issues:

- *Layers*. Translucent objects, sparse vegetation, and shadows break our strict one segment per pixel model. Permitting a pixel to belong to multiple segments, perhaps in a layered model, would avoid some of the problems we had with these issues.

- *Shadows*. Shadows are important enough to be segmented and tagged separately from the rest of the scene.

- *Shape and Figure-Ground*. One could study the depth-ordering of objects by labeling the figure and ground sides of each segmentation boundary. In addition, it is difficult to study the shape of objects without knowing to which object a contour belongs. We are currently adding figure-ground annotations to the segmentation dataset.

- *Hierarchies*. Segmentation is inherently hierarchical, so it would make sense to have subjects construct the hierarchy that they perceive. This additional information could be added to the current dataset through annotation, though there is not always a single hierarchy for an object.

2.5 Conclusion

We have presented a new dataset of segmentations of natural images by naïve human subjects containing $\sim$12,000 segmentations of $\sim$1,000 images. The images are taken from the Corel dataset. As a result, the images are of complex, natural scenes with no bias towards simplicity. The "difficult" phenomena of shadows, multiple scales, and texture are ubiquitous in the dataset, as they are in the natural world. The procedures we used to construct the dataset are fully disclosed in this document. In addition to the data itself, our segmentation tool is available for download from the Internet.

Although the segmentations of an image by different subjects are often superficially different, they appear qualitatively consistent. It is inconceivable that we would get identical segmentations from different subjects, due to the complexity of the task. However, assuming that the subjects share the same percept, segmentations of a single image should form an equivalence class. Our model of inter-subject variation is based on the notion of a common percept, so that the variation is characterized by refinement along with small boundary localization errors. The segmentation error measure LCE permits variation that can be explained by refinement, and penalizes all other sorts of variation. The data supports this as a model for human variation, verifying both that the dataset is consistent and that the error measure is a useful means of comparing two segmentations for equivalence.

Since there are multiple segmentations per image, the dataset does not provide a single ground truth segmentation for an image. Instead, it is the collection of human segmentations that constitutes the ground truth. Two of our motivations for constructing the dataset were to (1) show that the segmentation task is well defined, and (2) to provide a quantitative means of evaluating segmentation algorithms. In this chapter, we have shown that the task is indeed well defined by measuring the consistency of the human subjects. Since the subjects are consistent, we can use the dataset as the basis for a benchmark. The next chapter will treat the issue of finding good measures for benchmarking in detail.

With the introduction of a large segmentation dataset, we have laid the foundation for the quan-

titative evaluation of segmentation and boundary-detection algorithms. The dataset is also a rich source for the statistics of the boundaries and regions perceived by humans; it is likely to contain much information useful for developing computer vision algorithms, as well as understanding human vision. We look forward to its use by the scientific community.

Chapter 3

Segmentation Consistency Measures

In the previous chapter, we used a region-based segmentation consistency measure to determine that different segmentations of the same image by different subjects are consistent. The measure of consistency was based on the observations that segmentations are sampled from a perceptual hierarchy and that differences based on refinement are consistent with a single percept. This leniency was required because we did not have any control over the granularity of the segmentations provided by the human subjects.

One of the primary goals of constructing the dataset was to provide a means of benchmarking various grouping algorithms. These algorithms invariably have a parameter that controls the scale of grouping, since scale-selection remains a difficult and unsolved problem. To more robustly evaluate an algorithm, we therefore choose to evaluate it over all choices of its scale parameter. At each parameter setting, we will measure the degree to which the algorithm output predicts the human data.

In this task, our primary concern is one of discriminating more effective algorithms from less effective ones. We have found that instead of focusing on refinement as we did in Chapter 2, we get more robust measures of performance by testing an algorithm's output against the human data for equality. Though for any single image, one human subject may over-segment one area compared to another, this effect is greatly reduced if we use all subjects' segmentations of an image as the ground truth.

This chapter presents the evaluation techniques that we use in subsequent chapters to evaluate algorithms. Both boundary-based and region-based techniques are presented. The techniques will be illustrated in this chapter by comparing human segmentations to each another, in the same manner as in the previous chapter. This will simplify the presentation, while at the same time providing further evidence for the consistency of the segmentation dataset.

3.1 Statistical Methods

Before delving into the details of the various segmentation consistency measures, it will be useful to review two statistical evaluation techniques that will arise repeatedly in the remainder of this dissertation. Both techniques provide the means of comparing a classifier's output with the human ground truth dataset in order to quantify the classifier's performance.

3.1.1 Mutual Information

Assume that we can formulate a particular vision problem as a binary classification task, so that the ground truth human data provides the ideal classifier output $S \in \{0, 1\}$. An algorithm will produce $\hat{S}$, an estimate of S computed from the image data. In order to evaluate the accuracy of the estimate, we can compute the mutual information I between the classifier output $\hat{S}$ and the ground truth data S. Given the joint distribution $p(x, y) = P(S = x, \hat{S} = y)$, the mutual information is defined as the Kullback-Liebler divergence between the joint and the product of the marginals:

$$I(S; \hat{S}) = \int_{x,y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \tag{3.1}$$

The mutual information is non-negative. Higher values connote greater agreement with the ground truth, and therefore a better classifier.

Note that S is binary valued. If $\hat{S}$ is provided by a human segmentation that has been removed from the ground truth dataset — as is done in this chapter — then $\hat{S}$ will also be binary valued. If, however, $\hat{S}$ is provided by an algorithm, then it may be real-valued. In a probabilistic framework, for example, $\hat{S} \in [0, 1]$. In this case — which we will see in later chapters — we compute the joint distribution by binning the classifier's soft output.

3.1.2 Precision-Recall Curves

Though an information theoretic approach such as mutual information can produce a useful method for ranking algorithms relative to one another, it does not produce an intuitive performance measure. Thus, we turn to a richer evaluation tool: *precision-recall* curves. A standard evaluation technique in the information retrieval community [58], we have found the precision-recall framework to be both effective and intuitive. Furthermore, it is more appropriate for our purposes than the related *receiver operating characteristic* (ROC) curve or the Bayes risk, which is a summary statistic of the ROC curve. The problem with the ROC approach is that it considers the total error made by a classifier, from both classes weighted equally. When the classes are not balanced, the dominant class is easier to detect, and performance reported by ROC curves can be grossly exaggerated. In contrast,

precision and recall focus on the rarer class, and therefore do not reward the correct detection of the easy common case.

A precision-recall curve is a parameterized curve that captures the trade-off between accuracy and noise as a detector's threshold varies. *Precision* is the fraction of detections that are true positives rather than false positives, while *recall* is the fraction of true positives that are detected rather than missed. In probabilistic terms, precision is the probability that the detector's signal is valid, and recall is the probability that the ground truth data was detected.

These two measures are particularly meaningful for a classifier that is a detector of a sparse signal X, so that the two classes are X and $\tilde{X}$. For such a detector, we are interested in how many true X were missed (recall), and how many declared X were true (precision). A downstream application may be characterized in terms of how much true signal is required to succeed and how much noise can be tolerated. Recall gives the former, and precision the latter. A particular application can define a relative cost α between these quantities, which focuses attention at a specific point on the precision-recall curve. The *F-measure* [58], defined as:

$$F = \frac{1}{\alpha P^{-1} + (1-\alpha) R^{-1}} \tag{3.2}$$

captures this trade-off as the weighted harmonic mean of P and R. The F-measure is valued between 0 and 1, where larger values are more desirable. The location of the maximum F-measure along a precision-recall curve provides the optimal detector threshold for the application given α, which we set to 0.5 in our experiments. The curve's maximum F-measure can be used to characterize the curve with a single number. When two precision-recall curves do not intersect, the F-measure is a useful summary statistic.

In this chapter we will compare human segmentations to each other, so the "detector" will be binary valued. In this case, varying the detector threshold has no effect on precision and recall, so the precision-recall curve degenerates into a single point. In later chapters when we consider algorithmic detectors, we will see full precision-recall curves.

3.2 Boundary-Based Segmentation Consistency

By discarding segment membership information, a segmentation can be regarded as simply a boundary map. Are the subjects consistent when the segmentations are compared as boundary maps? We developed a boundary-based error measure with an eye towards later needing a way of comparing machine-generated boundary maps to the human segmentations.

3.2.1 Correspondence as Minimum-Cost Assignment

Given two segmentations, we will correspond little pieces of boundary, or *edgels*. An edgel is an oriented edge fragment which has an image-plane position (x, y, θ), and a length equal to 1 pixel unless otherwise noted. The error will be the proportion of edgels that cannot be corresponded. The naïve approach of simply corresponding collocated edgels will not work. As we wish to remain tolerant of localization error along boundaries, the matching must be soft with respect to position. Consequently, we resort to bipartite matching, where the cost of matching two edgels is proportional to their similarity in terms of spatial location and orientation.

The segmentations represent a partition of the pixels into disjoint sets. Any two adjacent pixels that are not in the same segment generate an edgel that is the boundary between the two pixels. The edgel itself has orientation $\theta = 0$ or $\theta = \pi/2$, but that is not the orientation we will use. Instead, we estimate the true boundary orientation as shown in Figure 3.1. Since we know the segment membership of each pixel, we can follow the segmentation boundary s steps in each direction out from the edgel of interest, continuing as long as the segments on each side of the boundary remain the same. Note that the search will stop at junctions because segment membership changes there. I used $s = 4$ to get the local edgel neighborhood of the edgel whose orientation we wish to estimate. The cloud of points given by the edgel centroids looks like the local boundary itself. The first eigenvector of the covariance matrix of these points yields the direction of maximal variance, and therefore the orientation of the boundary. Though we have no baseline with which to evaluate the orientation estimates, qualitative inspection verifies that the boundary orientations are estimated accurately at all but the highest curvature points, even at junctions.

We now need to measure the similarity between two segmentations represented as bags of oriented edgels. Consider the complete bipartite graph where the nodes represent edgels, and the edge weights are given by the similarity between the two edgels. The minimum-cost perfect matching of this graph provides a correspondence between the edgels of each segmentation.

There are a few practical details to work out. First, we must define the edge weights. Without orientation information, the most natural edge weights would the the Euclidean distance between the two edgels. In order to match intersections and corners accurately, we augment the edge weights with an orientation term:

$$w = \sqrt{\Delta x^2 + \Delta y^2} + \alpha \left(\frac{|\Delta\theta|}{\pi/2} \right) \tag{3.3}$$

The value of the scaling parameter α is specified below. Orientation differences $|\Delta\theta|$ are limited to the range $[0, \pi/2]$. A low edge weight requires both spatial proximity as well as similar orientation.

The second practical detail is that the two edgel sets are not likely to have the same cardinality,

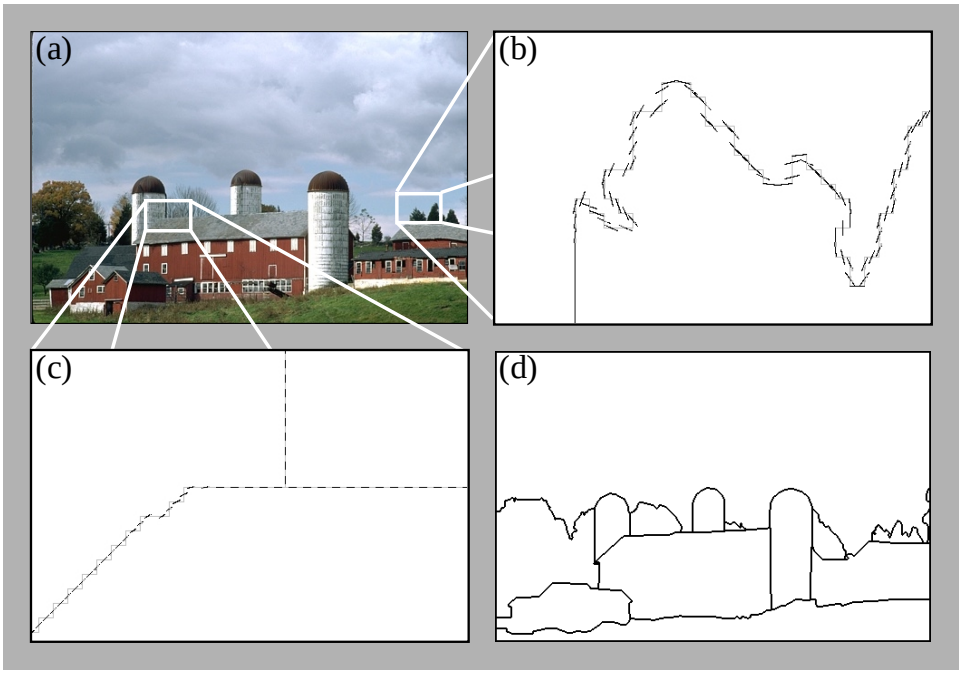


Figure 3.1: Estimating Human Segmentation Boundary Orientation. The segmentation boundaries run between pixels. We divide the boundary into edgels, where an edgel is the border shared by two adjacent pixels. In order to estimate the orientation of an edgel, we find the nearby edgels *on the same boundary*. The SVD of the centers of these nearby edgels yields an estimate of the edgel's orientation. (a) The original image. (d) The subject's segmentation. Panels (b) and (c) show the zoomed views of the areas marked in (a). In (b) and (d), segmentation boundaries are drawn in gray, and the edgels are drawn in black at their estimated orientation.

producing an asymmetric assignment problem. The standard trick for asymmetric assignment problems is to make them symmetric by padding the smaller side with *outlier nodes*. Edges to outlier nodes carry large weight, so that they are used in the assignment only when necessary. Nodes that match with outlier nodes are considered unmatched.

The assignment problem constructed thus far seems good until one tries to implement it. The problem has $2n$ nodes and n^2 edges, where n is the number of edgels in a segmentation. For our dataset, n commonly ranges from 1,000 to 10,000. Occasionally, we will wish to solve problems with n = 100,000. The best algorithms [29, 7] operate in the regime between $O(n^2)$ and $O(n^3)$, which is far too slow. By making the problem sparse and using the appropriate sparse algorithms, we can achieve tremendous speedup.

The vast majority of edges in the dense bipartite graph represent high-weight long-range connections between edgels and are superfluous for our task. We simply wish to find the edgels that correspond out to some small distance/orientation tolerance, and the edgels that don't. We can therefore replace the high weight edges with outlier connections without affecting the resulting thresholded assignment. In order to do this, we need to add outlier nodes on both sides of the graph, not just the smaller side. At this point, we have a bipartite graph that is extremely sparse between the "real" nodes, but still dense to and between the outlier nodes.

The overall sparsity has not changed much yet, but fortunately, the density of connections to and between outlier nodes is overkill. The outlier nodes are all identical, and so outlier connections in the final assignment can be freely permuted. This means that we can get away with sparse outlier connections in the bipartite graph before matching. In fact, we can use a surprisingly small number of connections per outlier node. The high connectivity properties of expander graphs permit this optimization.

We use Andrew Goldberg's CSA package [19] to solve the assignment problem in time that appears to be linear in the size of the graph, and therefore linear in the number of edgels. Goldberg's min-cost max-flow codes use his push-relabel algorithm [21, 10], which are the fastest available for the problem. In addition, the codes can easily handle our large problems.

One additional issue complicates the graph construction. CSA assumes the existence of a perfect matching. We adopted Goldberg's suggestion [20] of overlaying a high-cost perfect matching to ensure termination. We can then verify that the matching found by CSA does not include any of these high-cost safety-net edges.

In summary, we compare two segmentations S_1 and S_2 by computing a minimum cost assignment of their oriented edgels. Figure 3.2 illustrates the bipartite graph we construct for the assignment problem. An edgel in one segmentation is connected in the graph to edgels in the other

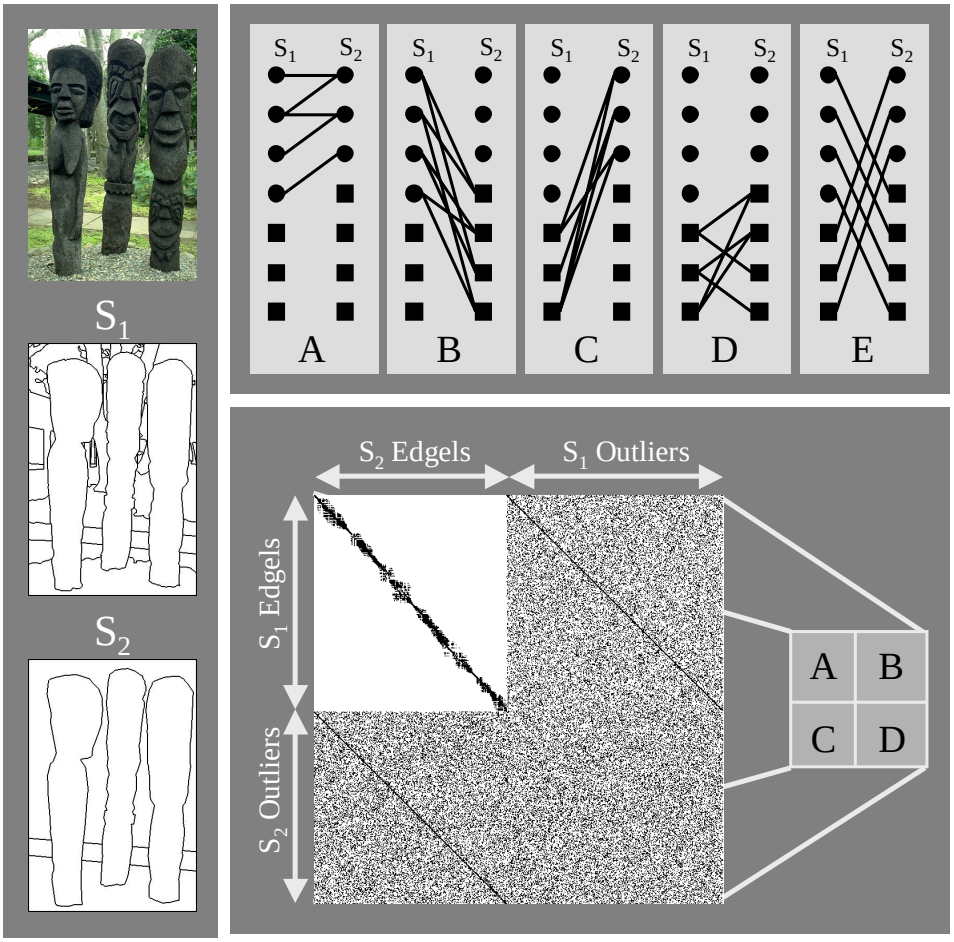


Figure 3.2: Bipartite Graph for Comparing Segmentations. We can compare two segmentations by matching edge elements. The figure shows the construction of the bipartite graph for this matching problem. The top panel contains an illustration of the 5 types of edges in the graph. The bottom panel contains the adjacency matrix for the graph. The two segmentations S_1 and S_2 contribute $n_1 = 3664$ and $n_2 = 4124$ nodes to the graph after pruning isolated edgels. After adding outlier nodes to both sides, we have a square $n \times n$ assignment problem, where $n = n_1 + n_2 = 7788$. The adjacency matrix for the bipartite graph has a block structure. Each block contains the corresponding edges from the top panel. The top-left block (A) contains the sparse local connections between edgels – the only "real" edges in the graph. Blocks B and C contain random outlier connections, and block D contains random outlier-to-outlier connections. The E edges lie on the diagonals of the B and C blocks, providing the safety-net high-cost perfect matching. The entire matrix has 64,470 non-zeros, for 0.1% density.

segmentation only if the Euclidean distance is less than some threshold d_{max}.[1] The edge weight is given by Equation 3.3 with $\alpha = d_{\text{max}}^{-1}$. If S_1 has n_1 edgels/nodes on the left side, and S_2 has n_2 edgels/nodes on the right side, then we add n_2 outlier nodes to the left and n_1 outlier nodes to the right. The high-cost perfect match overlay is added as parallel connections between nodes and outliers with a weight of 20,000*d_{max}. Each "real node" has d additional edges to randomly selected outlier nodes with weight 200*d_{max}. Finally, random outlier-to-outlier edges are added so that each outlier is connected to at least d outliers. We found $d = 6$ to be sufficient.

3.2.2 Comparing Segmentations

We will first apply the boundary-based error measure to pairs of human segmentations. Given two segmentations S_i and S_j, the error is given by the fraction of edgels that match to outlier nodes. Note that we get two numbers, since we have outliers on both sides of the match. Consider one of the segmentations S_j to be the ground truth, and the other segmentation S_i to be the signal. The fraction of matched S_i edgels gives us precision, since precision is the fraction of signal that agrees with ground truth; the fraction of matched S_j edgels gives us recall, since recall is the fraction of the ground truth contained in the signal:

$$S_i = \text{Signal} \tag{3.4}$$

$$S_j = \text{Ground Truth} \tag{3.5}$$

$$R_{ij} = P_{ji} = \text{Fraction of } S_j \text{ that matches } S_i \tag{3.6}$$

$$P_{ij} = R_{ji} = \text{Fraction of } S_i \text{ that matches } S_j \tag{3.7}$$

These labels for the outliers make more sense when the "signal" is the output of a machine vision system, as it will be in subsequent chapters. Both precision and recall are valued between 0 and 1, with larger values signifying greater consistency between S_i and S_j. Figure 3.3 shows the result of matching two segmentations from the dataset.

Figure 3.4 shows the distributions of precision and recall for both same-image and different-image segmentation pairs. Unlike the region-based error measure LCE, neither precision nor recall is tolerant to refinement, so two segmentations that are perfect mutual refinements of each other may have precision and recall scores of zero! Precision and recall measure the similarity between two segmentations as edge maps, so the values will be high only when the two segmentations have coincident boundary contours.

[1] We set d_{max} = 1% of the image diagonal.

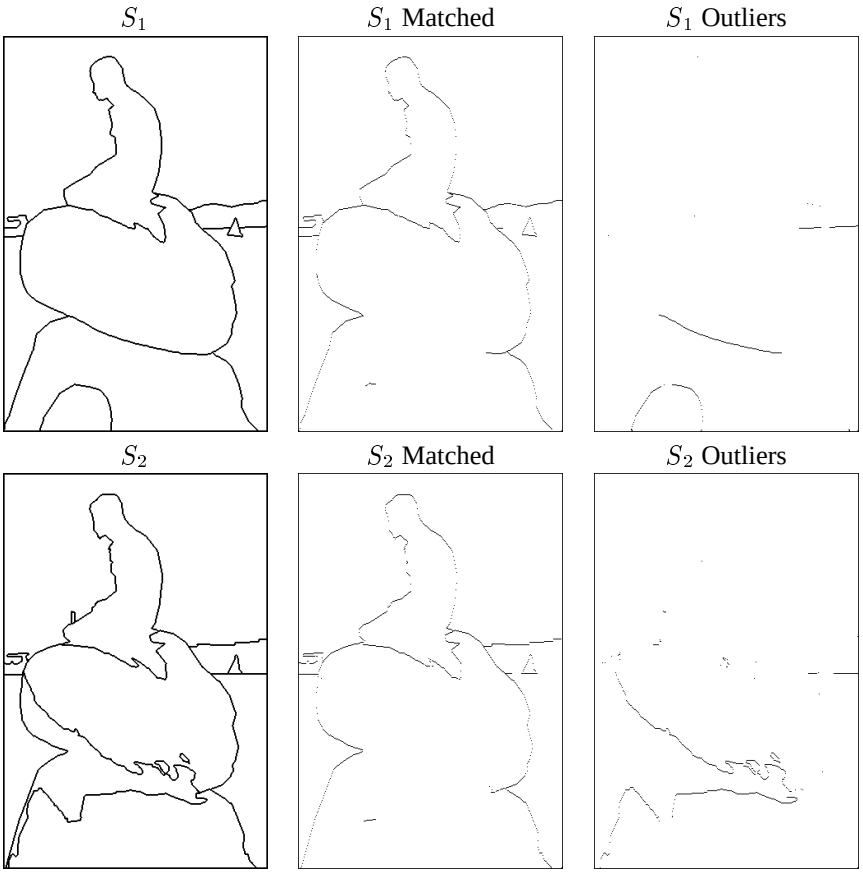


Figure 3.3: Edgel Matching Example. The left column shows the two segmentations S_1 and S_2 that we match by corresponding edgels. S_1 has 2631 edgels, and S_2 has 3355 edgels. The middle column shows the 2165 edgels that matched. The right column shows the unmatched, or outlier, edgels – 466 in S_1 and 1190 in S_2. The F-measure is 0.10.

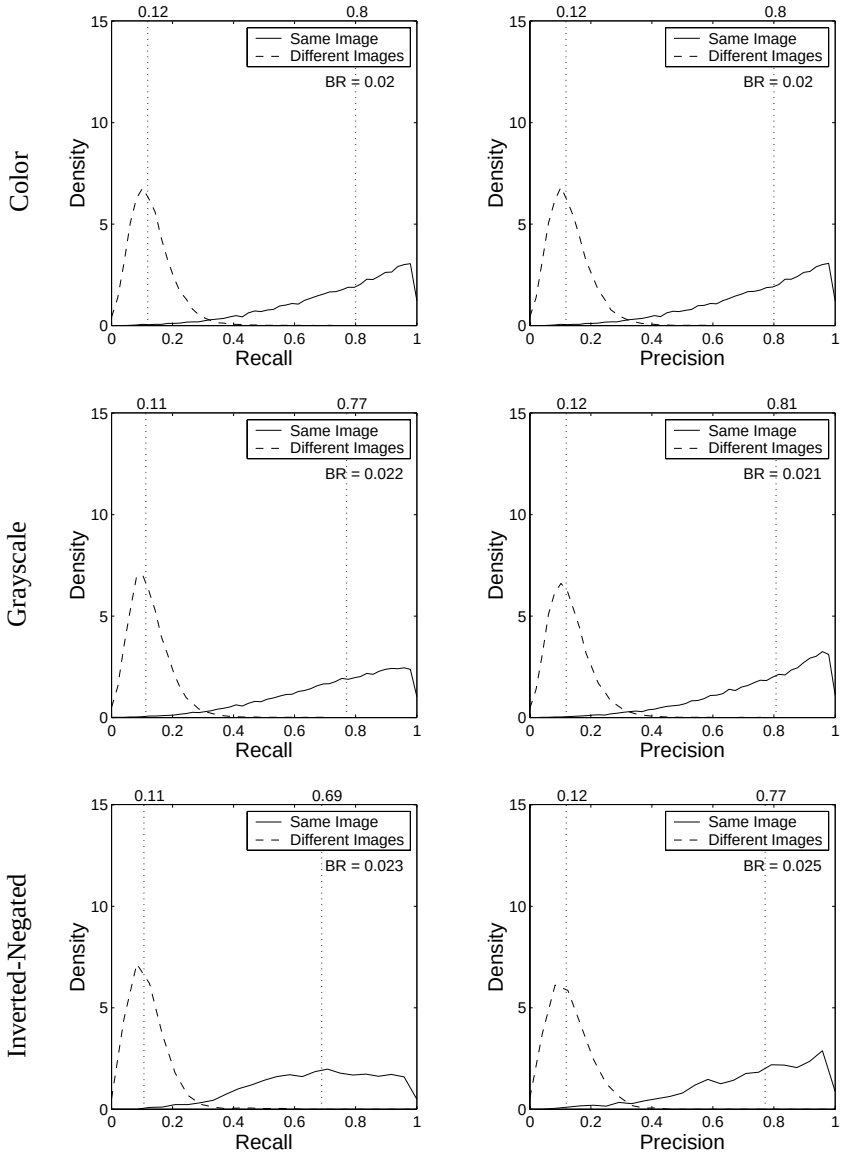


Figure 3.4: Distributions of Pairwise Recall and Precision for Edgels. The left column shows recall, and the right column shows precision. In each panel, the solid line denotes same-image pairs, and the dashed line different-image pairs. The median of each distribution is marked with a vertical dotted line. The first row shows color-color pairs, the second row color-grayscale pairs, and the third row color-invneg pairs. In each panel, we expect separable distributions because of the consistency of the human data. Note the similarity between color and grayscale segmentations, and the marked difference in recall for the inverted-negated segmentations. Lower recall indicates that the color segmentations contain boundaries that the inverted-negated segmentation do not have. Figure 3.6 shows the same distributions for the leave-one-out regime.

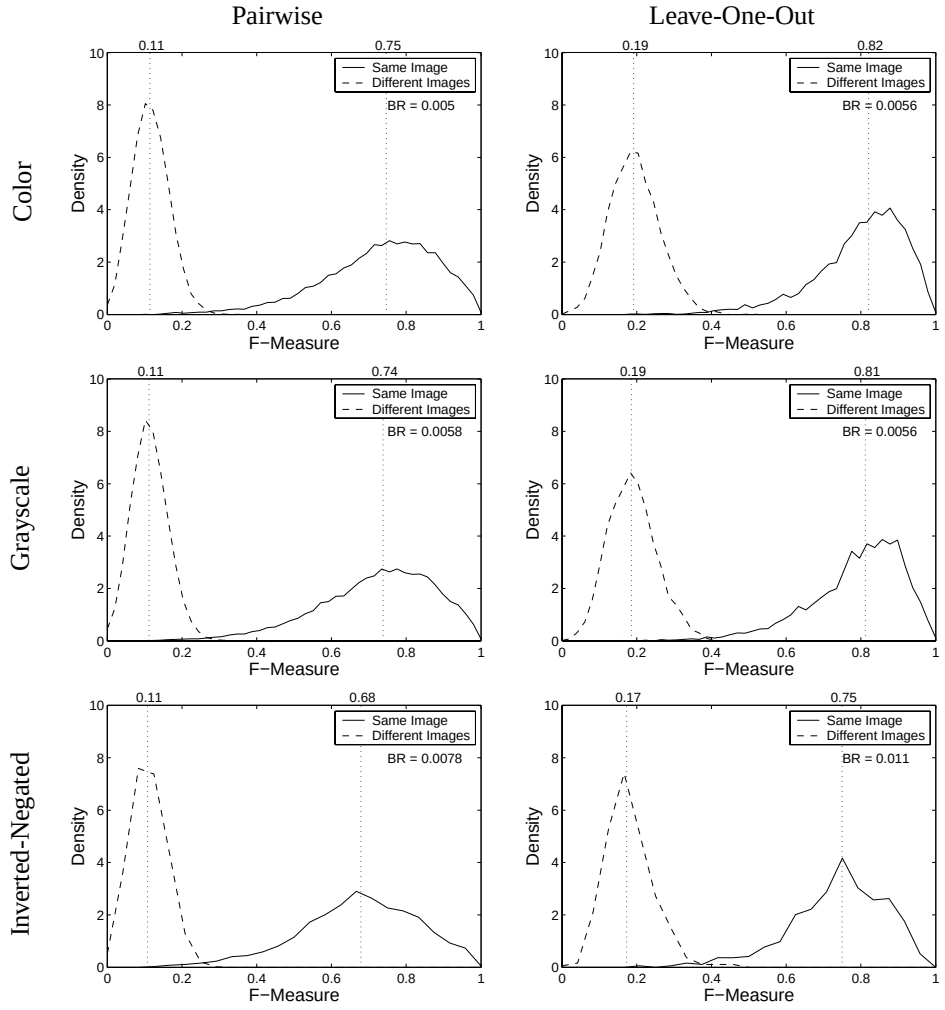


Figure 3.5: Distributions of F-Measure for Edgels. The F-measure combines the precision and recall shown in Figures 3.4 and 3.6. This measure does an exceptional job of distinguishing the same-image and different-image distributions for color and grayscale segmentations with a Bayes Risk of 0.5%. The discriminative ability of this measure can be attributed to the fact that segmentations of different images are unlikely to have coincident boundary contours, while segmentations of the same image seem to share at least 30% of their boundaries. Both the pairwise and leave-one-out methods are equally robust.

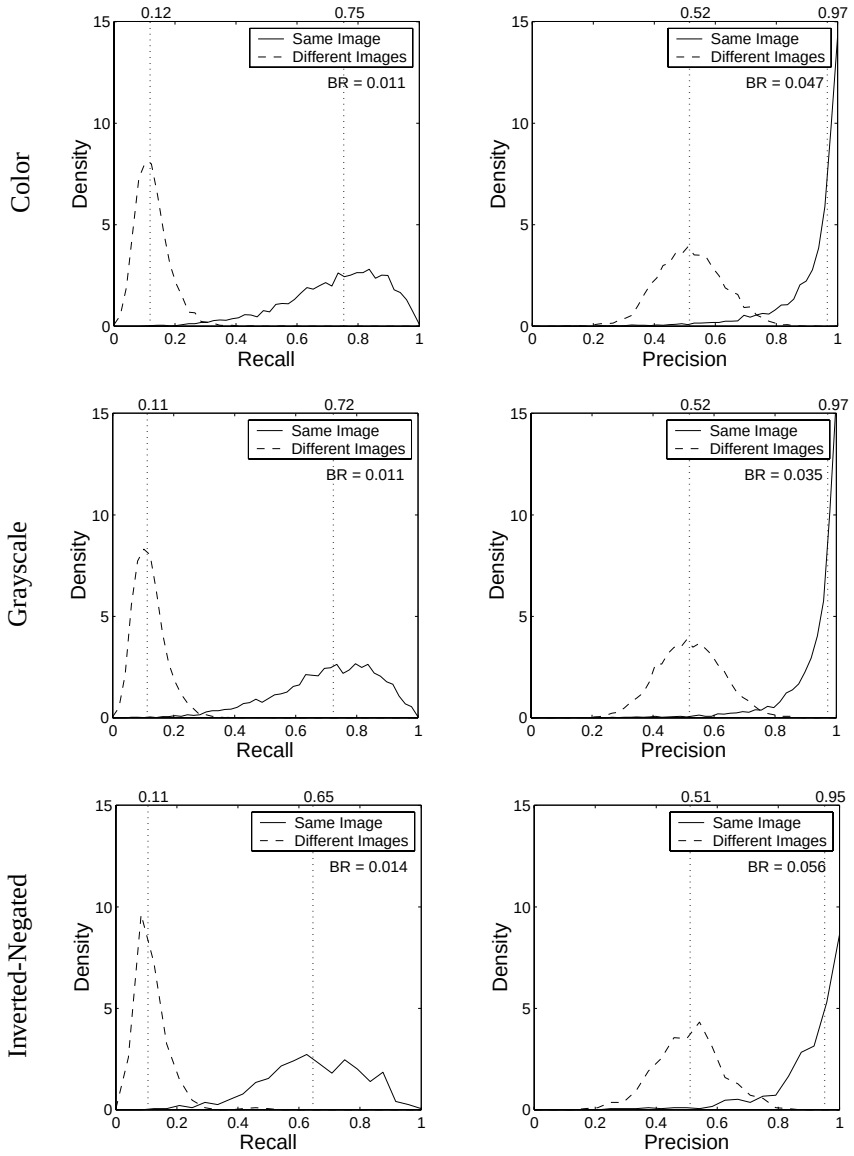


Figure 3.6: Distributions of Leave-One-Out Recall and Precision for Edgels. The layout is the same as for Figure 3.4. For the different-image distributions, each segmentation is compared to a set of randomly selected segmentations each from a different image. The size of the set is the same as what would be used for the same-image comparison. Note the marked increase in precision compared to Figure 3.4, since a union of human segmentations is more likely to contain all the boundaries marked by the left-out human. Recall decreases slightly for similar reasons. Again we see the same trend, that the inverted-negated segmentations show lower recall than the nearly indistinguishable color and grayscale segmentations.

The left column of Figure 3.5 shows the distributions of the F-measure for same-image and different image segmentation pairs. The F-measure does a surprisingly good job of distinguishing the two classes, despite its intolerance to refinement. It apparently takes only a small fraction of aligned boundaries to identify same-image pairs.

The three rows in Figures 3.4 and 3.5 compare the color, grayscale, and inverted-negated segmentations. In all cases, the color and grayscale segmentations are nearly identical. The inverted-negated segmentations, however, exhibit markedly lower recall, implying that the color and grayscale segmentations contain boundaries that the inverted-negated segmentation do not contain.

In addition to pairwise comparisons, we develop a *leave-one-out* methodology where one human is compared to the collection of other humans. Given k segmentations $\{S_1 \cdots S_k\}$, we want a measure that compares S_i to the set of segmentations S_j, $i \neq j$. Here we consider S_i the signal, and the other $k-1$ segmentation the ground truth. First, we compute the matching between each segmentation pair (S_i, S_j), $i \neq j$. These $k-1$ matchings yield $k-1$ precision and recall scores, P_{ij} and R_{ij}. The aggregate recall R_i is simply the mean of the individual recall values.

The leave-one-out precision is more complex. Precision should be the fraction of signal that matches with the ground truth. We declare that an edgel in S_i matches the ground truth if it matches an edgel in any of the S_j. The fraction of such edges yields the precision P_i for segmentation S_i. The following summarizes the leave-one-out precision and recall computations:

$$S_i = \text{Signal} \tag{3.8}$$

$$S_j = \text{Ground Truth}, \forall i \neq j \tag{3.9}$$

$$R_i = \frac{1}{k-1} \sum_{i \neq j} R_{ij} \tag{3.10}$$

$$P_i = \text{Fraction of } S_i \text{ that matches any of } S_j, i \neq j \tag{3.11}$$

The leave-one-out precision and recall distributions are shown in Figure 3.6. Note that both same-image and different-image distributions have shifted up the precision axis, since the with the one-to-many comparisons, it is more likely that an edgel in S_i matches some edgel in one of the S_j. Likewise, the recall marginals have shifted down, since it is more difficult for S_i to explain all of the S_j than it is to explain only one. The right column of Figure 3.5 shows the corresponding F-measure distributions. The distributions have shifted, and the Bayes threshold has moved, but the separability of the distributions remains about the same as in the left column where the comparisons are between pairs.

3.2.3 Discussion

This segmentation consistency measure, based on corresponding edge elements, is appropriate for evaluating machine-generated boundary maps. If an algorithm produces oriented edgels, then the approach outlined above can be used unchanged. However, many boundary detectors simply classify pixels as on-boundary or off-boundary. In this case, we can correspond boundary pixels instead of oriented edge elements, and omit the orientation penalty from the edge weights in the bipartite graph. If this approach is applied to the human-human comparisons, the distributions of precision, recall, and F-measure remain virtually unchanged. Some accuracy is lost around junctions, but the overall effect is small due to the relative scarcity of junctions compared to boundaries.

3.3 Region-Based Segmentation Consistency

In addition to evaluating algorithms that output boundary maps, we would also like a methodology for evaluating algorithms that output segmentations. A segmentation algorithm implicitly outputs a boundary map, and this boundary map can be evaluated using the method of the previous section, but a segmentation contains more information than a boundary map. In particular, the boundaries of a segmentation form closed contours, and so the pixels are divided into disjoint sets. In this section, we will develop three region-based methods for measuring segmentation consistency.

Why do we need multiple region-based consistency measures? The reason is that we do not know *a priori* which measures will be good for which tasks. Until we have extensive experience evaluating segmentation algorithms, it is not possible to determine which measure, if any, is superior. For example, we developed the LCE measure in Chapter 2 as a way of characterizing the consistency of different subjects' segmentations, but we also hoped to use it as a benchmarking measure. We discovered, however, that it is not well suited to the benchmarking task because a measure that permits refinement is too lenient for that purpose.

Consequently, we develop three region-based measures specifically for the task of benchmarking: a variant of LCE in Section 3.3.1, a measure based on mutual information in Section 3.3.2, and third measure based on precision-recall in Section 3.3.3. Both the mutual information and precision-recall approaches are based on proven statistical techniques in computer vision. Until one of the three region-based measures is proven to be reliable, segmentation evaluation work at this time should be done using multiple measures to validate results.

3.3.1 Region Overlap

The region-based consistency measure LCE, introduced in Section 2.2.2, is tolerant to refinement in either direction at each image pixel. If we simply replace the pixelwise minimum with a maximum,

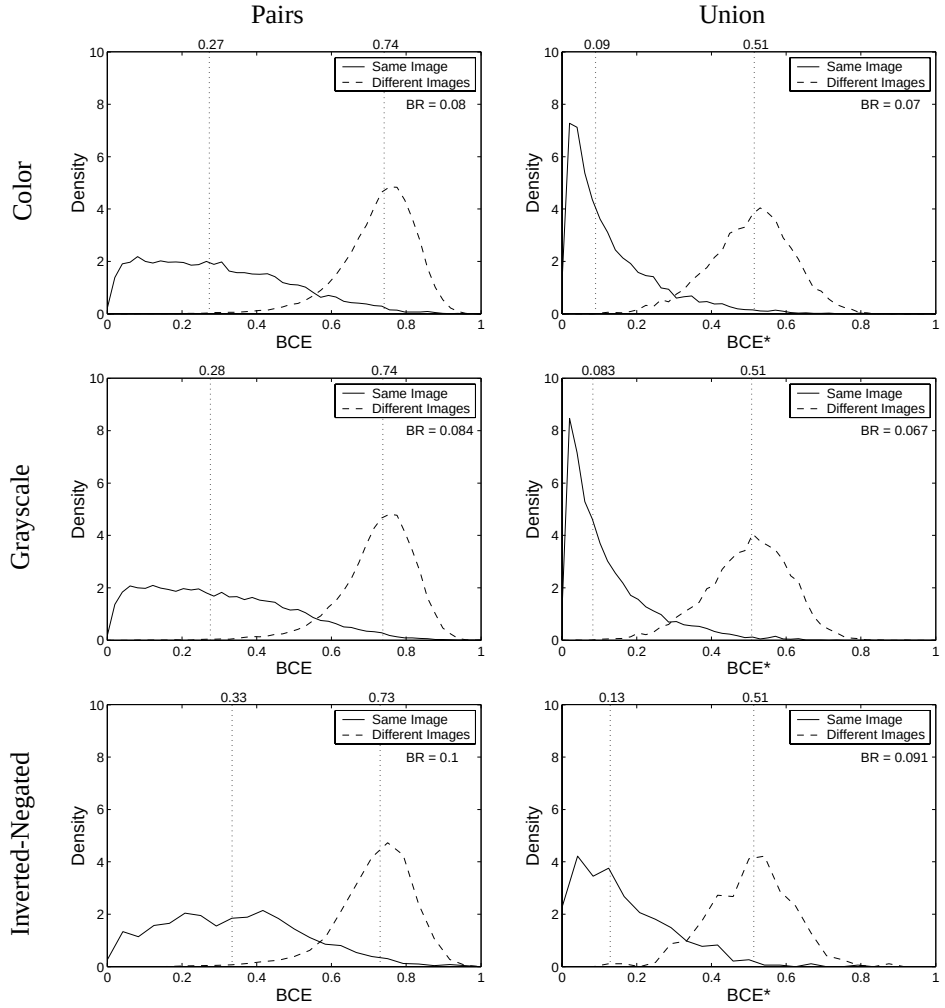


Figure 3.7: Distributions of BCE and BCE*. Since BCE does not tolerate refinement, the same-image comparisons yield higher error than with LCE, resulting in a marginal increase in overlap between same-image and different-image distributions. By using multiple segmentations as ground truth, BCE* lowers the separation and recovers the peak at zero error. Again, the grayscale and color segmentations are quite similar, but the inverted-negative segmentations are significantly different than the color segmentations.

we get a measure that does not tolerate refinement at all. The problem of degenerate segmentations "cheating" a benchmark disappear, and we get a measure that penalizes dissimilarity between segmentations proportional to the degree of region overlap. Applied to pairs of segmentations, the *Bidirectional Consistency Error* (BCE) is defined as:

$$\mathrm{BCE}(S_1, S_2) = \frac{1}{n} \sum_i \max \left\{ E(S_1, S_2, p_i), E(S_2, S_1, p_i) \right\} \tag{3.12}$$

In addition to considering the consistency of pairs of segmentations, we can ask if one subject's segmentation of an image is consistent with the collection of all other subjects' segmentations of that image. The BCE measure is easily extended to the "leave-one-out" regime by computing the minimum error at each pixel over each ground truth segmentation S_k:

$$\mathrm{BCE}^\star(S_j) = \frac{1}{n} \sum_i \min_{k \neq j} \left\{ \max\{E(S_j, S_k, p_i), E(S_k, S_j, p_i)\} \right\} \tag{3.13}$$

Figure 3.7 shows the distributions of BCE and BCE*. The layout of the figure is the same as in Figure 3.5 on page 47, with color-color comparisons in the top row, gray-color in the middle row, and invneg-color comparisons in the bottom row. From the left column (BCE), we see that though the same-image and different-images are still separated, the same-image distribution is not peaked at zero. The right column of the figure shows the distributions for BCE*. By measuring consistency with a set of human segmentations, BCE* better captures the consistency of same-image segmentations than BCE: the same-image and different-image distributions overlap less, and the same-image distributions are peaked at zero error. Once more, the grayscale and color segmentations remain nearly identical, while the inverted-negated segmentations exhibit measurable difference.

3.3.2 Mutual Information

Let us formulate the segmentation problem as a classification task, where we aim to classify pairs of pixels as belonging to either the same segment or different segments. Given the output of such a classifier — known as a *pixel affinity function* — one can perform segmentation by pairwise clustering. This is a common framework for image segmentation [67, 76, 65, 68, 13, 52, 17, 72]. To evaluate such an approach, we can evaluate the affinity function itself before clustering takes place, independent of what particular clustering method may be used. Any improvement to the affinity function will certainly ease the clustering task.

We need to compare a machine-generated affinity function $\hat{S}_{ij}$ to the ground truth affinity S_{ij}, where $S_{ij} = 1$ when pixels i and j belong to the same segment, and zero otherwise. If we compare the machine affinity $\hat{S}$ to only one human segmentation at a time, then the ground truth affinity S is given directly by the segmentation. We can compare $\hat{S}$ to a set of human segmentations by declaring

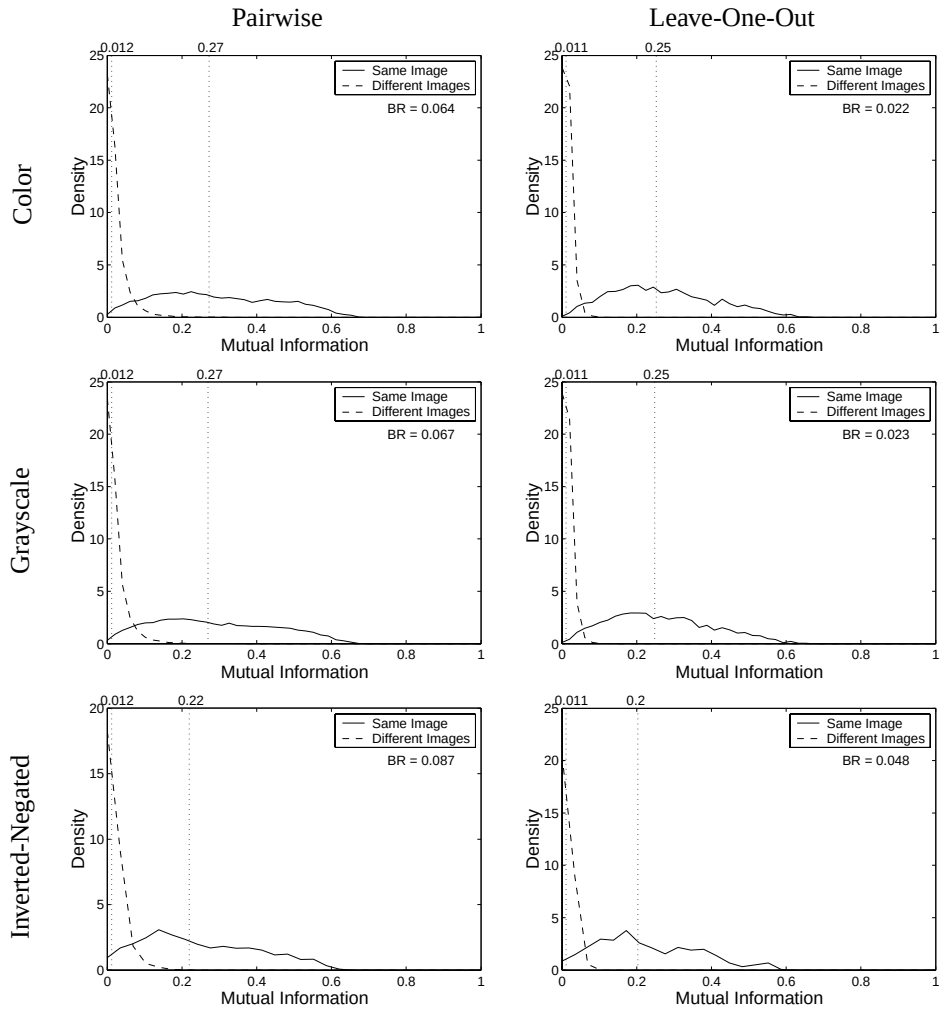


Figure 3.8: Mutual Information Distributions for Pairwise Affinity. In all cases, the ground truth is given by color segmentations. The left column shows the comparison for segmentation pairs, and the right column for leave-one-out comparisons. Either color (top row), grayscale (middle row), or inverted-negated (bottom row) are compared to the ground truth, and both same-image (solid lines) and different-image (dashed lines) distributions are shown in each panel. Median values are marked with a vertical dotted line. As expected, the different-image distributions are steeply peaked at zero, with better separation in the right column. The grayscale and color segmentations appear to have identical mutual information, while the inverted-negated segmentations share less information with the color segmentations.

$S = \min_k S^k$, so that a pixel pair is declared same-segment only if every subject k declared it to be same-segment.

We will explore both approaches by comparing human segmentations to each other. In the first case, we will compare pairs of segmentations where the ground truth is always a color segmentation, and the "signal" is either a color, grayscale, or inverted-negated segmentation. In the second case, the ground truth will be given by the set of color segmentations of an image, while the signal is still a single segmentation.

One approach we can take to evaluating an affinity function $\hat{S}$ is to compute the mutual information between it and the ground truth indicator S, as discussed in Section 3.1.1. Figure 3.8 shows the distributions for the mutual information between human segmentations. The left column shows the pairwise comparisons; the right column shows the leave-one-out comparison. Top to bottom, the rows show the result when the color, grayscale, and inverted-negated segmentations are used as the "signal", while always using the color segmentations as ground truth. As expected, the different-image distributions are steeply peaked at zero — more so for the leave-one-out regime, improving the separation in the right column. The grayscale and color segmentations appear to have identical mutual information, while the inverted-negated segmentations share less information with the color segmentations.

3.3.3 Precision-Recall for Regions

To complement the mutual information approach to comparing affinity functions, we also develop an approach based on precision and recall. The ground truth human segmentations define a set of same-segment pixel pairs that we wish to identify with a classifier. Given another human segmentation or machine-generated affinity function as "signal", we can determine how well it predicts the same-segment ground truth pairs.

In the case of comparing human segmentations, the signal is binary-valued. Precision is therefore $P(\hat{S}_{ij} = 1 | S_{ij} = 1)$, or the probability that a same-segment pair in the left-out human is a same-segment pair in the ground truth. Recall is the probability that a same-segment pair in the ground truth was in the left-out human, or $P(S_{ij} = 1 | \hat{S}_{ij} = 1)$.

Figure 3.9 shows the precision and recall distributions for the pairwise approach, and Figure 3.10 shows the distributions for the leave-one-out approach. The trends between the two are in contrast to those we observed in the boundary matching precision and recall. In this case, the union of human segmentations produces a stricter set of same-segment pairs, since all subjects must mark a pair in the same segment for $\hat{S} = 1$. Consequently, the leave-one-out distributions show increased recall and decreased precision. In both cases, the same-image distributions shift more than the different-image distributions, suggesting that the leave-one-out version of this measure is preferable. The

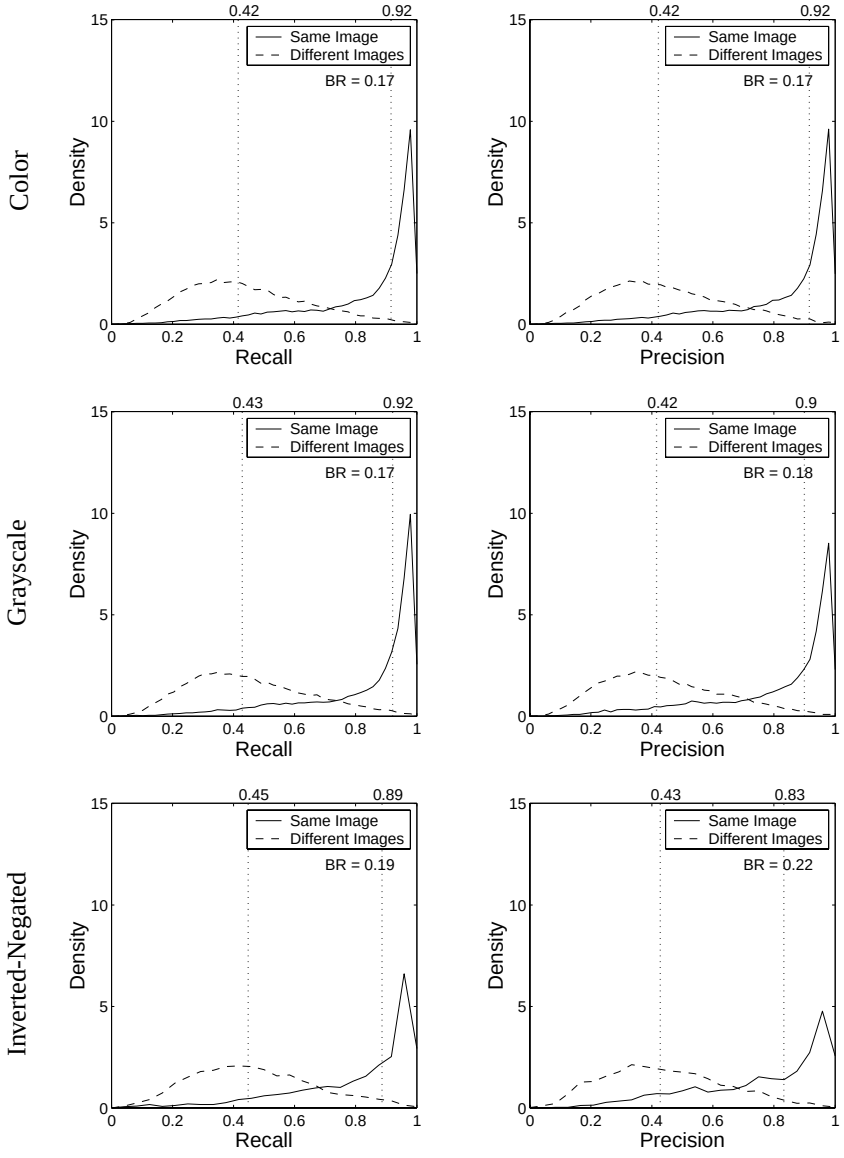


Figure 3.9: Distributions of Pairwise Recall and Precision for Pixel Affinity. Recall (left column) and precision (right column) distributions for same-image and different-image comparisons of human segmentations, for color (top row), grayscale (middle row), and inverted-negated (bottom row) segmentations compared to color segmentations.

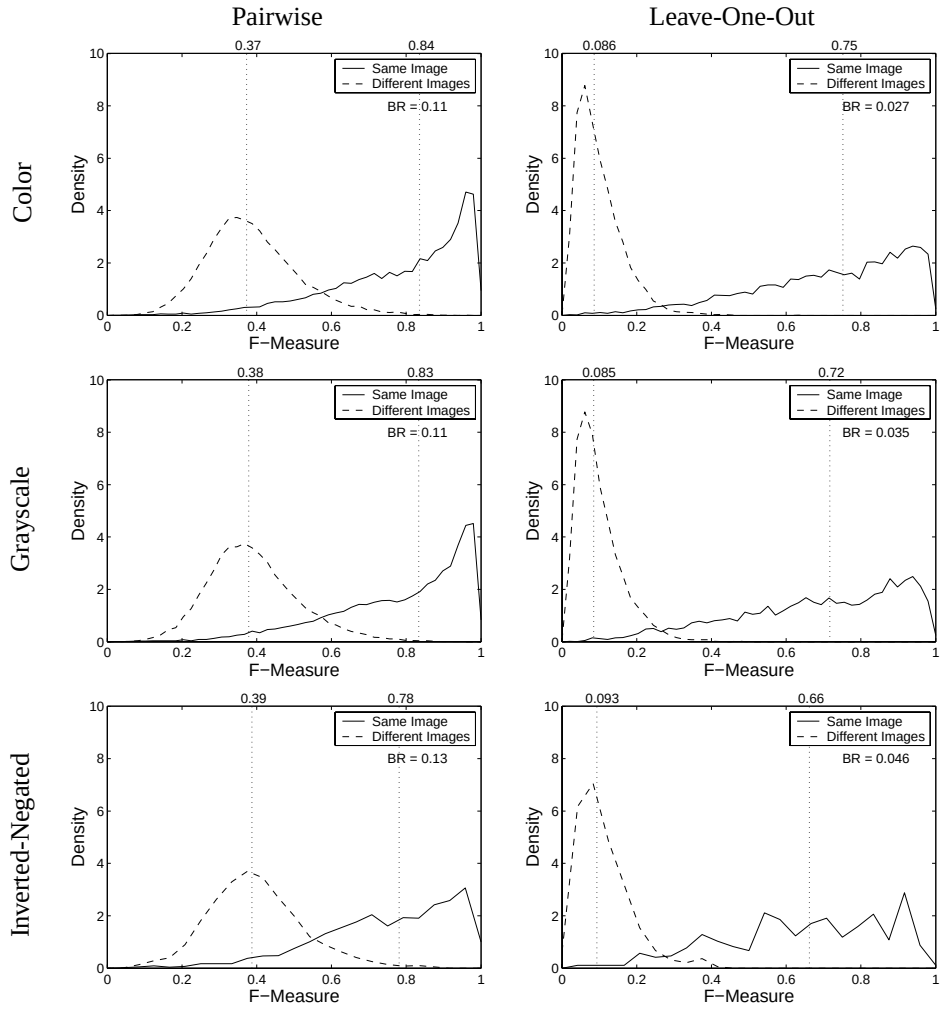


Figure 3.10: Distributions of F-Measure for Pixel Affinity. These F-Measure distributions combine the precision and recall from Figures 3.9 and 3.11. The leave-one-out distributions (right column) are better separated than the pairwise distributions (left column). In addition, the trend of increasing error from color (top row) to grayscale (middle row) to inverted-negated (bottom row) is again clear in the right column.

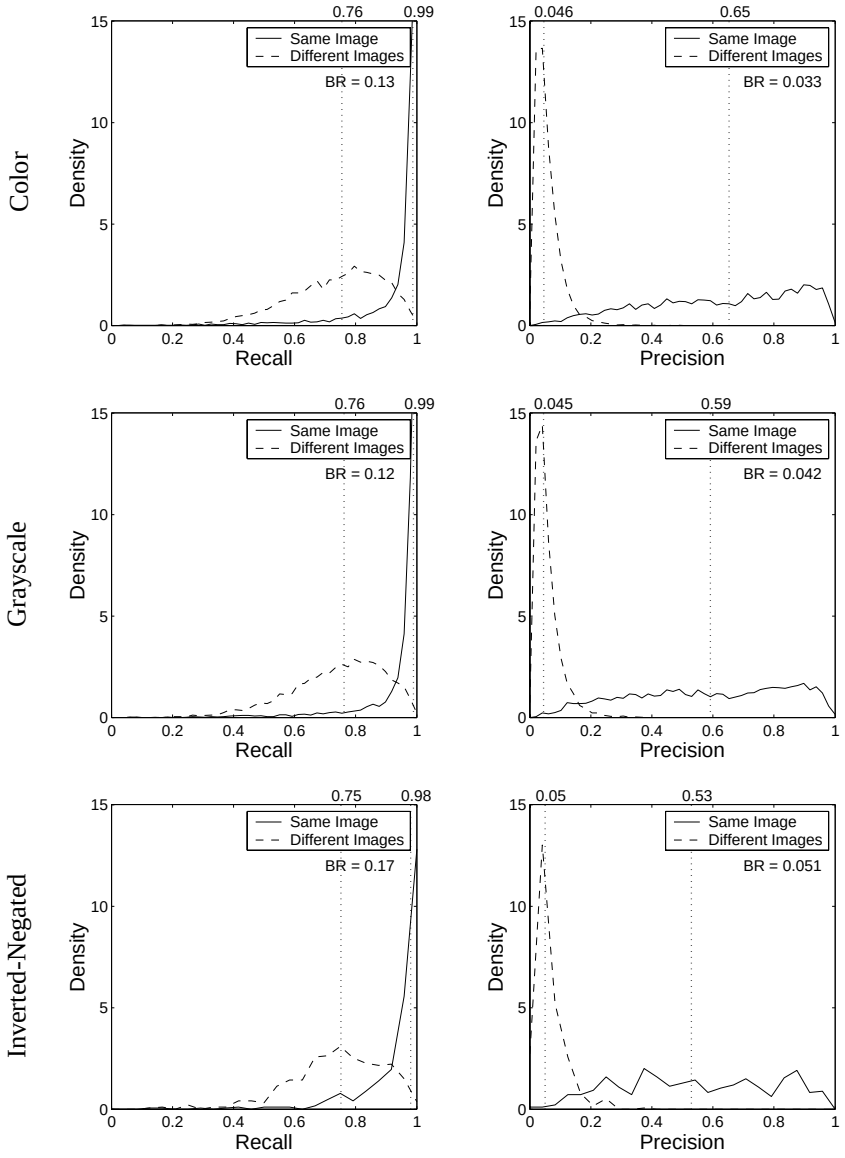


Figure 3.11: Distributions of Leave-One-Out Recall and Precision for Pixel Affinity. The layout is the same as for Figure 3.9, but this figure shows the human segmentation comparisons using the leave-one-out approach instead of the pairwise approach. Both precision and recall increase, though the same-image distributions are shifted to a greater degree, increasing discrimination. The three rows show little change in recall, but a steady decrease in precision. This trend indicates that as the subjects were denied information — fist color, then high-level information — they marked fewer segmentation boundaries, thereby adding more same-segment pairs to the "signal". This is the first time we have seen this trend for the grayscale versus color segmentations in addition to the inverted-negated versus color segmentations.

distributions of F-measure shown in Figure 3.10 also support this conclusion. Note once more the similarity of grayscale and color segmentations, and the measurable difference between inverted-negated and color segmentations.

3.4 Conclusion

In this chapter, we have introduced four new segmentation comparison techniques. The first is a boundary-based approach that directly corresponds boundary elements between one segmentation and a set of ground truth segmentations. The correspondences yield precision and recall measurements that can be combined with the F-measure. The intuitive nature of the correspondence procedure lends confidence to the approach. The other three are region-based approaches, two of which are based on the established statistical methods of mutual information and precision-recall.

If we wish to benchmark segmentation algorithms, then we have many choices. The boundary-based approach can be used to compare the region boundaries to the human data. Alternately, we can use any of the three region-based approaches. As in the popular pixel affinity and pairwise clustering formulation of image segmentation, a segmentation represents a binary-valued affinity function. This affinity function can then be evaluated using either mutual information or the precision-recall framework. Alternately, the BCE measure can be used as a measure of region overlap.

The multiple measures are important for two reasons. First, we must ensure that quantitative evaluations of segmentation algorithms are stable with respect to several measures. At this point, we do not have enough experience evaluating these algorithms to know which measure is preferable. Until that time, quantitative work should use as many different measures as possible.

In addition, the different measures allow us to evaluate different phases of segmentation algorithms, providing a range of "micro-benchmarks." For example, an image segmentation algorithm might take the following approach: (1) Find boundary elements from low-level cues, (2) Perform contour completion on the edge elements, (3) Define a pixel affinity function between pixels, and (4) Produce a segmentation by some pairwise grouping algorithm. We would like to evaluate each stage independently of the others using the segmentation dataset. The boundary-based precision-recall measures can evaluate stages (1) and (2). The mutual information and precision-recall affinity measures can evaluate stage (3), and all four measures can be used for evaluating stage (4).

The remaining chapters begin the project of rigorously optimizing and evaluating each stage of image segmentation. In Chapter 4 we will learn a boundary detector from the dataset, and then use the boundary-based measure to evaluate different algorithms. In Chapter 5 we will use the boundary detector along with other features to learn an affinity function, and evaluate alternatives using the mutual information and precision-recall measures. The evaluation of contour completion

and pairwise clustering algorithms we must leave for future work.

Chapter 4

Learning a Local Boundary Model

The goal of this chapter is to develop an algorithm to accurately detect and localize boundaries in natural scenes using local image measurements. We formulate features that respond to characteristic changes in brightness, color, and texture associated with natural boundaries. In order to combine the information from these features in an optimal way, we train a classifier using human labeled images as ground truth. The output of this classifier provides the posterior probability of a boundary at each image location and orientation. We present precision-recall curves showing that the resulting detector significantly outperforms existing approaches. Our two main results are (1) that cue combination can be performed adequately with a simple linear model, and (2) that a proper treatment of texture is required to detect boundaries in natural images.

4.1 Introduction

Consider the images and human-marked boundaries shown in Figure 4.1. How might we find these boundaries automatically? A complete solution is presently out of reach, since it would require us to model the high-level knowledge applied by the human subjects to find objects and their boundaries. However, general-purpose image segmentation based on low-level information and bottom-up processing can achieve surprisingly good results. We are interesting in pushing the bottom-up approach not because we believe it to be sufficient, but rather because any higher-level algorithm will benefit from gains at the low-level. Whether based on grouping pixels into regions [70, 37] or grouping edge fragments into contours [74, 57], a local boundary model is integral to any perceptual organization algorithm.

The image patches in the first column of Figure 4.2 show what information is available to a local boundary model. Though the patches lack global context, it is clear which contain boundaries and

Figure 4.1: Example Images and Human-Marked Segment Boundaries. Each image shows multiple (4 to 8) human segmentations. The boundaries are darker where more humans marked an edge.

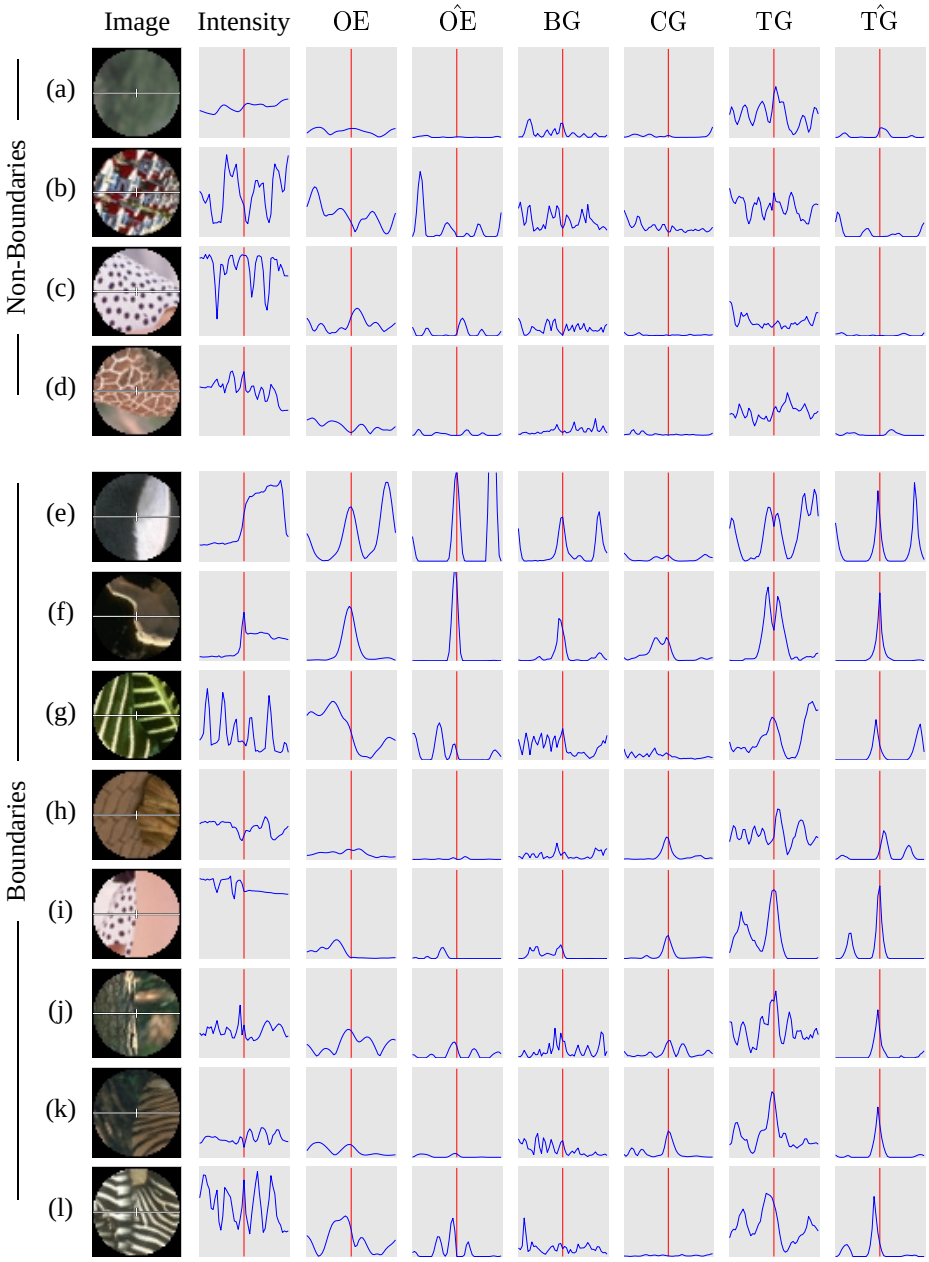


Figure 4.2: Local Image Features. In each row, the first panel shows an image patch. The following panels show feature profiles along the patch's horizontal diameter. The features are raw image intensity, raw oriented energy (OE), localized oriented energy ($\hat{\text{OE}}$), raw brightness gradient (BG), raw color gradient (CG), raw texture gradient (TG), and localized texture gradient ($\hat{\text{TG}}$). The vertical line in each profile marks the patch center. The scale of each feature has been chosen to maximize performance on the set of training images – 2% of the image diagonal (5.7 pixels) for OE, CG, and TG, and 1% of the image diagonal (3 pixels) for BG. The challenge is to combine these features in order to detect and localize boundaries.

which do not. The goal of this chapter is to use features extracted from the image patch to estimate the posterior probability of a boundary passing through the center point.

The traditional approach to this problem is to look for discontinuities in image brightness. For example, the widely employed Canny detector [6] models boundaries as brightness step edges. The second column of Figure 4.2 show that this is an inadequate model for boundaries in natural images where texture is a ubiquitous phenomenon. The Canny detector fires wildly inside textured regions where high-contrast edges are present, but no region boundary exists. In addition, it is unable to detect the boundary between textured regions when there is only a subtle change in average image brightness. Feature detectors based on the spatially averaged second moment matrix provide a partial solution to the problem of texture. The eigenspectrum of the matrix can distinguish simple edges from the multiple incident edges that can occur inside texture. Though this approach will suppress false positives in a limited class of textures, it will also suppress corners and texture edges.

These significant problems with simple boundary models have lead researchers to develop more complex boundary detectors that address the problems presented by texture, e.g. [60, 73]. Although these work well on the pure texture boundaries provided by synthetic Brodatz mosaics, they have problems in the vicinity of simple brightness edges. Texture descriptors over local windows that straddle a boundary have different statistics from windows contained in either of the neighboring regions, and different statistics from the window centered on the boundary. These differences inevitably result in either doubly-detected boundaries or thin halo-like regions along contours, for example see [60, 8, 24]. Just as a brightness edge model does not detect texture boundaries, a pure texture model does not detect brightness edges effectively.

Clearly, boundaries in natural images can be marked by changes in any of several cues including brightness, color, and texture. Evidence from psychophysics [59] suggests that humans combine multiple cues to improve their detection and localization of boundaries. There has been limited work in computational vision on addressing the difficult problem of cue combination. For example, Malik et al. [37] associate a measure of texturedness with each point in an image in order to suppress contour processing in textured regions and vice versa. However, their solution is full of ad-hoc design decisions and manual parameter settings.

In this chapter, we provide a more principled approach to cue combination by framing the task as a supervised learning problem. The dataset presented in Chapter 2 provides the ground truth label for each pixel as being on- or off-boundary. The task is then to model the probability of a pixel being on-boundary conditioned on some set of local image features. This quantitative approach to learning and evaluating boundary detectors is similar to the work of Konishi et al. [31] using the Sowerby dataset of English countryside scenes. Our work is distinguished by an explicit treatment of texture, enabling superior performance on a more diverse collection of natural images.

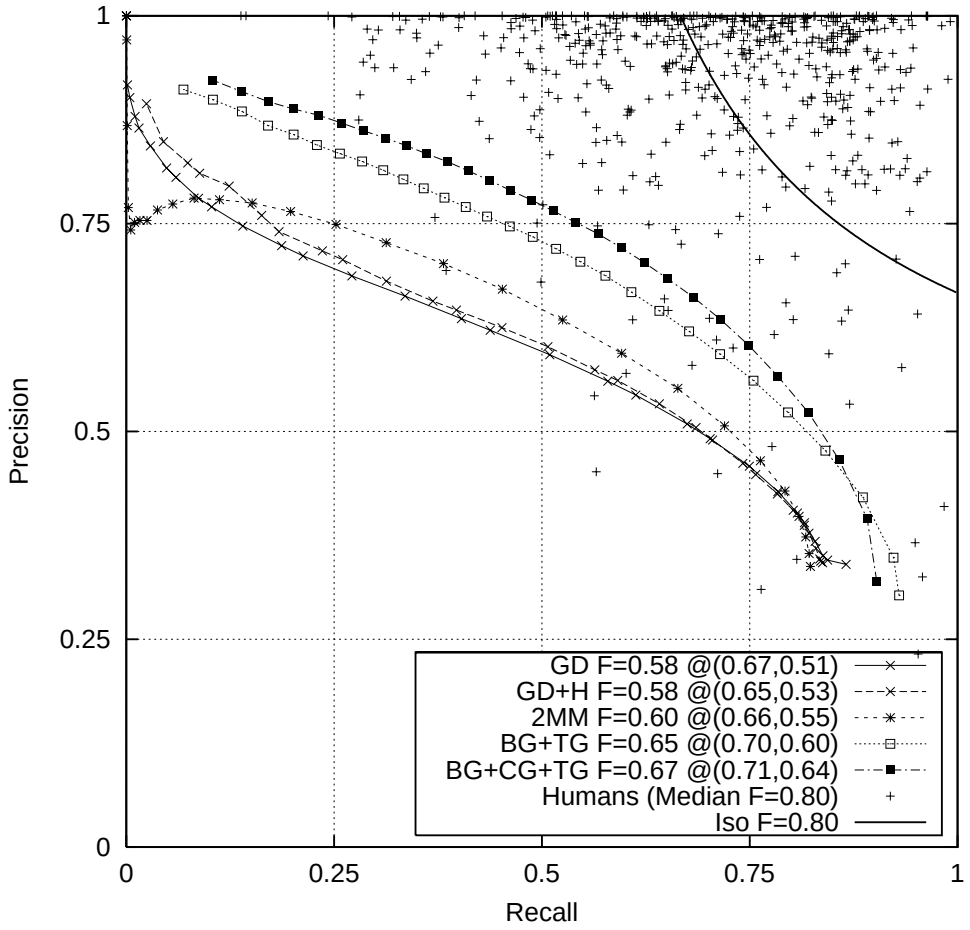


Figure 4.3: Two Decades of Boundary Detection Research. The performance of our boundary detector compared to classical boundary detection methods and to the human subjects' performance. *Precision-recall* curves, described in Section 3.1.2, are shown for five boundary detectors: (1) Gaussian derivative (GD), (2) Gaussian derivative with hysteresis thresholding (GD+H), the Canny detector, (3) A detector based on the second moment matrix (2MM), (4) our grayscale detector that combines brightness and texture (BG+TG), and (5) our color detector that combines brightness, color, and texture (BG+CG+TG). Each detector's precision-recall curve shows the tradeoff between accuracy and noise as the detector's threshold varies. Shown in the caption is each curve's maximal F-measure, valued from zero to one, along with the coordinates of the location of the maximum. Thus, the notation "F=0.67@(0.71,0.64)" means that the maximal F-measure of 67% occurs at 71% recall and 64% precision. Defined in Section 3.1.2, the F-measure is a summary statistic for a precision-recall curve. The points on the plot marked with "+" show the precision and recall of each ground truth human segmentation when compared to the other humans. The median F measure for the human subjects is 0.80. The solid curve shows the F=0.80 precision-recall curve, representing the frontier of human performance for this task.

By modeling texture and combining various local cues in a statistically optimal manner, we demonstrate a marked improvement over the state-of-the-art in boundary detection. Figure 4.3 shows the performance of our detector compared to the Canny detector, a detector based on the second moment matrix used by Konishi et al. [31], and the human subjects. The remainder of the chapter will show how this improvement was achieved. In Section 4.2 we describe the local brightness, color, and texture features used as input to our algorithm. Section 4.3 presents our training and testing methodology and the dataset of 12,000 human segmentations that provide the ground truth data. We apply this methodology in Section 4.4 to optimize each local cue independently, and in Section 4.5 to perform cue combination. Section 4.6 presents a quantitative comparison of our method to existing boundary detection methods. We conclude in Section 4.7.

4.2 Image Features

Our approach to boundary detection is to look at each pixel for local discontinuities in several feature channels, over a range of orientations and scales. We will consider two brightness features (oriented energy and brightness gradient), one color feature (color gradient), and one texture feature (texture gradient). Each feature has free parameters that we will learn from the training data.

4.2.1 Oriented Energy

In natural images, brightness edges are more than simple steps. Phenomena such as specularities, mutual illumination, and shading result in composite intensity profiles consisting of steps, peaks, and roofs. The oriented energy (OE) approach [46] can be used to detect and localize these composite edges [53]. OE is defined as:

$$\mathrm{OE}_{\theta,\sigma} = (I * f^e_{\theta,\sigma})^2 + (I * f^o_{\theta,\sigma})^2 \tag{4.1}$$

where $f^e_{\theta,\sigma}$ and $f^o_{\theta,\sigma}$ are a quadrature pair of even- and odd-symmetric filters at orientation θ and scale σ. Our even-symmetric filter is a Gaussian second-derivative, and the corresponding odd-symmetric filter is its Hilbert transform. $\mathrm{OE}_{\theta,\sigma}$ has maximum response for contours at orientation θ. The filters are elongated by a ratio of 3:1 along the putative boundary direction.

4.2.2 Gradient-Based Features

We include the oriented energy feature in our analysis because it is the standard means of detecting brightness edges in images. For more complex features, we introduce a gradient-based paradigm that we use for detecting local changes in color and texture, as well as brightness.

The approach is simple. A gradient function $G(x, y, \theta, r)$ is defined at a location (x, y) in the image, at an orientation θ, and at a scale r. Draw a circle of radius r, centered at (x, y), and divide it along the diameter at orientation θ. The gradient function compares the contents of the two resulting disc halves. A large difference between the disc halves indicates a discontinuity in the image along the disc's diameter.

How shall we describe and compare the two half-disc regions for each cue? Common to successful color and texture features in the literature is the approach of estimating the empirical distribution of pixel-based features over some region. Distributions of color in perceptual color spaces have been used successfully as region descriptors in the QBIC [48] and Blobworld [8] image retrieval systems. In addition, the compass operator of Ruzon and Tomasi [63, 64] uses color histogram comparisons to find corners and edges in color images. For texture analysis, there is an emerging consensus that an image should first be convolved with a bank of filters tuned to various orientations and spatial frequencies [14, 38]. The empirical distribution of filter responses has been shown to be powerful both for both texture synthesis [23] and texture discrimination [55].

For brightness and color gradients, we bin kernel density estimates of the distributions of pixel luminance and chrominance in each disc half. For the texture gradient, we compute histograms of vector quantized filter outputs in each disc half. In all three cases, the half-disc regions are described by histograms, which we compare with the χ^2 histogram difference operator [56]:

$$\chi^2(g, h) = \frac{1}{2} \sum \frac{(g_i - h_i)^2}{g_i + h_i} \tag{4.2}$$

Each gradient computation shares the step of computing a histogram difference at multiple orientations and scales at each pixel. The naïve implementation would involve much redundant computation. Appendix A presents efficient algorithms for computing the gradient features.

4.2.3 Brightness and Color Gradient Details

There are two common approaches to characterizing the difference between the color distributions for two sets of pixels. The first is based on density estimation using histograms. Both QBIC and Blobworld use fully three dimensional color histograms as region features, and compare histograms using a similarity measure such as L^1 norm, χ^2 difference, or the quadratic form distance function. Blobworld smoothes the histograms to prevent the aliasing of similar colors, while QBIC models the perceptual distance between bins explicitly.[1] For an image region with thousands of pixels, the

[1]The quadratic form distance function used in QBIC is $d(g, h) = (g - h)^T A(g - h)$, where g and h are the histograms to compare, and A is a matrix giving the similarity A_{ij} between two bins i and j. Niblack et al. indicate that this measure is superior for their task. We will not consider this histogram similarity function because it is computationally expensive, difficult to define A, and similar in spirit to the Earth Mover's distance.

full joint color distribution can be reasonably estimated with some smoothing, though the problem of defining a perceptually meaningful histogram difference operator is significant.

The second common approach avoids quantization artifacts by using the *Mallows* [40] or *Earth Mover's distance* (EMD) [36] to compare color distributions. This is potentially preferable for a local edge operator where one does not have a large number of samples from which to estimate distributions. In addition, the EMD explicitly accounts for the distance between points in the color space. If the space is a perceptual space where Euclidean distances correspond to perceptual distances, then one would imagine the EMD to be the ideal measure. Perceptual color spaces, however, have notorious problems with regard to perceptual uniformity: though small distances in a space such as CIELAB map well to perceptual distance, large distances are not meaningful in the sense that colors beyond some separation are perceptually equivalently different. Ruzon and Tomasi use an attenuated EMD to model this perceptual roll-off, but the EMD remains computationally expensive. For one-dimensional data, the it can be computed relatively efficiently by simply sorting the points. In higher dimensions, however, one must solve an assignment problem that increases complexity considerably.

We would like an approach that models the the color distribution accurately with respect to *human perception*, while remaining computationally simple. Our approach is based on binning kernel density estimates of the color distribution in CIELAB using a Gaussian kernel, and comparing histograms with the χ^2 difference. The χ^2 histogram difference does not make use of the perceptual distance between bin centers. Therefore, without smoothing, perceptually similar colors can produce disproportionately large χ^2 differences. Because the distance between points in CIELAB space is perceptually meaningful in a local neighborhood, binning a kernel density estimate whose kernel bandwidth σ matches the scale of this neighborhood means that perceptually similar colors will have similar histogram contributions. Beyond this scale, where color differences are perceptually incomparable, χ^2 will regard them as equally different. The combination of a kernel density estimate in CIELAB with the χ^2 histogram difference is a good match to the structure of human color perception.

For the brightness gradient we will compute histograms of L* values, which is a simple one-dimensional density estimation problem. The color gradient presents additional challenges because the pixel values are in the 2D space of a* and b*. We can apply the same methodology that we use for L*, using 2D kernels and 2D histograms to produce the joint color gradient CG^{ab}. However, in order to keep the computational cost reasonable, one must reduce both the number of kernel samples and the number of bins. This reduction compromises the quality of the density estimate, so instead we compute the marginal color gradients for a* and b* and take the full color gradient to be the sum of the corresponding marginal gradients: $\mathrm{CG}^{a+b} = \mathrm{CG}^{a} + \mathrm{CG}^{b}$. This simplification is

motivated not only by the much higher cost of computing CG^{ab}: The a* and b* axes correspond to the perceptually orthogonal red-green and yellow-blue color opponents found in the human visual system (see Palmer [50]). Section 4.4 presents the comparison of CG^{ab} to CG^{a+b}.

4.2.4 Texture Gradient Details

Analogous to the brightness and color gradient operators, we formulate a directional operator that measures the degree to which texture of scale r varies at an image location (x, y) in direction θ. Our texture gradient (TG) operator computes the texture dissimilarity in the two halves of a disk of radius r, centered on (x, y), and divided in two along a diameter at orientation θ. Oriented texture processing along these lines has been pursued by Rubner and Tomasi [60].

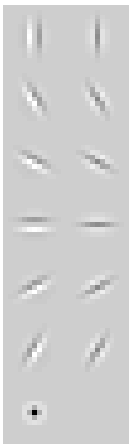
Figure 4.4a shows the filter bank that we use for texture processing. It contains six pairs of elongated, oriented filters, as well as a center-surround filter. The oriented filters are in even/odd quadrature pairs, and are the same filters we used to compute oriented energy. The even-symmetric filter is a Gaussian second derivative, and the odd-symmetric filter is its Hilbert transform. The center-surround filter is a difference of Gaussians. To each pixel we associate the vector of filter responses centered at the pixel. Note that unlike [37], we do not contrast-normalize the filter responses via Weber's Law for texture processing. Our experiments indicate that normalization does not improve performance, as it appears to amplify noise more than signal.

Each disc half contains a set of filter response vectors which we can imagine as two clouds of points in a feature space with dimensionality equal to the number of filters. One can use the empirical distributions of these point clouds as the texture descriptors, and then compare the descriptors to get the value of the texture gradient.

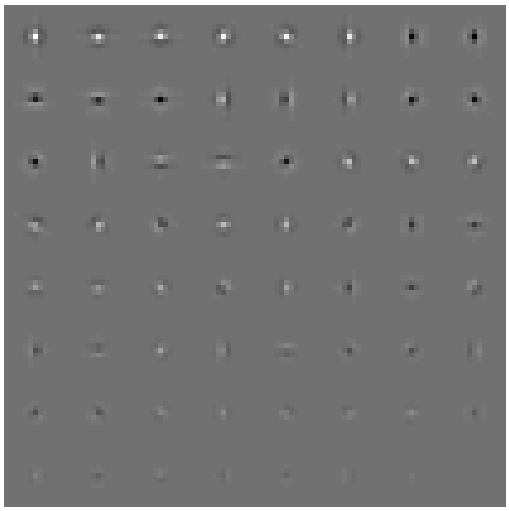
Many questions arise regarding the details of this approach. Should the filter bank contain multiple scales, and what should the scales be? How should we compare the distributions of filter responses? Should we use the Earth Mover's distance, or should we estimate the distributions? If the latter, should we estimate marginal or joint distributions, and with fixed or adaptive bins? How should we compare distributions: using some L^p-norm or the χ^2 distance? Puzicha et al. [55] evaluate a wide range of texture descriptors in this framework and examine many of these questions. We choose the approach developed by Malik et al., which is based on the idea of *textons*.

The texton approach is one that estimates the joint distribution of filter responses using adaptive bins. The filter response vectors are clustered using k-means. The clusters define Voronoi bins, and the cluster means – the *textons* – define texture primitives. The space covered by a texton's bin is called a *texton channel*. These texture primitives are simply linear combinations of the filters. Figure 4.4b shows example textons for $k = 64$ computed over the 200 images in the training set. After the textons have been defined, each pixel is assigned to the nearest texton. The texture dis-

(a) Filter Bank



(b) Universal Textons



(c) Image

(d) Texton Map

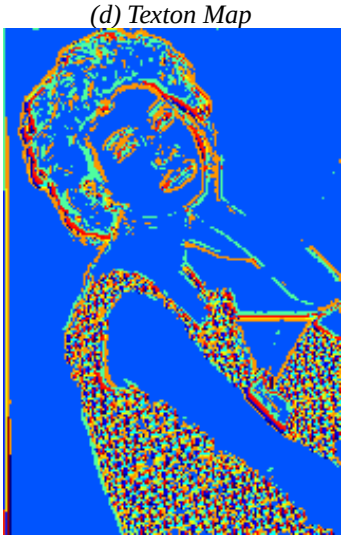


Figure 4.4: Textons. (a) The 13-element filter bank used for computing textons. (b) Example universal textons computed from the 200 training images, sorted by L1 norm for display purposes. (c) An image and (d) its associated texton map. Texton quality is best with a single scale filter bank containing small filters. Each pixel produces a 13-element response to the filter bank, and these responses are clustered with k-means. In this example, using 200 images with k=64 yields 64 universal textons. The textons identify basic structures such as steps, bars, and corners at various levels of intensity. If each pixel in the image shown in (c) is assigned to the nearest texton, and each texton is assigned a color, we obtain the texton map shown in (d). The elongated filters have 3:1 aspect, and the longer σ was set to 0.7% of the image diagonal (about 2 pixels).

similarities can then be computed by comparing the histograms of textons in the two disc halves. Figure 4.4 shows an image and the associated *texton map*, after each pixel is mapped to its texton channel. Some issues remain, namely which images to use to compute the textons, the choice of k, the procedure for computing the histograms, and the histogram comparison measure.

For computing textons, we can use a large image set such as our training set to compute *universal textons* that we will use for all test images. Alternately, we can compute *image-specific textons* by computing the textons separately for each test image. As for the choice of k, we will see that it depends on this choice between universal and image-specific textons as well as the scale r of the texture gradient. Section 4.4 presents experiments exploring both of these issues.

For computing the texton histograms, we use hard binning without smoothing. One can do soft binning by considering a pixel's distance to each texton. However, this type of soft binning is computationally expensive, and in our experiments, it has not proved worthwhile. The hard binning is not a problem because adjacent pixels have correlated filter responses due to the spatial extent of the filters. Consequently, the data is already somewhat smoothed, and pixels in a disc are likely to cover fewer bins ensuring more samples per bin.

Finally, the χ^2 distance is not the only viable measure of histogram distance for this task. Both Puzicha et al. [56] and Levina [35] evaluate various methods for comparing texture distributions, including L1 norm, χ^2 distance, and the Mallows or Earth Mover's distance. The optimal distance measure, however, depends on the task (matching or discrimination) and on the images used (Brodatz patches or natural images). Our experiments show that for local boundary detection in natural images, the χ^2 distance is marginally superior to the L1 norm, and significantly better than the Mallows distance.

4.2.5 Localization

The underlying function of boundary existence that we are trying to learn is tightly peaked around the location of image boundaries marked by humans. In contrast, Figure 4.2 on page 63 shows that the features we have discussed so far don't have this structure. As they pool information over some support, they produce smooth, spatially extended outputs. Since each pixel is classified independently, our spatially extended features are problematic for a classifier, as both on-boundary pixels and nearby off-boundary pixels will have large gradient values.

The texture gradient is particularly prone to this effect due to its large support. In addition, the TG produces multiple detections in the vicinity of brightness edges. The bands of textons present along such edges often produces a larger TG response on each side of the edge than directly on the edge. This double-peak problem is ubiquitous in texture edge detection and segmentation work [60, 8, 24], where it produces double detections of edges and sliver regions along region boundaries.

However, we are aware of no work that addresses this serious problem. Non-maxima suppression is typically used to narrow extended responses, but the multiple detection problem requires a more general solution. We will exploit the symmetric nature of the texture gradient response both to localize the edge accurately and to eliminate the double detections.

In order to make the spatial structure of boundaries available to the classifier, we transform the raw feature signals to emphasize local maxima. However, we must do this in a manner that smoothes out multiple detections. Given a feature $f(x)$ defined over spatial coordinate x orthogonal to the edge orientation, consider the derived feature $\hat{f}(x) = f(x)/d(x)$, where $d(x) = -|f'(x)|/f''(x)$ is the first-order approximation of the distance to the nearest maximum of $f(x)$. We use the smoothed and stabilized version:

$$\hat{f}(x) = \tilde{f}(x) \cdot \left(\frac{-f''(x)}{|f'(x)| + \epsilon} \right) \tag{4.3}$$

with ϵ chosen to optimize the performance of the feature. By incorporating the $1/d(x)$ localization term, $\hat{f}(x)$ will have narrower peaks than the raw $f(x)$. The $\tilde{f}(x)$ is an estimate of the underlying smooth "pure-texture" signal corrected for the double peaks. To robustly estimate the directional derivatives and the smoothed signal, we fit a cylindrical parabola over a circular window of radius r centered at each pixel. The least squares parabolic fit $ax^2 + bx + c$ provides directly the signal derivatives as $f''(x) = 2a$ and $f'(x) = b$, as well as $\tilde{f}(x) = c$. Thus, the localization function becomes $\hat{f} = -(2c^+a^+)/(|b| + \epsilon)$, where c and a require half-wave rectification.[2]

The last two columns of Figure 4.2 on page 63 show the result of applying this transformation to the texture gradient, which clearly has the effect of reducing noise, tightly localizing the boundaries, and coalescing double detections. We found that the localization procedure does not improve the brightness and color gradient features, and so our final feature set consists of $\{\hat{OE}, BG, CG, \hat{TG}\}$, each at 8 orientations and 3 half-octave scales.

4.3 Evaluation Methodology

Our system will ultimately combine the cues of the previous section into a single function $P_b(x, y, \theta)$ which gives the posterior probability of a boundary at each pixel (x, y) and orientation θ. In order to optimize this system and compare it to other systems, we need a methodology for judging the quality of a boundary detector. We formulate the boundary-detection task as a classification problem of discriminating non-boundary from boundary pixels, and apply the *precision-recall* framework of Chapter 3 using human-marked boundaries as ground truth. The segmentation dataset contains 5-10

[2] Windowed parabolic fitting is known as 2nd-order Savitsky-Golay filtering, or LOESS smoothing. We also considered Gaussian derivative filters $\{G_r, G'_r, G''_r\}$ to estimate $\{f_r, f'_r, f''_r\}$ with similar results.

segmentations for each of 1000 images. We use 200 images and associated segmentations as the training data, and a different 100 images and associated segmentations as the test dataset.

Precision-recall curves were first used for evaluating edge detectors by Abdou and Pratt [1]. A similar approach was taken by Bowyer et al. [3] for boundary detector evaluation with ROC curves. Other methods of evaluating boundary detectors in a quantitative framework exist, such as the Chernoff information used by Konishi et al. [31]. Though the information theoretic approach approach can lead to a useful method for ranking algorithms relative to one another, the resulting measure is not intuitive.

The precision-recall framework does produce effective and intuitive measures of performance. Furthermore, it is more appropriate than the related ROC curve or Bayes risk framework for this task since our two classes — boundary and non-boundary — are unbalanced. Both ROC curves and Bayes risk use the overall classification error, which includes the errors made by the detector for both classes. Since there are many more non-boundaries than boundaries, and since non-boundaries are more easily detected than boundaries, the ROC approach exaggerates performance. In contrast, the precision-recall framework focuses on the rarer class, and does not reward the correct detection of non-boundaries.

Remember that precision is the probability that the detector's signal is valid, and recall is the probability that the ground truth data was detected. These two measures are particularly meaningful in the context of boundary detection when we consider applications that make use of boundary maps, such as stereo or object recognition. It is reasonable to characterize these algorithms in terms of how much true signal is required to succeed (recall), and how much noise can be tolerated (precision).

Precision and recall are appealing measures, but to compute them we must determine which true positives are correctly detected, and which detections are false. Each point on the precision-recall curve is computed from the detector's output at a particular threshold. In addition, we have binary boundary maps as ground truth from the human subjects. For the moment, let us consider how to compute the precision and recall of a single thresholded machine boundary map given a single human boundary map. One could simply correspond coincident boundary pixels and declare all unmatched pixels to be either false positives or misses. This approach would not tolerate any localization error, however, and would consequently be a poor indicator of performance. From Figure 4.1 on page 62, it is clear that the assignment of machine boundary pixels to ground truth boundaries must tolerate localization errors, since the ground truth data contains boundary localization errors.

The approach of Martin et al. [42] is to add a modicum of slop to the naïve correspondence procedure in order to permit small localization errors at the cost of permitting multiple detections. However, an *explicit* correspondence of machine and human boundary pixels is the only way to robustly count the hits, misses, and false positives that we need to compute precision and recall.

In particular, it is important to compute the correspondence explicitly in order to penalize multiple detections, single detection being one of the three goals of boundary detection formalized in Canny's work [6] along with good detection and good localization.

Chapter 3 details the correspondence computation, which provides us the means of computing the precision and recall for a single human segmentation. The segmentation dataset, however, provides multiple human segmentations for each image, so that the ground truth is defined by a collection of 5 to 10 human segmentations. Simply unioning the humans' boundary maps is not effective because of the localization errors present in the dataset itself. The proper way to combine the human boundary maps would likely require additional correspondences, or perhaps the estimation of a generative model specifying the humans' detection and localization error processes along with the hidden true signal.

Fortunately, we are able to finesse these issues in the following manner. First, we correspond the machine boundary map with each human map in turn. Only those machine boundary pixels that match no human boundary are counted as false positives, so that the fraction of machine boundary pixels that are false positives gives the precision. The hit rate is simply averaged over the different humans, so that to achieve perfect recall the machine boundary map must explain all of the human data up to some tolerance d_{max}. Our intention is that this approach to estimating precision and recall matches as closely as possible the approach one would take if one were to compute them visually. All three desirable properties of a boundary detector – detection, localization, single detection – are encouraged by the method, and the results are splendid.

In summary, we have a method for describing the quality of a boundary detector that produces soft boundary maps of the form $P_b(x, y)$ or $P_b(x, y, \theta)$. For the latter, we take the maximum over θ to reduce the case to the former. Given the soft boundary image $P_b(x, y)$, we produce a precision-recall curve. Each point on the curve is computed independently by first thresholding P_b to produce a binary boundary map, and then matching this machine boundary map against each of the human boundary maps in the ground truth segmentation dataset. The precision-recall curve is a rich descriptor of performance. When a single performance value is required or is sufficient, precision and recall can be combined with the F-measure. The F-measure curve is usually unimodal, so the curve's maximal F-measure may be reported as a summary of the detector's performance. We now turn to applying this evaluation method to optimizing our boundary detector and comparing our approach to the standard methods.

4.4 Cue Optimization

Before combining the brightness, color, and texture cues into a single detector, we first optimize each cue individually. By applying coordinate ascent with high precision and recall as the objective, we can optimize each cue with respect to the ground truth dataset so that no change in any single parameter improves performance. I do not present the complete set of experiments, but only those that afford interesting observations.

Each of the four cues – oriented energy (OE), brightness gradient (BG), color gradient (CG), and texture gradient (TG) – has a scale parameter. In the case of OE, the scale σ is the bandwidth of the quadrature filter pair. For the others, the scale r is the radius of the disc. We determined the optimal one octave range for each cue. In units of percentage of the image diagonal, the ranges are 1.4% to 2.8% for OE, CG, and TG, and 0.75% to 1.5% for BG. These scales are optimal, independent of whether or not we use the localization procedure of Section 4.2.5. The middle scale always performs best, except in the case of raw OE where the largest scale is superior.

Figures 4.5 and 4.6 shows the precision-recall (PR) curves for each cue at the optimal scales both with and without localization applied. In addition, each plot shows the PR curve for the combination of the three scales. Each curve requires a $P_b(x, y)$ function that is obtained from fitting a logistic model to the training dataset. We evaluate the P_b function on the test set to produce the $P_b(x, y)$ images from which the curve is generated.

The ϵ for each cue's localization function was optimized separately to 0.01 for TG and 0.1 for all other cues. The figures show that localization is not required for BG and CG, but helpful for both OE and TG. The localization function has two potential benefits. It narrows peaks in the signal, and it merges multiple detections. From Figure 4.2 on page 63, we see that the scale of OE is rather large so that localization is effective at narrowing the wide response. TG suffers from both multiple detections and a wide response, both of which are ameliorated by the localization procedure. Neither BG nor CG require either alteration offered by the localization procedure, partially because they have other parameters that we used to optimize their responses.

Figure 4.7 shows our optimization of the kernel size used in the density estimation computations for BG and CG. For these features, we compare the distributions of pixel values in two half discs, whether those values are brightness (L*) or color (a*b*). First, consider the color gradient $CG^{\,a+b}$ computed over the marginal distributions of a* and b*. With a disc radius ranging from 4 to 8 pixels, kernels are critical in obtaining low-variance estimates of the distributions. In the figure, we vary the Gaussian kernel's sigma from 1.25% to 40% of the diameter of the domain. In addition, the number of bins was varied inversely in order to keep the number of samples per bin constant, and at a minimum of two per bin. The kernel was clipped at 2σ and sampled at 23 points. The dominant PR curve on each plot indicates that the optimal parameter for BG is $\sigma = 0.2$ (with 12 bins) and

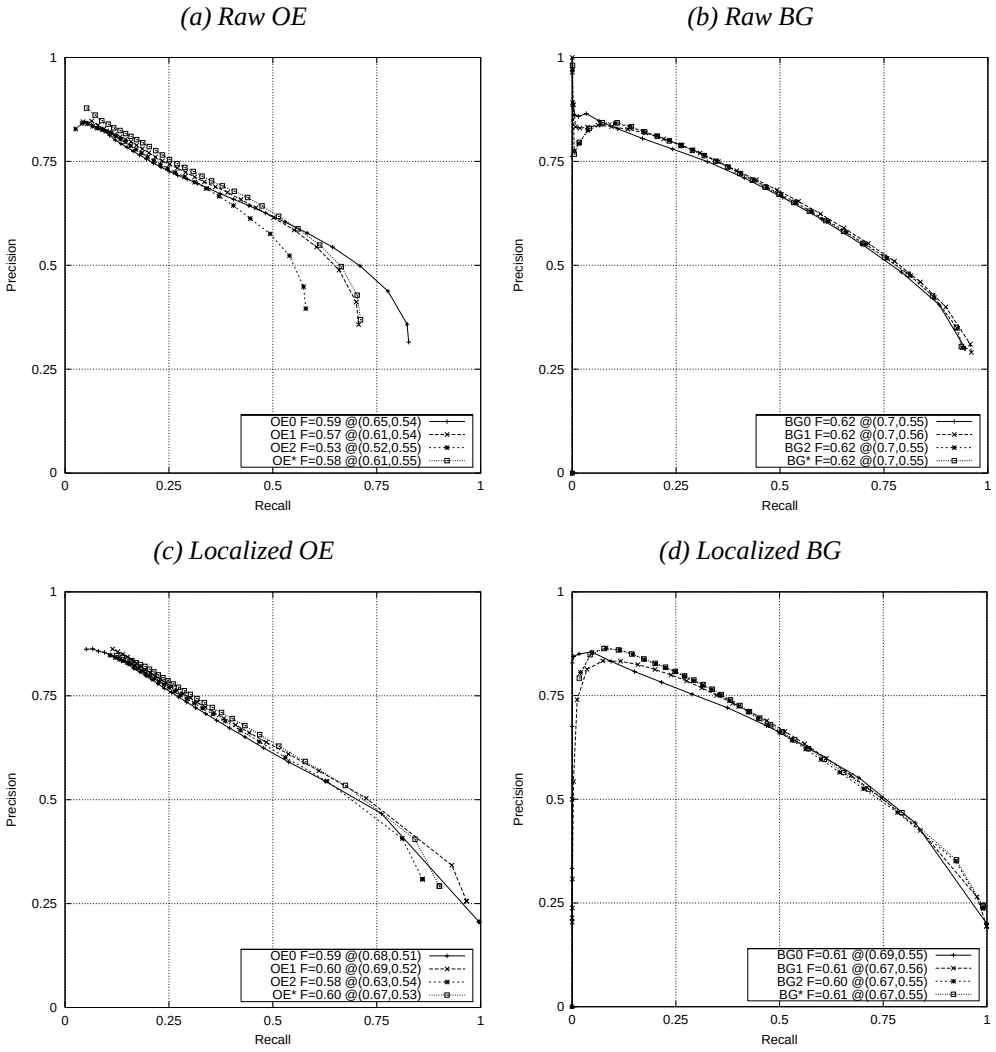


Figure 4.5: Boundary Localization for OE and BG. Performance of raw features (top row) and localized features (bottom row). Curves toward the top (lower noise) and right (more recovered signal) are better. Each curve is parameterized by P_b and is scored by its maximal F-measure, the value and location of which are shown in the legend. Each panel in this figure shows four curves: one curve for each of three half-octave spaced scales of the feature, along with one curve for the combination of the three scales. The starting scales for OE and BG are 1.4% and 0.75% of the image diagonal, respectively. With the exception of Figure 4.12, we use logistic regression to model P_b. In this figure, we see that the localization procedure is marginally helpful for OE, and unnecessary for BG. Multiple scales help neither OE nor BG. Note that each feature's ϵ and scale parameters were optimized against the training set using the precision-recall methodology. Figure 4.6 shows the same graphs for CG and TG.

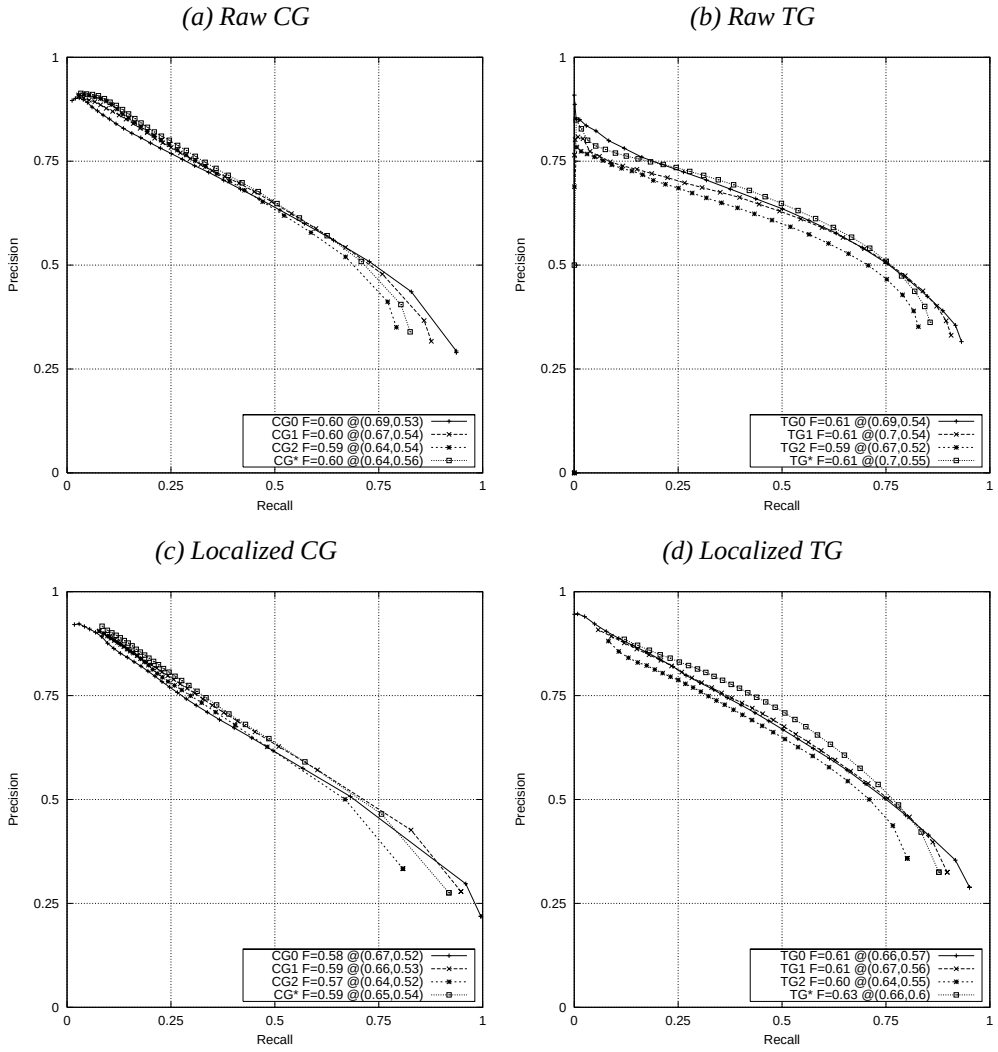
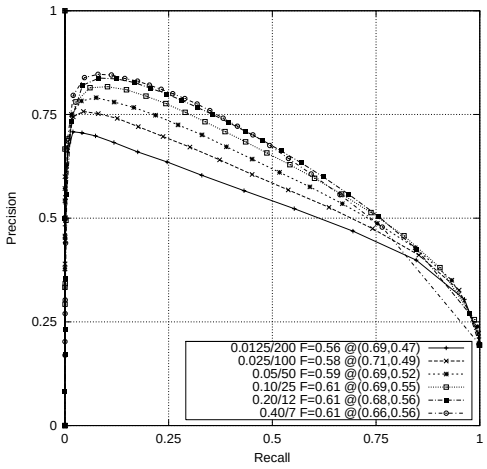


Figure 4.6: Boundary Localization for CG and TG. Performance of raw features (top row) and localized features (bottom row). The layout is identical to Figure 4.5, which shows the curves for OE and BG. The starting scales for CG and TG are both 1.4% of the image diagonal. In this figure, we see that the localization procedure is unnecessary for CG, but extremely helpful for TG. The performance gain for TG is due to the elimination of double-detections along with good localization, as is evident from Figure 4.2 on page 63. In addition, TG is the only feature for which there is benefit from combining scales. Note that each feature's ϵ and scale parameters were optimized against the training set.

(a) Brightness Gradient



(b) Color Gradient

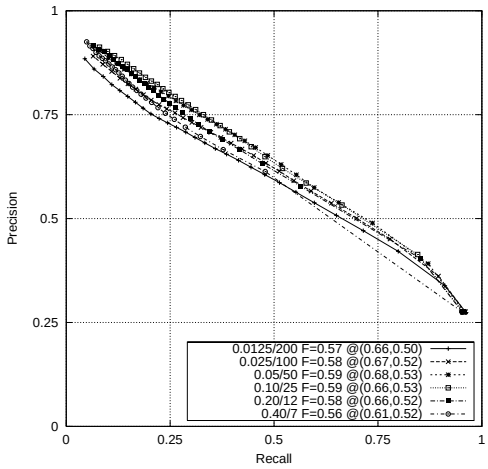


Figure 4.7: Bandwidth for BG and CG Kernel Density Estimates. Both BG and CG operate by comparing the distributions of 1976 CIE L*a*b* pixel values in each half of a disc. We estimate the 1D distributions of L*, a*, and b* with histograms of kernel density estimates. Each curve is labeled with σ and bin count. The accessible ranges of L*, a*, and b* are scaled to $[0, 1]$. The kernel was clipped at 2σ and sampled at 23 points. The bin count was adjusted so that there would be no fewer than 2 samples per bin. The best values are $\sigma = 0.2$ for BG (12 bins), and $\sigma = 0.1$ for CG (25 bins).

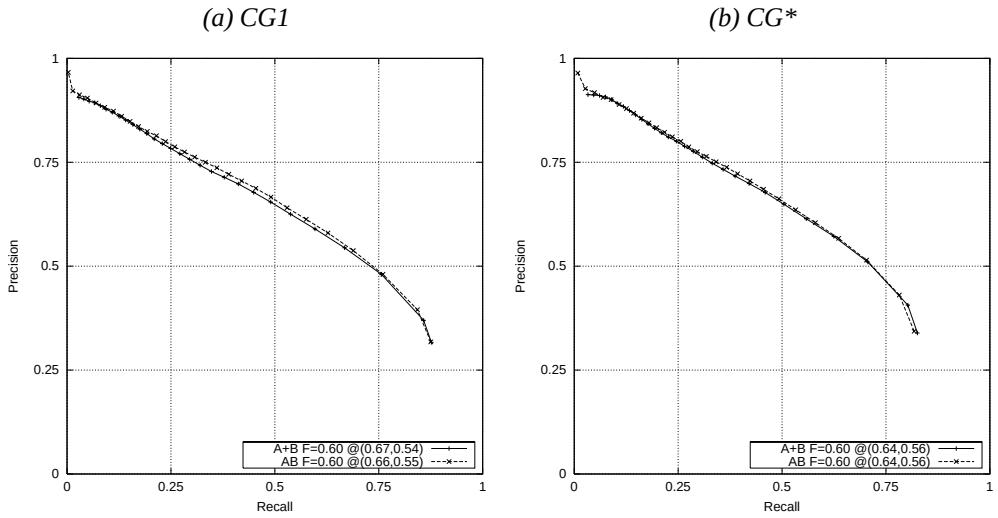


Figure 4.8: Marginal vs. Joint Estimates of CG. (a) shows the middle scale of the color gradient, and (b) shows the three scales combined. Our inclination in estimating pixel color distributions was to estimate the 2D joint distribution of a* and b*. However, the 2D kernel density estimation proved to be computationally expensive. Since the a* and b* axes in the 1976 CIE L*a*b* color space were designed to mimic the blue-yellow green-red color opponency found the human visual cortex, one might expect the joint color distribution to contain little *perceptual* information not present in the marginal distributions of a* and b*. The curves labeled "AB" show the color gradient computed using the joint histogram (CG^{ab}); the curves labeled "A+B" show the color gradient computed computed as $(\mathrm{CG}^{a} + \mathrm{CG}^{b})$. The number of bins in each dimension is 25 for both experiments, so that the CG^{ab} computation requires 25x more bins and 25x the compute time. The cue quality is virtually identical, and so we adopt the marginal CG approach.

$\sigma = 0.1$ for CG (with 25 bins).

The experiments in Figure 4.7 used the separated version of the color gradient CG^{a+b} rather than the joint version CG^{ab}. Figure 4.8 shows the comparison between these two methods of computing the color gradient. Whether using a single scale of CG or multiple scales, the difference between CG^{a+b} and CG^{ab} is minimal. The joint approach is far more expensive computationally due to the additional dimension in the kernels and histograms. The number of bins in each dimension was kept constant at 25 for the comparison, so the computational costs differed by 25x, requiring tens of minutes for CG^{ab}. If computational expense is kept constant, then the marginal method is superior because of the higher resolution afforded in the density estimate. In all cases, the marginal approach to computing the color gradient is preferable.

The texture gradient cue also has some additional parameters beyond r and ϵ to tune, related to the texture representation and comparison. The purpose of TG is to quantify the difference in

the distribution of filter responses in the two disc halves. Many questions arise and are discussed in Section 4.2.4. For filters, we use the same even and odd-symmetric filters that use for oriented energy – a second derivative Gaussian and its Hilbert transform – at six orientations along with a center-surround DOG. We experimented with multi-scale filter banks, but found agreement with Levina [35] that a single-scale filter bank at the smallest scale was preferable. Figure 4.4(a) on page 70 shows the filter bank we used for texture estimation. As for distribution estimation issues, we follow the *texton* approach of Malik et al. [37] which estimates the joint distribution with adaptive bins by clustering the filter responses with k-means, and compares histograms using the χ^2 measure. We verified that none of L^1, L^2, or L^∞ norm performs better. In addition, we determined that the Mallows distance computed on the marginal raw filter outputs performed poorly. The Mallows distance on the joint distribution is computationally infeasible, requiring the solution to an assignment problem.

After settling on the approach of comparing texton histograms with the χ^2 distance measure, two issues remain. First, we must choose between image-specific and universal textons. For image-specific textons, we recompute the adaptive texton bins for each test image separately. For universal textons, we compute a standard set of textons from the 200 training images. The computational cost of each approach is approximately equal, since the per-image k-means problems are small, and one can use fewer textons in the image-specific case. Figure 4.4(b) shows universal textons for $k = 64$. The second remaining issue is that we must choose the number of textons, or the k parameter for k-means.

Figure 4.9 shows experiments optimizing the number of textons, and Figure 4.10 shows experiments comparing image-specific to universal textons. From these figures, we see that the choice between image specific and universal textons is not important for the quality of the classifier. We use image-specific textons for convenience, though universal textons are perhaps more appealing in that they can be used to characterize textures in an image-independent manner. Image-independent descriptions of texture would be useful for image retrieval and object recognition applications. Figure 4.9 also reveals two scaling rules for the optimal number of textons. First, the optimal number of textons for universal textons is roughly double that required for image specific textons. Second, the optimal number of textons scales linearly with the area of the disc. The former scaling is expected, to avoid over-fitting in the image-specific case. The latter scaling rule keeps the number of samples per texton bin constant, which reduces over-fitting for the smaller TG scales.

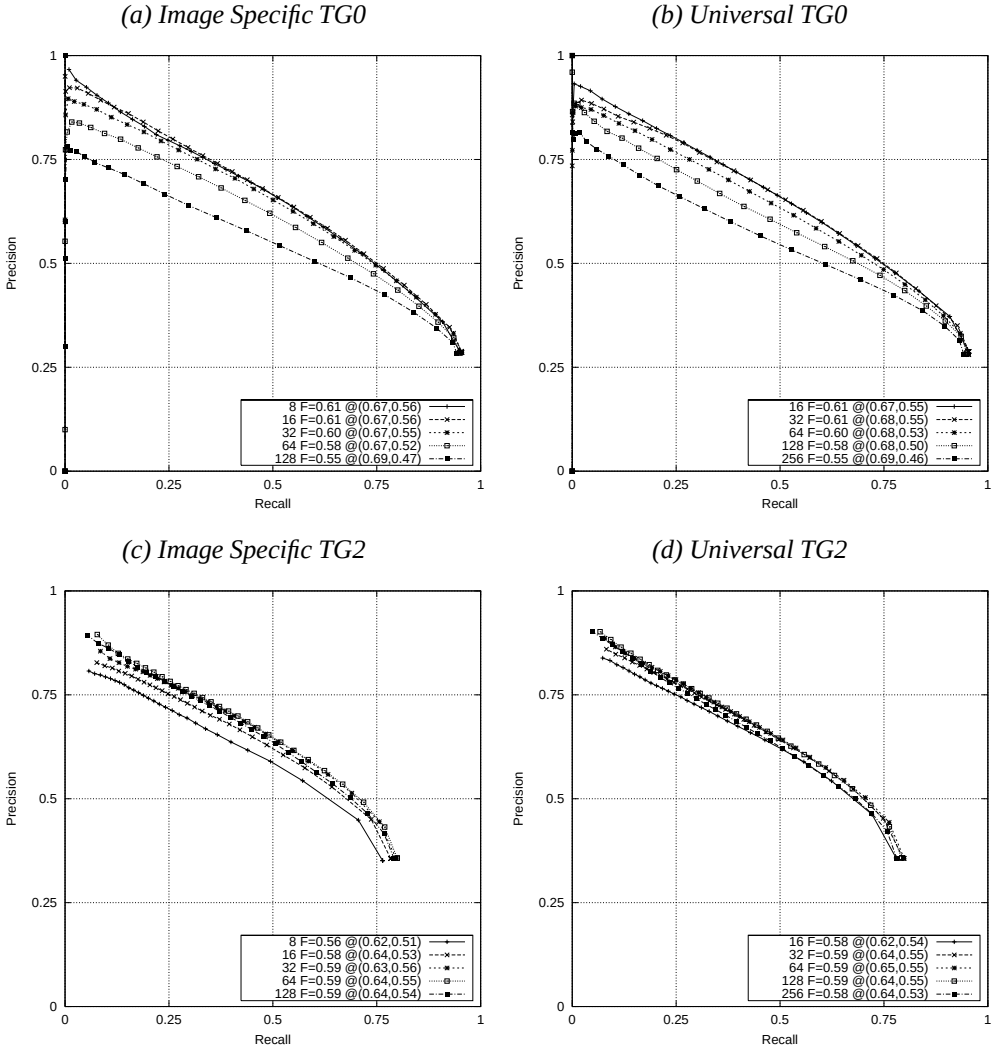


Figure 4.9: Optimizing the Number of Textons. We can compute textons on a per-image basis, or universally on a canonical image set. (a) and (c) show the performance of the small and large scales of TG for 8 to 128 image specific textons; (b) and (d) show the performance of the same TG scales for 16 to 256 universal textons. The optimal number of universal textons is double the number for image specific textons. In addition, smaller scales of TG require fewer textons. The scaling is roughly linear in the area of the TG disc, keeping the number of samples per bin constant. Results are insensitive to within a factor of two of the optimal number.

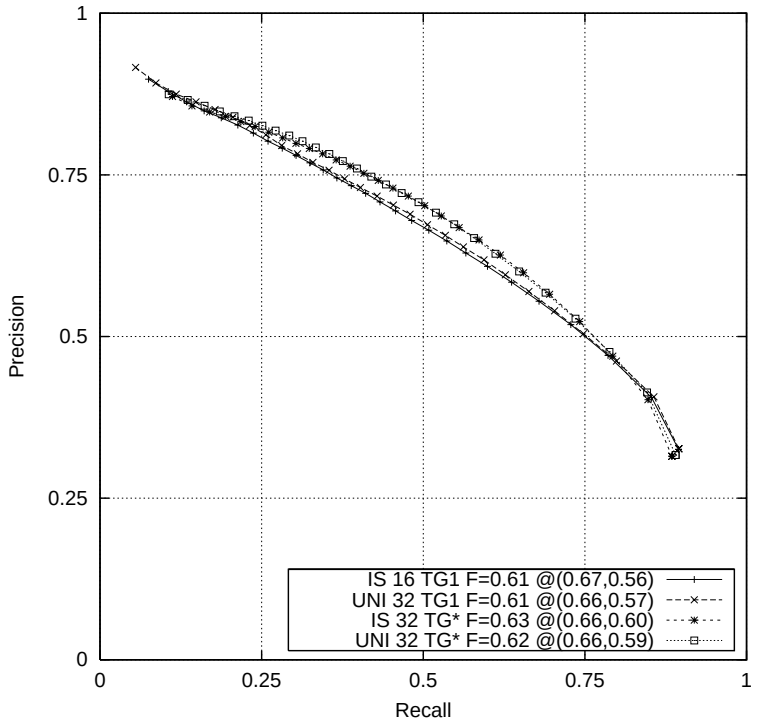


Figure 4.10: Image Specific vs. Universal Textons. This graph shows the performance of image specific vs. universal textons for the middle TG scale along with the combined TG scales. The choice between image-specific and universal textons is not critical. In our experiments, we use image-specific textons with k=$\{12,24,48\}$. For an application such as object recognition, one would likely prefer the measure of texture provided by universal textons, which can be compared across images.

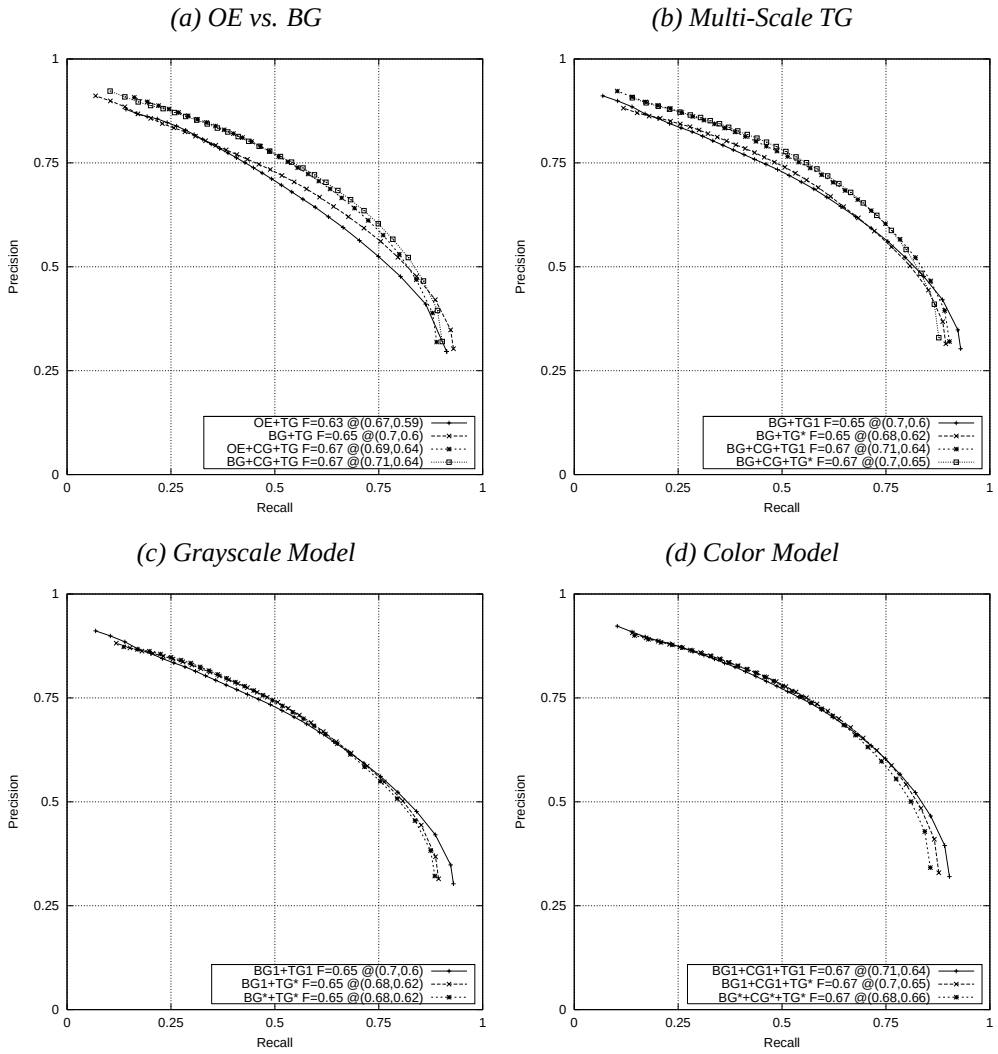
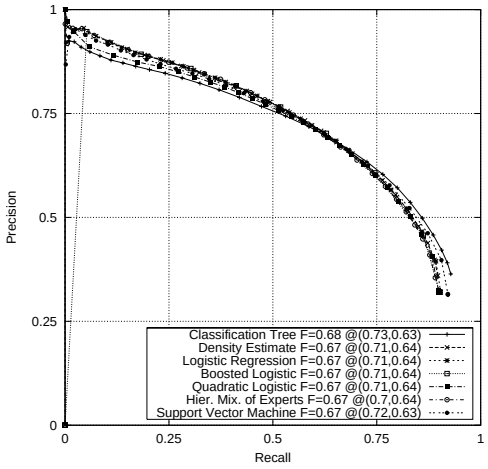


Figure 4.11: Cue Combination. After optimizing the parameters of each cue independently, we seek to combine the cues effectively. (a) shows that whether or not we include CG, we are always better off using BG as our brightness cue instead of OE. Note that though the curve is not shown, using OE and BG together is not beneficial. (b) Although we saw in Figure 4.6 that we benefit from using multiple scales of TG, the benefit is significantly reduced when BG is included. This is because BG contains some ability to discriminate fine scale textures. (c) Our non-color model of choice is simply the combination of a single scale of BG with a single scale of TG. (d) Our color model of choice also includes only a single scale of each of the BG, CG, and TG features.

(a) One Scale Per Cue



(b) Three Scales Per Cue

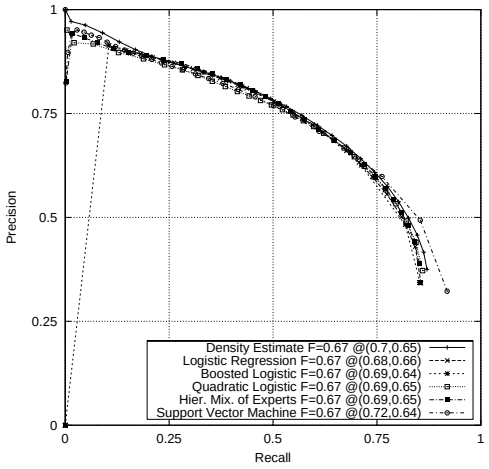


Figure 4.12: Choice of Classifier. Until this point, all results have been shown using the logistic regression model. This model is appealing because it is compact, robust, stable, interpretable, and quick to both train and evaluate. However, its linear decision boundary precludes any potentially interesting cross-cue gating effects. In this figure, we show the result of applying various more powerful models on (a) one scale of each of BG, CG, and TG, and (b) all three scales of each feature (9 total features). In neither case does the choice of classifier make much difference. In both cases, the logistic regression performs well. The addition of multiple scales does not improve performance. The logistic is still the model of choice.

4.5 Cue Combination

There is some previous work on learning boundary models. Will et al. [73] learned texture edge models for synthetic Brodatz mosaics. Meila and Shi [44] presented a framework for learning segmentations from labeled examples. Most compelling is the work of Konishi et al. [31], where boundary models were trained from human-labeled ground truth. After optimizing the performance of each cue, we face the problem of combining the cues into a single detector. We approach the task of cue combination as a supervised learning problem, where we will learn the combination rules from the ground truth data.

Figure 4.11 presents the first set of cue combination experiments. The first task is to determine whether any of the cues is redundant given the others. Until this point, we have presented four cues, two of which – OE and BG – both detect discontinuities in brightness. Panel (a) of the figure shows that BG is a superior cue to OE, whether used in conjunction with the texture gradient alone or with the texture and color gradients together. In addition, since we do not gain anything by using OE and BG in conjunction (not shown), we can safely drop OE from the list of cues.

We have the option of computing each cue at multiple scales. Figures 4.5 and 4.6 showed that only the texture gradient contains significant independent information at the different scales. The benefit of using multiple TG scales may not remain when TG is combined with other cues, however. Panel (b) of Figure 4.11 shows the effect of using multiple TG scales in conjunction with BG and CG. In both the BG and BG+CG cases, multiple TG scales improve performance only marginally. The remaining two panels of Figure 4.11 show the effect of adding multiple BG and CG scales to the model. In neither case do multiple scales improve overall performance. In some cases (see Figure 4.11(d)), performance can degrade as additional scales may introduce more noise than signal.

In order to keep the final system as simple as possible, we will retain only the middle scale of each feature. It is surprising that multi-scale cues are not a benefit, however. An explanation is suggested by Figures 4.6(d) on page 77 and 4.11(b) on page 83, where we see that the multiple scales of TG have independent information, but the benefit of multiple TG scales vanishes when BG is used. The brightness gradient operates at small scales, and is capable of low-order texture discrimination. At the smallest scales, there is not enough information for high-order texture analysis anyway, so BG is a good small-scale texture feature. The texture gradient identifies the more complex, larger scale textures.

Until this point, all results were generated with a logistic model:

$$p(x) = \frac{1}{1 + e^{-\beta^T x}} \tag{4.4}$$

We will show that the logistic model is a good choice by comparing a wide array of classifiers,

each trained on the human segmentation dataset. With more powerful models, we hoped to discover some interesting cross-cue and cross-scale gating effects. For example, one might discount the simpler edge detection of BG when TG is low because the brightness edges are likely to correspond to edges interior to textured areas. In addition, the optimal mixing function for the various cues could well be non-linear, with each cue treated as an expert for a certain class of boundaries. These are the classifiers that we used:

- ***Density Estimation*** We do density estimation with adaptive bins provided by vector quantization using k-means. Each k-means centroid provides the density estimate of its Voronoi cell as the fraction of on-boundary samples in the cell. We use k=128 bins and average the estimates from 10 runs to reduce variance.

- ***Classification Trees*** The domain is partitioned hierarchically with top-down axis-parallel splits. When a cell is split, it is split in half along a single dimension. Cells are split greedily so as to maximize the information gained at each step. The effect of this heuristic is to split nodes so that the two classes become separated as much as possible. A 5% bound on the error of the density estimate is enforced by splitting cells only when both classes have at least 400 points present.

- ***Logistic Regression*** This is the simplest of our classifiers, and the one perhaps most easily implemented by neurons in the visual cortex. Initialization is random, and convergence is fast and reliable by maximizing the likelihood with about 5 Newton-Raphson iterations. We also consider two variants: quadratic combinations of features, and boosting using the confidence-rated generalization of AdaBoost by Schapire and Singer [66]. No more than 10 rounds of boosting are required for this problem.

- ***Hierarchical Mixtures of Experts*** The HME model of Jordan and Jacobs [30] is a mixture model where both the experts at the leaves and the internal nodes that compose the gating network are logistic functions. We consider small binary trees up to a depth of 3 (8 experts). The model is initialized in a greedy, top-down manner and fit with EM. 200 iterations were required for the log likelihood to converge. Appendix B presents the multinomial logistic model that we used in our HME implementation.

- ***Support Vector Machines*** We use the SVM package `l` [9] to do soft-margin classification using Gaussian kernels. The optimal parameters were ν=0.2 and σ=0.2. In this parameterization of SVMs, ν provides the expected fraction of support vectors, which is also an estimate of the degree of class overlap in the data. The high degree of class overlap in our problem also explains the need for a relatively large kernel.

The ground truth boundary data is based on the dataset of [43] which provides about 5 human segmentations for each of 1000 natural images from the Corel image database. We used 200 images for training and algorithm development. The 100 test images were used only to generate the final results for this chapter. Chapter 2 established that the segmentations of a single image by the different subjects are highly consistent, so we consider all human-marked boundaries valid. We declare an image location (x, y, θ) to be on-boundary if it is within $\Delta x \leq \sqrt{8}$ pixels and $\Delta\theta \leq 30$ degrees of any human-marked boundary. The remainder are labeled off-boundary.

This classification task is characterized by relatively low dimension, a large amount of data (100M samples for our 240x160-pixel images), poor class separability, and a 10:1 class ratio. The maximum feasible amount of data, uniformly sampled, is given to each classifier. This varies from 50M samples for the density estimation and logistic to 20K samples for the SVM and HME. Note that a high degree of class overlap in *any* low-level feature space is inevitable because the human subjects make use of both global constraints and high-level information to resolve locally ambiguous boundaries.

Figure 4.12(a) shows the performance of the seven classifiers using only the middle scale of BG, CG, and TG. The PR curves all cross approximately at the maximal F measure point, and so all the classifiers are equivalent as measured by the F measure. The classification tree and SVM are able to achieve marginally higher performance in the high recall and low precision regime, but they perform worse in the low recall and high precision area. Overall, the performance of all the classifiers is approximately equal, but other issues affect model choice such as representational compactness, stability, bias, variance, cost of training, and cost of evaluation.

The non-parametric models achieve the highest performance, as they are able to make use of the large amount of training data to provide unbiased estimates of the posterior, at the cost of opacity and a large model representation. The plain logistic is stable and quick to train, and produces a compact and intuitive model. In addition, the figure shows that the logistic's bias does not hurt performance. When given sufficient training data and time, all the variants on the logistic – the quadratic logistic, boosted logistic, and HME – provided minor performance gains. However, the many EM iterations needed to fit the HME required us to subsample the training data heavily to keep training time within reasonable limits.

The support vector machine was a disappointment. Training time is super-linear in the number of samples, so the training data had to be heavily subsampled. The large class overlap produced models with 25% of the training samples as support vectors, so that the resulting model was opaque, large, and exceedingly slow to evaluate. In addition, we found the SVM to be brittle with respect to its parameters ν and σ. Even at the optimal settings, the training would occasionally produce nonsense models. Minute variations from the optimal settings would produce infeasible problems.

We conclude that the SVM is poorly suited to a problem that does not have separable training data.

Panel (b) of Figure 4.12 shows the performance of each classifier, except the classification tree, when all three scales are included for each of the three features. The results are much as before, with virtually no difference between the different models. Balancing considerations of performance, model complexity, and computational cost, we favor the logistic model and its variants.

4.6 Results

Having settled on a grayscale boundary model using a single scale each of BG and TG, and a color model that adds a single scale of CG, we seek to compare these models to classical models and the state-of-the-art. The model that we present as a baseline is MATLAB's implementation of the Canny [6] edge detector. We consider the detector both with and without hysteresis. To our knowledge, there is no work proving the benefit of hysteresis thresholding for natural images. We will call the Canny detector without hysteresis "GD", as it is simply a Gaussian derivative filter with non-maxima suppression. With hysteresis, the operator is called "GD+H".

The GD and GD+H detectors each have a single parameter to tune – the σ of the Gaussian derivative filters. Figure 4.13(a) and (b) show the PR curves for various choices of σ. For both cases, $\sigma = 1$ pixel is a good choice. Note that the detector threshold is not a parameter that we need to fit, since it is the parameter of the PR curves.

For the state of the art detector, we use a detector derived from the spatially-averaged second moment matrix (2MM). It has long been known that the eigenspectrum of the second moment matrix provides interesting features. For example, both eigenvalues being large may indicate a corner or junction. This observation is the basis of the Plessey or Harris-Stephens [22] corner detector and the Förstner corner detector [15]. One large and one small eigenvalue may indicate a simple edge. The Nitzberg edge detector [49] used by Konishi et al. [31] is based on the difference between the eigenvalues. What is common to these approaches is that they are all based on simple features derived from the eigenvalues of the 2MM.

We apply the same training/test methodology to our 2MM detector as we do to our own detectors, but we use the full eigenspectrum as a feature vector. From the 200 training images, we obtain on- and off-boundary labels for pixels and train a logistic model using both eigenvalues of the 2MM as features. Figure 4.14 shows the model trained in this manner. Panel (a) shows the distribution of the training data in feature space. Panel (b) shows the empirical posterior, and panel (c) shows the fitted posterior from the logistic model. In addition, in order to perform non-maxima suppression on the 2MM output, we calculated the orientation of the operator's response from the leading eigenvector.

The 2MM detector also has scale parameters. The inner scale is the scale at which image deriva-

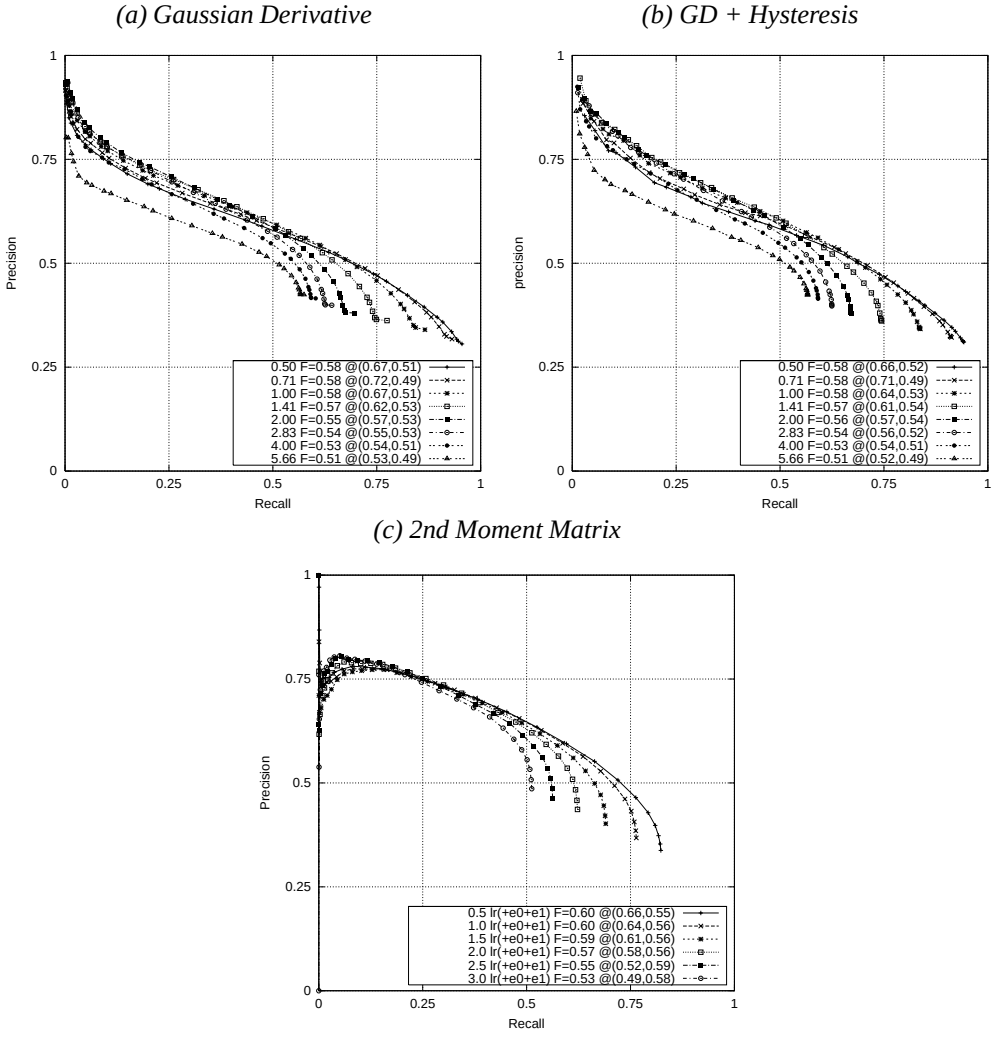


Figure 4.13: Choosing σ for the Classical Edge Operators. The Gaussian derivative (GD) operator (a) without hysteresis and (b) with hysteresis, and (c) the second moment matrix (2MM) operator, fitted as in Figure 4.14. In (a) and (b), σ ranges from 0.5 to 5.66 at half-octave scales. In (c), σ ranges from 0.5 to 3 at an increment of 0.5. From these experiments, we choose the optimal scales of σ=1 for GD regardless of hysteresis, and σ=0.5 for 2MM.

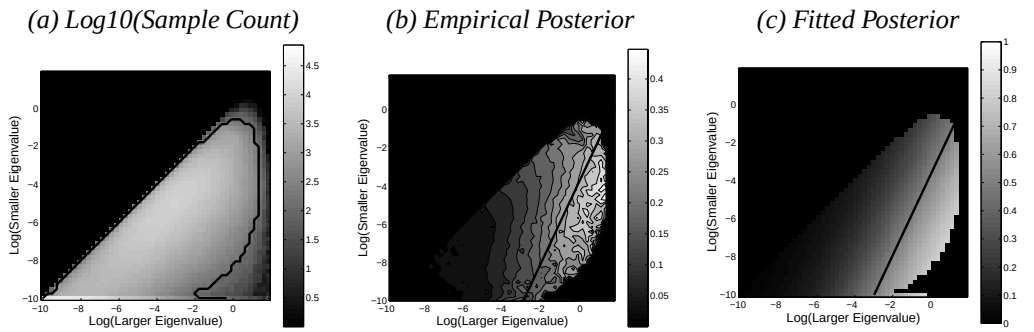


Figure 4.14: Optimizing the 2nd Moment Matrix Model (2MM). For this model, the two features are the smaller and larger eigenvalues of the locally averaged second moment matrix. (a) shows the histogram of samples from the 200 training images along with the 100 samples/bin contour. (b) shows the empirical posterior probability of a boundary, and (c) shows the fitted posterior using logistic regression. The linear decision boundary of the fitted logistic is drawn in both (b) and (c). The coefficients of the fitted logistic are -0.27 for the larger eigenvalue and 0.58 for the smaller eigenvalue, with an offset of -1.

tives are estimated. We set the inner scale to a minimum value, estimating the derivatives with the typical 3x3 [-1,0,1] filters. Figure 4.13(c) shows the optimization over the outer scale parameter, which is the scale at which the derivatives are spatially averaged. Only a modest amount of blur is required ($\sigma = 0.5$ pixels). Note that some blur is required, or the second eigenvalue vanishes. Less smoothing is not possible due to pixel resolution.

In Figure 4.15, we give a summary comparison of the BG, CG, and TG detectors, along with two combinations: BG+TG for grayscale images, and BG+CG+TG for color images. It is clear that each feature contains a significant amount of independent information. Figure 4.3 on page 65 shows the comparison between the two Gaussian derivative operators (GD and GD+H), the second moment matrix operator (2MM), our grayscale BG+TG operator, and our color BG+CG+TG operator. [3] First, note that hysteresis does impart a marginal improvement to the plain GD operator, though the difference is pronounced only at very low recall rates. The 2MM operator does mark a significant improvement over the Canny detector, except at low recall. The main benefit of the 2MM operator is that it does not fire where both eigenvalues are large – note the opposite signs of the coefficients in the model. As a result, it does not fire where energy at multiple orientations coincide at a pixel, such as at corners or inside certain textures. Thus, 2MM reduces the number of false positives from

[3] The logistic coefficients for the BG+TG operator are 0.50 for BG and 0.52 for TG with an offset of -2.81. The coefficients for the color model are 0.31 for BG, 0.53 for CG, and 0.44 for TG, with an offset of -3.08. The features are normalized to have unit variance. Feature standard deviations are 0.13 for BG, 0.077 for CG, and 0.063 for TG.

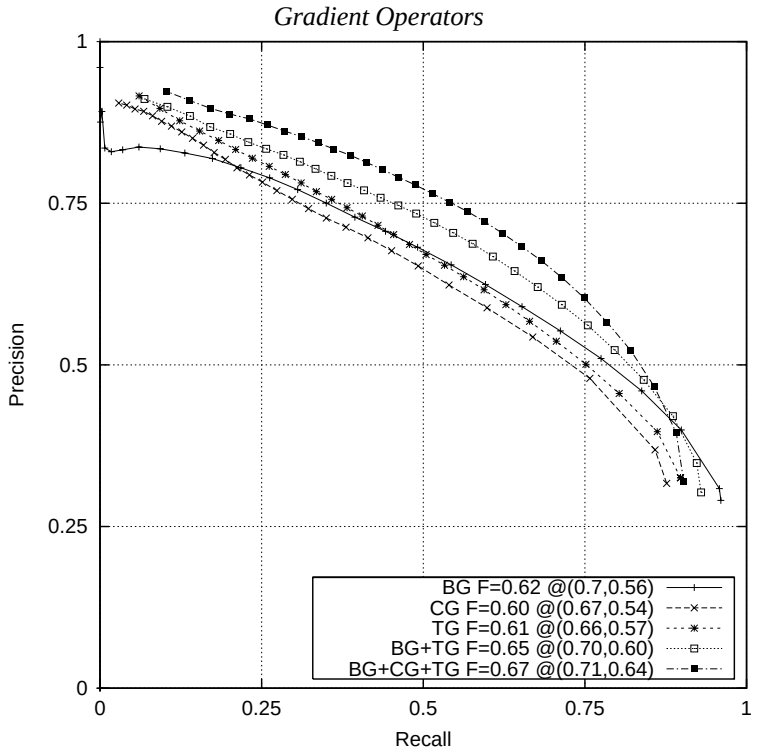


Figure 4.15: Detector Comparison. The performance of the edge detectors proposed in this chapter, both independently and in combination.

high contrast texture.

The operators based on BG and TG significantly outperform both classical and state-of-the-art edge detectors. The main reason for the improved performance is a robust treatment of texture. Neither GM nor 2MM can detect texture boundaries. For the same reason that 2MM suppresses false positives inside textured areas, it also suppresses edges between textured areas.

Figure 4.3 also shows the performance of the human subjects in the segmentation dataset. Each plotted point shows the precision and recall of a single human segmentation when it is compared to the other humans' segmentations of the same image. The median human F-measure is 0.80. The solid line in the upper right corner of the figure shows the iso-F-measure line for 0.80, representing the F-measure frontier of human performance.

Each of the curves in Figure 4.3 uses a fixed distance tolerance $d_{\max}$ = 1% of the image diagonal (2.88 pixels). Figure 4.16 shows the precision-recall curves for each detector as this tolerance changes. The digital pixel grid forces a discretization of this parameter, so the figure shows the result for $d_{\max} = \{\sqrt{2}, 2, \sqrt{5}, \sqrt{10}\}$. Since the curves do not intersect and are roughly parallel, the F-measure captures the differences effectively. Figure 4.17 shows how the F-measure changes as a function of $d_{\max}$ for each detector and for the human subjects. If a detector's localization were good to within 1 pixel, then the detector's curve would be flat. In contrast, all of the machine curves reveal localization error greater than that shown by the human subjects. Additional work on local boundary detection will no doubt narrow the gap between machine and human performance, but large gains will ultimately require higher-level algorithms.

We present qualitative results in Figures 4.18 and 4.19. The first figure shows various versions of our detectors along with the humans' boundaries. The second figure shows a comparison between the GD+H, 2MM, and BG+TG detectors, also alongside the human's boundaries. Each machine detector image shows the soft boundary map after non-maxima suppression, and after taking the maximum over θ. In Figure 4.18, we see the complementary information contained in the three channels, and the effective combination by the logistic model. For example, color is used when present in (b,c,i) to improve the detector output. Figure 4.19 shows how the BG+TG detector has eliminated false positives from texture while retaining good localization of boundaries. This effect is particularly prominent in image (e).

Overshoot artifacts are visible in the TG-only images in Figure 4.18(a-b), though they are suppressed somewhat after cue combination. These artifacts hint at the next issue we must face in a local boundary model: integrating high-level information. In order to integrate texture and color, we must use operators with a large support. Such large supports cause problems at high-curvature boundaries where the boundary does not follow the disc's diameter, and where the support crosses multiple regions. One would ideally clip the operator support at object boundaries in order to clar-

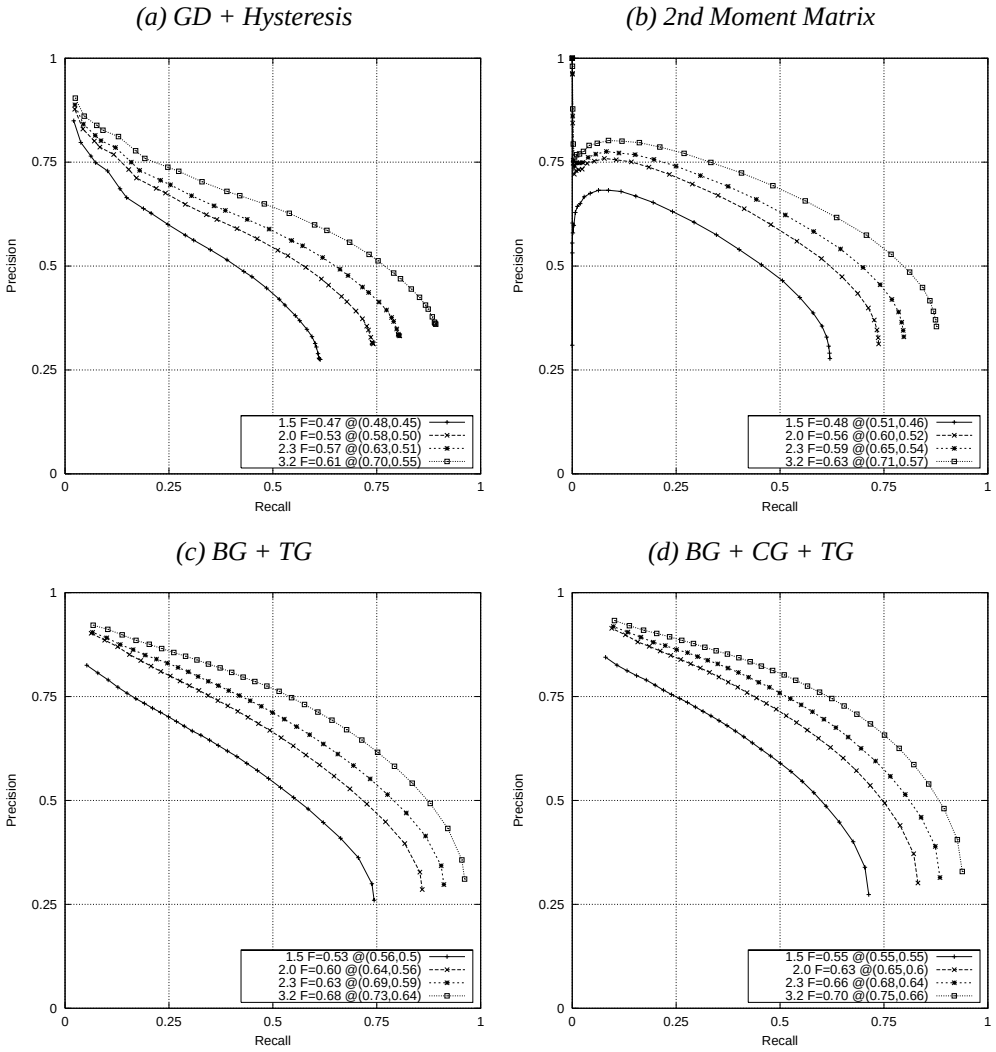


Figure 4.16: Detectors at Various Distance Tolerances. The panels show precision-recall curves for several detector as the matching tolerance varies from $\sqrt{2}$ to $\sqrt{10}$ pixels. Since the curves do not intersect, the F-measure is a good representation of performance regardless of threshold. Figure 4.17 shows F-measure curves for this data.

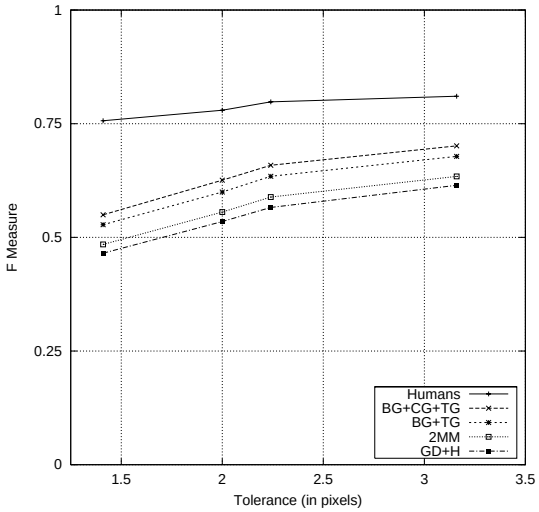


Figure 4.17: F-Measure for Detectors at Various Distance Tolerances. This graph shows the relationship between F-measure and the distance tolerance for the four detectors, along with the median human performance. The human curve is flatter than the machine curves, showing that the humans' localization is good. The gap between human and machine performance can be reduced but not closed by better local boundary models. Both mid-level cues and high-level object-specific knowledge are likely required to approach the performance of the human subjects.

ify the signal. However, with feedback one must initially proceed from the low level, which is the purpose of this chapter.

4.7 Conclusion

We have defined a novel set of brightness, color, and texture cues appropriate for constructing a local boundary model. By using a large dataset of human-labeled boundaries in natural images, we have formulated the task of cue combination for local boundary detection as a supervised learning problem. This approach models the true posterior probability of a boundary at every image location and orientation, which is particularly useful for higher-level algorithms. The choice of classifier for modeling the posterior probability of a boundary based on local cues is not important; a simple linear model is sufficiently powerful. Based on a quantitative evaluation on 100 natural images, our detector outperforms existing methods, indicating that a proper treatment of texture is essential for detecting boundaries in natural images.

Figure 4.18: Boundary Images for Our Detectors. Rows 2 to 4 show real-valued probability-of-boundary (P_b) images after non-maxima suppression for the three cues. The complementary information in each of the three BG, CG, and TG channels is successfully integrated by the logistic in row five. The boundaries in the human segmentations shown in the last row are darker where more subjects marked a boundary.

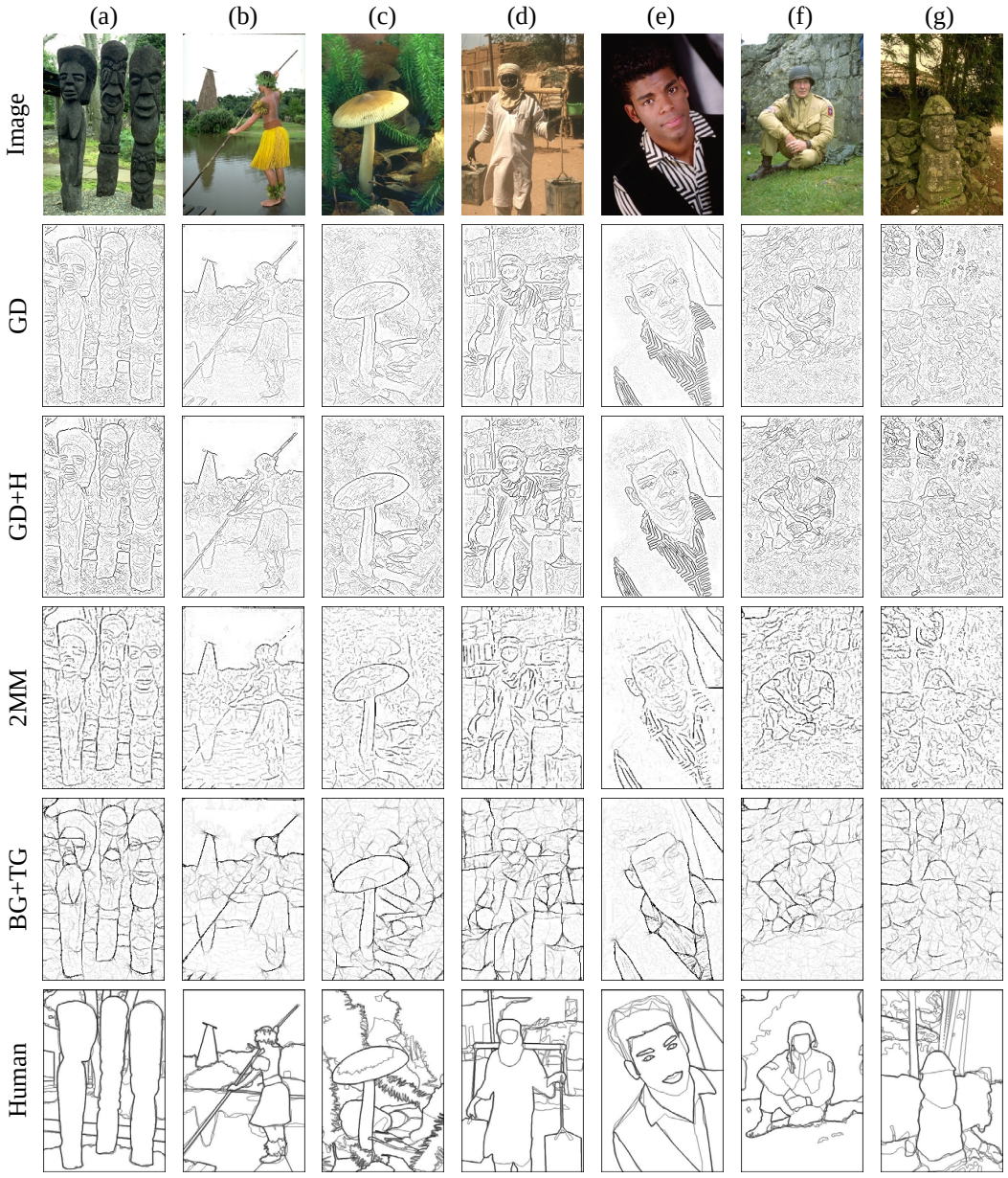


Figure 4.19: Boundary Images for Three Grayscale Detectors. These are the same images as in Figure 4.18. Rows 2 to 4 show P_b images for the Gaussian derivative with hysteresis (GD+H), the 2nd moment matrix (2MM), and our brightness+texture detector (BG+TG). The human segmentations are shown once more for comparison. The BG+TG detector benefits from a suppression of edges on the interior of textured regions, and from operating at a large scale without sacrificing localization.

Chapter 5

Learning a Pixel Affinity Model

In this chapter, we examine the problem of combining region and boundary cues for natural image segmentation. We employ a large database of manually segmented images in order to learn an optimal affinity function between pairs of pixels. These pairwise affinities can then be used to cluster the pixels into visually coherent groups. Region cues are computed as the similarity in brightness, color, and texture between image patches. Boundary cues are incorporated by looking for the presence of an "intervening contour", a large gradient along a straight line connecting two pixels.

We first use the dataset of human segmentations to individually optimize parameters of the patch and gradient features for brightness, color, and texture cues. We then quantitatively measure the power of different feature combinations by computing the precision and recall of classifiers trained using those features. The mutual information between the output of the classifiers and the same-segment indicator function provides an alternative evaluation technique that yields identical conclusions.

As expected, the best classifier makes use of brightness, color, and texture features in both patch and gradient forms. Figure 5.1 shows the performance of our best affinity models compared to the human ground truth. We find that for brightness, the gradient cue outperforms the patch similarity. In contrast, using color patch similarity yields better results than using color gradients. Texture is the most powerful of the three channels, with both patches and gradients carrying significant independent information. Interestingly, the proximity of the two pixels does not add any information beyond that provided by the similarity cues. We also find that the convexity assumptions made by the intervening contour approach are supported by the ecological statistics of the dataset.

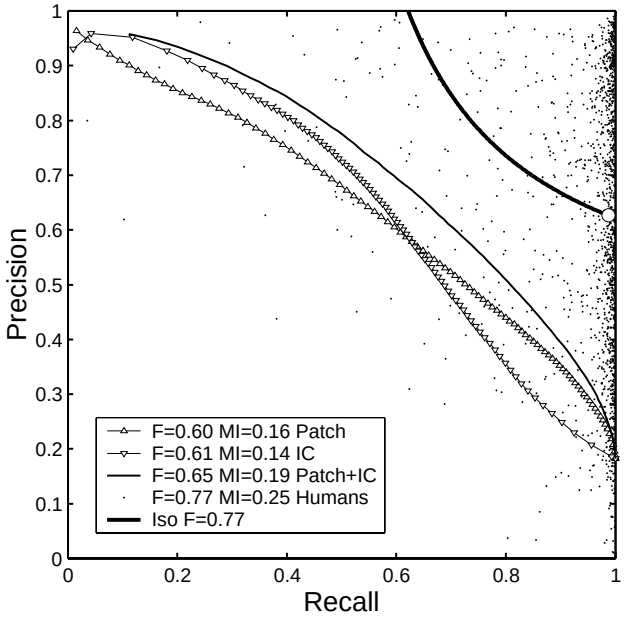


Figure 5.1: Performance of Pixel Affinity Models. This figure shows the performance of our best pixel affinity models compared to the affinity defined by the human data. The dots show the precision and recall of each of 1366 human segmentations in the 250-image test set when compared to the other humans' segmentation of the same image. The large dot marks the median recall (99%) and precision (63%) of the humans. The iso-F-measure curve at F=77% is extended from this point to represent the frontier of human performance for this task. The three remaining curves represent our patch-only model, contour-only model, and patch+contour model. Neither patches nor contours are sufficient, as there is significant independent information in the patch and contour cues. The model used to combine features is a logistic function with quadratic terms, which performs best among classifiers tried on this dataset.

5.1 Introduction

Boundaries and regions are closely intertwined. A closed boundary generates a region, and every image region has a boundary. Psychophysical experiments suggest that humans use both boundary and region cues to perform segmentation [75]. In order to build a vision system capable of parsing natural images into coherent units corresponding to surfaces and objects, it is clearly desirable to make global use of both boundary and region information.

Historically, researchers have focused separately on the sub-problems of boundary and region grouping. Region based approaches are motivated by the Gestalt notion of grouping by similarity. They typically involve integrating features such as color or texture over local patches of the image [14, 23, 56] and then comparing different patches [48, 55]. However, smooth changes in texture or brightness caused by shading and perspective within regions pose a problem for this approach since two distant patches can be quite dissimilar despite belonging to the same image segment. To overcome these difficulties, gradient-based approaches detect local edge fragments marked by sharp, localized changes in some image feature [6,46,38,53,42]. The fragments can then be linked together in order to identify extended contours [51, 74, 12].

Less work has dealt directly with the problem of finding an appropriate intermediate representation in order to incorporate non-closed boundary fragments into segmentation. Mathematical formulations outlined by [18, 47, 45] along with algorithms such as [39, 28] have attempted to unify boundary and region information. More recently, [37, 69] have demonstrated the practical utility of integrating both in order to segment images of natural scenes.

There are some widely held "folk-beliefs" regarding the various cues used for image segmentation:

1. Brightness gradients (caused by shading) and texture gradients (caused by perspective) necessitate a boundary-based approach.
2. Edge detectors are confused by texture, so one must use patch-based similarity for texture segmentation.
3. Color integrated over local patches is a robust and powerful cue.
4. Proximity is a good indicator for grouping.

These statements have not been tested empirically, however. By using a dataset of human segmentations as ground truth, we are able to provide quantitative results regarding the ecological statistics [1] of patch- and gradient-based cues and gauge their relative effectiveness.

[1] Our approach follows the lines of Egon Brunswik's suggestion nearly 50 years ago that the Gestalt factors made sense because they reflected the statistics of natural scenes [4].

We treat the problem of integrating both gradient and patch information for segmentation within the framework of pairwise clustering [67, 76, 65, 68, 13, 52, 17, 72]. In contrast to central clustering techniques such as k-means or mixtures-of-Gaussians, which compare each pixel (or other image element) to some small set of prototypes, pairwise techniques rely on the evaluation of an affinity function between each pair of image pixels. Although pairwise techniques tend to be more computationally expensive, they have the advantage of removing the constraint that pixels be explicitly embedded in some normed vector space where Euclidean or Mahalanobis distances "make sense". Instead, pixels are implicitly described by their similarity to every other pixel in the image.

The pairwise framework allows patch and gradient information equal footing in the following way. Associate a descriptor to each pixel that captures color, brightness and texture in a neighborhood of the pixel. The patch-based similarity between two pixels is a function of the difference in their descriptors. A gradient is computed as the change in these local descriptors between nearby pixels. For each pair of pixels, record the magnitude of the gradient encountered along a straight path connecting the two pixels in the image plane. Large gradients indicate the presence of an "intervening contour" [34] and suggest that the pixels do not belong to the same segment. The pairwise affinity between the i-th and j-th pixel is given by a function whose arguments are the similarity between the i-th and j-th local descriptors and the gradients along the path from i and j.

Most applications of pairwise clustering to segmentation have made use of heuristically derived affinity functions, such as Malik et al. [37]. Meilă and Shi [44] make a natural proposal to learn optimal pairwise affinities from training data. In the results presented here, nearly all free parameters — filter scales, histogram binning and quantization, descriptor windowing, combination of gradient features, and so on — have been carefully optimized with respect to training data. Our goal in this chapter is to explicitly model the posterior probability of two pixels belonging to the same image segment conditioned on photometric properties of the image. Figure 5.2 shows examples of both ground truth affinity functions and affinity models learned from data.

We provide two general schemes for evaluating the effectiveness of different combinations of features. The first is to train a classifier that declares two pixels as lying in the same or different segments given some set of features. Classifier performance is then evaluated by considering the trade-off between precision and recall. The second approach is to compute the mutual information between the classifier output and the same-segment indicator provided by the human segmentations. These two schemes are in strong agreement, which lends force to our findings:

- Segmentations of the same image by different humans are quite consistent with each other. "Fine" segmentations tend to be "coarse" segmentations with regions that have been refined by breaking them into roughly convex parts.
- The ecological statistics of the dataset show that regions are mostly convex, validating the

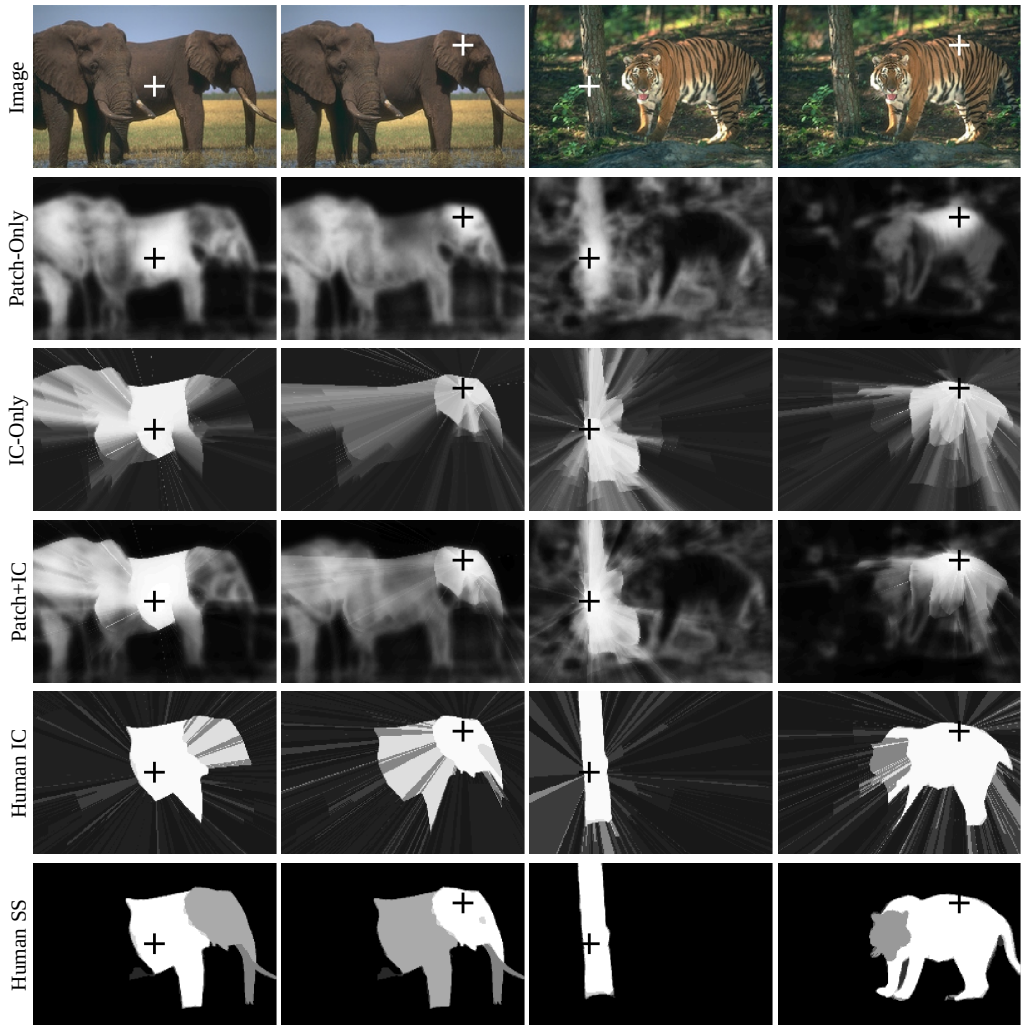


Figure 5.2: Pixel Affinity Images. The first row shows an image with one pixel selected. The remaining rows show the similarity between that pixel and all other pixels in the image, where white is most similar. Rows 2 to 4 show our patch-only, contour-only, and patch+contour affinity models. Rows 5 and 6 show the pixel similarity given the ground truth data, where white corresponds to more agreement between humans. Row 6 shows simply the same-segment indicator function, while row 5 is computed using intervening contour on the human boundary maps.

assumptions made by the intervening contour approach.

- Intervening contour and patch comparisons both provide significant, independent information about whether two pixels belong in the same segment.
- The color cue is best captured using patches, while for brightness one should use gradients. For texture, both gradients and patches are valuable.
- The proximity of two pixels does not provide any information not given by the patch-based or gradient-based similarity. It is simply a result, not a cause, of grouping.

5.2 Methodology

We formulate the problem of learning the pixel affinity function as a classification problem of discriminating same-segment pixel pairs from different-segment pairs. Let S_{ij} be the true same-segment indicator so that $S_{ij}=1$ when pixels i and j are in the same segment, and $S_{ij}=0$ when pixels i and j are in different segments.

The dataset presented in Chapter 2 provides the ground truth segmentation data. We use the color segmentations for 500 images, divided into test and training sets of 250 images each. Each image has been segmented by at least 5 subjects, so the ground truth S_{ij} is defined by a set of human segmentations. We declare two pixels to lie in the same segment only if all subjects declare them to lie in the same segment.

Given a classifier output $\hat{S}_{ij}$, we can evaluate the classifier's performance in two ways. Our first evaluation technique uses the *precision-recall* (PR) framework, as in Chapter 4. *Precision* measures the probability that two pixels declared by the classifier to be in the same segment are in the same segment, i.e. $P(S_{ij}{=}1\,|\,\hat{S}_{ij}{=}1)$. *Recall* measures the probability that a same segment pair is detected, i.e. $P(\hat{S}_{ij}{=}1\,|\,S_{ij}{=}1)$.

The second approach to evaluating a classifier measures the mutual information between the classifier output $\hat{S}$ and the ground truth data S, as discussed in Section 3.1.1. Given the joint distribution $p(x,y)=P(S=x,\hat{S}=y)$, the mutual information I is defined as the Kullback-Liebler divergence between the joint and the product of the marginals, so $I(S;\,\hat{S})=\int_{x,y}p(x,y)\log\frac{p(x,y)}{p(x)p(y)}$. We compute the joint distribution by binning the soft classifier output.

Chapter 3 provides target performance numbers for our affinity models. In that chapter, we evaluated the ground truth affinity functions generated by human segmentations. The median F-measure for the human subjects was 0.76, and the median mutual information 0.25.

5.3 Features

We will model the affinity between two pixels as a function of both patch-based and gradient-based features. In each case, we can use brightness, color, or texture, producing a total of six features. We also consider the distance between the pixels as a seventh feature.

5.3.1 Patch-Based Features

Given a pair of pixels, we wish to measure the brightness, color, and texture similarity between circular neighborhoods of some radius centered at each pixel. These features are similar to the gradient features used in Chapter 4. The difference is that instead of computing the similarity between two adjacent half discs, here we compute the similarity between two whole discs.

For the brightness and color patch features, all of the same issues that were presented in Section 4.2.3 arise regarding histogram comparison techniques and kernel density estimation. We take the same approach here of using the 1976 CIE L*a*b* color space separated into luminance and chrominance channels. We model brightness and color distributions with histograms constructed by binning kernel density estimates, and compare histograms with the χ^2 histogram difference operator. For the brightness cue, we use the L* histogram for each pixel. For color, we compute separate a* and b* histograms, and sum their respective χ^2 contributions.

For the patch-based texture feature, we compare the distributions of filter responses in the two discs. As in Section 4.2.4, we compare texton distributions with χ^2, where the textons are computed by clustering filter bank responses with k-means. The texture descriptor for a pixel is therefore a k-bin histogram over the pixels in a disc of radius r centered on the pixel.

All of the patch-based features have parameters that require tuning, such as the radius of the discs, the binning parameters for brightness and color, and the texton parameters for texture. Section 5.4.2 covers the experiments that tune these parameters with respect to the training data.

5.3.2 Gradient-Based Features

Given a pair of pixels, consider the straight-line path connecting them in the image plane. If the pixels lie in different segments, then we expect to find, somewhere along the line, a photometric discontinuity or *intervening contour* [34]. If no such discontinuity is encountered, then the affinity between the pixels should be large.

In order to compute the intervening contour cue, we require a boundary detector that works robustly on natural images. For this we employ the gradient-based boundary detector of Chapter 4. The output of the detector is a P_b image that provides the posterior probability of a boundary at each pixel. We consider the three P_b images computed using brightness, color, and texture gradients

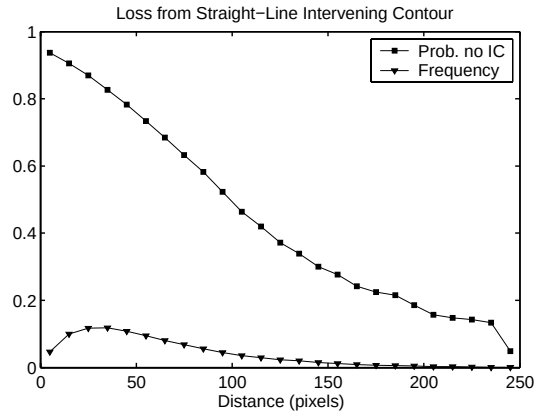


Figure 5.3: Loss from Straight-Line Intervening Contour. The top curve shows the probability — in the human segmentations — that there is no intervening contour between two same-segment pixels, conditioned on image plane distance. An intervening contour of this sort can occur only from concavities. Therefore, this curve shows the degree to which segments are convex as a function of scale. The lower curve shows the distribution of the same-segment pairs used in the computation.

individually, as well as the P_b image that combines the three cues into a single boundary map. The combined model uses a logistic function trained on the dataset, which is well motivated by evidence in psychophysics that humans make use of multiple cues in localizing contours [59] perhaps using a linear combination [32].

We compute the intervening contour cue for two pixels i and j from the P_b values that occur along the straight line path $\Gamma(t)$ connecting the two pixels. We consider the family of measures $L^p(\Gamma) = (\sum_t P_b(\Gamma(t))^p)^{1/p}$ for $p = \{0, 1, 2, 4, \infty\}$, as well as the mean of $P_b(\Gamma(t))$. The next section will cover the choice of the intervening contour function, as well as the best way to combine the contour information from the brightness, color, and texture channels.

5.4 Findings

5.4.1 Validating Intervening Contour

Although the ecological statistics of natural images indicate that regions tend to be convex [16], the presence of an intervening contour does not necessarily indicate that two pixels belong in different segments. Concavities introduce intervening contours between same-segment pixel pairs. In this section, we analyze the frequency with which this happens.

Given the union of boundary maps for all human segmentations of an image, we measure the probability that same-segment pairs have no intervening boundary contour. Figure 5.3 shows this

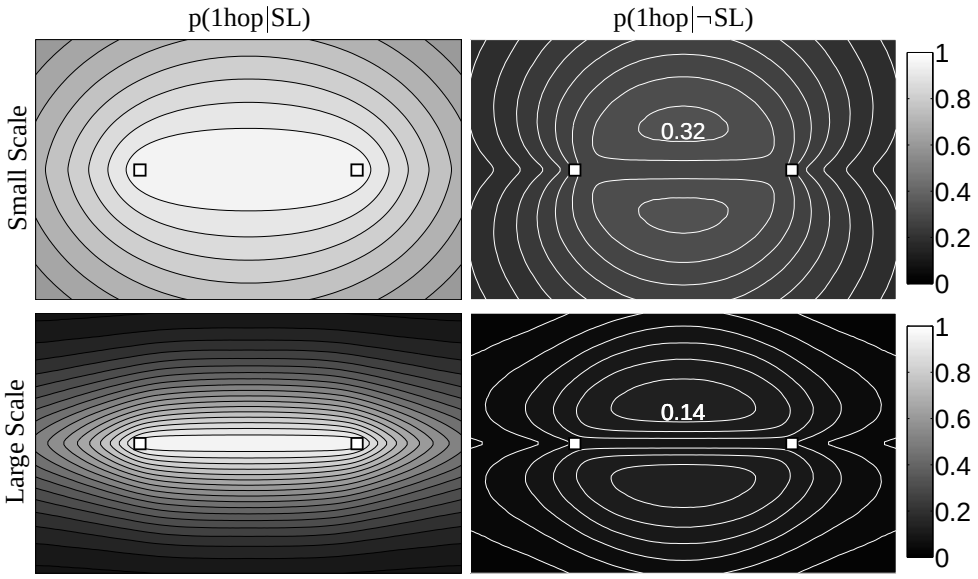


Figure 5.4: One-Hop Paths. Each panel shows two pixels, marked with squares, that all humans declared to be in the same segment. The intensity at each point represents the empirical probability that a one-hop path through that point does not intersect a boundary contour. The left column is conditioned on there being an unobstructed straight-line (SL) path between the pixels, while the right column shows the probabilities when the SL path is obstructed. The top row shows data gathered from pixel pairs with small separation; the bottom row for pairs with large separation.

probability as a function of pixel separation, along with the frequency of same-segment pairs at each distance. If the regions were convex, then the curve would be fixed at one. We see that straight-line intervening contour is a good approximation to the same-segment indicator for small distances: 49% of pairs are in the $>75\%$ range.

When straight-line intervening contour fails for a same-segment pair, there exists a more complex path connecting the pixels. Consider the set of paths that consist of two straight-line segments, which we call *one-hop* paths. The situations where one-hop paths succeed but straight-line paths fail can give us some intuition about how much can be gained by examining paths more complex than straight line paths.

Figure 5.4 shows the empirical spatial distribution of one-hop paths between same-segment pixel pairs, using the union of human segmentations. The probability of a one-hop path existing is conditioned on (left) there being a straight-line path with no intervening contour and (right) on there being no straight-line path. If the human subjects' regions were completely convex, then the right column images would be zero. Instead, we see that when the straight-line intervening contour fails, there is a small but significant probability that a more complex one-hop path will succeed, and the probability

of such a path is larger for smaller scales. There is clearly some benefit from the more complex paths due to concavities in the regions. However, the degree to which an algorithm could take advantage of the more powerful one-hop version of intervening contour depends on the frequency with which the one-hop paths find holes in the estimated boundary map. In any case, the figure makes clear that the simple straight-line path is a good first-order approximation to the connectivity of same-segment pairs.

Since the straight-line version of intervening contour will underestimate connectivity in concave regions, it may have a tendency toward over-segmentation. Figure 5.5 shows the effect on precision and recall for the human data when we add the constraint that same-segment pairs must have no intervening boundary contour. As in Chapter 3, we are comparing a left-out human to the union of the remaining humans. On average, the union segmentation will be more detailed than the left-out human. The figure shows a increase in median precision from 66% to 75%, indicating that intervening contour tends to break up non-convex segments in a manner similar to the human subjects. This lends confidence to an approach to perceptual organization of first finding convex object pieces through low-level processes, and then grouping the object pieces with into whole objects using higher-level cues.

5.4.2 Performance of Patches

Each of the brightness, color, and texture patch features has several parameters that require tuning. We optimized each patch feature independently via coordinate ascent to maximize performance on the training data. Figure 5.6 shows the result of the coordinate ascent experiments, where no change in any single parameter further improves performance.

For brightness and color, a radius of 5.76 pixels was optimal, though performance is similar for larger and smaller discs. In contrast, the texture disc radius has greater impact on performance, and the optimal radius is much larger at 16.1 pixels.

The brightness and color patches also have parameters related to the binned kernel density estimates. The binning parameters for brightness are important for performance, while the color binning parameters are less critical. A larger σ indicates that small differences in the cue are less perceptually significant—or at least less useful for this task.

Apart from the disc radius, the texture patch cue has additional parameters related to the texton computation. The upper right table in Figure 5.6 shows the optimization over the number of textons, with 512 being optimal. In general, we found that the number of textons should be approximately half the number of pixels in the disc. In addition, we find agreement with [42, 35] that the filter bank should contain a single scale, and that the scale should be as small as possible.

Figure 5.7 shows the performance of classifiers trained on each patch feature individually, along

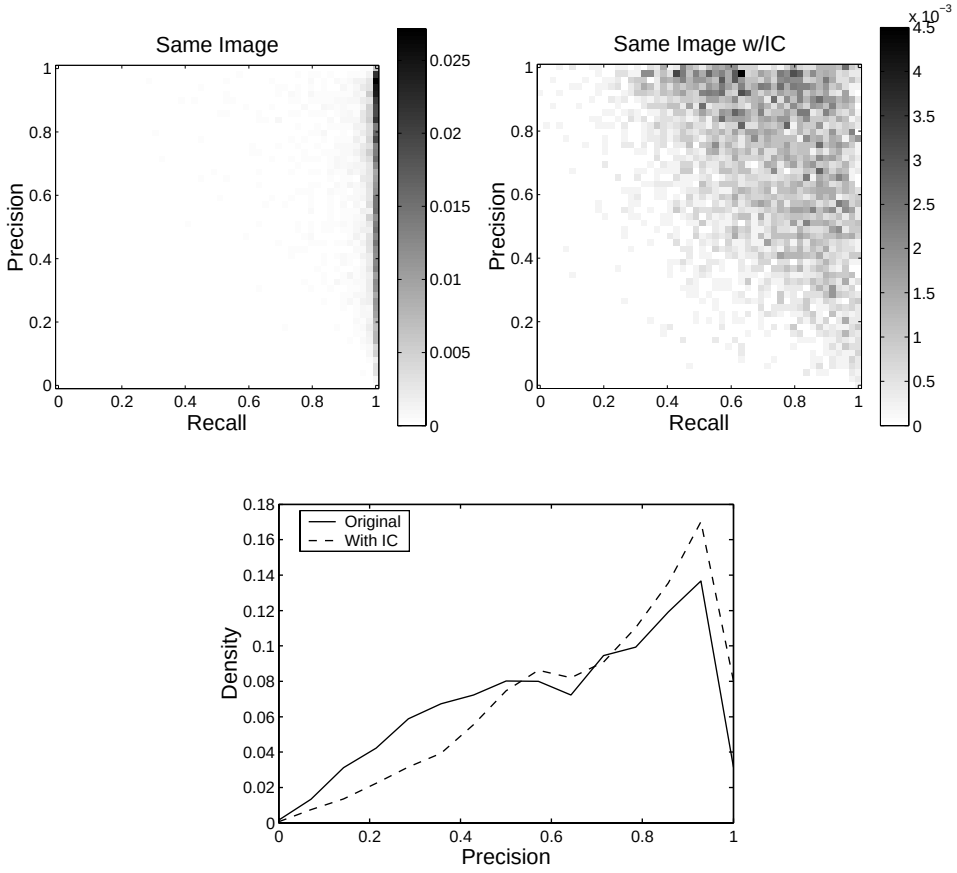


Figure 5.5: Intervening Contour and Refinement. The potential over-segmentation caused by the intervening contour approach agrees with the refinement of objects by human observers. The top left graph shows the distribution of precision and recall for all color segmentations in the dataset, evaluated using the leave-one-out method. The right graph adds the constraint that the same-segment pairs from the left-out human must not have an intervening boundary contour. The recall naturally decreases from adding a constraint to the "signal". From the marginal distributions shown in the lower graph, we see that precision *increases* with the added constraint. Because the union segmentation is, on average, a refinement of the left-out segmentation, intervening contour tends to break non-convex regions in a manner similar to the human subjects.

Patch Radius

	Brightness		Color		Texture	
Radius	F	MI	F	MI	F	MI
0.010	0.46	.069	0.49	.093	-	-
0.014	0.46	.071	0.50	.096	-	-
0.020	**0.46**	**.074**	**0.50**	**.097**	-	-
0.028	0.46	.073	0.50	.097	0.50	.097
0.040	0.46	.071	0.49	.095	0.53	0.11
0.056	0.45	.067	0.48	.091	**0.55**	**0.12**
0.080	-	-	-	-	0.53	0.11
0.112	-	-	-	-	0.50	.089

Kernel Density Estimate

		Brightness		Color	
Sigma	Bins	F	MI	F	MI
0.025	100	0.40	.041	0.48	.085
0.05	50	0.41	.047	0.50	.094
0.1	25	0.44	.059	**0.50**	**.097**
0.2	12	0.46	.070	0.491	.094
0.4	**6**	**0.46**	**.073**	0.49	.087
0.8	3	0.45	.062	0.48	.083

Number of Textons

Num	F	MI
32	0.48	.081
64	0.50	.093
128	0.52	0.10
256	0.53	0.11
512	**0.55**	**0.12**
1024	0.52	0.10
2048	0.52	0.10

Texton Filter Bank

Scale	F	MI
0.007	**0.55**	**0.12**
0.010	0.55	0.11
0.014	0.53	0.11
0.020	0.51	.092
0.028	0.47	.072
0.007-0.014	0.55	0.12
0.010-0.020	0.53	0.11
0.014-0.028	0.51	.091

Figure 5.6: Optimization of Patch Features. The parameters of the patches were optimized on the 250-image training set so that no change in any single parameter improves performance. The optimal patch sizes and filter scales are in units of the image diagonal, which is 288 pixels for our 240x160 images. The accessible ranges of the L*a*b* color axes were scaled to $[0, 1]$, which is the scale for the σ parameter. The Gaussian kernel was sampled at 21 points from $[-2\sigma, 2\sigma]$. We must reduce the number of bins as σ increases to keep the number of samples per bin constant. In the lower right table, the multi-scale texton filter bank contains three half-octave scales covering the range shown.

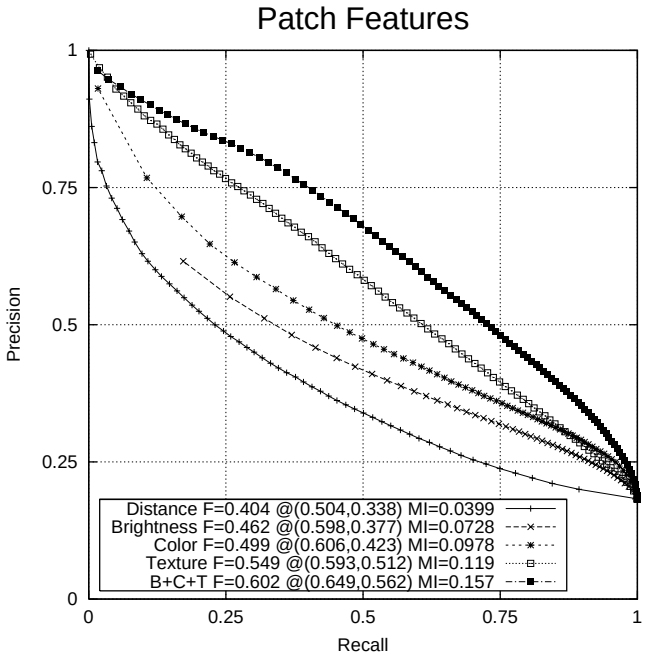


Figure 5.7: Precision-Recall of Patch Features. The lower curve shows the performance of the distance cue for reference. The next three curves show the performance of the brightness, color, and texture cues respectively. The top curve shows the result of combining the three cues with a logistic model. Texture is the most powerful cue, but much benefit is gained by using all three.

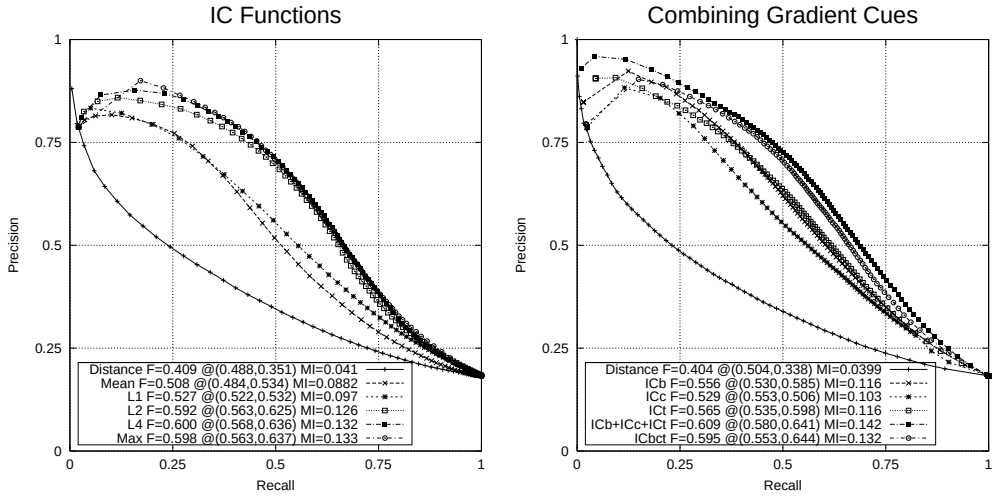


Figure 5.8: Intervening Contour Cues. The comparison of different intervening contour functions (left) shows that L^∞ is superior. The right graph explores the two ways we can combine different intervening contour cues into a single contour cue. We can either compute the intervening contour feature for brightness (ICb), color (ICc), and texture (ICt) separately and then combine with a classifier (ICb+ICc+ICt), or we can use the P_b function that combines the three channels into a single boundary map for the intervening contour feature (ICbct). We achieve better performance by computing separate contour cues.

with a classifier that uses all three. It is clear that each patch feature contains independent information, and that texture is the most powerful cue.

5.4.3 Performance of Gradients

The gradient cues are based on P_b images, which give the posterior probability of a boundary at each image location. The P_b function can incorporate any or all of the brightness, color, and texture cues, though consider for the moment the version that uses all three. Which intervening contour function should we use? The left panel of Figure 5.8 shows the performance of various functions including the mean, and the range of $L^p(\Gamma)$ functions from sum to max. The L^∞ version is clearly the best approach. Both the mean and the sum perform significantly worse. The results are the same no matter which cues the P_b function uses. Note that the max does not include any encoding of distance.

The right panel of Figure 5.8 compares the two ways in which we can combine the contour cues. We can either compute the intervening contour feature for brightness, color, and texture separately and then combine with a classifier, or we can use the P_b function that combines the three channels into a single boundary map for the intervening contour feature (ICbct). We achieve slightly better

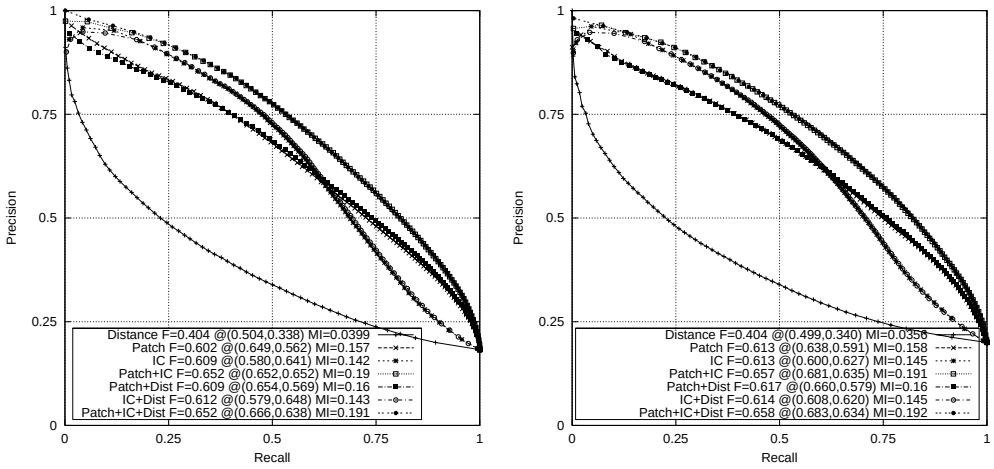


Figure 5.9: Distance as a Grouping Cue. In this figure, we investigate the utility of the distance between two pixels as a cue for grouping. The Gestalt school identified proximity as a grouping cue, however, in all cases the classifier performance is the same whether or not distance is used. The right panel shows the same experiment with the test and training sets swapped. We performed all our experiments with swapped sets. Results were always consistent, with the F-measure and mutual information accurate to within two decimal places.

performance using the three individual contour cues than using the combined P_b.

5.4.4 Cue Combination

We now have seven prospective cues for our model of the pixel affinity, though we expect some to be redundant. The cues are brightness, color, and texture patches, intervening contour from the same three channels, and the distance between the two pixels in the image plane. We first evaluate the power of the distance cue in Figure 5.9. Whether we use a patch-only model, a contour-only model, or a patch+contour model, the result is always the same: distance does not add any information beyond that already provided by similarity cues.

We expect that the superiority of patch versus contour cues to differ depending on the feature channel. Smooth shading and foreshortening effects may favor brightness and texture gradients, while it is "well known" that color patches are a stable cue. Figure 5.10 shows the patch-only, contour-only, and patch+contour models for each of the brightness, color, and texture channels. As expected, the brightness patch proves to be far weaker than the brightness contour cue, with only marginal benefit from combining the two. Neither patches nor contours seem to dominate the color or texture channels. However, both texture cues appear quite powerful with independent information.

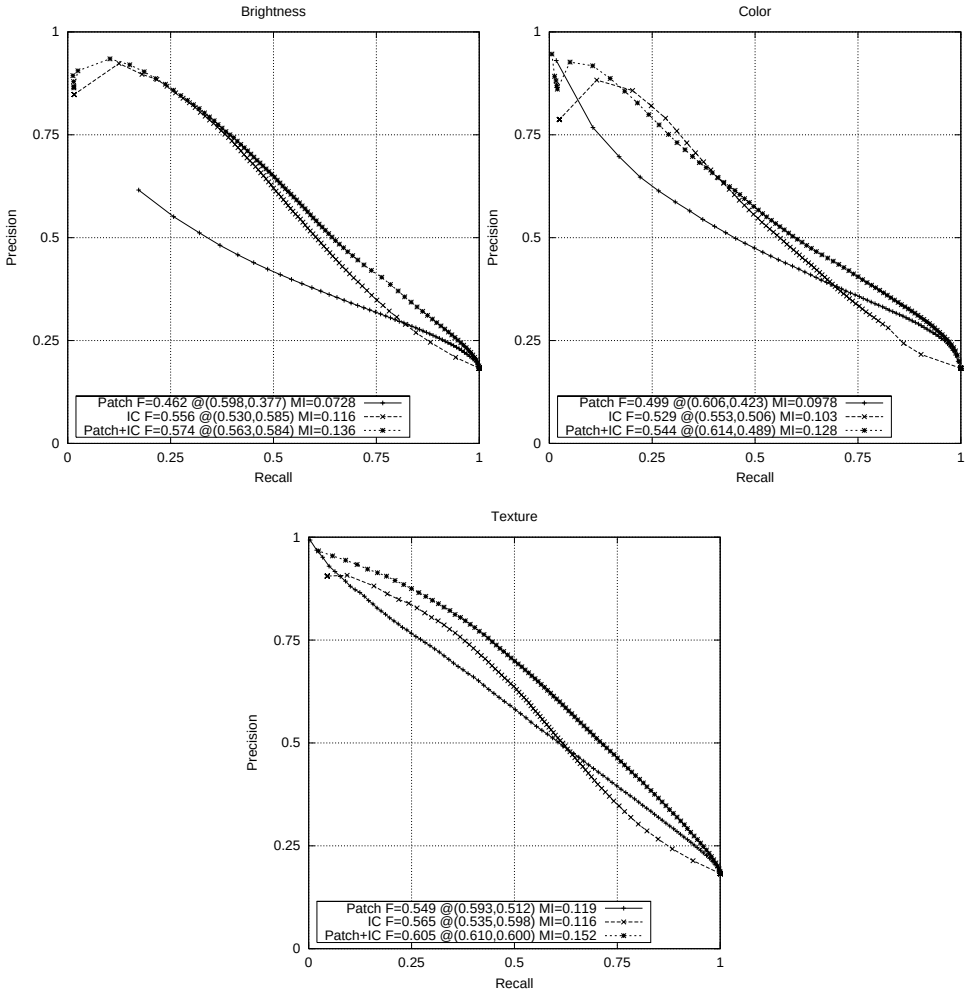


Figure 5.10: Per-Cue Patch and Gradient Combinations. The three plots show classifiers that use either brightness (top left), color (top right), or texture (bottom). Each plot shows the performance of a classifier using the patch cue, the gradient cue, and both together. The brightness patch appears an especially weak cue, which can be expected from the frequency of shading gradients in images. Both texture patches and texture gradients are powerful cues, and their combination is everywhere superior to using one alone.

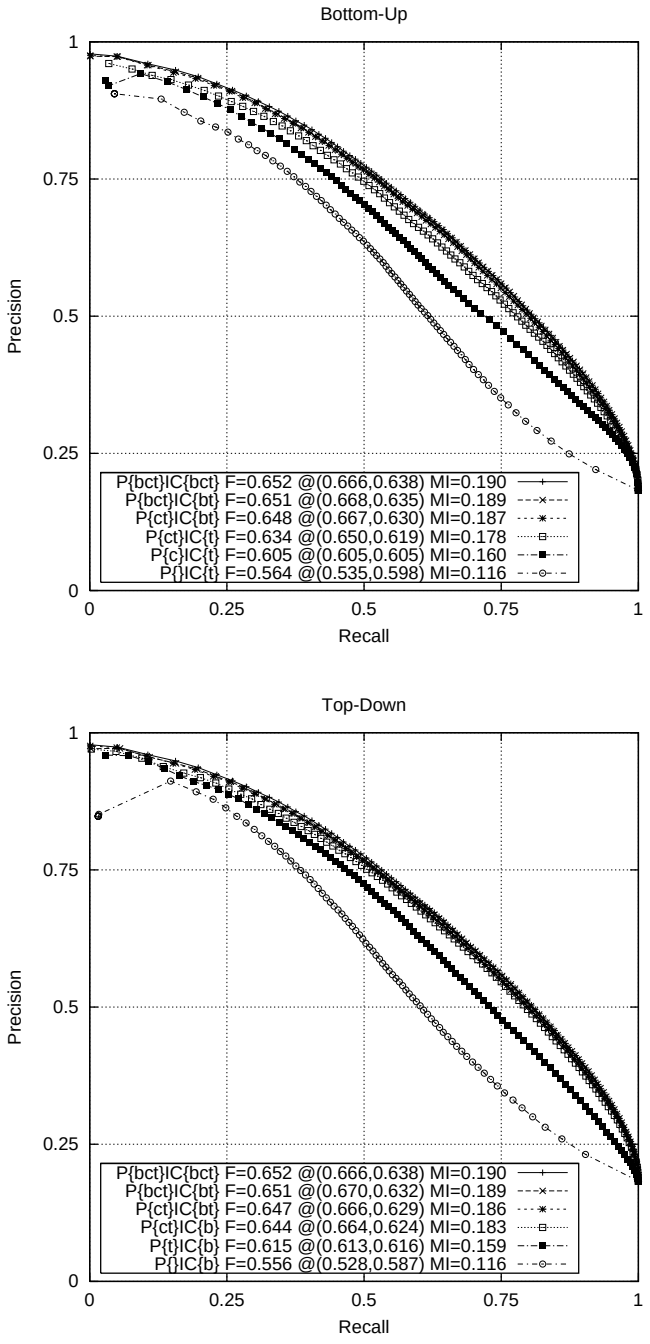


Figure 5.11: Feature Pruning. In the upper panel, we start with no features and add one feature at a time to the model in order to maximize performance in a greedy manner. In the lower panel, we start with all six features, and greedily remove the worst feature, one feature at a time. In both cases, the model of choice uses the brightness gradient, color patches, and both texture cues. The brightness patch and color gradient are weak cues, while the color patch and both texture cues are powerful.

In order to determine the most fruitful combination of cues, we executed both top-down and bottom-up feature pruning experiments. Figure 5.11 shows the result. In both cases, the model that maximizes performance using the fewest cues is the 4-cue model containing the brightness contour cue, the color patch cue, and both texture cues. All three feature channels — brightness, color, and texture — are represented, with particular emphasis on texture. From the bottom-up pruning, it is clear that the texture cues are the most powerful along with color patches. It is interesting to see that at all stages in the pruning experiments, the model contains a balance between patch and contour cues, as well as a balance between the three feature channels.

5.4.5 Choice of Classifier

As in Chapter 4, the choice of classifier is not important. Performance was always nearly identical whether we used a non-parametric density estimation method, or parametric models based on logistic regression, including simple logistic regression, logistic regression with quadratic features, or hierarchical mixtures of experts. To a first order approximation, a linear combination of features is sufficient. We favor the logistic with quadratic terms since it yields a slight improvement over the linear logistic function with little added computational cost.

5.5 Conclusion

We have shown how to combine patch and contour information into a model of pixel affinity for the purpose of image segmentation. For both patches and contours, we formulate brightness, color, and texture cues based on histogram differences. Contour cues are constructed in the intervening contour framework, which is justified by the ecological statistics of human segmentations. The six cues are carefully optimized with respect to a large dataset of manually segmented natural images, and then combined with a classifier trained on the ground truth data.

We can now revisit the so-called "folk-beliefs" listed in the introduction:

- *Brightness gradients (caused by shading) and texture gradients (caused by perspective) necessitate a boundary-based approach.* This is not true, since the patch-based features are as powerful as the boundary-based features.

- *Edge detectors are confused by texture, so one must use patch-based similarity for texture segmentation.* This is patently false, as Chapter 4 demonstrates that we can incorporate texture into a boundary detector. In fact, Figure 5.11 shows that in this chapter, the texture gradient is our most powerful cue!

- *Color integrated over local patches is a robust and powerful cue.* It does appear that color patches are more effective than color edges when used in conjunction with other cues: the most effective three-cue models in Figure 5.11 include the color patch feature. However, Figure 5.10 indicates that when color is used alone, the color gradient is more effective than color patches. This is likely due to the existence of color gradients across objects. Though not as common as brightness gradients, which essentially cripple the brightness patch cue, color gradients do exist in the natural world. In addition to albedo-induced gradients, color gradients are caused by mutual illumination and shading.

- *Proximity is a good indicator for grouping.* Proximity is far less powerful than any of our patch-based or gradient-based cues. In addition, the proximity cue does not improve the performance of models that use other cues. It appears that proximity does not add any information for grouping beyond that provided by local image-based features.

Figure 5.1 on page 98 shows the performance of our final pixel affinity models compared to the human dataset. Both precision-recall and mutual information measures indicate that patches and gradients contain significant independent information. The combined model outperforms the patch-only and gradient-only models, but the result falls short of human performance. The gap may be closed by incorporating mid- and high-level cues into the model of pixel affinity, such as closure, convexity, symmetry, and familiar configuration.

Chapter 6

Summary and Conclusion

I have presented the Berkeley Segmentation Dataset, a large and unique dataset of natural images segmented by human subjects. Figure 6.1 shows a sample of the data. The images are of complex, natural scenes, and there are multiple segmentations of each image. I have shown in Chapter 2 that the segmentations of an image by different subjects are consistent under a model of refinement. Even without allowing refinement, however, the different segmentations are surprisingly similar. This agreement is visible in Figure 6.1, where darker lines signify that a boundary was marked by more subjects. We confidently reject the common assertion that segmentation is an ill-posed problem.

The segmentation data is a valuable source of information for the ecological statistics of human vision. As we assume that the human visual system exploits statistical regularity in the visual stimulus, we aim to do the same in our machine vision algorithms. To this end, I have outlined in this thesis a process by which we can systematically learn models of perceptual organization from data. The human data provides ground truth labels for various sub-problems. Statistical learning machines provide the means of fitting computational models in a statistically optimal manner.

We have taken significant steps toward a completely data-driven segmentation algorithm by learning a local boundary model in Chapter 4 and a pixel affinity model in Chapter 5. The results of the learned boundary model show that the data-driven approach is valuable. Figure 6.2 shows the performance of the circa 1980 Canny detector with and without hysteresis, the circa 1990 state-of-the-art Nitzberg/Förster/Harris edge detector based on the second moment matrix, and our circa 2000 detector. All of the models were trained exhaustively on the same dataset.

First note that hysteresis improves the Canny detector only marginally. We are confident that the majority of boundary detection work that populates the literature of the 1980s operates in this narrow range of performance. The second moment matrix detector is a slight improvement, but our gradient-

Figure 6.1: Example Images and Human-Marked Segment Boundaries. Each image shows multiple (4 to 8) human segmentations. The boundaries are darker where more humans marked an edge.

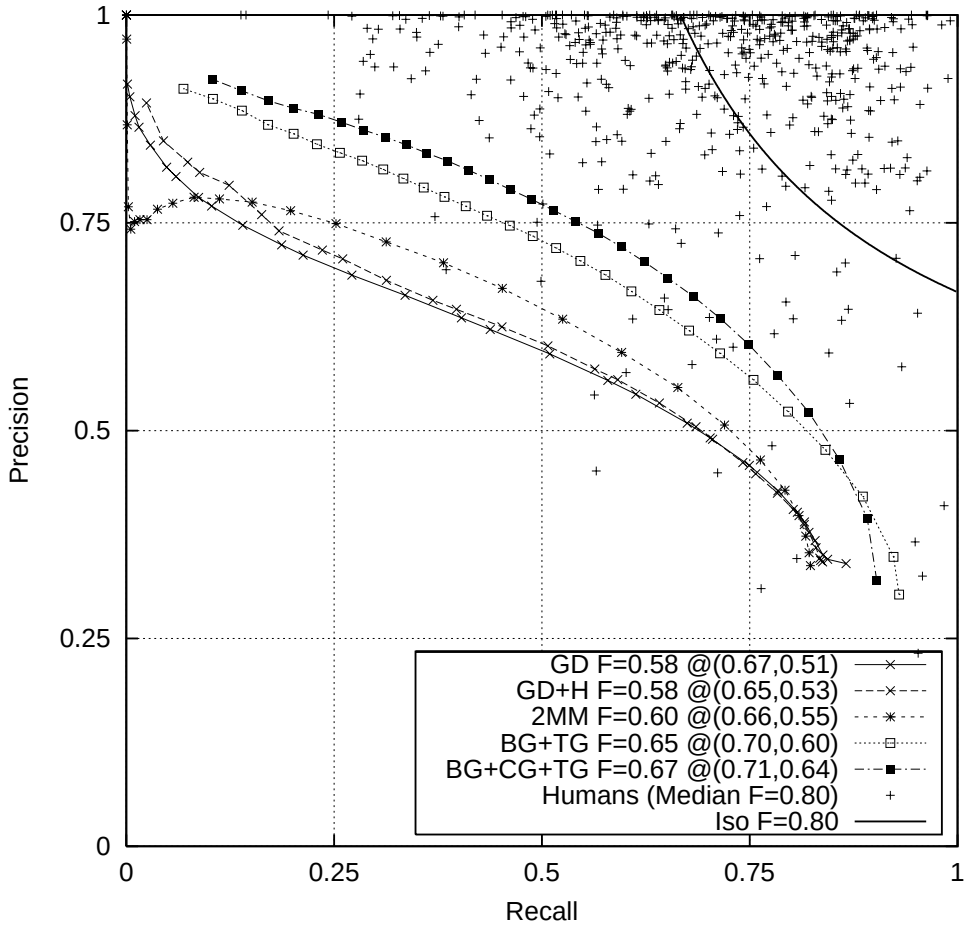


Figure 6.2: Two Decades of Boundary Detection Research. This figure shows the performance of our boundary detectors compared to classical boundary detection methods and to the human subjects' performance. *Precision-recall* curves, described in Section 3.1.2, are shown for five boundary detectors: (1) Gaussian derivative (GD), (2) Gaussian derivative with hysteresis thresholding (GD+H), the Canny detector, (3) A detector based on the second moment matrix (2MM), (4) our grayscale detector that combines brightness and texture (BG+TG), and (5) our color detector that combines brightness, color, and texture (BG+CG+TG). Each detector's curve shows the tradeoff between accuracy and noise as the detector's threshold varies. Shown in the caption is each curve's maximal F-measure, valued from zero to one, along with the coordinates of the location of the maximum. The points on the plot marked with "+" show the precision and recall of each ground truth human segmentation when compared to the other humans. The median F-measure for the human subjects is 0.80. The solid curve shows the F=0.80 precision-recall curve, representing the frontier of human performance for this task.

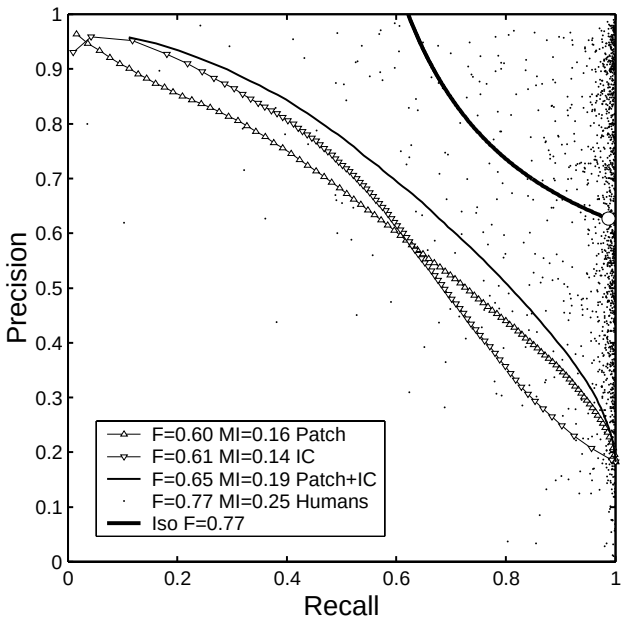


Figure 6.3: Performance of Pixel Affinity Models. This figure shows the performance of our best pixel affinity models compared to the human subjects. The dots show the precision and recall of each of 1366 human segmentations in the 250-image test set when compared to the other humans' segmentations of the same image. The large dot marks the median recall (99%) and precision (63%) of the humans. The iso-F-measure curve at F=77% is extended from this point to represent the frontier of human performance for this task. The three remaining curves represent our patch-only model, contour-only model, and patch+contour model. Neither patches nor contours are sufficient, as there is significant independent information in the patch and contour cues. The model is a logistic function with quadratic terms, which performs best among classifiers tried on this dataset.

based models show a marked increase in performance. The improvement in the gradient detector comes from two factors: (1) an explicit model of texture edges, and (2) an effective combination of brightness, color, and texture cues. It is interesting to note that it does not matter which model one uses to combine the three cues. A simple logistic model performs as well as decision trees, density estimation, boosted logistic regression, hierarchical mixtures of experts, and support vector machines.

One contribution of this work is that we have significantly improved the quality of boundary detection on natural images. However, it is perhaps the methodology that is most important. It is only through rigorous empirical evaluation using precision-recall curves that we were able to guide our gradient-based boundary model to higher performance. Without the boundary-detection micro-benchmark provided by the dataset, we would not have achieved the gains that we did.

Now that we have a boundary detector that works well on complex, natural images, we take the next step towards segmentation. Working in Chapter 5 within the popular framework of pairwise clustering for segmentation, we used the human data to learn a model of pixel affinity. This model is based on both the gradient-based features of Chapter 4, and simple patch-based features. We formulated seven features for the pixel affinity model: image plane distance, patch similarity for brightness, color, and texture, and intervening contour features based on the brightness, color, and texture gradients. Figure 6.3 shows the precision-recall curves for three of our affinity models compared to the human data: patch-only, gradient-only, and patch+gradient. It is clear that both patches and gradients provide independent information for the task. The gap to human performance must be bridged by additional mid-level and high-level cues.

Our primary goal in Chapter 5 was to determine which features were useful for defining pixel affinity. The model that maximizes performance by using the fewest features uses both texture cues, the color patches, and the brightness gradient. In performing the experiments to optimize each cue and explore cue combinations, we reach conclusions that contradict several commonly held beliefs about segmentation:

1. Neither a patch-based nor gradient-based approach is superior for segmentation. Both contain significant independent information, so our model of pixel affinity uses both patch and gradient cues.

2. Edge detectors *can* deal with texture. If this was not clear from Chapter 4, then it should be clear given that the most powerful cue in Chapter 5 was the texture gradient. Texture is therefore not a reason to favor patch-based approaches.

3. Color patches are not a silver bullet for segmentation. If color is used alone, then the color gradient performs better. If other cues are used, then the color patches are preferable, but the texture cues are far more powerful. Color is not a strong signal in natural images both because saturated colors are scarce, and because of the presence of color gradients.

4. Proximity is not a good indicator for grouping. All of our other cues soundly out-performed proximity. Once one considers the content of the image, proximity does not provide additional information.

In addition to these conclusions, we also find that the intervening contour approach is supported by the dataset in the sense that segments are convex, especially at small scales. Furthermore, we find that intervening contour tends to break up segments in a manner similar to the way our subjects refine segmentations. This finding supports an approach to segmentation that first finds convex object parts, and then groups the object parts into whole objects.

That our results contradict several common beliefs in computer vision shows the true value of an empirical approach to computer vision. Introspection and intuition are valuable tools, but they can lead to erroneous conclusions if ideas are not tested scientifically.

The next logical step in this line of work is to derive full segmentation algorithms from the data, and to enforce additional Gestalt grouping cues such as curvilinear continuity, closure, convexity, symmetry, and familiar configuration. This is future work, but we have developed the necessary quantitative methods in Chapter 3. For example, contour completion algorithms can be evaluated by measuring how much the boundary map improves, using precision-recall curves computed by edgel matching. Pairwise clustering algorithms can be evaluated by how much they improve the pixel affinity function, using either mutual information or precision-recall. Given that we have micro-benchmarks for both boundary detection and pixel affinity, we can quantitatively evaluate arbitrary black-box improvements to these modules.

None of this work is possible without human data. We feel that computer vision must become more data-driven. Though time-consuming, we must study human subjects and create datasets. Fortunately, we found that a single dataset can suffice for evaluating many aspects of a task if the dataset is large, high-level, and based on human perception. For image segmentation, we collected high-level object-based segmentations, and were therefore able to create micro-benchmarks for the different elements of segmentation algorithms. This provided the perfect environment for developing robust low-level modules, since the high-level nature of the data introduces "noise" with the proper ecological statistics.

We are confident that this empirical approach will continue to produce significant improvements in general-purpose image segmentation, as well as further insights into perceptual organization. Of particular interest are the problems of figure-ground assignment, hierarchy in perceptual organization, and class-specific recognition, though all areas of computational vision would benefit from a data-driven methodology. As computer vision researchers, we aim to understand human vision and build systems that achieve human-level performance on vision tasks. It seems only sensible to mark the target with data, search the data for patterns, and measure our progress with benchmarks.

Appendix A

Computing Gradient Features

The most computationally expensive part of the gradient computations of Chapters 4 and 5 is the computation of the half-disc feature histograms. At each pixel, we must compute two histograms over semi-circular neighborhoods at several orientations and several scales. Properly structured, this computation can be done efficiently.

The most significant speedup is achieved by optimizing the loop over orientations. Assuming that we wish to compute the gradient at n evenly spaced orientations, we can divide the disc into $2n$ pie slices. If we compute the pixel histogram for each pie slice, then any half-disc histogram is simply the sum of n adjacent pie slice histograms. In addition, we can compute the histograms for orientation $i + 1$ incrementally from the histograms from orientation i by subtracting the last slice and adding the next slice as we spin the disc. Note also that the initial step of computing the pie slice histograms can be optimized by precomputing a slice membership mask.

For the texture gradient, these optimizations are sufficient. However, the soft binning required by the brightness and color gradients present other opportunities for speedup. Each pixel contributes multiple points to the histogram, one for each kernel sample. Precomputing kernel offsets and values is effective, though this approach is slow if the number of kernel samples is large. If there are more kernel samples than bins, then one should precompute the total histogram contribution from each pixel.

Other loops admit additional optimization opportunities. In the same way that we split the disc by orientation into pie slices, one could additionally split the disc into concentric rings corresponding to the multiple scales. Since our half-octave scales produce an area increment for the disc of 2x per scale, our computation is dominated by the largest scale. A smaller scale increment would justify this optimization.

There is still much redundant computation as we sweep the disc across a scan-line. The pie slice

histograms change slowly between adjacent pixels, especially when the number of orientations is not large. It is possible to compute the slice histograms incrementally by computing slice update masks. For large radii, this optimization achieves an order of magnitude speedup.

Appendix B

Multinomial Logistic Regression

The hierarchical mixtures of experts (HME) model of Jordan and Jacobs [30] is a mixture model where both the experts at the leaves and the internal nodes that comprise the gating network are logistic functions. One can build an HME model with a binary logistic regression module, however in that case one is restricted to a binary HME model with a binary tree gating network. We were interested in more general models, and so we implemented a general multinomial HME. To do this, we required an implementation of multinomial logistic regression. However, we could not find an implementation, and the literature did not contain a rigorous derivation that we could find. This appendix presents the derivation for the Newton-Raphson update rules for fitting a multinomial logistic model.

Let the probability that data point x was generated by class j be:

$$p_j(x, \beta) = \frac{e^{\beta_j^T x}}{\sum_m e^{\beta_m^T x}} \tag{B.1}$$

β denotes the collection $\{\beta_1, \beta_2, \ldots, \beta_K\}$.

The gradient of this function (which we will use later) is:

$$\begin{aligned}
\frac{\partial}{\partial \beta_i} p_j(x, \beta) &= \frac{\partial}{\partial \beta_i} \frac{e^{\beta_j^T x}}{\sum_m e^{\beta_m^T x}} \\
&= \frac{\left(\delta_{ij} x e^{\beta_j^T x}\right)\left(\sum_m e^{\beta_m^T x}\right) - \left(e^{\beta_j^T x}\right)\left(x e^{\beta_i^T x}\right)}{\left(\sum_m e^{\beta_m^T x}\right)^2} \\
&= \frac{e^{\beta_j^T x}}{\sum_m e^{\beta_m^T x}} \left[\delta_{ij} - \frac{e^{\beta_i^T x}}{\sum_m e^{\beta_m^T x}}\right] x \\
&= p_j(x, \beta) \left[\delta_{ij} - p_i(x, \beta)\right] x
\end{aligned} \tag{B.2}$$

To fit this model we need the likelihood for N IID samples $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ with sample weights $\{w_1, \ldots, w_n\}$. y_n is a soft class membership vector with unit L1 norm where y_n^k gives the probability of membership of x_n to class k. The w_n are scalar weights in $[0, 1]$.

The likelihood is:

$$L(\beta) = \prod_n \prod_k p_k(x_n, \beta)^{w_n y_n^k} \tag{B.3}$$

The log likelihood is:

$$\ell(\beta) = \sum_n \sum_k w_n y_n^k \log p_k(x_n, \beta) \tag{B.4}$$

We would like to maximize the log likelihood using the Newton-Raphson method, for which we will need the gradient and Hessian of ℓ with respect to each β_j.

First, the gradient vector:

$$\begin{aligned}
\frac{\partial \ell}{\partial \beta_j} &= \frac{\partial}{\partial \beta_j} \sum_n \sum_k w_n y_n^k \log p_k(x_n, \beta) \\
&= \sum_n \sum_k w_n \frac{y_n^k}{p_k(x_n, \beta)} \frac{\partial}{\partial \beta_j} p_k(x_n, \beta) \\
&= \sum_n \sum_k w_n \frac{y_n^k}{p_k(x_n, \beta)} p_k(x_n, \beta) \left[\delta_{jk} - p_j(x_n, \beta)\right] x_n \\
&= \sum_n w_n x_n \left[\sum_k y_n^k \delta_{jk} - p_j(x_n, \beta) \sum_k y_n^k\right] \\
&= \sum_n w_n x_n \left[y_n^j - p_j(x_n, \beta)\right]
\end{aligned} \tag{B.5}$$

Second, the Hessian matrix:

$$\begin{aligned}
\frac{\partial^2 \ell}{\partial \beta_i \partial \beta_j^T} &= \frac{\partial}{\partial \beta_i} \left(\frac{\partial \ell}{\partial \beta_j}\right)^T \\
&= \frac{\partial}{\partial \beta_i} \sum_n w_n x_n^T \left[y_n^k - p_j(x_n, \beta)\right] \\
&= -\sum_n w_n x_n^T \frac{\partial}{\partial \beta_i} p_j(x_n, \beta) \\
&= -\sum_n w_n p_j(x_n, \beta) \left[\delta_{ij} - p_i(x_n, \beta)\right] x_n x_n^T
\end{aligned} \tag{B.6}$$

The Newton-Raphson update step is:

$$\beta^{t+1} = \beta^t - \left(\frac{\partial^2 \ell(\beta^t)}{\partial \beta^2}\right)^{-1} \frac{\partial \ell(\beta^t)}{\partial \beta} \tag{B.7}$$

The update step in matrix notation is:

$$\theta^{t+1} = \theta^t - \mathbf{H}^{-1}\mathbf{g} \tag{B.8}$$

where H and g are the Hessian and gradient for the pooled β_j:

$$\theta^T = [\beta_1^T \beta_2^T \cdots \beta_K^T] \tag{B.9}$$

$$\mathbf{g}^T = [g_1^T g_2^T \cdots g_K^T] \tag{B.10}$$

$$\mathbf{H} = \begin{bmatrix} H_{11} & H_{12} & \cdots & H_{1K} \\ H_{21} & H_{22} & & H_{2K} \\ \vdots & & \ddots & \vdots \\ H_{K1} & H_{K2} & \cdots & H_{KK} \end{bmatrix} \tag{B.11}$$

$$g_i = \frac{\partial \ell}{\partial \beta_i} \tag{B.12}$$

$$H_{ij} = \frac{\partial^2 \ell}{\partial \beta_i \partial \beta_j^T} \tag{B.13}$$

We can write the gradient and Hessian sub-blocks in matrix notation:

$$g_i = \mathbf{X}\mathbf{W}(\mathbf{y}^i - p_i(\mathbf{X}, \beta))^T \tag{B.14}$$

$$H_{ij} = -\mathbf{X}\mathbf{W}\mathbf{Z}_{ij}\mathbf{X}^T \tag{B.15}$$

where $\mathbf{y}^i = [y_1^i y_2^i \cdots y_N^i]$, $\mathbf{X} = [x_1 x_2 \cdots x_N]$, $\mathbf{W}$ is a diagonal matrix containing the sample weights w_n, and $\mathbf{Z}_{ij}$ is a diagonal matrix with its n'th entry equal to:

$$\mathbf{Z}_{ij}^n = p_j(x_n, \beta)\,[\delta_{ij} - p_i(x_n, \beta)] \tag{B.16}$$

B.1 Implementation Details

The HME requires a particularly robust logistic regression implementation. In an over-specified HME, the unneeded experts tend to be marginalized by the gating network, producing degenerate and numerically unstable problems to the experts. Simply implementing the Newton-Raphson iteration is not sufficient. Two problems can arise.

The first problem is that of overshoot. The Newton-Raphson update step assumes that the function is locally close to parabolic. This is normally a reasonable assumption, however it is possible to construct a situation where Newton-Raphson will not converge. The situation is easy to detect, however. Since the likelihood should decrease at each iteration, an increase in the likelihood can

only occur if the step overshot a local minimum. Step-size halving ensures rapid convergence.

The second problem is that the Hessian can become ill-conditioned, so that its inverse cannot be computed. This condition can be detected when the condition number of the Hessian is large. The Levenberg-Marquardt method deals with this situation by iteratively increasing the magnitude of the diagonal entries until the Hessian is sufficiently conditioned.

Bibliography

[1] I. Abdou and W. Pratt. Quantitative design and evaluation of enhancement/thresholding edge detectors. *Proc. of the IEEE*, 67(5):753–763, May 1979.

[2] L. Alvarez, Y. Gousseau, and J. Morel. Scales in natural images and a consequence on their bounded variation norm. *Scale-Space Theories in Computer Vision*, pages 247–258, 1999.

[3] K. Bowyer, C. Kranenburg, and S. Dougherty. Edge detector evaluation using empirical ROC curves. *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, I:354–359, 1999.

[4] E. Brunswik and J. Kamiya. Ecological validity of proximity and other Gestalt factors. *Am. J. Psych.*, 66:20–32, 1953.

[5] R. W. Buccigrossi and E. P. Simoncelli. Image compression via joint statistical characterization in the wavelet domain. *IEEE Trans. on Image Proc.*, 8(12):1688–1701, December 1999.

[6] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[7] G. Carpaneto, S. Martello, and P. Toth. Algorithms and codes for the assignment problem. *Annals of operations research*, 13:193–223, 1988.

[8] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(8):1026–1038, 2002.

[9] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[10] B. V. Cherkassky and A. V. Goldberg. On implementing push-relabel method for the maximum flow problem. *4th Integer Programming and Combinatorial Optimization Conference*, pages 157–171, May 1995.

[11] Berkeley Segmentation Dataset, 2002. http://www.cs.berkeley.edu/projects/vision/bsds.

[12] J. Elder and S. Zucker. Computing contour closures. *Europ. Conf. Comput. Vision*, I:399–412, April 1996.

[13] P. Felzenszwalb and D. Huttenlocher. Image segmentation using local variation. *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, pages 98–104, 1998.

[14] I. Fogel and D. Sagi. Gabor filters as texture discriminator. *Bio. Cybernetics*, 61:103–113, 1989.

[15] W. Förstner and E. Gulch. A fast operator for detection and precise locations of distinct points, corners, and centres of circular features. *Intercommission Conference on Fast Processing of Photogrammetric Data*, pages 281–305, 1987.

[16] C. Fowlkes, D. Martin, and J. Malik. Understanding Gestalt cues and ecological statistics using a database of human segmented images. *Perceptual Organization for Computer Vision Workshop, Intl. Conf. on Computer Vision*, 2001.

[17] Y. Gdalyahu, D. Weinshall, and M. Werman. Stochastic image segmentation by typical cuts. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(10):1053–1074, 2001.

[18] S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution, and the Bayesian retoration of images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6:721–41, November 1984.

[19] A. Goldberg. *CSA: an efficient implementation of a scaling push-relabel algorithm for the assignment problem*. Software available at http://www.avglab.com/andrew/soft.html.

[20] A. Goldberg, 12 July 2002. Personal Communication.

[21] A. Goldberg and R. Kennedy. An efficient cost scaling algorithm for the assignment problem. *Mathematical Programming*, 71:153–178, December 1995.

[22] C. Harris and M. J. Stephens. A combined corner and edge detector. *Proc. 4th Alvey Vision Conference*, pages 147–151, 1988.

[23] D. Heeger and J. Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of SIGGRAPH '95*, pages 229–238, 1995.

[24] L. Hermes, T. Zoller, and J. Buhmann. Parametric distributional clustering for image segmentation. *Europ. Conf. Comput. Vision*, pages 577–591, 2002.

[25] J. Huang. *Statistics of Natural Images and Models*. PhD thesis, Brown University, May 2000.

[26] J. Huang, A. B. Lee, and D. Mumford. Statistics of range images. *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, I:324–331, 2000.

[27] J. Huang and D. Mumford. Statistics of natural images and models. *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, I:541–547, 1999.

[28] I. Jermyn and H. Ishikawa. Globally optimal regions and boundaries as minimum ratio weight cycles. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(10):1075–1088, 2001.

[29] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38:325–340, 1987.

[30] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[31] S. Konishi, A. L. Yuille, J. Coughlan, and S. C. Zhu. Fundamental bounds on edge detection: an information theoretic evaluation of different edge cues. *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, pages 573–579, 1999.

[32] M. Landy and H. Kojima. Ideal cue combination for localizing texture-defined edges. *J. Opt. Soc. Am. A*, 18(9):2307–2320, September 2001.

[33] A. B. Lee and D. Mumford. Scale-invariant random-collage model for natural images. *Proc. IEEE Workshop on Statistical and Computational Theories of Vision*, 1999.

[34] T. Leung and J. Malik. Contour continuity in region-based image segmentation. *Europ. Conf. Comput. Vision*, I:544–59, June 1998.

[35] E. Levina. *Statistical Issues in Texture Analysis*. PhD thesis, University of California, Berkeley, 2002.

[36] E. Levina and P. Bickel. The Earth Mover's distance is the Mallows distance: some insights from statistics. *Int'l. Conf. Computer Vision*, II:251–256, 2001.

[37] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *Int'l. Journal of Computer Vision*, 43(1):7–27, June 2001.

[38] J. Malik and P. Perona. Preattentive texture discrimination with early vision mechanisms. *J. Opt. Soc. Am.*, 7(2):923–932, May 1990.

[39] R. Malladi, J. Sethian, and B. Vemuri. Shape modelling with front propogation: A level set approach. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(2):158–175, 1995.

[40] C. Mallows. A note on asymptotic joint normality. *Annals of Mathematical Statistics*, 43(2):508–515, 1972.

[41] C. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT Press, 2001.

[42] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using brightness and texture. *Neural Information Processing Systems*, 2002. In press.

[43] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *Int'l. Conf. Computer Vision*, II:416–423, July 2001.

[44] M. Meilă and J. Shi. Learning segmentation by random walks. *Neural Information Processing Systems*, 13:873–879, 2000.

[45] J. Morel and S. Solimini. *Variational Methods in Image Segmentation*. Birkhäuser, 1995.

[46] M. Morrone and D. Burr. Feature detection in human vision: a phase dependent energy model. *Proc. R. Soc. Lond. B*, 235:221–2245, 1988.

[47] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions, and associated variational problems. *Comm. in Pure and Applied Math.*, pages 577–684, 1989.

[48] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The QBIC project: Querying image by content using color, texture, and shape. *Proceedings of the SPIE - The International Society for Optical Engineering v.1908 (Storage and Retrieval for Image and Video Databases)*, 1908:173–187, 1993.

[49] M. Nitzberg, D. Mumford, and T. Shiota. *Filtering, Segmentation, and Depth*. Springer-Verlag, 1993.

[50] S. Palmer. *Vision Science*. MIT Press, 1999.

[51] P. Parent and S. Zucker. Trace inference, curvature consistency, and curve detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 11(8):823–839, August 1989.

[52] P. Perona and W. Freeman. A factorization approach to grouping. *Europ. Conf. Comput. Vision*, pages 655–670, June 1998.

[53] P. Perona and J. Malik. Detecting and localizing edges composed of steps, peaks and roofs. *Int'l. Conf. Computer Vision*, pages 52–57, December 1990.

[54] NASA Clickworkers Project, 2002. http://clickworkers.arc.nasa.gov.

[55] J. Puzicha, T. Hofmann, and J. Buhmann. Non-parametric similarity measures for unsupervised texture segmentation and image retrieval. *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, pages 267–72, June 1997.

[56] J. Puzicha, Y. Rubner, C. Tomasi, and J. Buhmann. Empirical evaluation of dissimilarity measures for color and texture. *Int'l. Conf. Computer Vision*, pages 1165–1173, 1999.

[57] X. Ren and J. Malik. A probabilistic multi-scale model for contour completion based on image statistics. *Europ. Conf. Comput. Vision*, I:312–327, 2002.

[58] C. Van Rijsbergen. *Information Retrieval, 2nd ed.* Dept. of Comp. Sci., Univ. of Glasgow, 1979.

[59] J. Rivest and P. Cavanagh. Localizing contours defined by more than one attribute. *Vision Research*, 36(1):53–66, 1996.

[60] Y. Rubner and C. Tomasi. Coalescing texture descriptors. *ARPA Image Understanding Workshop*, 1996.

[61] D. Ruderman. The statistics of natural images. *Network*, 5(4):517–548, 1994.

[62] D. Ruderman. Origins of scaling in natural images. *Vision Research*, 37(23):3385–3395, 1997.

[63] M. Ruzon and C. Tomasi. Color edge detection with the compass operator. *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, pages 160–166, 1999.

[64] M. Ruzon and C. Tomasi. Corner detection in textured color images. *Int'l. Conf. Computer Vision*, pages 1039–1045, 1999.

[65] S. Sarkar and K. Boyer. Quantitative measures of change based on feature organization: Eigenvalues and eigenvectors. *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, 71(1):110–136, 1996.

[66] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[67] G. Scott and H. Longuet-Higgins. Feature grouping by relocalisation of eigenvectors of the proximity matrix. *Proceedings of the British Machine Vision Conference*, pages 103–108, 1990.

[68] J. Shi and J. Malik. Normalized cuts and image segmentation. *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, pages 731–7, June 1997.

[69] Z. Tu and S. Zhu. Image segmentation by data-driven markov chain monte carlo. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(5):657–673, May 2002.

[70] Z. Tu, S. Zhu, and H. Shum. Image segmentation by data driven markov chain monte carlo. *Int'l. Conf. Computer Vision*, II:131–138, July 2001.

[71] J. H. van Hateren and A. Van der Schaaf. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proc. of the Royal Soc. London B*, 265:359–366, 1998.

[72] Y. Weiss. Segmentation using eigenvectors: a unifying view. *Int'l. Conf. Computer Vision*, pages 975–982, 1999.

[73] S. Will, L. Hermes, J. M. Buhmann, and J. Puzicha. On learning texture edge detectors. *Proc. Int'l. Conf. Image Processing*, pages 877–880, 2000.

[74] L. Williams and D. Jacobs. Stochastic completion fields: a neural model of illusory contour shape and salience. *Int'l. Conf. Computer Vision*, pages 408–415, June 1995.

[75] S. Wolfson and M. Landy. Examining edge- and region-based texture analysis mechanisms. *Vision Research*, 38(3):439–446, 1998.

[76] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(11):1101–13, November 1993.