Contour Detection and Image Segmentation

by

Michael Randolph Maire

B.S. (California Institute of Technology) 2003

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, Chair
Professor Trevor Darrell
Professor Stephen Palmer

Fall 2009

The dissertation of Michael Randolph Maire is approved:

Professor Jitendra Malik, Chair — Date

Professor Trevor Darrell — Date

Professor Stephen Palmer — Date

University of California, Berkeley

Fall 2009

Contour Detection and Image Segmentation

by

Michael Randolph Maire

Abstract

Contour Detection and Image Segmentation

by

Michael Randolph Maire

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Jitendra Malik, Chair

This thesis investigates two fundamental problems in computer vision: contour detection and image segmentation. We present new state-of-the-art algorithms for both of these tasks. Our segmentation algorithm consists of generic machinery for transforming the output of any contour detector into a hierarchical region tree. In this manner, we reduce the problem of image segmentation to that of contour detection. Extensive experimental evaluation demonstrates that both our contour detection and segmentation methods significantly outperform competing algorithms.

Our approach to contour detection couples multiscale local brightness, color, and texture cues to a powerful globalization framework using spectral clustering. The local cues, computed by applying oriented gradient operators at every location in the image, define an affinity matrix representing the similarity between pixels. From this matrix, we derive a generalized eigenproblem and solve for a fixed number of eigenvectors which encode contour information. Using a classifier to recombine this signal with the local cues, we obtain a large improvement over alternative globalization schemes built on top of similar cues.

To produce high-quality image segmentations, we link this contour detector with a generic grouping algorithm consisting of two steps. First, we introduce a new im-

age transformation called the Oriented Watershed Transform for constructing a set of initial regions from an oriented contour signal. Second, using an agglomerative clustering procedure, we form these regions into a hierarchy which can be represented by an Ultrametric Contour Map, the real-valued image obtained by weighting each boundary by its scale of disappearance. This approach outperforms existing image segmentation algorithms on measures of both boundary and segment quality. These hierarchical segmentations can optionally be further refined by user-specified annotations.

While the majority of this work focuses on processing static images, we also develop extensions for video. In particular, we augment the set of static cues used for contour detection with a low-level motion cue to create an enhanced boundary detector. Using optical flow in conjunction with this detector enables the determination of occlusion boundaries and assignment of figure/ground labels in video.

Professor Jitendra Malik, Chair Date

Acknowledgements

Thanks first and most of all to my advisor, Jitendra Malik, for his mentorship over the course of my graduate studies. His incredible enthusiasm has been a source of energy without which this work would not be the same. His seemingly limitless supply of new ideas and dedication to solving the grand challenges of computer vision have been an inspiration.

Thanks to Trevor Darrell and Steve Palmer for serving on both my thesis and qualifying committees, and to Ruzena Bajcsy for serving on my qualifying committee.

I have had the privilege of meeting many outstanding people during my time at Berkeley and it has been a pleasure to know them both personally and professionally. Thanks especially to my co-authors and collaborators whose efforts contributed directly to my thesis work: Pablo Arbeláez, Charless Fowlkes, Patrik Sundberg, and Thomas Brox. Thanks also to my additional co-authors and all of my fellow graduate students in the vision group over the years. You have helped make this an exciting and rewarding journey.

Thanks to David Forsyth for his guidance during my first months as a graduate student.

Thanks to Pietro Perona for starting me on this path as an undergraduate.

Prior to arriving at Berkeley, I was fortunate enough to encounter an assortment of excellent teachers, both as an undergraduate at the California Institute of Technology, and as a high school student in the Science, Mathematics, and Computer Science Magnet Program at Montgomery Blair High School in Maryland. Thanks to each of them for helping to build my scientific career.

Finally, thanks to my friends and family for all of their love and support.

to my parents

Contents

Chapter 1

Introduction

1.1 Motivation

Computer vision seeks to enhance the ability of machines to understand the visual world through the development of algorithms for tasks such as object recognition, tracking, and 3D reconstruction from image and video data. These tasks are complex enough that it is often not sufficient to simply regard the raw images as training examples and apply the latest machine learning algorithms. Rather, there is structure in natural images which should be exploited in conjunction with learning techniques. This thesis concentrates on the creation of algorithms for perceptual organization, which extract this structure, transforming an image into an intermediate representation which in turn can provide far superior input to modules for high-level tasks such as recognition.

In this spirit, we first focus our attention on contour detection, which has long been a fundamental problem in computer vision. Early approaches include the Canny edge detector, which was tuned to respond to sharp discontinuities in image brightness. Contours convey key information about object shape and serve as the basis for a

variety of local image descriptors, such as SIFT [Lowe, 1999], shape context [Belongie *et al.*, 2002], and geometric blur [Berg and Malik, 2001], used in recognition systems. Hence, improvements at the contour detection stage can impact the quality of the descriptors used in a wide range of vision algorithms.

However, we want to separate the development of a general contour detection algorithm from any particular application. The recent creation of the Berkeley Segmentation Dataset (BSDS) [Martin *et al.*, 2001] has provided a much needed metric for evaluating contour detector performance through comparison to human-drawn ground-truth boundaries. By making use of the benchmarking methodology of [Martin *et al.*, 2004], we can judge contour quality independently of the systems that will eventually make use of the contours.

While contours and shape-based features have seen much success, there is a sense that low-level machinery which produces contours alone is not enough. Often, a partition of the image into meaningful segments is desired. Segments may reduce the computational complexity of later stages by transforming an image consisting of millions of pixels into hundreds of regions. Furthermore, they come with significant structure and ideally correspond to objects, object parts, or otherwise geometrically coherent scene elements. This advantage is likely to be increasingly important in scaling recognition algorithms to deal with the full variety of the visual world, including the perhaps tens of thousands of object categories [Biederman, 1987] humans are capable of distinguishing.

Additional value exists in producing hierarchical image segmentations rather than single-level partitions. Important information often exists in segments at multiple scales. For example, an image of a person contains a face region nested within a region corresponding to the entire body. Hierarchies are better suited to capturing such object-part relationships and provide a step toward obtaining a parse of the image.

1.2 Thesis Outline

Chapter 2 covers our development of a contour detection algorithm which provides the best performance to date on the Berkeley Segmentation Dataset (BSDS) benchmark [Martin *et al.*, 2004]. This chapter first presents a review of the local visual cues on top of which we build this detector. As an important contribution, we show that these cues can be computed efficiently, in time independent of the local operator scale. The main portion of the chapter then introduces our novel globalization scheme based on spectral clustering. We turn an affinity matrix computed from the local cues into a "spectral" contour signal which we recombine with the local signal. We present both images and benchmark results illustrating the improved performance of the combined detector.

Chapter 3 describes a new algorithm for transforming the output of any contour detector into a hierarchical segmentation. This technique allows us to leverage the contour detector developed in Chapter 2 in order to produce high-quality image segmentations. These segmentations respect the boundaries produced by the contour detector, while guaranteeing closure and forming a region hierarchy. Benchmarks demonstrate a large jump in performance compared to all existing segmentation algorithms. Though our primary contribution in this chapter is a fully automatic segmentation algorithm, we also show that it can be used as preprocessing step for user-assisted image segmentation. With a minimum annotation effort, a user can extract customized regions from our hierarchy. In closing the chapter, we analyze some initial experiments aimed at quantifying the overlap between regions in our hierarchy and actual objects of interest in a scene.

While Chapters 2 and 3 concern only static images, Chapter 4 explores perceptual organization for video data. It begins by introducing a new biologically motivated local cue for motion. We argue for this cue's use as a complement to the static

cues exploited in the previous chapters and show that it enables successful detection of boundaries in the absence of static cues. Using this enhanced contour detector together with an optical flow algorithm, we address the problem of finding and labeling occlusion boundaries from video. On these tasks, we also report favorable benchmark results.

Chapter 2

Contour Detection

2.1 Introduction

In this chapter, we discuss the detection of contours in natural images and report the results of our new, high-performance algorithm for this task [Maire *et al.*, 2008]. To compare contour detection algorithms in a quantitative manner, we rely on the Berkeley Segmentation Dataset (BSDS) [Martin *et al.*, 2001] and boundary-based evaluation methodology developed by [Martin *et al.*, 2004].

The BSDS consists of 300 natural images, each of which has been manually segmented by a number of different human subjects. Figure 2.1 shows a sample of the dataset. The ground-truth data for this large collection shows the diversity, yet high consistency, of human segmentation. Much of the difference between subjects can be explained in terms of the level of detail they chose to depict. The human-drawn boundaries tend to correspond to object outlines rather than interior edges. A program capable of producing similar output is therefore likely to prove useful for automatic visual processing.

The benchmarking framework introduced by [Martin *et al.*, 2004] operates by

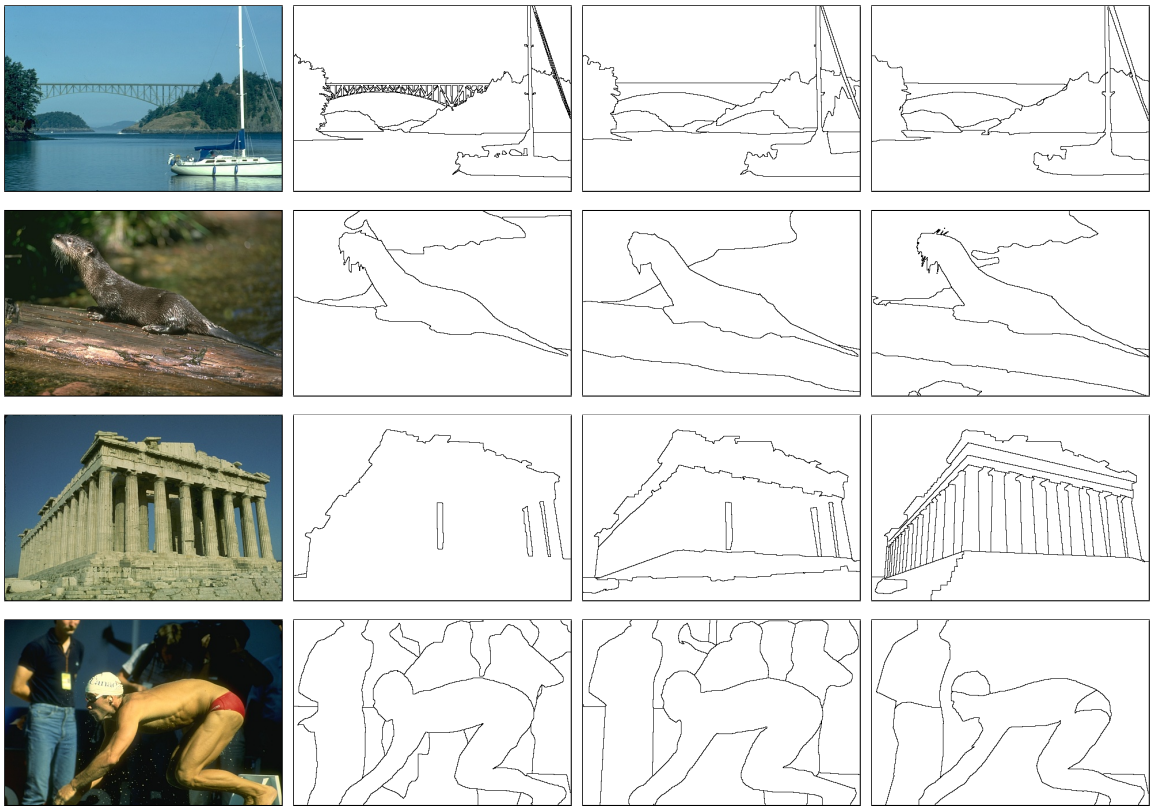

Figure 2.1: **Berkeley Segmentation Dataset** [Martin *et al.*, 2001]**. From left to right:** Original image, and hand-drawn ground-truth segmentations by three different human subjects. The entire dataset consists of 200 training and 100 test images, each with multiple ground-truth segmentations.

comparing machine generated contours to the human ground-truth data and has become a standard, as demonstrated by its widespread use [Ren *et al.*, 2005; Felzenszwalb and McAllester, 2006; Dollar *et al.*, 2006; Arbeláez, 2006; Zhu *et al.*, 2007; Mairal *et al.*, 2008; Ren, 2008; Maire *et al.*, 2008]. This framework considers two aspects of detection performance. Precision measures the fraction of true positives in the contours produced by a detector. Recall measures the fraction of ground-truth boundaries detected. For detectors that provide real-valued outputs, one obtains a curve parameterized by detection threshold, quantifying performance across operating

regimes. At each threshold, the benchmark computes a correspondence between the contour pixels declared by the detector and the human boundaries, enforcing proximity between the matched pixels. Results are averaged across the multiple hand-drawn boundary maps for each test image. The global F-measure, defined as the harmonic mean of precision and recall, provides a useful summary score for the algorithm.

Figure 2.2 summarizes the significant progress made in contour detection in the last few years [Felzenszwalb and McAllester, 2006; Ren *et al.*, 2005; Martin *et al.*, 2004; Dollar *et al.*, 2006; Zhu *et al.*, 2007; Mairal *et al.*, 2008; Ren, 2008] in terms of the BSDS benchmark. Our detector [Maire *et al.*, 2008] obtains the highest F-measure (0.70) to date and compares favorably with these other leading techniques, providing equal or better precision for most choices of recall.

Much of the extensive literature on contour detection predates the development of the quantitative benchmarking framework outlined above. One family of methods aims at quantifying the presence of a boundary at a given image location through local measurements. Early local approaches, such as the Canny detector [Canny, 1986], model edges as sharp discontinuities in the brightness channel. A richer description can be obtained by considering the response of the image to a family of filters of different scales and orientations. An example is the Oriented Energy approach [Morrone and Owens, 1987; Perona and Malik, 1990; Freeman and Adelson, 1991], in which the filter bank is composed of quadrature pairs of even and odd symmetric filters. [Lindeberg, 1998] proposes an automatic scale selection mechanism. More recent approaches take into account color and texture information and make use of learning techniques for cue combination [Martin *et al.*, 2004; Dollar *et al.*, 2006].

Another family of methods relies on integrating global image information into the grouping process. Spectral graph theory [Chung, 1997] has often been used for this purpose, particularly, the Normalized Cuts criterion [Shi and Malik, 2000; Malik *et*

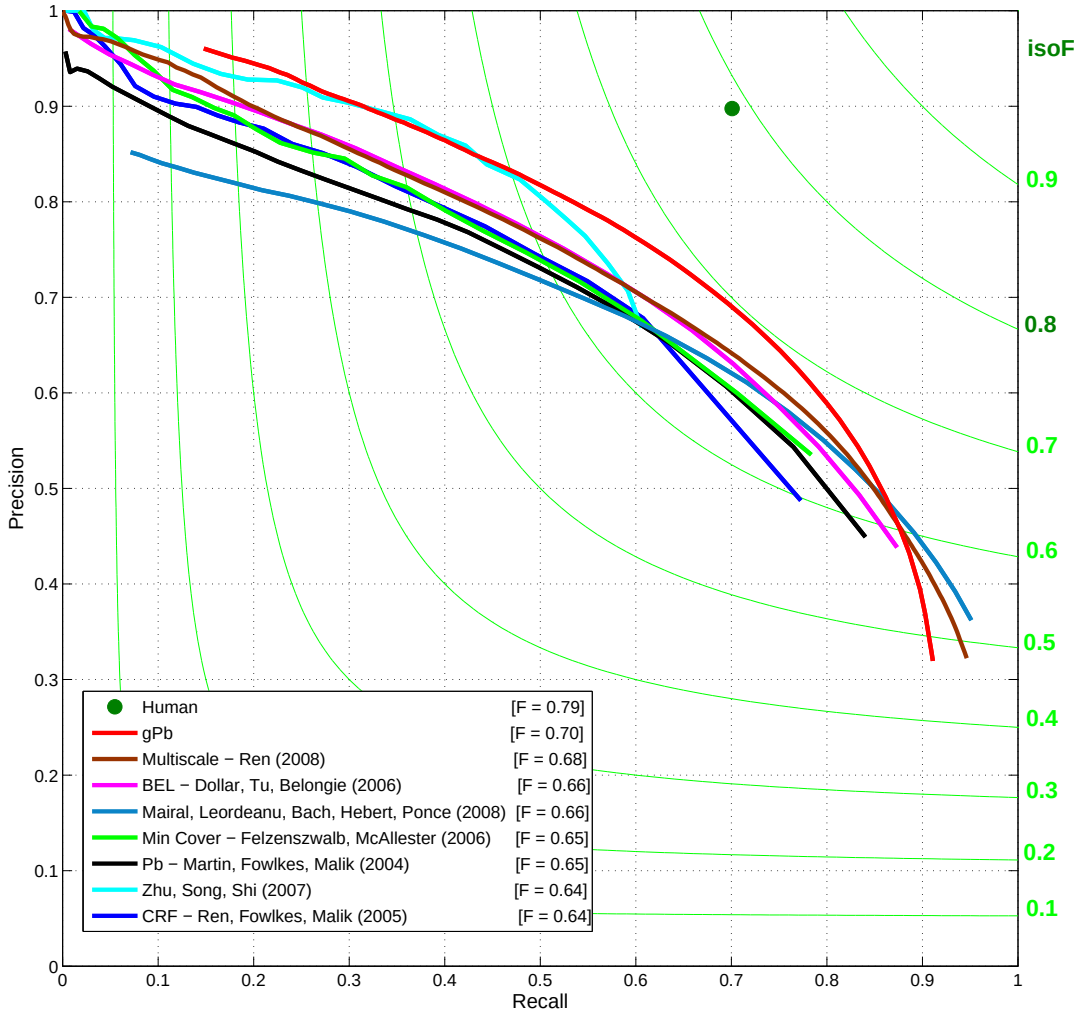

Figure 2.2: **Evaluation of contour detectors on the Berkeley Segmentation Dataset Benchmark** [Martin *et al.*, 2004]**.** Leading approaches to contour detection are ranked according to their F-measure $(2 \cdot Precision \cdot Recall/(Precision + Recall))$ with respect to human ground-truth boundaries. Iso-F curves are shown in green. The gPb detector [Maire *et al.*, 2008] performs significantly better than other algorithms across almost the entire operating regime. Average agreement between human subjects is indicated by the green dot.

al., 2001]. In this framework, given an affinity matrix W whose entries encode the similarity between pixels, one defines $D_{ii} = \sum_j W_{ij}$ and solves for the generalized eigenvectors of the linear system:

$$(D - W)\mathbf{v} = \lambda D\mathbf{v} \tag{2.1}$$

Traditionally, after this step, clustering is applied to obtain a segmentation into regions. This approach often breaks uniform regions where the eigenvectors have smooth gradients. One solution is to reweight the affinity matrix [Tolliver and Miller, 2006]; others have proposed alternative graph partitioning formulations [Fowlkes and Malik, 2004; Wang *et al.*, 2005; Yu, 2005]. Recently, [Zhu *et al.*, 2007] proposed detecting closed topological cycles in a directed edgel graph by considering the complex eigenvectors of the normalized random walk matrix. Although contour detection methods based on spectral partitioning have been reported to do well in the high precision / low recall regime, their performance is generally poor in the high recall / low precision regime [Fowlkes and Malik, 2004; Yu, 2005].

There is of course a much larger tradition in boundary detection and region segmentation. Classic approaches include the variational formulation introduced by Mumford and Shah [Mumford and Shah, 1989], level-set methods [Sethian, 1999] and techniques based on perceptual organization of contour outputs [Mahamud *et al.*, 2003; Williams and Jacobs, 1995].

Several of the recent high-performance methods we benchmark against share as a common feature their use of the local edge detection operators developed by [Martin *et al.*, 2004], which is a characteristic of our approach as well. Exceptions to this rule are [Dollar *et al.*, 2006] and [Mairal *et al.*, 2008]. Rather than rely on a few hand-crafted features, [Dollar *et al.*, 2006] propose a Boosted Edge Learning (BEL) algorithm which attempts to learn an edge classifier in the form of a probabilistic boosting tree [Tu,

2005] from thousands of simple features computed on image patches. An advantage of this approach is that it may be possible to handle cues such as parallelism and completion in the initial classification stage. [Mairal *et al.*, 2008] create both generic and class-specific edge detectors by learning discriminative sparse representations of local image patches. For each class, they learn a discriminative dictionary and use the reconstruction error obtained with each dictionary as feature input to a final classifier.

The remaining methods benchmarked in Figure 2.2 are distinguished primarily by another level of processing or globalization that utilizes the local detector output, and secondarily by if and how they integrate multiscale information. [Ren *et al.*, 2005] use the Conditional Random Fields (CRF) framework as a globalization stage to enforce curvilinear continuity of contours. After thresholding the locally detected contours, they compute a constrained Delaunay triangulation (CDT) yielding a graph consisting of the detected contours along with the new "completion" edges introduced by the triangulation. The CDT is scale-invariant and tends to fill short gaps in the detected contours. By associating a random variable with each contour and each completion edge, they define a CRF with edge potentials in terms of detector response and vertex potentials in terms of junction type and continuation smoothness. They use loopy belief propagation [Weiss, 2000] to compute expectations.

In [Felzenszwalb and McAllester, 2006], the authors use a different strategy for extracting salient smooth curves from the output of a local contour detector. They consider the set of short oriented line segments that connect pixels in the image to their neighboring pixels. Each such segment is either part of a curve or is a background segment. They assume that curves are drawn from a Markov process, that the prior distribution on curves favors few curves per scene, and that detector responses are conditionally independent given the labeling of line segments. Finding the optimal line segment labeling then translates into a general weighted min-cover problem in which the elements being covered are the line segments themselves and the

objects covering them are drawn from the set of all possible curves and all possible background line segments. Since this problem is NP-hard, the authors develop a greedy approximate solution using a "cost per pixel" heuristic.

While [Martin *et al.*, 2004] claim not to have seen much benefit from combining multiple scales of their local operator to improve contour detection, [Ren, 2008] achieves such a boost by defining additional localization and relative contrast cues in terms of the multiscale detector output. For each scale, the localization cue captures the distance from a pixel to the nearest peak response. The relative contrast cue normalizes the response at each pixel in terms of the local neighborhood. Inclusion of these signals contributes positively to the performance of a logistic regression classifier for boundary detection. In Section 2.2.2, we present a simpler multiscale cue combination approach that also improves performance. Section 2.3 then presents the new globalization method we run on top of the multiscale local detector.

2.2 Local Cues

As a starting point for contour detection, we consider the work of [Martin *et al.*, 2004], who define a function $Pb(x, y, \theta)$ that predicts the posterior probability of a boundary with orientation θ at each image pixel (x, y) by measuring the difference in local image brightness, color, and texture channels. In this section, we review these cues, introduce our own multiscale version of the Pb detector, and show how it can be computed efficiently.

2.2.1 Brightness, Color, Texture Gradients

The basic building block of the Pb contour detector is the computation of an oriented gradient signal $G(x, y, \theta)$ from an intensity image I. This computation proceeds by

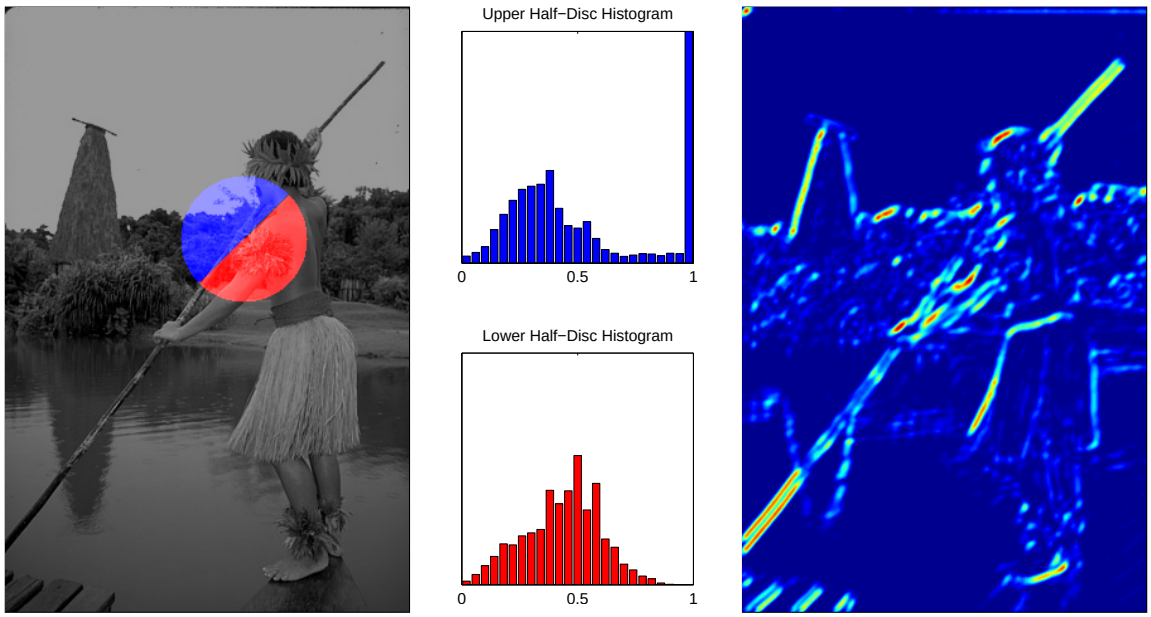

Figure 2.3: **Oriented gradient of histograms.** Given an intensity image, consider a circular disc centered at each pixel and split by a diameter at angle θ. We compute histograms of intensity values in each half-disc and output the χ^2 distance between them as the gradient magnitude. The blue and red histograms shown in the middle panel are the histograms of the pixel values in the blue and red half-disc regions, respectively, in the left image. The rightmost panel shows an example result for a disc of radius 5 pixels at orientation $\theta = \frac{\pi}{4}$ after applying a second-order Savitzky-Golay smoothing filter to the raw histogram difference output. Note that the leftmost panel displays a larger disc (radius 50 pixels) for illustrative purposes.

placing a circular disc at location (x, y) split into two half-discs by a diameter at angle θ. For each half-disc, we histogram the intensity values of the pixels of I covered by it. The gradient magnitude G at location (x, y) is defined by the χ^2 distance between the two half-disc histograms g and h:

$$\chi^2(g, h) = \frac{1}{2} \sum_i \frac{(g(i) - h(i))^2}{g(i) + h(i)} \tag{2.2}$$

Figure 2.4: **Filters for creating textons.** Our texton filter set consists of eight oriented even- and odd-symmetric Gaussian derivative filters together with a center-surround (difference of Gaussians) filter.

We then apply second-order Savitzky-Golay filtering [Savitzky and Golay, 1964] to enhance local maxima and smooth out multiple detection peaks in the direction orthogonal to θ. This is equivalent to fitting a cylindrical parabola, whose axis is orientated along direction θ, to a local 2D window surrounding each pixel and replacing the response at the pixel with that estimated by the fit.

Figure 2.3 shows an example. This computation is motivated by the intuition that contours correspond to image discontinuities and histograms provide a robust mechanism for modeling the content of an image region. A strong oriented gradient response means a pixel is likely to lie on the boundary between two distinct regions.

The Pb detector combines the oriented gradient signals obtained from transforming an input image into four separate feature channels and processing each channel independently. The first three correspond to the channels of the CIE Lab colorspace, which we refer to as the brightness, color a, and color b channels. For grayscale images, the brightness channel is the image itself and no color channels are used.

The fourth channel is a texture channel, which assigns each pixel a texton id. We apply the same oriented gradient machinery to the image channel in which each pixel value is its corresponding integer texton id. Equivalently, the histogram in each half-disc is a histogram of which textons lie within that half-disc.

The pixel to texton assignments themselves are computed by another filtering stage which occurs prior to the computation of the oriented gradient of histograms. This stage converts the input image to grayscale and convolves it with the set of 17

Gaussian derivative and center-surround filters shown in Figure 2.4. Each pixel is associated with a (17-dimensional) vector of responses, containing one entry for each filter. These vectors are then clustered using K-means. The cluster centers define a set of image-specific textons and each pixel is assigned the integer id in $[1, K]$ of the closest cluster center. In our experiments, we found choosing $K = 32$ textons to be sufficient.

We now form an image where each pixel has an integer value in $[1, K]$, as determined by its assigned texton id. An example of such an image can be seen in the upper right panel of Figure 2.5. On this image, we compute differences of histograms in oriented half-discs in the same manner as for the brightness and color channels. When computing histograms for the texture channel, one histogram bin is allocated per texton and it counts the number of pixels in the half-disc region assigned that texton id.

2.2.2 Multiscale Cue Combination

While the previous subsection is entirely review, this section introduces our multiscale extension of the *Pb* detector developed by [Martin *et al.*, 2004]. As discussed in Section 2.1, [Ren, 2008] introduces a different multiscale extension in work contemporaneous with that presented here. [Ren, 2008] suggests possible reasons [Martin *et al.*, 2004] did not see performance improvements in their original multiscale experiments, including their use of smaller images in the experiments and their choice of scales.

In order to detect fine as well as coarse structures, we consider gradients at three scales: $[\frac{\sigma}{2}, \sigma, 2\sigma]$ for each of the brightness, color, and texture channels. Figure 2.5 shows an example of the oriented gradients obtained for each channel. For the brightness channel, we use $\sigma = 5$ pixels, while for color and texture we use $\sigma = 10$ pixels.

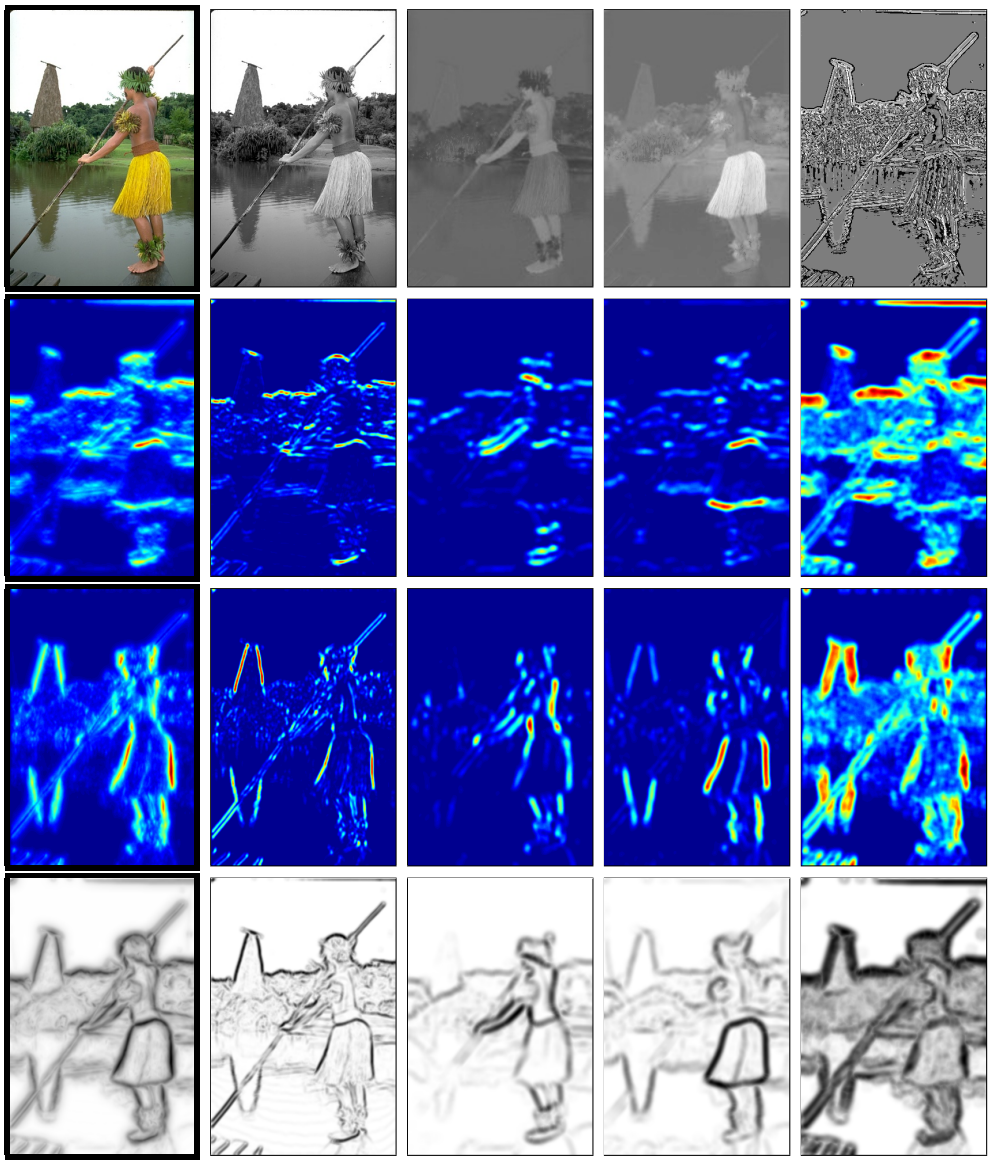

Figure 2.5: **Multiscale Pb. Top row, left to right:** Original image, followed by the brightness and color a and b channels of Lab color space, and the texton channel computed using image-specific textons. **Columns:** Below each channel, we display the oriented gradient of histograms (as outlined in Figure 2.3) for $\theta = 0$ and $\theta = \frac{\pi}{2}$ (horizontal and vertical), and the maximum response over eight orientations in $[0, \pi)$ (bottom row). Beneath the original image, we display the combination of oriented gradients across all four channels and across three scales. The lower left panel shows mPb, the final output of the multiscale contour detector.

We then linearly combine these local cues into a single multiscale oriented signal:

$$mPb(x, y, \theta) = \sum_s \sum_i \alpha_{i,s} G_{i,\sigma(i,s)}(x, y, \theta) \tag{2.3}$$

where s indexes scales, i indexes feature channels (brightness, color a, color b, texture), and $G_{i,\sigma(i,s)}(x, y, \theta)$ measures the histogram difference in channel i between two halves of a disc of radius $\sigma(i, s)$ centered at (x, y) and divided by a diameter at angle θ. In our experiments, we sample θ at eight equally spaced orientations in the interval $[0, \pi)$. Taking the maximum response over orientations yields a measure of boundary strength at each pixel:

$$mPb(x, y) = \max_\theta \{mPb(x, y, \theta)\} \tag{2.4}$$

An optional non-maximum suppression step [Canny, 1986] produces thinned, real-valued contours.

In contrast to [Martin *et al.*, 2004] and [Ren, 2008] which use a logistic regression classifier to combine cues, we learn the weights $\alpha_{i,s}$ by gradient ascent on the F-measure using the 200 training images of the BSDS and their corresponding ground-truth.

2.2.3 Efficient Computation

Computing the oriented gradient of histograms (Figure 2.3) directly as outlined in the previous section is expensive. In particular, for an N pixel image and a disc of radius r it takes $O(Nr^2)$ time to compute as a region of size equal to the area of the disc is examined at every pixel. This entire procedure is repeated 32 times (4 channels with 8 orientations) for each of 3 choices of r, with the cost of the largest scale dominating. [Martin *et al.*, 2004] suggest ways to speed up this computation,

including incremental updating of the histograms as the disc is swept across the image. However, this strategy still requires $O(Nr)$ time. We present an algorithm for the oriented gradient of histograms computation that runs in $O(N)$ time, independent of the radius r.

Following Figure 2.6, we can approximate each half-disc by a series of rectangles. It turns out that a single rectangle is a sufficiently good approximation for our purposes. Now, instead of rotating our rectangular regions, we pre-rotate the image so that we are concerned with computing a histogram of the values within axis-aligned rectangles. This can be done in time independent of the size of the rectangle using integral images.

We process each histogram bin separately. Let I denote the rotated intensity image and let $I^b(x, y)$ be 1 if $I(x, y)$ falls in histogram bin b and 0 otherwise. Compute the integral image J^b as the cumulative sum across rows of the cumulative sum across columns of I^b. The total energy in an axis-aligned rectangle with points P, Q, R, and S as its upper-left, upper-right, lower-left, and lower-right corners, respectively, falling in histogram bin b is:

$$h(b) = J^b(P) + J^b(S) - J^b(Q) - J^b(R) \tag{2.5}$$

It takes $O(N)$ time to pre-rotate the image, and $O(N)$ to compute each of the $O(B)$ integral images, where B is the number of bins in the histogram. Once these are computed, there is $O(B)$ work per rectangle, of which there are $O(N)$. Rotating the output back to the original coordinate frame takes an additional $O(N)$ work. The total complexity is thus $O(NB)$ instead of $O(Nr^2)$ (actually instead of $O(Nr^2 + NB)$ since we always had to compute χ^2 distances between N histograms). Since B is a fixed constant, the computation time no longer grows as we increase the scale r.

This algorithm runs in time $O(NB)$ as long as we use at most a constant number of rectangular boxes to approximate each half-disc. For an intuition as to why a

Figure 2.6: **Efficient computation of the oriented gradient of histograms. Left:** The two half-discs of interest can be approximated arbitrarily well by a series of rectangular boxes. We found a single box of equal area to the half-disc to be a sufficient approximation. **Middle:** Replacing the circular disc of Figure 2.3 with the approximation reduces the problem to computing the histograms within rectangular regions. **Right:** Instead of rotating the rectangles, rotate the image and use the integral image trick to compute the histograms efficiently. Rotate the final result to map it back to the original coordinate frame.

single rectangle turns out to be sufficient, look again at the overlap of the rectangle with the half disc in the lower left of Figure 2.6. The majority of the pixels used in forming the histogram lie within both the rectangle and the disc, and those pixels that differ are far from the center of the disc (the pixel at which we are computing the gradient). Thus, we are only slightly changing the shape of the region we use for context around each pixel. Figure 2.7 shows that the result using the single rectangle approximation is visually indistinguishable from that using the original half-disc.

Results reported in this chapter use the exact half-discs because they were ob-

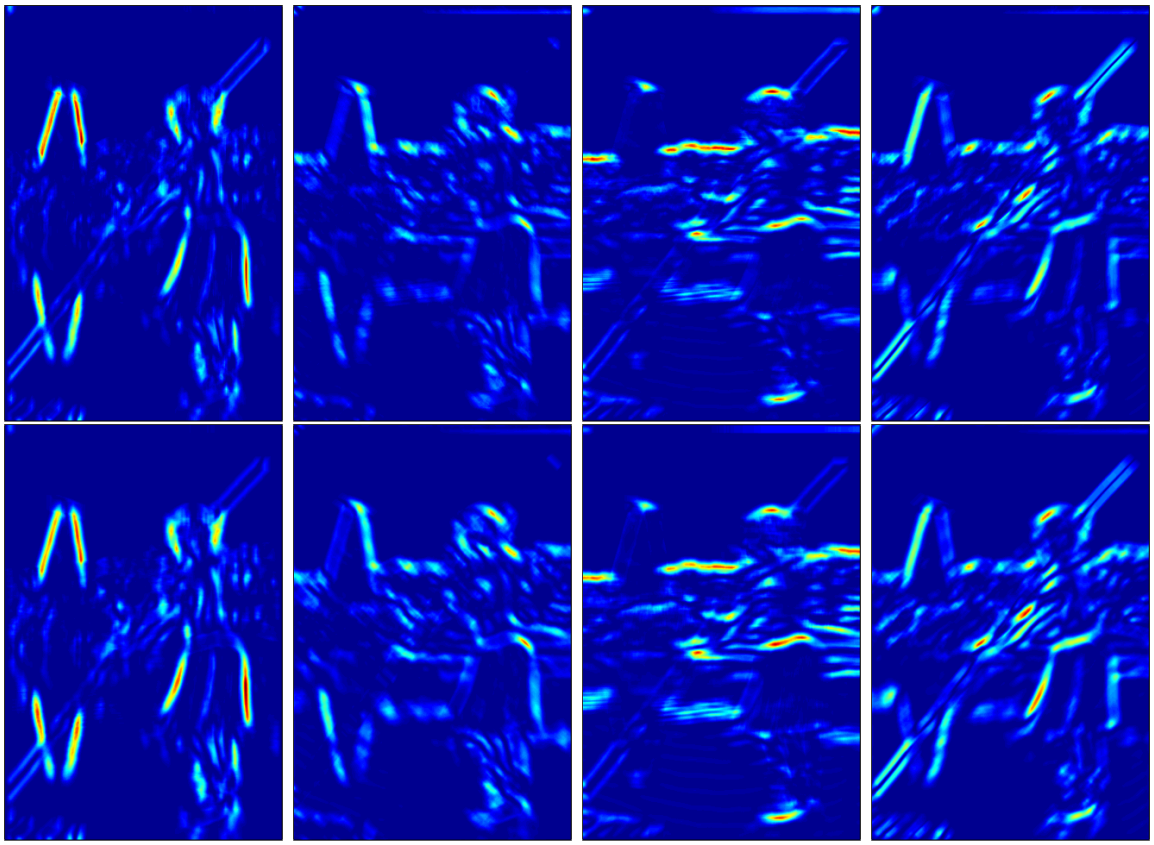

Figure 2.7: **Comparison of half-disc and rectangular regions for computing the oriented gradient of histograms. Top row:** Results of using the $O(Nr^2)$ time algorithm to compute the difference of histograms in oriented half-discs at each pixel. Shown is the output for processing the brightness channel displayed in Figure 2.6 using a disc of radius $r = 10$ pixels at four distinct orientations (one per column). N is the total number of pixels. **Bottom row:** Approximating each half-disc with a single rectangle (of height 9 pixels so that the rectangle area best matches the disc area), as shown in Figure 2.6, and using integral histograms allows us to compute nearly identical results in only $O(N)$ time. In both cases, we show the raw histogram difference output before application of a smoothing filter in order to clearly demonstrate the similarity of the results.

tained prior to our implementation of the faster rectangular approximation. As Chapter 3 builds upon these results, we also use the exact version for consistency. The motion gradient channel introduced in Chapter 4 is computed using the fast approximation.

Note that the same image rotation technique can be used for computing convolutions with any oriented separable filter, such as the oriented Gaussian derivative filters used for textons or the second-order Savitzky-Golay filters used for spatial smoothing of our oriented gradient output. Rotating the image, convolving with two 1D filters, and inverting the rotation is more efficient than convolving with a rotated 2D filter. Moreover, in this case, no approximation is required as these operations are equivalent up to the numerical accuracy of the interpolation done when rotating the image. This means that all of the filtering performed as part of the local cue computation can be done in $O(Nr)$ time instead of $O(Nr^2)$ time where here $r = max(w, h)$ and w and h are the width and height of the 2D filter matrix. For large r, the computation time can be further reduced by using the Fast Fourier Transform to calculate the convolution.

The entire local cue computation is also easily parallelized and there are a number of choices for doing so. The image can be partitioned into overlapping subimages, each of which is processed in parallel. In addition, the 96 intermediate results (3 scales of 4 channels with 8 orientations) can all be computed in parallel as they are independent subproblems. [Catanzaro *et al.*, 2009] have created parallel GPU implementations of the multiscale contour detector we discussed here and the globalization stage we introduce in the next section. They also exploit the integral histogram trick introduced here, with the single rectangle approximation, while replicating the same precision-recall curve on the BSDS benchmark. The speed improvements due to both the reduction in computational complexity outlined above and parallelization make our contour detector into a practical tool.

2.3 Globalization

Spectral clustering lies at the heart of our globalization machinery. The key element differentiating the algorithm described in this section from other approaches [Shi and Malik, 2000; Tolliver and Miller, 2006] is the soft manner in which we use the eigenvectors obtained from spectral partitioning.

As input to the spectral clustering stage, we construct a sparse symmetric affinity matrix W using the *intervening contour* cue [Fowlkes *et al.*, 2003; Fowlkes and Malik, 2004; Leung and Malik, 1998], the maximal value of mPb along a line connecting two pixels. We connect all pixels i and j within a fixed radius r with affinity:

$$W_{ij} = \exp\left(-\max_{p \in \overline{ij}}\{mPb(p)\}/\sigma\right) \tag{2.6}$$

where $\overline{ij}$ is the line segment connecting i and j and σ is a constant. In experiments, we set $r = 5$ pixels and $\sigma = 0.1$.

In order to introduce global information, we define $D_{ii} = \sum_j W_{ij}$ and solve for the generalized eigenvectors $\{\mathbf{v_0}, \mathbf{v_1}, ..., \mathbf{v_n}\}$ of the system $(D - W)\mathbf{v} = \lambda D\mathbf{v}$ (Equation 2.1), corresponding to the $n + 1$ smallest eigenvalues $0 = \lambda_0 \leq \lambda_1 \leq ... \leq \lambda_n$. Figure 2.8 displays an example with eigenvectors $\mathbf{v_1}$ through $\mathbf{v_4}$. In practice, we use $n = 16$.

At this point, the standard Normalized Cuts approach associates with each pixel a length n descriptor formed from entries of the n eigenvectors and uses a clustering algorithm such as K-means to create a hard partition of the image. Unfortunately, this can lead to an incorrect segmentation as large uniform regions in which the eigenvectors vary smoothly are broken up. Figure 2.8 shows an example for which such gradual variation in the eigenvectors across the sky region results in an incorrect partition.

Figure 2.8: **Spectral Pb. Left:** Original image. **Middle Left:** The thinned non-max suppressed multiscale Pb signal defines a sparse affinity matrix, connecting each pixel i to all others within a fixed radius. Pixels i and j have a low affinity as a strong boundary separates them, whereas i and k have high affinity. **Middle:** First four generalized eigenvectors resulting from spectral clustering. **Middle Right:** Partitioning the image by running K-means clustering on the eigenvectors erroneously breaks smooth regions. **Right:** Instead, we compute gradients of the eigenvectors, transforming them back into a contour signal.

To circumvent this difficulty, we observe that the eigenvectors themselves carry contour information. Treating each eigenvector $\mathbf{v_k}$ as an image, we convolve with Gaussian directional derivative filters at multiple orientations θ, obtaining an oriented signal $sPb_{\mathbf{v_k}}(x, y, \theta)$. Taking derivatives in this manner ignores the smooth variations that previously lead to errors. The information from different eigenvectors is then combined to provide the "spectral" component of our boundary detector:

$$sPb(x, y, \theta) = \sum_{k=1}^{n} \frac{1}{\sqrt{\lambda_k}} \cdot sPb_{\mathbf{v_k}}(x, y, \theta) \tag{2.7}$$

where the choice of the weights is motivated by the physical interpretation of generalized eigensystems as mass-spring systems [Belongie and Malik, 1998]. Figures 2.8 and 2.9 present examples.

The signals mPb and sPb convey different information, as the former fires at all the edges while the latter extracts only the most salient curves in the image. We

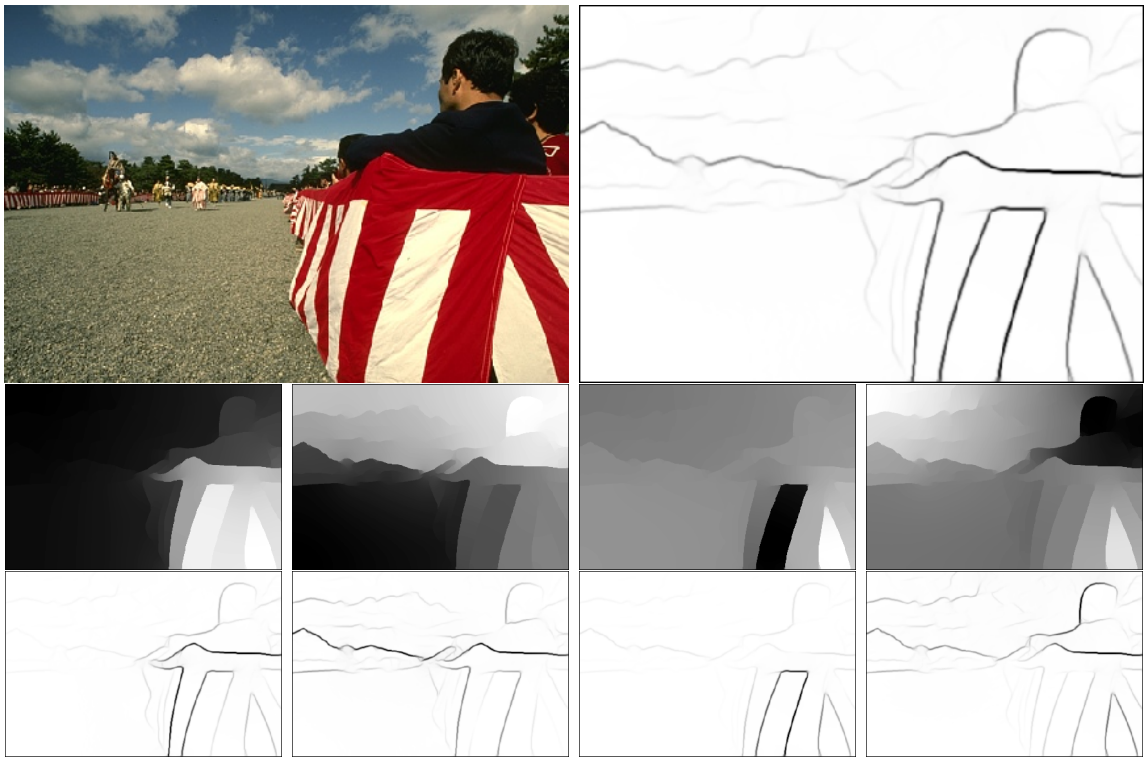

Figure 2.9: **Eigenvectors carry contour information. Top:** Original image and maximum response of spectral Pb over orientations, $sPb(x, y) = \max_\theta sPb(x, y, \theta)$. **Middle:** First four generalized eigenvectors used in creating spectral Pb. **Bottom:** Maximum response over orientations θ of $sPb_{\mathbf{v_k}}(x, y, \theta)$ for each eigenvector $\mathbf{v_k}$ shown above.

found that a simple linear combination is enough to benefit from both behaviors. Our final **globalized probability of boundary** is then written as a weighted sum of local and spectral signals, which is subsequently rescaled using a sigmoid:

$$gPb(x, y, \theta) = \sum_s \sum_i \beta_{i,s} G_{i,\sigma(i,s)}(x, y, \theta) + \gamma \cdot sPb(x, y, \theta) \tag{2.8}$$

As with mPb (Equation 2.3), the weights $\beta_{i,s}$ and γ are learned by gradient ascent on the F-measure using the BSDS training images.

2.4 Results

Figure 2.10 breaks down the contributions of the multiscale and spectral signals to the performance of gPb. These precision-recall curves show that the reduction of false positives due to the use of global information in sPb is concentrated in the high thresholds, while gPb takes the best of both worlds, relying on sPb in the high precision regime and on mPb in the high recall regime.

Figure 2.2 presents a comparison of the gPb contour detector to other algorithms on the BSDS benchmark. The mean improvement in precision of gPb with respect to the single scale Pb is 10% in the recall range $[0.1, 0.9]$. Qualitatively, this improvement translates into the reduction of clutter edges and completion of contours in the output, as shown in Figure 2.11.

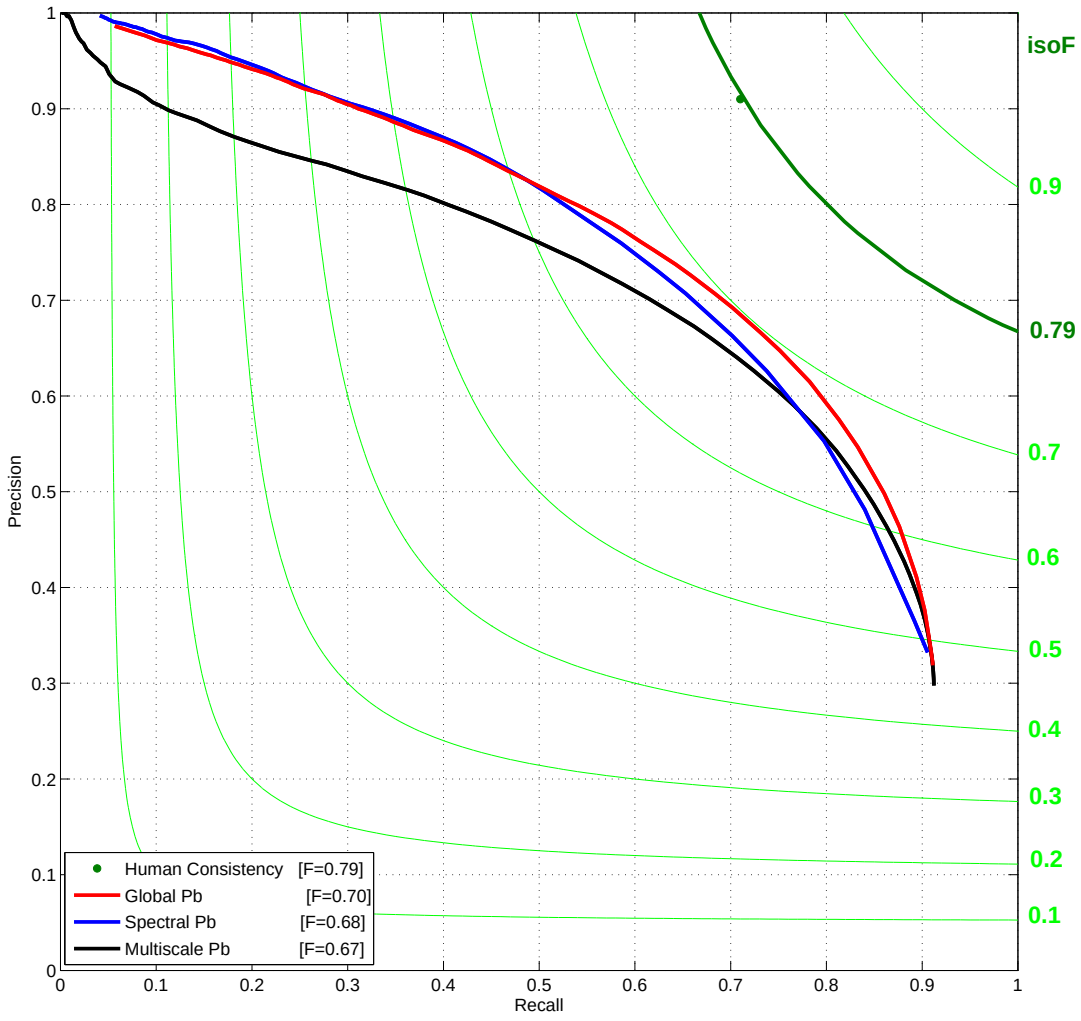

Figure 2.10: **Globalization improves contour detection.** The spectral Pb detector, derived from the eigenvectors of a spectral partitioning algorithm, improves the precision of the local multiscale Pb signal used as input. Global Pb, a learned combination of the two, provides uniformly better performance.

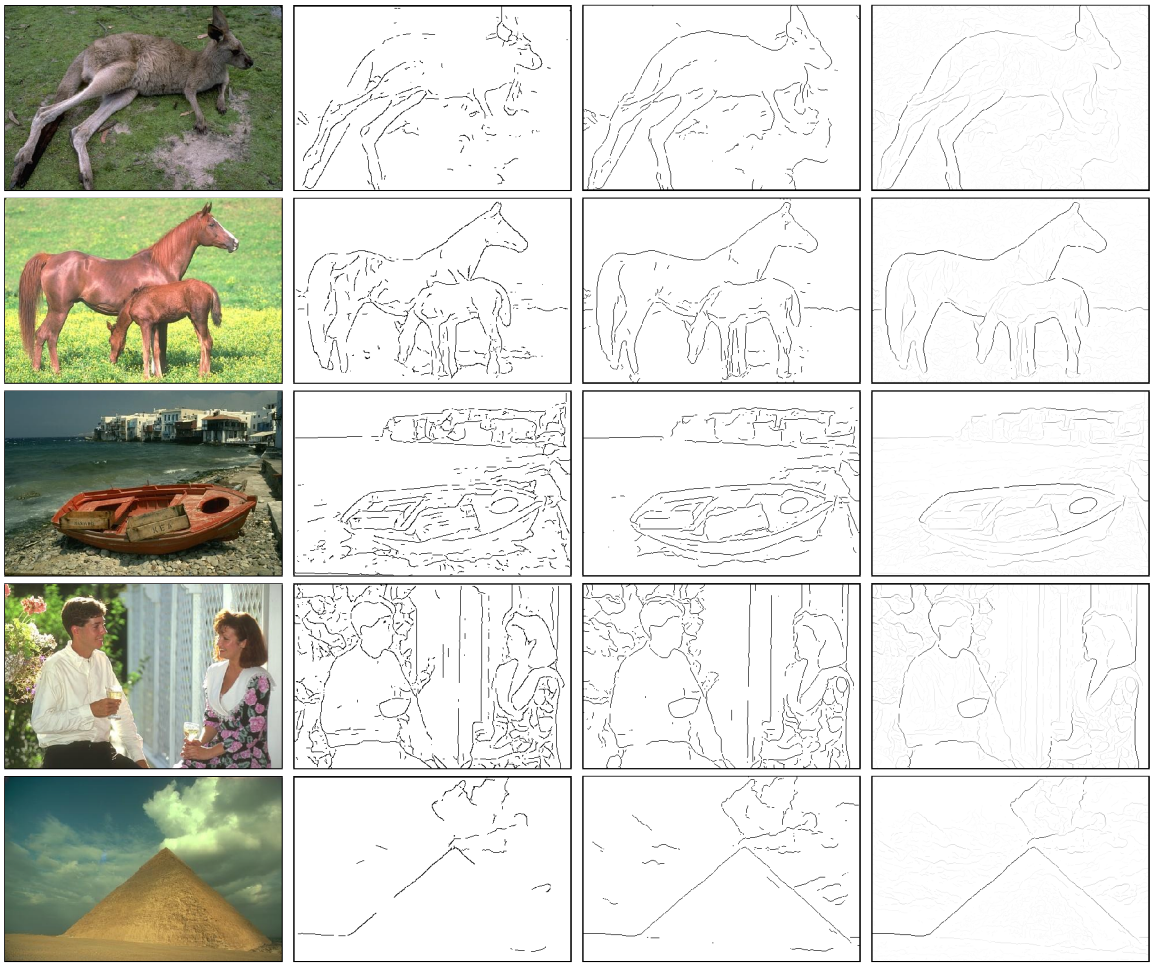

Figure 2.11: **Benefits of globalization.** When compared with the local detector Pb, our detector gPb reduces clutter and completes contours. **From left to right:** Original image, thresholded Pb, thresholded gPb, and gPb. The thresholds shown correspond to the points of maximal F-measure on the curves in Figure 2.2.

Chapter 3

Segmentation

3.1 Introduction

Applications such as object recognition [Rabinovich *et al.*, 2007; Malisiewicz and Efros, 2007; Ahuja and Todorovic, 2008; Gu *et al.*, 2009] and monocular inference of 3D structure [Hoiem *et al.*, 2005; Saxena *et al.*, 2008] have led to a renewed interest in algorithms for automatic segmentation of an image into closed regions. Segments come with their own scale estimates and provide natural domains for computing features used in recognition. Many visual tasks can also benefit from the reduction in complexity achieved by transforming an image with millions of pixels into a few hundred or thousand "superpixels" [Ren and Malik, 2003].

A broad family of approaches to segmentation involve integrating features such as brightness, color, or texture over local image patches and then clustering those features based on, *e.g.*, fitting mixture models [Belongie *et al.*, 1998; Yang *et al.*, 2008], mode-finding [Comaniciu and Meer, 2002], or graph partitioning [Shi and Malik, 2000; Malik *et al.*, 2001; Tolliver and Miller, 2006; Felzenszwalb and Huttenlocher, 2004]. While this is by no means the only approach taken (see *e.g.* the vast literature inspired

by variational formulations [Mumford and Shah, 1989; Morel and Solimini, 1995] and level set techniques [Malladi *et al.*, 1995]), three algorithms in this category appear to be the most widely used as sources of image segments in recent applications, due to a combination of reasonable performance and publicly available implementations:

- **Felzenszwalb and Huttenlocher (Felz-Hutt)**

  The graph based region merging algorithm advocated by [Felzenszwalb and Huttenlocher, 2004] attempts to partition image pixels into components such that the resulting segmentation is neither too coarse nor too fine. Given a graph in which pixels are nodes and edge weights measure the dissimilarity between nodes (*e.g.* color differences), each node is initially placed in its own component. Define the internal difference of a component $Int(C)$ as the largest weight in the minimum spanning tree of C. Considering edges in non-decreasing order by weight, each step of the algorithm merges components C_1 and C_2 connected by the current edge if the edge weight (equivalently difference between components) is less than $min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2))$ where $\tau(C) = k/|C|$. k is a scale parameter that can be used to set a preference for component size. Merging stops when the difference between components exceeds the internal component difference.

- **Mean Shift** [Comaniciu and Meer, 2002]

  In [Comaniciu and Meer, 2002] pixels are represented in the joint spatial-range domain by concatenating their spatial coordinates and color values into a single vector. Applying mean shift filtering in this domain yields a convergence point for each pixel. Regions are formed by grouping together all pixels whose convergence points are closer than h_s in the spatial domain and h_r in the range domain, where h_s and h_r are respective bandwidth parameters. Additional merging can also be performed to enforce a constraint on minimum region area.

- **Multiscale Normalized Cuts (NCuts)** [Cour *et al.*, 2005]

  In the Normalized Cuts framework [Shi and Malik, 2000] image cues are used to define an affinity matrix W whose entries encode pixel similarity. Image segmentation is performed by partitioning the graph $G = (V, E, W)$ with pixels as nodes V and edge weights W using the generalized eigenvectors of the linear system $(D - W)x = \lambda Dx$, where $D_{ii} = \sum_j W_{ij}$. While our algorithm integrates eigenvector information in a soft manner as explained in Section 2.3, the standard approach directly uses the eigenvectors for segmentation.

  We compare against the latest variant of such an approach [Cour *et al.*, 2005]. The fact that W must be sparse in order to avoid a prohibitively expensive computation, limits the naive implementation to using only local pixel affinities. [Cour *et al.*, 2005] solve this limitation by computing sparse affinity matrices at multiple scales, setting up cross-scale constraints, and deriving a new eigenproblem for this constrained multiscale Normalized Cut.

There does not appear to be a consensus about which of these algorithms is best. [Felzenszwalb and Huttenlocher, 2004] is typically used in high recall settings to create a gross oversegmentation into thousands of superpixels. Mean Shift and Normalized Cuts provide better precision, but often produce artifacts by breaking large uniform regions (*e.g.* sky) into chunks.

The problem of oversegmentation is common across approaches based on feature clustering since smooth changes in texture or brightness due to perspective or shading can cause patches to appear dissimilar despite belonging to the same image region. Contour detection ignores such smooth variations by directly searching for locations in the image where brightness or other features undergo rapid local changes [Canny, 1986; Perona and Malik, 1990]. These high-gradient edge fragments can then be linked together in order to identify extended, smooth contours [Parent and Zucker, 1989;

Williams and Jacobs, 1995; Elder and Zucker, 1996; Ren *et al.*, 2008].

Despite the advances in contour detection discussed in Chapter 2 and summarized by Figure 2.2, without some mechanism for enforcing closure, a segmentation built up from locally detected contours will often mistakenly join regions due to leaks in the bounding contour, resulting in an under-segmentation.

In this chapter, we propose an algorithm, first reported in [Arbeláez *et al.*, 2009], that produces a hierarchical segmentation from the output of any contour detector, while avoiding these difficulties. We introduce a new variant of the watershed transform [Beucher and Meyer, 1992; Najman and Schmitt, 1996], the Oriented Watershed Transform (OWT), for producing a set of initial regions from contour detector output. We then construct an Ultrametric Contour Map (UCM) [Arbeláez, 2006] from the boundaries of these initial regions. This sequence of operations (OWT-UCM) can be seen as generic machinery for going from contours to a hierarchical region tree. Contours encoded in the resulting hierarchical segmentation retain real-valued weights indicating their likelihood of being a true boundary. For a given threshold, the output is a set of closed contours that can be treated as either a segmentation or as a boundary detector for the purposes of benchmarking.

To establish the value of the OWT-UCM algorithm, we examine a number of different benchmark metrics and standard datasets for both boundary and region detection. Based on this extensive testing, we report two important results, illustrated in Figure 3.1 and Figure 3.2, respectively:

- Weighted boundary maps can be converted into hierarchical segmentations without loss of boundary precision or recall. (Section 3.2)

- Using the gPb contour detector [Maire *et al.*, 2008] as input, our method, gPb-owt-ucm provides a powerful mid-level grouping mechanism which outperforms all existing segmentation algorithms. (Section 3.4)

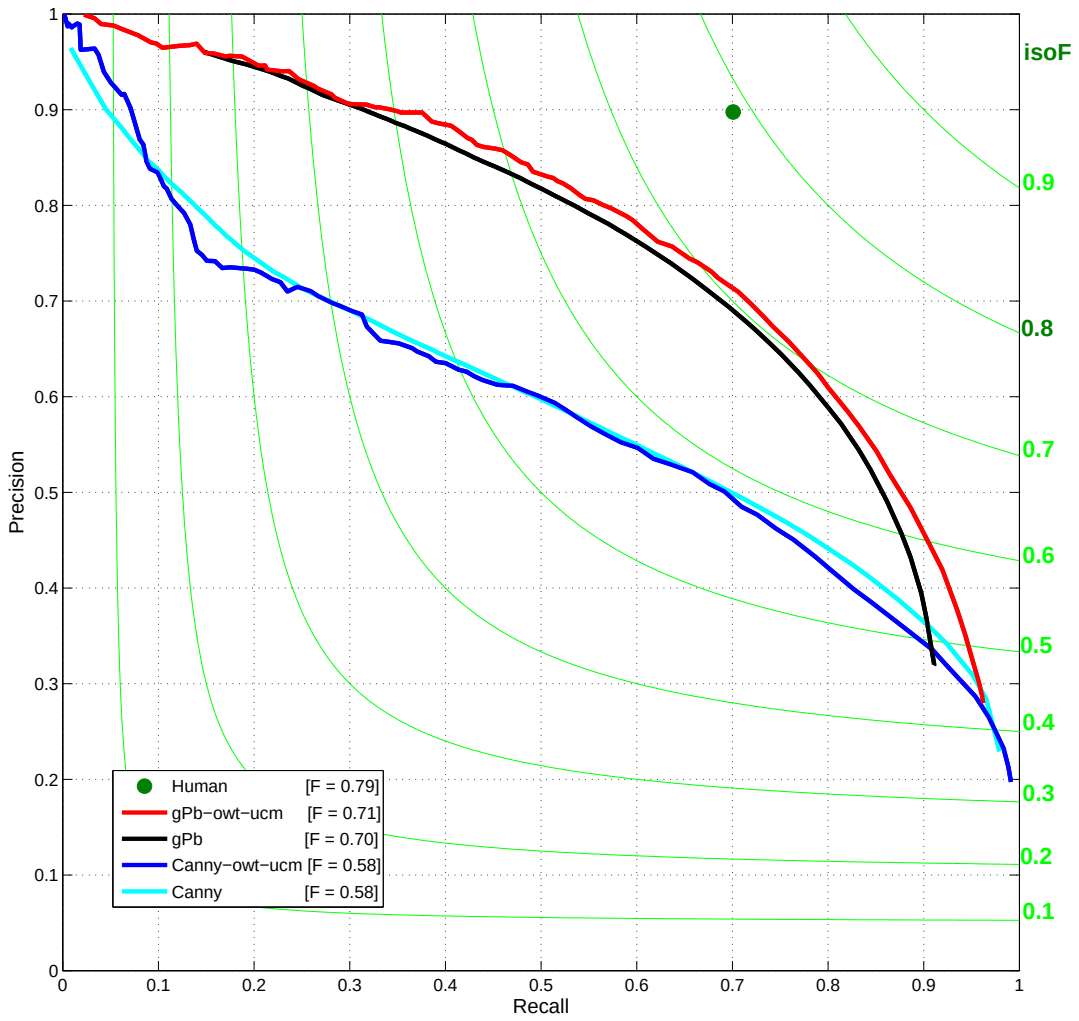

Figure 3.1: **The OWT-UCM algorithm preserves contour detector quality.** Our algorithm (OWT-UCM) produces a hierarchical segmentation from the output of any contour detector. Comparing the resulting segment boundaries to the original contours shows that this method constructs regions without losing performance on the Berkeley Segmentation Dataset (BSDS) boundary benchmark [Martin *et al.*, 2004]. In fact, we obtain a boost in performance when using the gPb detector [Maire *et al.*, 2008] as input. The quality of the contour detector (gPb vs Canny) on which we build significantly influences the quality of the resulting segmentation.

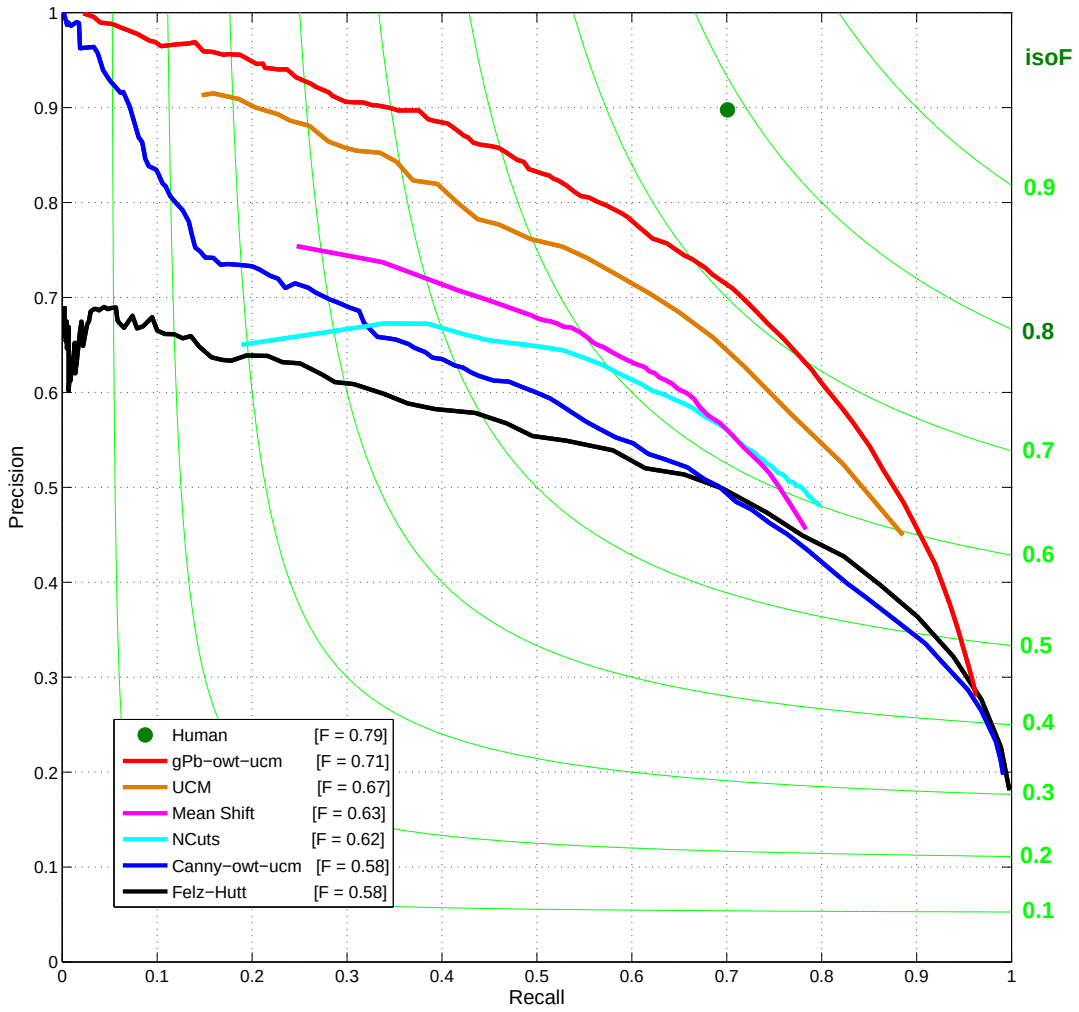

Figure 3.2: **Evaluation of segmentation algorithms on the Berkeley Segmentation Dataset Benchmark** [Martin *et al.*, 2004]**.** Paired with the gPb contour detector as input, our hierarchical segmentation algorithm gPb-owt-ucm produces segments whose boundaries match human ground-truth better than those produced by alternative segmentation approaches such as Mean Shift, Normalized Cuts, or region-merging (Felz-Hutt).

3.2 Contours to Hierarchical Regions

We consider a contour detector, whose output $E(x, y, \theta)$ predicts the probability of an image boundary at location (x, y) and orientation θ. We build hierarchical regions by exploiting the information in this contour signal using a sequence of two transformations, the Oriented Watershed Transform (OWT) and Ultrametric Contour Map (UCM), detailed below.

3.2.1 Oriented Watershed Transform

Using the contour signal, we first construct a finest partition for the hierarchy, an oversegmentation whose regions determine the highest level of detail considered. This is done by computing $E(x, y) = \max_\theta E(x, y, \theta)$, the maximal response of the contour detector over orientations. We take the regional minima of $E(x, y)$ as seed locations for homogeneous segments and apply the watershed transform used in mathematical morphology [Beucher and Meyer, 1992; Najman and Schmitt, 1996] on the topographic surface defined by $E(x, y)$. The catchment basins of the minima, denoted $\mathcal{P}_0$, provide the regions of the finest partition and the corresponding watershed arcs, $\mathcal{K}_0$, the possible locations of the boundaries.

Figure 3.3 shows an example of the standard watershed transform. Unfortunately, simply weighting each arc by the mean value of $E(x, y)$ for the pixels on the arc can introduce artifacts. The root cause of this problem is the fact that the contour detector produces a spatially extended response around strong boundaries. For example, a pixel could lie near but not on a strong vertical contour. If this pixel also happens to belong to a horizontal watershed arc, that arc would be erroneously upweighted. Several such cases can be seen in Figure 3.3. As we flood from all local minima, the initial watershed oversegmentation contains many arcs that should be weak, yet intersect nearby strong boundaries.

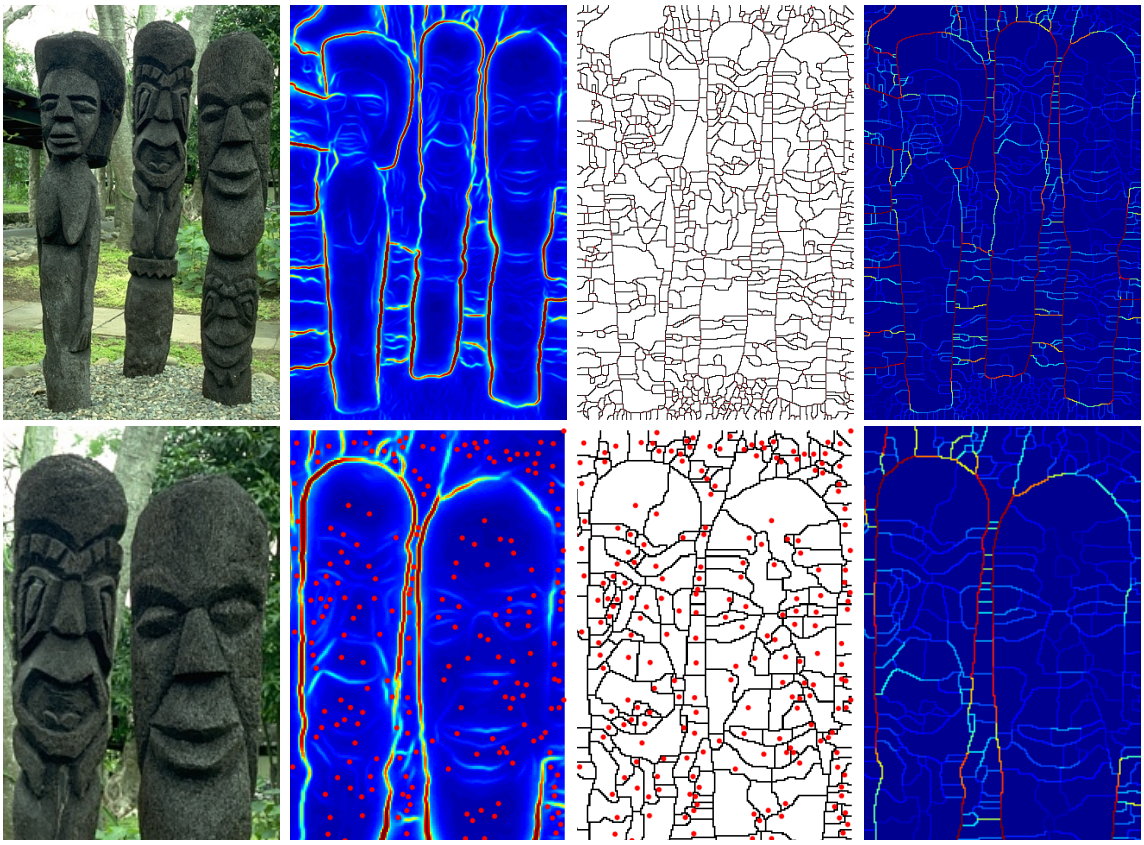

Figure 3.3: **Watershed Transform. Left:** Image. **Middle Left:** Boundary strength $E(x, y)$. We regard $E(x, y)$ as a topographic surface and flood it from its local minima. **Middle Right:** This process partitions the image into catchment basins $\mathcal{P}_0$ and arcs $\mathcal{K}_0$. There is exactly one basin per local minimum and the arcs coincide with the locations where the floods originating from distinct minima meet. The bottom row displays a magnified view for the upper-right portion of the image with each local minimum marked with a red dot. **Right:** Each arc weighted by the mean value of $E(x, y)$ along it. This weighting scheme produces artifacts, such as the strong horizontal contours in the small gap between the two statues seen in the magnified view.

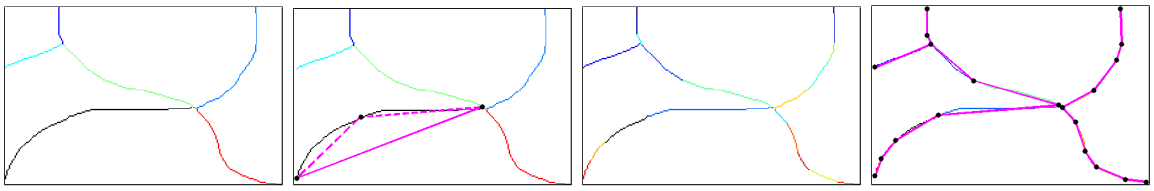

Figure 3.4: **Contour subdivision. Left:** Initial arcs color-coded. **Middle Left:** For each arc, we consider the straight line segment connecting its endpoints. If the distance from any point on the arc to this line segment is greater than a fixed fraction of the segment length, we subdivide the arc at the maximally distant point. An example is shown for one arc, with the dashed segments indicating the new subdivision. **Middle Right:** The final set of arcs resulting from recursive application of the scale-invariant subdivision procedure. **Right:** Approximating straight line segments overlaid on the subdivided arcs.

To correct this problem, we enforce consistency between the strength of the boundaries of $\mathcal{K}_0$ and the underlying $E(x, y, \theta)$ signal in a modified procedure, which we call the Oriented Watershed Transform (OWT). As the first step in this reweighting process, we estimate an orientation at each pixel on an arc from the local geometry of the arc itself. These orientations are obtained by approximating the watershed arcs with line segments as shown in Figure 3.4. We recursively subdivide any arc which is not well fit by the line segment connecting its endpoints. By expressing the approximation criteria in terms of the maximum distance of a point on the arc from the line segment as a fraction of the line segment length, we obtain a scale-invariant subdivision. We assign each pixel (x, y) on a subdivided arc the orientation $o(x, y) \in [0, \pi)$ of the corresponding line segment.

Next, we use the oriented contour detector output $E(x, y, \theta)$, to assign each arc pixel (x, y) a boundary strength of $E(x, y, o(x, y))$. Here we quantize $o(x, y)$ in the same manner as θ, so this operation is a simple lookup. Finally, each original arc in $\mathcal{K}_0$ is assigned weight equal to average boundary strength of the pixels it contains. Figure 3.5 illustrates how this reweighting scheme removes artifacts.

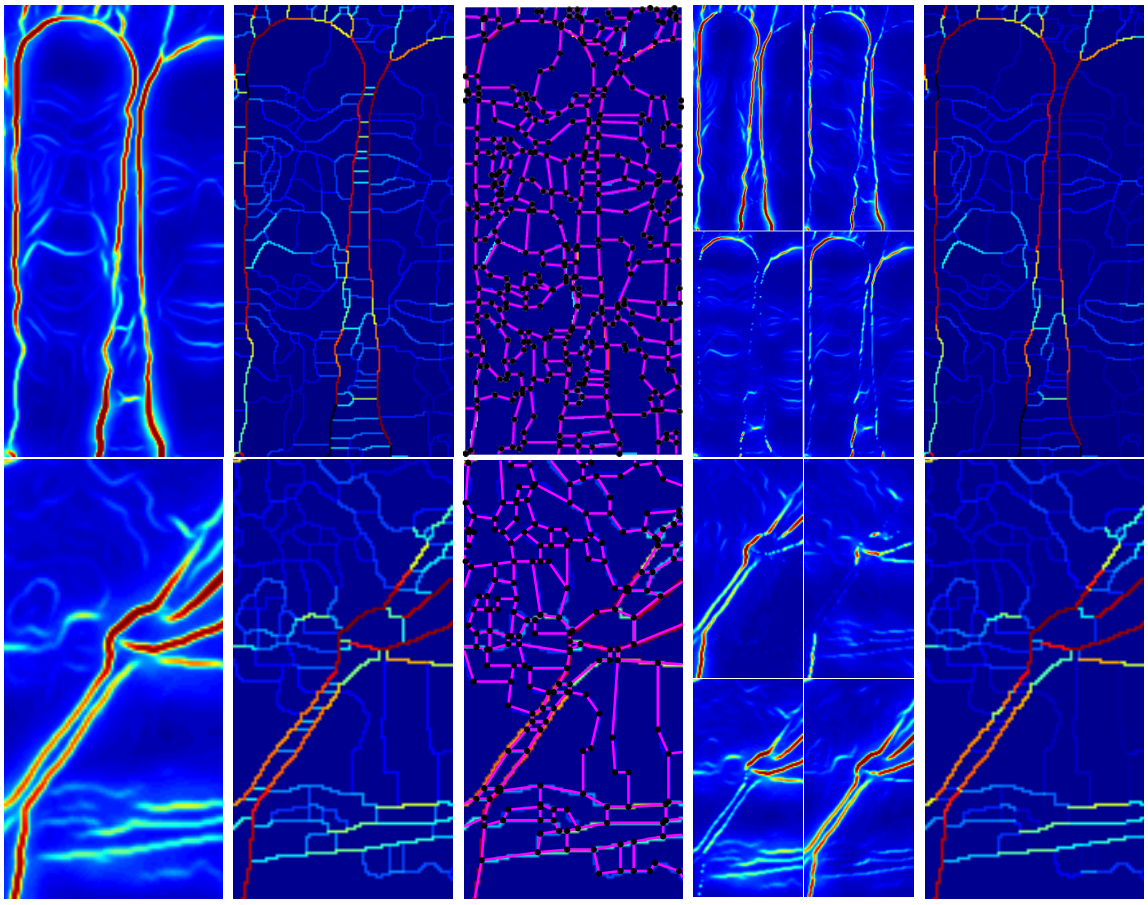

Figure 3.5: **Oriented Watershed Transform. Left:** Input boundary signal $E(x, y) = \max_\theta E(x, y, \theta)$. **Middle Left:** Watershed arcs computed from $E(x, y)$. Note that thin regions give rise to artifacts. **Middle:** Watershed arcs with an approximating straight line segment subdivision overlaid. We compute this subdivision in a scale-invariant manner by recursively breaking an arc at the point maximally distant from the straight line segment connecting its endpoints, as shown in Figure 3.4. Subdivision terminates when the distance from the line segment to every point on the arc is less than a fixed fraction of the segment length. **Middle Right:** Oriented boundary strength $E(x, y, \theta)$ for four orientations θ. In practice, we sample eight orientations. **Right:** Watershed arcs reweighted according to E at the orientation of their associated line segments. Artifacts, such as the horizontal contours breaking the long skinny regions, are suppressed as their orientations do not agree with the underlying $E(x, y, \theta)$ signal.

3.2.2 Ultrametric Contour Map

Contours have the advantage that it is fairly straightforward to represent uncertainty in the presence of a true underlying contour, *i.e.* by associating a binary random variable to it. One can interpret the boundary strength assigned to an arc by the Oriented Watershed Transform (OWT) of the previous section as an estimate of the probability of that arc being a true contour.

It is not immediately obvious how to represent uncertainty about a segmentation. One possibility, which we exploit here, is the Ultrametric Contour Map (UCM) [Arbeláez, 2006] which defines a duality between closed, non-self-intersecting weighted contours and a hierarchy of regions. The base level of this hierarchy respects even weak contours and is thus an oversegmentation of the image. Upper levels of the hierarchy respect only strong contours, resulting in an under-segmentation. Moving between levels offers a continuous trade-off between these extremes. Making this shift in representation from a single segmentation to a nested collection of segmentations frees later processing stages to use information from multiple levels or select a level based on additional knowledge.

Our hierarchy is constructed by a greedy graph-based region merging algorithm. We define an initial graph $G = (\mathcal{P}_0, \mathcal{K}_0, W(\mathcal{K}_0))$, where the nodes are the regions $\mathcal{P}_0$, the links are the arcs $\mathcal{K}_0$ separating adjacent regions, and the weights $W(\mathcal{K}_0)$ are a measure of similarity between regions. The algorithm proceeds by sorting the links by similarity and iteratively merging the most similar regions. Specifically:

1. Select minimum weight contour $C^* = \text{argmin}_{C \in \mathcal{K}_0} W(C)$.
2. Let $R_1, R_2 \in \mathcal{P}_0$ be the regions separated by C^*.
3. Set $R = R_1 \cup R_2$, and update $\mathcal{P}_0 \leftarrow \mathcal{P}_0 \backslash \{R_1, R_2\} \cup \{R\}$, and $\mathcal{K}_0 \leftarrow \mathcal{K}_0 \backslash \{C^*\}$.
4. Stop if $\mathcal{K}_0$ is empty. Otherwise, update weights $W(\mathcal{K}_0)$ and repeat.

This process produces a tree of regions, where the leaves are the initial elements of $\mathcal{P}_0$, the root is the entire image domain, and the regions are ordered by the inclusion relation.

We define similarity between two adjacent regions as the average strength of their common boundary in $\mathcal{K}_0$, with weights $W(\mathcal{K}_0)$ initialized by the OWT. Since at every step of the algorithm all remaining contours must have strength greater than or equal to those previously removed, the weight of the contour currently being removed cannot decrease during the merging process. Hence, the constructed region tree has the structure of an indexed hierarchy and can be described by a dendrogram, where the height $H(R)$ of each region R is the value of the similarity at which it first appears. Stated equivalently, $H(R) = W(C)$ where C is the contour whose removal formed R. The hierarchy also yields a metric on $\mathcal{P}_0 \times \mathcal{P}_0$, with the distance between two regions given by the height of the smallest segment containing them:

$$D(R_1, R_2) = \min\{H(R) : R_1, R_2 \subseteq R\} \tag{3.1}$$

This distance satisfies the ultrametric property:

$$D(R_1, R_2) \leq \max(D(R_1, R), D(R, R_2)) \tag{3.2}$$

since if R is merged with R_1 before R_2, then $D(R_1, R_2) = D(R, R_2)$, or if R is merged with R_2 before R_1, then $D(R_1, R_2) = D(R_1, R)$. As a consequence, the whole hierarchy can be represented as an Ultrametric Contour Map (UCM) [Arbeláez, 2006], the real-valued image obtained by weighting each boundary between two regions by its scale of disappearance.

Figure 3.6 presents an example of our method. The UCM is a weighted contour image that, by construction, has the remarkable property of producing a set of closed

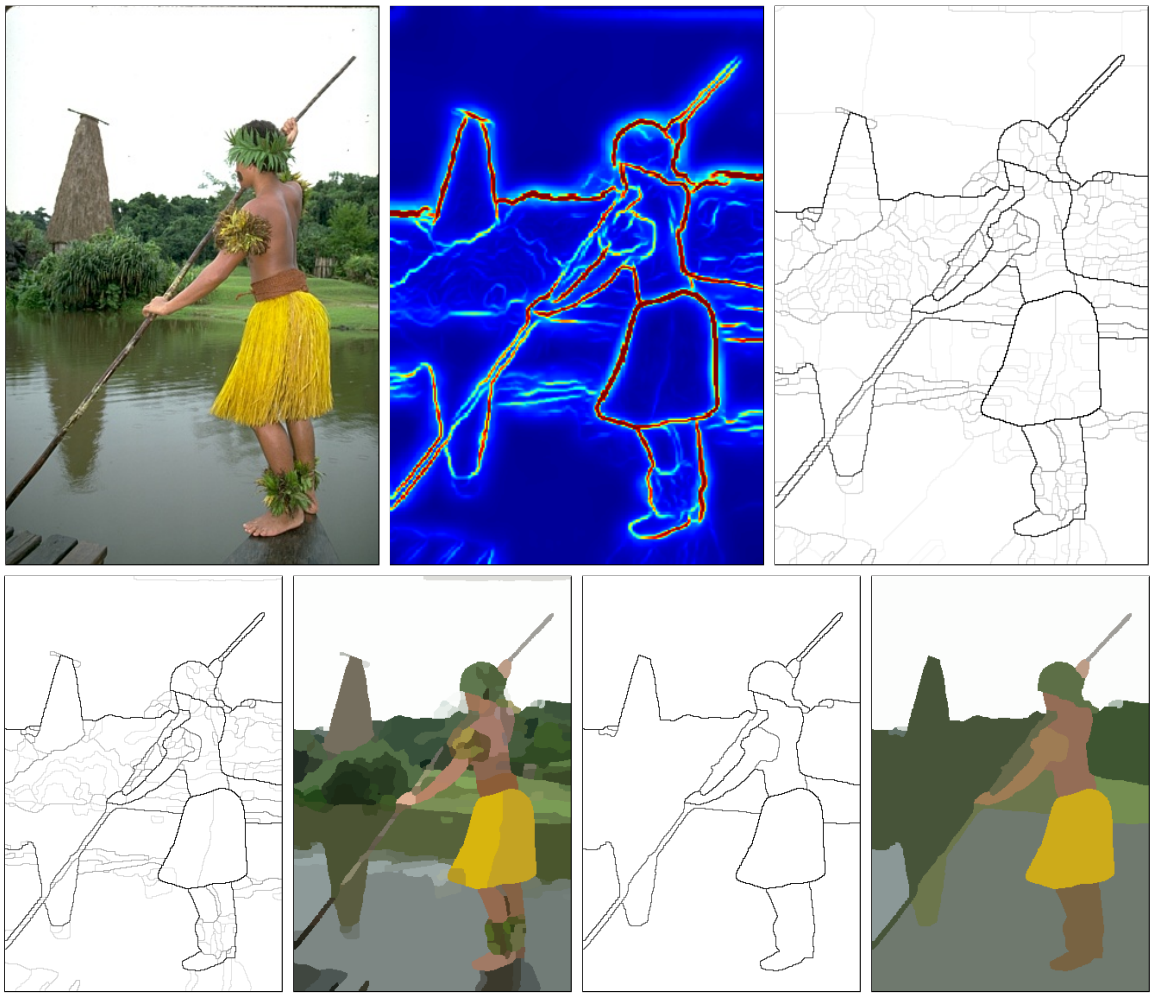

Figure 3.6: **Hierarchical segmentation from contours. Top Left:** Original image. **Top Middle:** Maximal response of contour detector gPb over orientations. **Top Right:** Weighted contours resulting from the Oriented Watershed Transform - Ultrametric Contour Map (OWT-UCM) algorithm using gPb as input. This single weighted image encodes the entire hierarchical segmentation. By construction, applying any threshold to it is guaranteed to yield a set of closed contours (the ones with weights above the threshold), which in turn define a segmentation. Moreover, the segmentations are nested. Increasing the threshold is equivalent to removing contours and merging the regions they separated. **Bottom:** Contours and corresponding segmentations obtained by thresholding the UCM at levels 0.1 (left), and 0.5 (right), with segments represented by their mean color.

curves for any threshold. Conversely, it is a convenient representation of the region tree since the segmentation at a scale k can be easily retrieved by thresholding the UCM at level k. Since our notion of scale is the average contour strength, the UCM values reflect the contrast between neighboring regions.

3.3 Results

While the OWT-UCM algorithm can use any source of contours, *e.g.* the Canny edge detector before thresholding, for the input $E(x, y, \theta)$ signal, for best results, we employ the gPb detector [Maire *et al.*, 2008] introduced in the previous chapter.

We report experiments using both gPb as well as the baseline Canny detector, and refer to the resulting segmentation algorithms as gPb-owt-ucm and Canny-owt-ucm, respectively. Figures 3.7 and 3.8 illustrates results of gPb-owt-ucm on images from the Berkeley Segmentation Dataset (BSDS) [Martin *et al.*, 2001].

3.4 Evaluation

To provide a basis of comparison for the performance of the OWT-UCM algorithm, we make use of the Felzenszwalb and Huttenlocher (Felz-Hutt) region merging [Felzenszwalb and Huttenlocher, 2004], Mean Shift [Comaniciu and Meer, 2002], and Multi-scale Normalized Cuts (NCuts) [Cour *et al.*, 2005] segmentation methods. We evaluate each method using multiple benchmark criteria.

3.4.1 Benchmarks

Following Chapter 2, we again make use of the Berkeley Segmentation Dataset for evaluation, as this large collection captures the diversity, yet high consistency, of

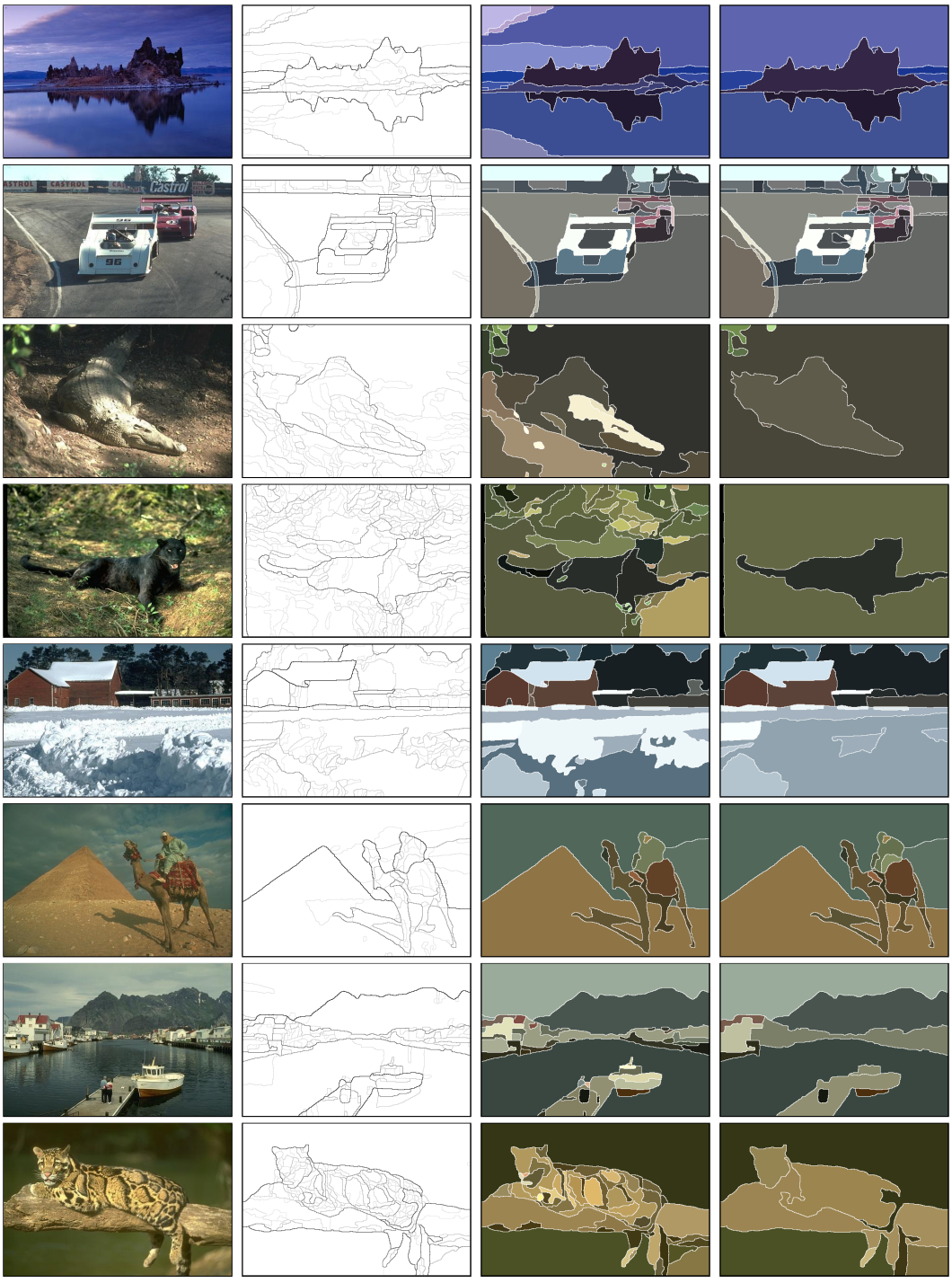

Figure 3.7: **Hierarchical segmentation results. From left to right:** Original image, Ultrametric Contour Map (UCM) produced by gPb-owt-ucm, and segmentations obtained by thresholding at the optimal dataset scale (ODS) and optimal image scale (OIS). All images are from the BSDS test set.

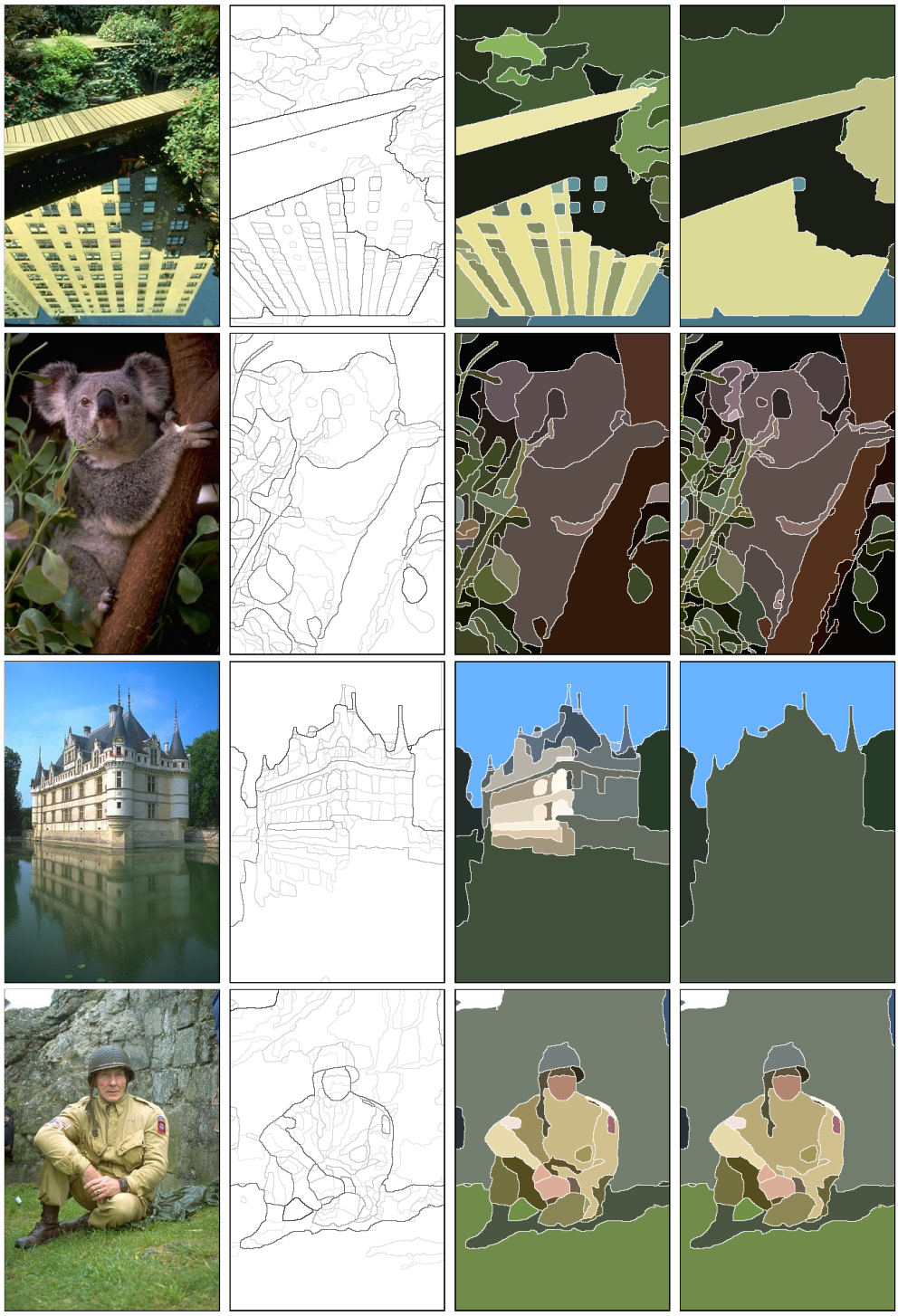

Figure 3.8: **Additional hierarchical segmentation results. From left to right:** Original image, gPb-owt-ucm, and segmentations obtained at the optimal dataset scale (ODS) and optimal image scale (OIS). All images are from the BSDS test set.

human segmentation. However, in addition to the standard metric for boundary quality, we also examine metrics for comparing regions against human ground-truth.

3.4.1.1 Precision-Recall on Boundaries

Remember from Section 2.1, that the the boundary-based evaluation methodology developed by [Martin *et al.*, 2004] measures detector performance in terms of precision and recall, where precision is the fraction of true positives and recall the fraction of ground-truth boundaries detected. The global F-measure, or harmonic mean of precision and recall at the optimal detector threshold (scale), provides a useful summary score.

In our experiments, we report three different quantities for an algorithm: the **Optimal Dataset Scale (ODS)** or best F-measure on the dataset for a fixed scale, the **Optimal Image Scale (OIS)** or aggregate F-measure on the dataset for the best scale in each image, and the **Average Precision (AP)** on the full recall range (equivalently, the area under the precision-recall curve), shown in Table 3.1 for the BSDS.

This benchmarking system possesses the appealing property that it allows the comparison of region-based segmentation and contour detection methods in the same framework. Any segmentation algorithm automatically provides contours in the form of the boundaries of the regions in the segmentation. Figure 3.1 takes advantage of this fact to compare the boundaries produced by the OWT-UCM algorithm to the contours used as input. This comparison shows that we can transform contours to regions while preserving boundary quality and actually obtaining a small improvement in the case of gPb-owt-ucm.

However, for segmentation algorithms, a methodology that directly evaluates the quality of the segments is also desirable. Some types of errors, *e.g.* a missing pixel in the boundary between two regions, may not be reflected in the boundary bench-

Method	ODS	OIS	AP
human	0.79	0.79	–
gPb-owt-ucm	**0.71**	**0.74**	**0.77**
Mean Shift	0.63	0.66	0.62
NCuts	0.62	0.66	0.59
Canny-owt-ucm	0.58	0.63	0.59
Felz-Hutt	0.58	0.62	0.54
gPb	0.70	0.72	0.75
Canny	0.58	0.62	0.60

Table 3.1: **Boundary benchmarks on the BSDS.** We benchmark boundaries produced by five different segmentation methods (upper table) and two contour detectors (lower table). Shown are the F-measures when choosing an optimal scale for the entire dataset (ODS) or per image (OIS), as well as the average precision (AP). Figures 3.1 and 3.2 show the full precision-recall curves for the boundaries produced by these algorithms.

mark, but can have substantial consequences for segmentation quality, *e.g.* incorrectly merging two large regions. It can also be argued that the boundary benchmark favors contour detectors over segmentation methods, since the former are not burdened with the constraint of producing closed curves. We therefore also consider various region-based metrics.

3.4.1.2 Variation of Information

This metric was introduced for the purpose of clustering comparison. It measures the distance between two segmentations in terms of their average conditional entropy given by:

$$VI(C, C') = H(C) + H(C') - 2I(C, C') \tag{3.3}$$

where H and I represent respectively the entropies and mutual information between two clusterings of data C and C'. In our case, the two clusterings are test and ground-truth segmentations. Although VI possesses some interesting theoretical properties [Meila, 2005], its perceptual meaning and applicability in the presence of several

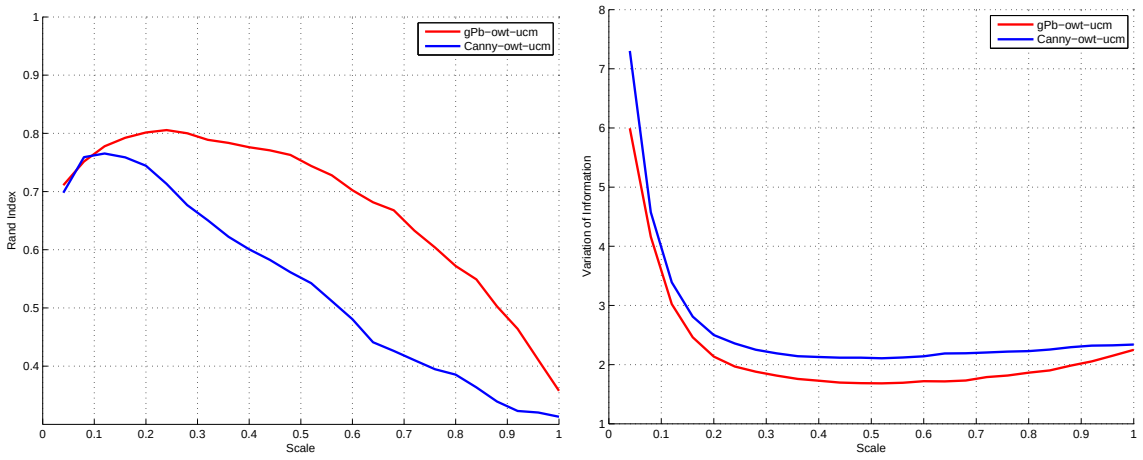

Figure 3.9: **Evaluating regions on the BSDS.** Contour detector influence on segmentation quality is evident when benchmarking the regions of the resulting hierarchical segmentation. **Left:** Probabilistic Rand Index. **Right:** Variation of Information.

ground-truth segmentations remains unclear.

3.4.1.3 Rand Index

Originally, the Rand Index [Rand, 1971] was introduced for general clustering evaluation. It operates by comparing the compatibility of assignments between pairs of elements in the clusters. In our case, the Rand Index between test and ground-truth segmentations S and G is given by the sum of the number of pairs of pixels that have the same label in S and G and those that have different labels in both segmentations, divided by the total number of pairs of pixels. Variants of the Rand Index have been proposed [Unnikrishnan *et al.*, 2007; Yang *et al.*, 2008] for dealing with the case of multiple ground-truth segmentations. Given a set of ground-truth segmentations $\{G_k\}$, the Probabilistic Rand Index is defined as:

$$PRI(S, \{G_k\}) = \frac{1}{T} \sum_{i<j} [c_{ij} p_{ij} + (1 - c_{ij})(1 - p_{ij})] \tag{3.4}$$

Method	ODS	OIS	Best		PRI	VI
human	0.73	0.73	–		0.87	1.16
gPb-owt-ucm	**0.58**	**0.64**	**0.74**		**0.81**	**1.68**
Mean Shift	0.54	0.58	0.64		0.78	1.83
Felz-Hutt	0.51	0.58	0.68		0.77	2.15
Canny-owt-ucm	0.48	0.56	0.67		0.77	2.11
NCuts	0.44	0.53	0.66		0.75	2.18

Table 3.2: **Region benchmarks on the BSDS.** For each segmentation method, the leftmost three columns report the score of the covering of ground-truth segments according to optimal dataset scale (ODS), optimal image scale (OIS), or Best covering criteria. The rightmost two columns compare the segmentation methods against ground-truth using the probabilistic Rand Index (PRI) and Variation of Information (VI) benchmarks, respectively.

where c_{ij} is the event that pixels i and j have the same label and p_{ij} its probability. T is the total number of pixel pairs considered. When the sample mean is used to estimate p_{ij}, Equation 3.4 amounts to averaging the Rand Index among different ground-truth segmentations. However, the PRI has been reported to suffer from a small dynamic range [Unnikrishnan *et al.*, 2007; Yang *et al.*, 2008], and its values across images and algorithms are often very similar. In [Unnikrishnan *et al.*, 2007], this drawback is addressed by normalization with an empirical estimation of its expected value.

3.4.1.4 Segmentation Covering

The **overlap** between two regions R and R', defined as:

$$\mathcal{O}(R, R') = \frac{|R \cap R'|}{|R \cup R'|} \tag{3.5}$$

has been used for the evaluation of the pixel-wise classification task in recognition [Malisiewicz and Efros, 2007; Everingham *et al.*, 2008].

We define the **covering** of a segmentation S by a segmentation S' as:

$$\mathcal{C}(S' \to S) = \frac{1}{N} \sum_{R \in S} |R| \cdot \max_{R' \in S'} \mathcal{O}(R, R') \tag{3.6}$$

where N denotes the total number of pixels in the image.

Similarly, the covering of a machine segmentation S by a family of ground truth segmentations $\{G_i\}$ is defined by, first covering S separately with each human map $\{G_i\}$ in turn, and then averaging over the different humans, so that to achieve perfect covering the machine segmentation must explain all of the human data.

We can then define two quality descriptors for regions: the covering of S by $\{G_i\}$ and the covering of $\{G_i\}$ by S. We include results for the latter. For a family of machine segmentations $\{S_i\}$, corresponding to different scales of a hierarchical algorithm or different sets of parameters, we report the **Optimal Dataset Scale (ODS)**, **Optimal Image Scale (OIS)**, and the **Best** possible covering of the ground-truth by segments in $\{S_i\}$.

Figure 3.9 and Table 3.2 present region benchmarks on the BSDS. While the relative ranking of segmentation algorithms remains fairly consistent across different benchmark criteria, the boundary benchmark (Table 3.1) appears most capable of discriminating performance.

3.4.2 Additional Datasets

We concentrated experiments on the BSDS because it is the most complete dataset available for our purposes, has been used in several publications, and has the advantage of providing several human-labeled segmentations per image. Table 3.3 reports the comparison between Canny-owt-ucm and *gPb*-owt-ucm on two other publicly available datasets:

MSRC	ODS	OIS	Best
gPb-owt-ucm	**0.66**	**0.75**	**0.78**
Canny-owt-ucm	0.57	0.68	0.72
PASCAL 2008	ODS	OIS	Best
gPb-owt-ucm	**0.45**	**0.58**	**0.61**
Canny-owt-ucm	0.40	0.53	0.55

Table 3.3: **Region benchmarks on MSRC and PASCAL 2008.** Shown are scores for the segment covering criteria used in Table 3.2.

- **MSRC** [Shotton *et al.*, 2006]

  The MSRC object recognition database is composed of 591 natural images with objects belonging to 21 classes. We evaluate performance using the ground-truth object instance labeling of [Malisiewicz and Efros, 2007], which is cleaner and more precise than the original data.

- **PASCAL 2008** [Everingham *et al.*, 2008]

  We use the train and validation sets of the segmentation task on the PASCAL challenge 2008, composed of 1023 images. This is one of the most difficult and varied datasets for recognition. We evaluate performance with respect to the object instance labels provided. Note that only objects belonging to the 20 categories of the challenge are labeled, and 76% of all pixels are unlabeled.

The gPb-owt-ucm segmentation algorithm offers the best performance on every dataset and for every benchmark criterion we tested. In addition, it is straightforward, fast, has no parameters to tune, and, as discussed in the next section, supports interactive user refinement. Our generic segmentation machinery has found use in optical flow [Brox *et al.*, 2009] and object recognition [Gu *et al.*, 2009] applications.

3.5 Interactive Segmentation

Until now, we have only discussed fully automatic image segmentation. Human assisted segmentation is relevant for many applications, and recent approaches rely on the graph-cuts formalism [Boykov and Jolly, 2001; Rother *et al.*, 2004; Li *et al.*, 2004] or other energy minimization procedure [Bagon *et al.*, 2008] to extract single foreground regions.

For example, [Boykov and Jolly, 2001] cast the task of determining binary foreground/background pixel assignments in terms of a cost function with both unary and pairwise potentials. The unary potentials encode agreement with estimated foreground or background region models and the pairwise potentials bias neighboring pixels not separated by a strong boundary to have the same label. They transform this system into an equivalent minimum cut/maximum flow graph partitioning problem through the addition of a source node representing the foreground and a sink node representing the background. Edge weights between pixel nodes are defined by the pairwise potentials, while the weights between pixel nodes and the source and sink nodes are determined by the unary potentials. User-specified hard labeling constraints are enforced by connecting a pixel to the source or sink with sufficiently large weight. The minimum cut of the resulting graph can be computed efficiently and produces a cost-optimizing assignment.

It turns out that the segmentation trees generated by the OWT-UCM algorithm provide a natural starting point for user-assisted refinement. Following the procedure of [Arbeláez and Cohen, 2008], we can extend a partial labeling of regions to a full one by assigning to each unlabeled region the label of its closest labeled region, as determined by the ultrametric distance (Equation 3.1). Computing the full labeling is simply a matter of propagating information in a single pass along the segmentation tree. Each unlabeled region receives the label of the first labeled region merged with

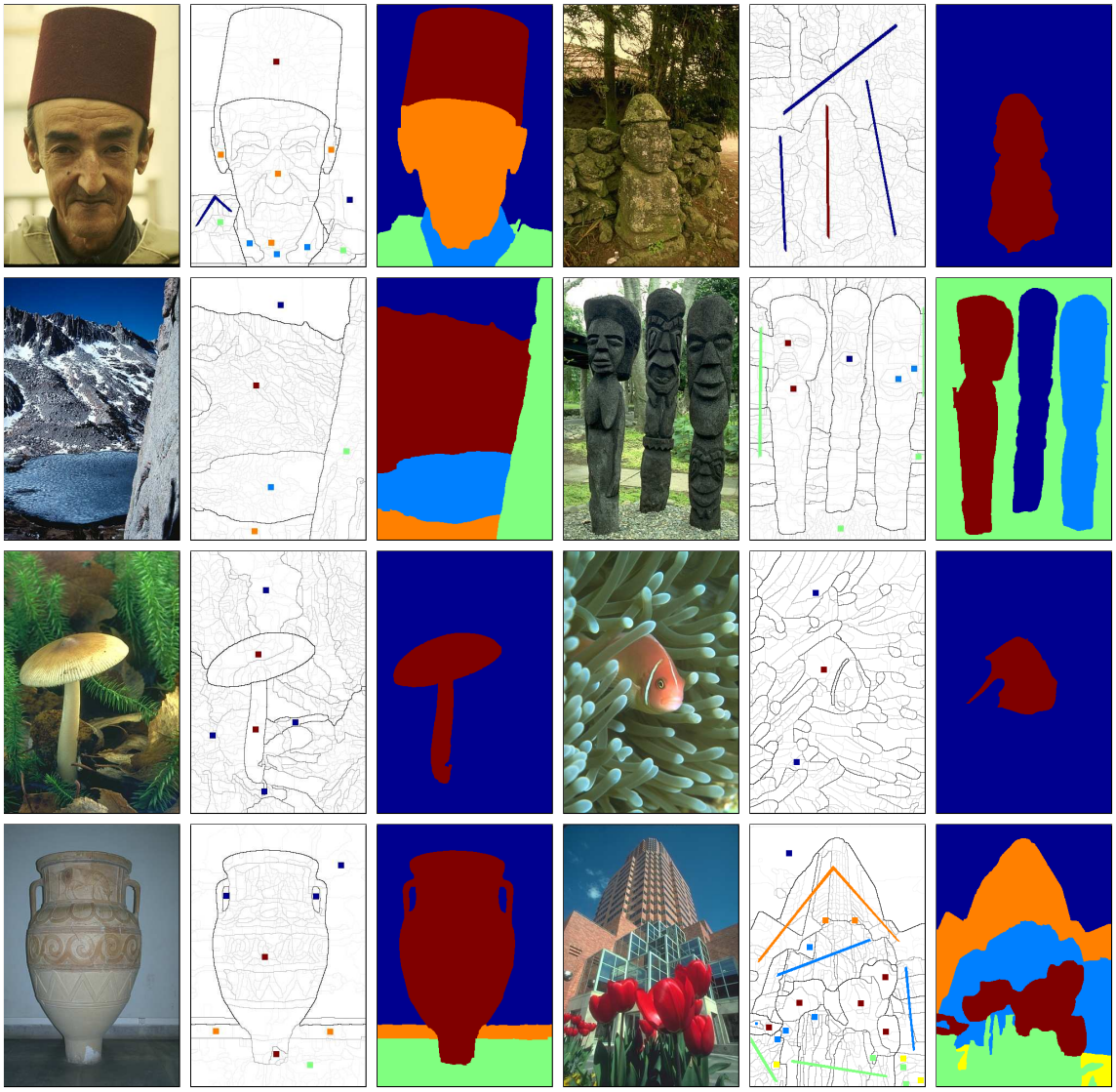

Figure 3.10: **Interactive segmentation. Left:** Original image. **Middle:** UCM produced by gPb-owt-ucm (grayscale) with additional user annotations (color dots and lines). **Right:** The region hierarchy defined by the UCM allows us to automatically propagate annotations to unlabeled segments, resulting in the desired labeling of the image with minimal user effort.

it. This procedure, illustrated in Figure 3.10, allows a user to obtain high quality results with minimal annotation.

3.6 Regions vs Bounding Boxes

Scanning approaches, which exhaustively sample image windows over a large range of positions, scales, and aspect ratios, feeding each to a classifier, have been a dominant strategy in many object recognition settings. This includes face [Viola and Jones, 2004] and pedestrian [Dalal and Triggs, 2005] detection, and the PASCAL challenge [Felzenszwalb *et al.*, 2008]. As the number of categories a recognition system is expected to handle increases, it is unclear whether such a brute force approach is capable of scaling well. Classifier cascades [Viola and Jones, 2004] or branch and bound search strategies [Lampert *et al.*, 2008] may help, but it seems unnecessary to place the entire burden on the classification machinery without taking advantage of perceptual cues.

Here, we conduct some initial experiments in an attempt to determine whether automatically generated regions can serve as a replacement for the scanning window approach. In particular, do regions provide sufficient coverage of objects in a scene so that one could feed the regions to a classifier instead of sampling windows? Furthermore, can this be accomplished with relatively few (as compared to sliding windows) regions per scene?

As a test, we consider the segmentation subset of the PASCAL 2008 dataset [Everingham *et al.*, 2008] and ask what is the best overlap, as defined by the intersection over union criterion (Equation 3.5), of a region with the human-marked ground-truth segmentation? We ask the same question for bounding boxes. The answer should yield an idea of the maximum recall one can expect when using either regions or sliding windows, respectively, as interest operators for an object detector.

Region Selection Method	Mean Overlap	Average # Regions per Image
Bounding Box (oracle)	0.62	–
Best Box (oracle)	0.66	–
gPb-owt-ucm tree (all scales)	**0.67**	2138
gPb-owt-ucm tree (scale 1)	0.60	1552
gPb-owt-ucm tree (scale 0.5)	0.61	452
gPb-owt-ucm tree (scale 0.25)	0.61	134

Table 3.4: **Regions provide recall comparable to sliding windows on PASCAL 2008.** Selecting the region best overlapping with the ground-truth segmentation achieves about the same degree of overlap as the optimal box chosen by an oracle. This is true when one is given the freedom to select the region from any of three hierarchical segmentation trees, constructed from resized versions of the original image.

Now, let's bias this benchmark in favor of sliding windows by assuming we have an oracle that tells us the true bounding box (location, size, and aspect ratio) for each ground-truth region. We compare this oracle to our actual algorithm (gPb-owt-ucm) for generating regions. The only penalty the bounding box oracle pays is for inclusion of background pixels. Let's also consider a "best box" oracle that is allowed to shrink the bounding box inward, maintaining the same aspect, to improve the ratio of object to background pixels.

Since objects may appear at different scales, we pick the best overlapping region appearing anywhere in the hierarchical segmentation tree output of the gPb-owt-ucm algorithm. Moreover, since the PASCAL dataset contains a larger range of object scale than BSDS, we compute three different segmentation trees, one each on the original image and images downsampled by factors of 0.5 and 0.25 in each dimension. The cost of examining more regions in this manner will have to be justified by better coverage.

Table 3.4 presents the results of this experiment. In computing the mean overlap scores, we weight each ground-truth object by its total area, so that detecting large objects is more important than detecting small ones. Selecting the best region over

the three segmentation trees on average performs slightly better than the best box oracle and significantly better than the bounding box oracle. Figures 3.11 and 3.12 provide a visual comparison.

Furthermore, to achieve the performance of the oracle in practice, one would have to sample bounding boxes over location, scale, and aspect ratio, producing far more candidates than the number of regions considered. Also note that of the on average 2138 regions examined per image across all scales, some are similar to one another. Thus, the list of candidate regions could likely be further pruned without much of a decrease in overlap score.

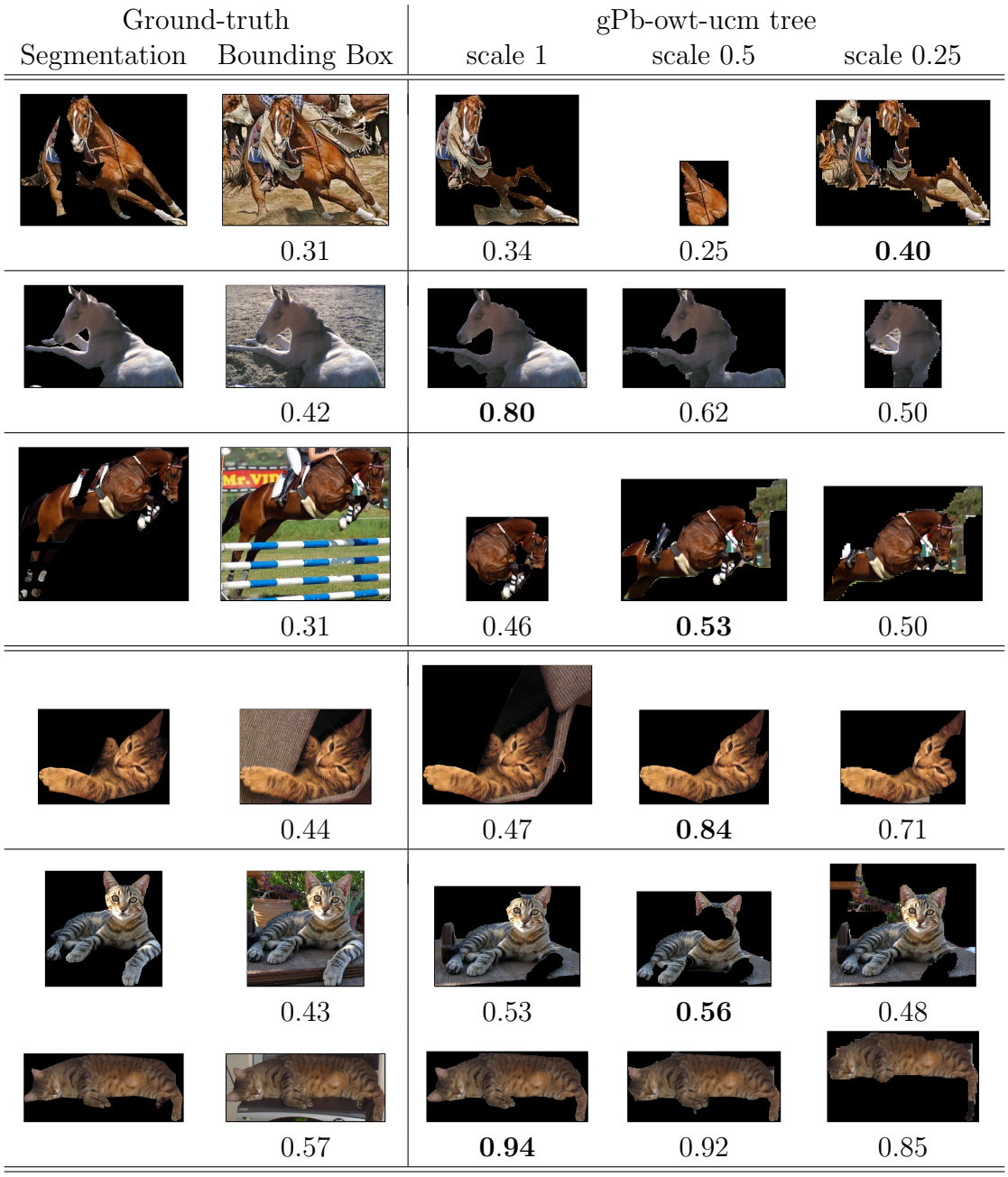

Figure 3.11: **Region selection results on PASCAL 2008.** From each of three different hierarchical segmentation trees, created by running the gPb-owt-ucm algorithm on rescaled versions of the original image, we select the region best overlapping the ground-truth segmentation. Overlap (intersection divided by union) scores are displayed below each region. The best region tends to score better than the ground-truth bounding box.

Ground-truth		gPb-owt-ucm tree		
Segmentation	Bounding Box	scale 1	scale 0.5	scale 0.25
	 0.70	 0.83	 0.77	 **0.85**
	 **0.62**	 0.56	 0.54	 0.46
	 0.51	 **0.86**	 0.84	 0.65
	 0.44	 0.51	 **0.53**	 0.51
	 0.39	 0.55	 0.48	 **0.60**
	 0.34	 0.41	 0.48	 **0.76**
	 **0.61**	 0.38	 0.57	 0.59
	 0.58	 0.67	 0.68	 **0.78**

Figure 3.12: **Additional region selection results on PASCAL 2008.**

Chapter 4

Extensions to Video

4.1 Introduction

Biological vision systems make use of optical flow for a number of purposes, such as egomotion estimation and scene structure recovery, the latter including both metric depth estimates and ordinal relationships like figure/ground. In this chapter, we focus particularly on the role of motion for grouping and figure/ground assignment. The importance of motion cues in these tasks is a classic point in the psychophysical literature. Koffka stated the Gestalt principle of "common fate" where similarly moving points are perceived as coherent entities [Koffka, 1935], and grouping based on motion was emphasized by numerous other works including Gibson, who also pointed out occlusion/disocclusion phenomena. In contrast to color and stereopsis, which also help to separate different objects, motion is a cue shared by all visual species - a fact that emphasizes its importance in biological systems.

In the computational vision literature, much work has dealt with the problem of optical flow estimation over the past three decades, and there have been numerous approaches to make use of the optical flow field for grouping [Darrell and Pentland, 1991;

Shi and Malik, 1998; Smith *et al.*, 2004; Xiao and Shah, 2005; Cremers and Soatto, 2005]. Most of them are similar to the work of [Wang and Adelson, 1994], which proposes to partition the image into motion layers by clustering similar optical flow vectors according to a parametric motion model. While this approach is attractive in the sense that it directly provides object regions, there are many cases that are not properly captured by parametric models. To deal with this shortcoming, [Weiss, 1997] suggested a nonparametric version of layered motion, where each layer is described by a smooth flow field. Similar techniques based on level sets have been presented in [Amiaz and Kiryati, 2006; Brox *et al.*, 2006]. The problem with these nonparametric layer models is the susceptibility of the EM procedure to local minima, particularly in areas of the image with little structure.

An alternative strategy is to detect occlusion edges and to infer layers from these edges afterwards, if needed. As demonstrated in the previous chapter, such a strategy works well for segmentation of static images [Arbeláez *et al.*, 2009], and it makes even more sense for grouping based on motion cues, where additional difficulties due to the aperture problem limit the reliability of typical EM style procedures.

Only a few papers have dealt with explicitly detecting occlusion boundaries based on motion cues and assigning figure/ground labels to both sides of these boundaries. In [Black and Fleet, 2000], initial motion boundaries are obtained from a motion estimator and then serve a probabilistic state model that can finally distinguish occlusion boundaries from internal boundaries and can assign figure/ground labels to the regions on both sides of the edge. A particle filter approach is employed to deal with the complex, multi-modal distributions in the high-dimensional state space. In [Gaucher and Medioni, 1999], tensor voting yields optical flow estimates together with an uncertainty measure based on the homogeneity of the votes. Occlusion boundaries are assumed to be maxima in the uncertainty measure. Both [Black and Fleet, 2000] and [Gaucher and Medioni, 1999] do not make use of static cues. Since the optical

flow as a secondary feature requires integration over a spatial domain to deal with the aperture problem, edges based only on motion estimates are usually inaccurate and dislocated in the normal direction of the occluding edge.

In contrast, [Smith *et al.*, 2004] present a method that relies only on static edges and their motion. Edges are computed by the Canny edge detector and an affine motion model for edge fragments is computed via block matching. The authors also provide a depth ordering of both sides of an edge by reasoning at t-junctions. The work of [Stein and Hebert, 2009] also makes use of a static edge detector from [Martin *et al.*, 2004] to obtain an initial set of potential occlusion boundaries. They then learn a classifier that distinguishes occlusion boundaries from internal edges based on both static and motion features.

We make three distinct contributions. First, we extend the gPb boundary detector [Maire *et al.*, 2008] of Chapter 2 to exploit motion cues in video. Only low-level motion cues are taken into account at this stage, as the goal is simply to augment the static cues and provide a more robust set of boundaries to our module for occlusion reasoning. Second, we propose a technique for distinguishing occlusion boundaries from internal boundaries based only on optical flow that clearly outperforms the most recent work by [Stein and Hebert, 2009] on their data set. Finally, we assign figure/ground labels to both sides of each occlusion boundary and we provide a new labeled dataset for evaluating performance on this task. This dataset can be regarded as a video counterpart to the Berkeley Segmentation data set on static images [Martin *et al.*, 2001]. Given the importance of occlusion boundaries and figure/ground assignments for visual perception, we believe this data will be helpful in comparing alternative algorithms and advancing research in the field.

4.2 Motion Gradient

Edge detection was one of the earliest problems addressed in computer vision, yet current contour detectors, discussed in Chapter 2, are designed for analyzing single images and rely only on static cues such as brightness, color, and texture. When working with video, we have the additional cue of motion and would like to take advantage of it at all levels of processing involved in figure/ground labeling, starting with the boundary detection task itself.

We find biological motivation for this strategy as there is evidence that motion serves as an important low-level cue for human vision. For example, Figure 4.1 illustrates the extreme case of a uniformly textured object moving in front of a background with the same uniform texture. Here, there are no static cues, yet humans easily perceive the motion of the foreground object when viewing the video sequence. We compute a new cue, which we refer to as the motion gradient, that correctly detects the boundary of this moving object in each frame.

Our motion gradient signal can be thought of as a temporal analog to the brightness, color, and texture gradients introduced in Section 2.2.1. To compute the motion gradient for grayscale video frame I_t, we first compute temporal derivatives with respect to the previous and subsequent frames:

$$D^- = I_t - I_{t-1} \tag{4.1}$$

$$D^+ = I_t - I_{t+1} \tag{4.2}$$

Then, we consider the oriented gradient operator from Section 2.2.1, $G_r(x, y, \theta)$, which measures the χ^2 difference of the histograms of values in the two halves of a radius r disc centered at (x, y) and divided by a diameter at angle θ. We sample θ for 8 orientations in the interval $[0, \pi)$ and apply this gradient operator to D^- and D^+ to

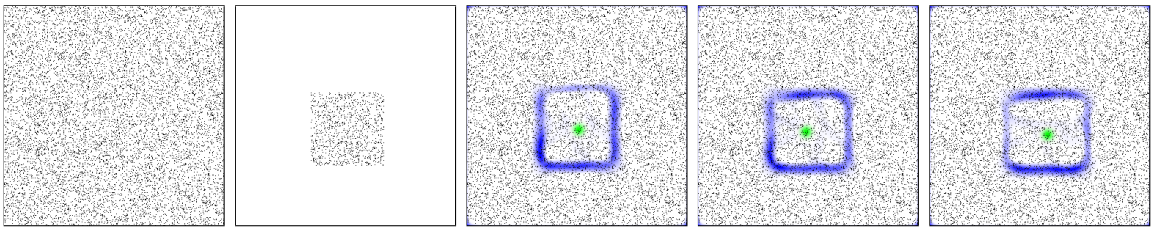

Figure 4.1: **Boundary detection in the absence of static cues.** Consider a textured background (left) and an identically textured foreground object (middle-left) moving in front of it. Three frames of this motion sequence are shown (middle to right). Humans easily detect the moving square from the video sequence despite the uniform appearance of any single frame. Our motion gradient cue (shown in blue) successfully detects the object boundary. Ground-truth locations of the object center are marked in green.

produce motion gradients $MG^-(x, y, \theta)$ and $MG^+(x, y, \theta)$, respectively.

Gradients MG^- and MG^+ both contain double responses to moving boundaries as there is a large temporal difference between any two consecutive frames at both the old and new locations of a boundary. However, these double responses occur at different spatial locations in MG^- and MG^+. Each moving edge is detected at three spatial locations, two of which appear in MG^- and two of which appear in MG^+. The detection common to both MG^- and MG^+ is the true location of the edge at time t. By taking the geometric mean of these signals we suppress the spurious responses while preserving the correct one, resulting in the motion gradient:

$$MG(x, y, \theta) = \sqrt{MG^-(x, y, \theta) \cdot MG^+(x, y, \theta)} \tag{4.3}$$

Figure 4.2 shows an example of the motion gradient applied to a real video sequence. Note that it is not our intention to detect occlusion boundaries at this stage. Rather, the motion gradient is designed to respond strongly at any edge which moves relative to the camera. For this reason, MG picks up the surface markings on the tennis court. At the same time, it serves to enhance the actual occlusion boundaries

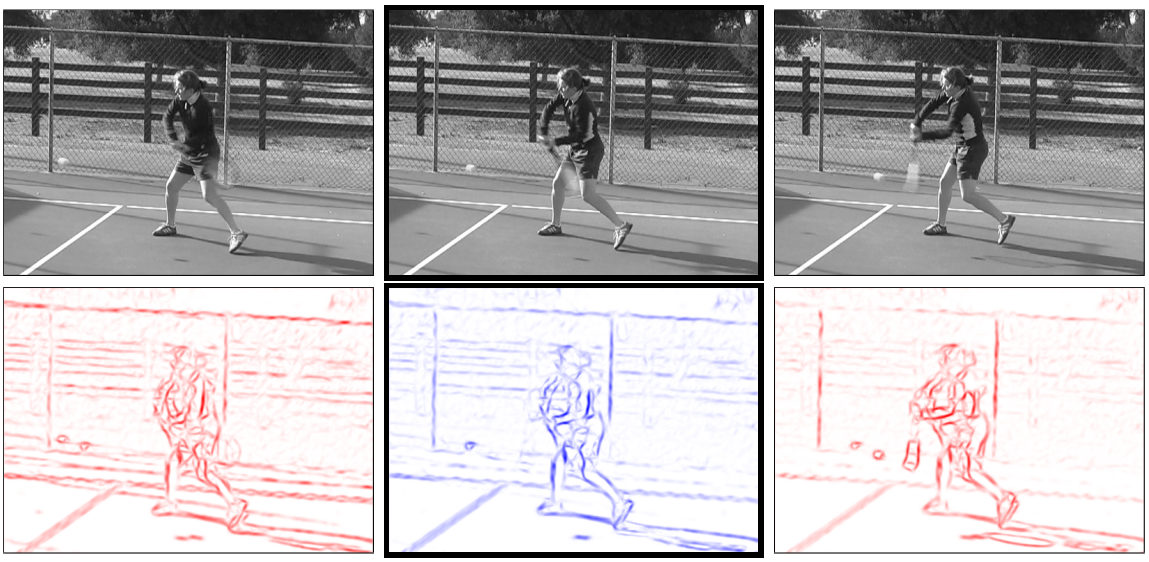

Figure 4.2: **Motion gradient. Top:** Three consecutive frames from video of a tennis player. The large frame to frame displacement of the tennis ball, tennis racket, and person's arms and legs make this a challenging sequence. **Bottom Left and Right (red):** Gradient operator applied to the temporal differences between the center frame and those immediately before and after. We show $MG^{-}(x,y) = \max_{\theta}\{MG^{-}(x,y,\theta)\}$ and $MG^{+}(x,y) = \max_{\theta}\{MG^{+}(x,y,\theta)\}$, respectively. Both MG^{-} and MG^{+} contain double images of the moving boundaries. **Bottom Center (blue):** A motion boundary in the current frame should be detected as differing from both the previous frame and subsequent frame. We compute MG using the geometric mean of MG^{-} and MG^{+} as a soft "and" operation and display $MG(x,y) = \max_{\theta}\{MG(x,y,\theta)\}$. Double boundaries are eliminated and those surviving in MG are correctly aligned with the current frame.

on the person and provides robustness against cases in which static cues are weak or nonexistent (Figure 4.1). The main concern is to utilize the motion gradient to improve the quality and reliability of the boundary map upon which our occlusion reasoning machinery will depend.

Following the framework of Chapter 2, we add the motion gradient as an additional channel alongside the static brightness, color, and texture cues, and pass these local cues through the same sequence of steps for combining them with the result of a spectral partitioning process. We use gPb+mg to refer to contours created using

both the static cues and motion gradient. We then utilize the OWT-UCM machinery of Chapter 3 to transform the output of the gPb+mg boundary detector into an Ultrametric Contour Map (UCM) defining a hierarchical segmentation. This data structure determines the edge fragments and associated regions that we classify in the occlusion reasoning and figure/ground stages.

4.3 Occlusion Boundary Detection

In contrast to other recent work [Stein and Hebert, 2009], our occlusion reasoning stage utilizes a straightforward classification technique whose power lies in its ability to exploit two key components: the boundary detector discussed in the previous section, and the variational optical flow method of [Brox *et al.*, 2004]. Given both reliable prior boundaries gPb+mg and a reliable, dense optical flow field $(u, v)^\top := \mathbf{w} : (\Omega \subset \mathbb{R}^2) \rightarrow \mathbb{R}^2$, we are able to judge whether an edge is an occlusion boundary by looking at the difference between the flows in the regions on either side of that edge.

We begin by estimating the neighboring region motions at each point on every edge fragment from $\mathbf{w}$. We need to ensure that the estimate not be polluted by the motion of the edge itself, or by the motion of the opposing region. In order to do this, we define a weighted filter $w_i(x, y)$ for each edge point $\mathbf{p}_i = (x_i, y_i)$ and region R neighboring $\mathbf{p}_i$ as follows:

$$w_i^R(x, y) = \frac{1}{Z} \exp\left(-\frac{(x - x_i)^2 + (y - y_i)^2}{2\sigma_w^2}\right) \delta(r(x, y), R) i_d(x, y) \tag{4.4}$$

where δ is the Kronecker delta function, $r(x, y)$ is the region assignment for pixel location (x, y), and $i_d(x, y)$ is 1 for pixels internal to any region, defined as being at least d pixels away from any region boundary, and 0 otherwise. In this work, we use

$d = 2$. While it is reasonable to vary σ_w, combining windowing functions of different scales, in our experiments we did not observe a significant change when doing so. This is most likely due to the optical flow method from [Brox *et al.*, 2004] already considering multiple scales. Experiments reported here all use $\sigma_w = 4$.

Given the weights w_i, we use weighted least squares to fit a plane to the u and v flow components as functions of x and y, minimizing:

$$e_u^R = \sum_{(x,y)} w_i^R(x,y)(A_u^R x + B_u^R y + C_u^R - u(x,y))^2 \tag{4.5}$$

$$e_v^R = \sum_{(x,y)} w_i^R(x,y)(A_v^R x + B_v^R y + C_v^R - v(x,y))^2 \tag{4.6}$$

and finally describe the flow of a region R by this affine model:

$$(u_i^R, v_i^R) = (A_u^R x_i + B_u^R y_i + C_u^R, A_v^R x_i + B_v^R y_i + C_v^R) \tag{4.7}$$

This step is critical, since we do not measure the region flow at any points closer than d pixels away from the boundary, and many common camera and object motions lead to flow vectors which vary spatially. In addition, the resulting error terms e_u^R and e_v^R serve as measures of the certainty of the flow estimate.

For each edge point indexed by i, we compute the two flow vectors $\mathbf{w}_i^{r_1} := (u_i^{r_1}, v_i^{r_1})$ and $\mathbf{w}_i^{r_2} := (u_i^{r_2}, v_i^{r_2})$ on the edge point from the affine coefficients from above. Given these vectors and their associated errors, we can test the hypothesis that the vectors are the same. Here we assume that the flow vectors are drawn from a Gaussian distribution, and use a t-test to compare them. The test statistic is:

$$t^2 = \frac{1}{2}\mathbf{d}^\top \mathbf{S}^{-1}\mathbf{d} \tag{4.8}$$

where $\mathbf{d} = \mathbf{w}_i^{r_1} - \mathbf{w}_i^{r_2}$ is the difference between the flow vectors and $\mathbf{S}$ is the pooled

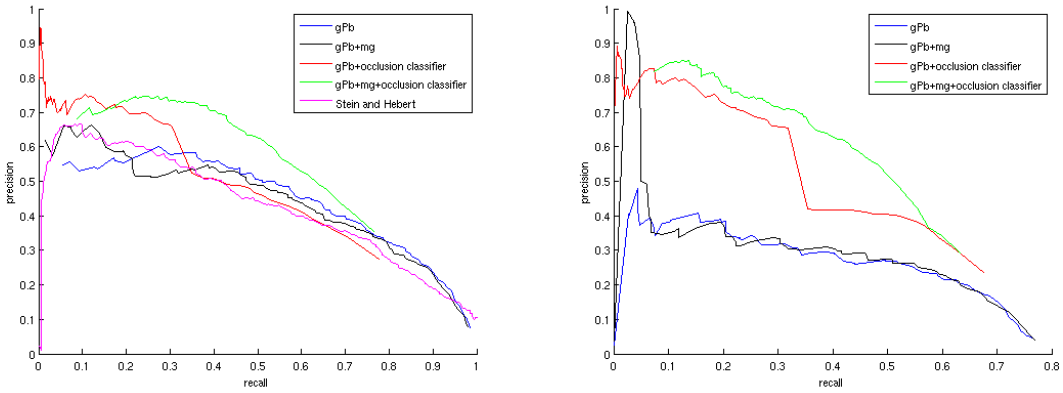

Figure 4.3: **Occlusion boundary detection benchmark. Left:** Precision-recall curves for the occlusion boundary detection task reported by [Stein and Hebert, 2009]. On the same dataset, our occlusion boundary detection algorithm significantly outperforms the state-of-the-art results reported by Stein and Hebert. It is important to note here that while we include the curve reported by [Stein and Hebert, 2009], neither the exact benchmarking code nor the output of Stein and Hebert's algorithm was publicly available. To obtain their curve, they map ground-truth labels to edge fragments in an oversegmentation and plot the classification accuracy on the fragments. We present results using the BSDS [Martin *et al.*, 2004] benchmark code, which requires pixel-to-pixel correspondence. We believe it is fair to say that our benchmarking technique is stricter than that of [Stein and Hebert, 2009] and that the performance gap is at least as large as illustrated here. **Right:** Results on our own dataset. Both our motion gradient cue mg and occlusion classifier contribute to improving performance.

covariance matrix, in which we have included the error terms from above:

$$\mathbf{S} = \sum_{k=1}^{2} (\mathbf{w}^{r_k} - \bar{\mathbf{w}})(\mathbf{w}^{r_k} - \bar{\mathbf{w}})^\top + \begin{pmatrix} e_u^{r_1} + e_u^{r_2} & 0 \\ 0 & e_v^{r_1} + e_v^{r_2} \end{pmatrix} \tag{4.9}$$

where $\bar{\mathbf{w}}$ denotes the average of the two vectors $\mathbf{w}^{r_1}$ and $\mathbf{w}^{r_2}$.

Given t^2, we can compute the probability $p(\text{motion}) = Pr(x \geq t^2)$ by looking up the cdf of the T^2 distribution. This value measures the probability that we would have observed values at least as different as the ones we did, and we use this one-sided

test to identify boundaries which we have strong reason to believe are moving. Note that a low value of $p(\text{motion})$ does not mean that the edge is not moving, it simply means we have no evidence that it is moving. Finally, we use the strength of the edge as measured by the boundary detector (either gPb or gPb+mg) as a prior p_0, resulting in the following per-pixel measurement of whether this edge is a moving occlusion edge:

$$p(\text{occlusion}) = \frac{p(\text{motion})p_0}{p(\text{motion})p_0 + (1 - p(\text{motion}))(1 - p_0)} \tag{4.10}$$

where the denominator is a normalization term to account for the fact that the t-test produces a one-sided probability.

It should be noted that this method not only suppresses internal edges, but non-moving true occlusion edges as well. Despite this, as Figure 4.3 shows, it clearly outperforms both the original boundary detector and [Stein and Hebert, 2009] on their occlusion benchmark. It is also worth noting that while [Stein and Hebert, 2009] used video sequences of a dozen frames or more, we get our result using only two frames (three frames if using gPb+mg). Figure 4.4 displays the results of our occlusion classifier on some frames from the dataset of [Stein and Hebert, 2009] as well as our own dataset.

If desired, we can use these occlusion edges to improve the optical flow field by reducing smoothing across these edges. As Figure 4.5 shows, this increases the performance of the variational optical flow method on the standard Middlebury benchmark [Baker *et al.*, 2007]. In contrast, reducing smoothing across all gPb edges, including internal edges, is actually inferior to the original method from [Brox *et al.*, 2004].

Figure 4.4: **Occlusion boundary detection results. From left to right:** Two consecutive video frames, Ultrametric Contour Map for the first frame created from the gPb+mg contour detector, and boundaries reweighted according to our occlusion classifier.

Optical flow evaluation results **Statistics:** Average SD R2.5 R5.0 R10.0 A50 A75 A95
Error type: angle end-point interpolation normalized interpolation

Average angle error	avg. rank	Army (Hidden texture) GT all	Army im0 all	Army im1 all	Army im1 disc	Army im1 untext	Mequon (Hidden texture) GT all	Mequon im0 all	Mequon im1 all	Mequon im1 disc	Mequon im1 untext	Schefflera (Hidden texture) GT all	Schefflera im0 all	Schefflera im1 all	Schefflera im1 disc	Schefflera im1 untext	Wooden (Hidden texture) GT all	Wooden im0 all	Wooden im1 all	Wooden im1 disc	Wooden im1 untext	Grove (Synthetic) GT all	Grove im0 all	Grove im1 all	Grove im1 disc	Grove im1 untext	Urban (Synthetic) GT all	Urban im0 all	Urban im1 all	Urban im1 disc	Urban im1 untext	Yosemite (Synthetic) GT all	Yosemite im0 all	Yosemite im1 all	Yosemite im1 disc	Yosemite im1 untext	Teddy (Stereo) GT all	Teddy im0 all	Teddy im1 all	Teddy im1 disc	Teddy im1 untext
ComplOF [27]	5.2	4.44 10	11.2 8	4.04 10	2.51 2	9.77 3	**1.74** 1	3.93 4	10.6 4	**2.04** 1	3.87 8	18.8 5	2.19 7	**3.17** 1	**4.00** 1	2.92 3	4.64 9	13.8 3	3.64 7	2.17 6	3.36 3	2.51 15	3.08 2	7.04 2	3.65 9
Adaptive [26]	5.2	**3.29** 1	**9.43** 1	**2.28** 1	3.10 6	11.4 7	2.46 7	6.58 12	15.7 12	2.52 5	**3.14** 1	**15.6** 1	**1.56** 1	3.67 7	4.46 6	3.48 7	**3.32** 1	13.0 2	**2.38** 1	2.76 13	4.39 13	1.93 10	3.58 4	8.18 4	2.88 2
Aniso. Huber-L1 [28]	7.1	3.71 3	10.1 3	3.08 3	4.36 16	13.0 10	3.77 15	6.92 15	15.3 10	3.60 15	3.54 2	15.9 2	2.04 5	3.38 3	4.45 5	2.47 2	3.88 3	**12.9** 1	2.74 2	3.37 18	4.36 12	2.85 17	3.16 3	7.52 3	2.90 3
Spatially variant [22]	7.6	3.73 4	10.2 6	3.33 5	3.02 5	11.0 6	2.67 8	5.36 8	13.8 9	2.35 2	3.67 3	19.3 7	1.84 3	3.81 10	4.81 17	3.69 10	4.48 8	16.0 9	3.90 8	2.11 4	3.26 2	2.12 12	4.66 12	9.41 11	4.35 14
TV-L1-improved [20]	8.6	3.36 2	9.63 2	2.62 2	2.82 4	10.7 5	2.23 3	6.50 11	15.8 13	2.73 6	3.80 6	21.3 14	1.76 2	3.34 2	4.38 4	**2.39** 1	5.97 11	18.1 17	5.67 18	3.57 19	4.92 20	3.43 21	4.01 8	9.84 12	3.44 7
occedge [31]	8.9	4.42 8	12.4 11	3.90 8	3.86 11	13.2 11	3.32 11	5.00 7	13.0 8	3.30 11	4.45 11	20.7 11	2.37 8	3.84 11	4.67 12	4.39 22	3.75 2	15.9 8	3.30 4	2.19 7	4.00 11	1.17 2	4.33 9	9.20 7	3.19 5
Multicue MRF [24]	9.7	4.50 12	10.1 3	4.18 14	2.52 3	**7.07** 1	2.36 6	**3.09** 1	**7.41** 1	2.36 3	4.46 12	20.8 12	2.73 13	3.51 5	4.11 2	4.06 15	6.08 13	15.6 7	5.40 13	5.25 26	5.36 22	9.02 26	3.63 5	8.39 5	4.15 12
baseline [29]	9.9	4.44 10	12.4 11	4.22 15	3.72 10	13.5 12	3.06 9	4.97 6	13.3 7	3.11 8	4.58 14	22.0 16	2.37 8	3.79 9	4.60 9	4.33 21	3.91 4	17.0 13	3.45 6	2.22 3	3.79 7	1.19 4	4.62 11	10.0 13	3.38 6
F-TV-L1 [18]	11.1	5.44 16	12.5 14	5.69 20	5.46 17	15.0 17	4.03 17	7.48 17	16.3 15	3.42 14	5.08 16	23.3 19	2.81 14	3.42 4	4.34 3	3.03 4	4.05 6	15.1 5	3.18 3	2.43 10	3.92 9	1.87 9	3.90 7	9.35 10	**2.61** 1
pb [30]	11.5	4.57 13	12.6 15	4.26 16	4.12 14	13.8 14	3.47 13	4.87 5	12.5 6	3.31 12	4.51 13	20.8 12	2.45 10	3.91 15	4.79 16	4.29 19	4.22 7	16.1 10	3.96 9	2.95 15	4.55 16	1.46 6	4.38 10	9.29 9	3.12 4
DPOF [21]	12.2	5.63 17	10.9 7	4.16 13	4.05 12	12.1 8	3.31 10	3.87 3	8.82 2	3.17 10	4.34 9	16.2 3	3.13 16	3.95 16	4.78 15	4.17 17	6.69 19	15.2 6	6.27 18	5.62 27	6.89 28	6.60 25	**2.44** 1	**4.83** 1	3.74 11
Fusion [9]	13.2	4.43 9	13.7 17	4.08 11	**2.47** 1	8.91 2	2.24 4	3.70 2	9.68 3	3.12 9	3.68 4	19.8 8	2.54 12	4.26 19	5.16 18	4.31 20	6.32 15	16.8 12	6.15 17	4.55 23	5.78 24	3.10 20	7.12 22	13.6 22	7.86 23

Figure 4.5: **Middlebury flow benchmark results.** Reducing spatial integration at the detected occlusion boundaries improves the accuracy of the optical flow field (occedge), whereas an edge detector that does not distinguish between occlusion edges and internal edges (pb) yields inferior results than the baseline method from [Brox *et al.*, 2004].

4.4 Figure/Ground Labeling

Once we know that a moving edge is an occlusion edge, we can determine which side of the edge is the occluding foreground object and which side belongs to the background, thereby obtaining a depth ordering of the attached regions. There have been previous attempts on assigning figure/ground labels to both sides of an edge based on static cues [Ren *et al.*, 2006; Hoiem *et al.*, 2007], which is a difficult task with many ambiguities. The task becomes trivial in a stereo setting with estimated disparities. In a monocular setting, motion cues provide rich information to decide upon figure/ground. Obviously, an occluding edge moves the same way as the occluding region next to it. Thus, a decision can be inferred from the motion of the two regions as well as the motion of the occluding edge.

In the previous section, we already discussed how the motion of a region can be obtained from an optical flow field without too much interference from edge motion. Thus, all we need here for the figure/ground assignment is the motion directly on the edge. [Smith *et al.*, 2004] and [Liu *et al.*, 2006] both show how to compute the

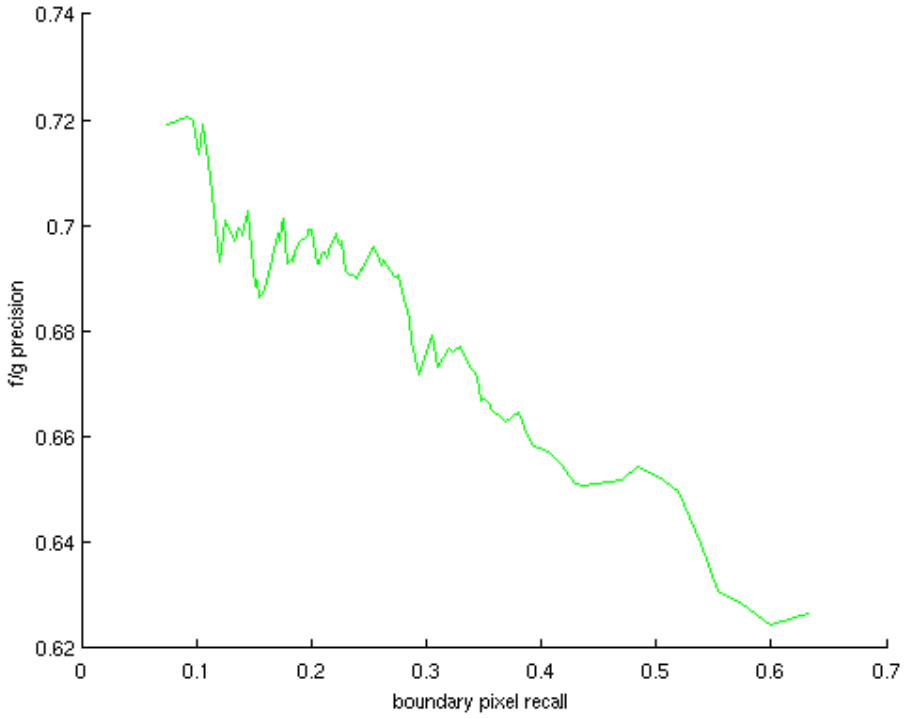

Figure 4.6: **Figure/ground classification performance.** We show figure/ground performance as a function of occlusion edge recall. In our framework, occlusion edge detection and figure/ground classification both depend on motion. The edges which are most easily detected are also the ones where figure/ground is most easily assigned. As a result, classification performance decreases along with occlusion edge detection confidence.

motion of an edge. In particular [Liu *et al.*, 2006] takes some effort to estimate the edge motion from edge cues only. Moreover, they deal with special cases such as illusory boundaries. Here, we are interested in a methodology that is most consistent with the way we estimated the motion of the regions, in order to compare them. Thus, it is natural to simply consider the optical flow on the edge in order to obtain the edge motion (u_i^e, v_i^e) at each edge pixel i. Actually, this makes sense because the flow estimates obtained by the variational technique are most reliable on edges.

For each edge fragment we compute two distances, each between the motion of

Figure 4.7: **Figure/ground classification results.** We show figure/ground classification results for detected occlusion boundaries with strength above 0.2 for a subset of frames from Figure 4.4. Green and red markings indicate the figure and ground sides of each boundary, respectively.

the edge and the respective region:

$$d^2(e, r_k) = \frac{1}{n} \sum_{i=1}^{n} \left[(u_i^e - u_i^{r_k})^2 + (v_i^e - v_i^{r_k})^2 \right] \tag{4.11}$$

where n is the number of edge pixels on this edge fragment. Finally, we assign the edge to the region with the smaller distance.

As there is no dataset yet for a quantitative evaluation of this task, we created

one consisting of 20 images with labeled occlusion boundaries as well as non-occlusion boundaries, and figure/ground assignments associated to occlusion edges. Figure 4.6 shows a plot of the performance on this dataset. The results are far above chance level, which is at 0.5. Some sample figure/ground assignments are shown in Figure 4.7.

Occlusion boundaries and figure/ground knowledge may prove to be a promising way to extract objects from video in an unsupervised fashion. Fully unsupervised segmentation from static images is difficult, yet motion is a reliable cue for this task. Further development of an automated video segmentation engine could provide a boost to object recognition, enabling the discovery and learning of object categories from vast archives of video data.

Bibliography

[Ahuja and Todorovic, 2008] Narendra Ahuja and Sinisa Todorovic. Connected segmentation tree: A joint representation of region layout and hierarchy. *CVPR*, 2008.

[Amiaz and Kiryati, 2006] Tomer Amiaz and Nahum Kiryati. Piecewise-smooth dense optical flow via level sets. *IJCV*, 2006.

[Arbeláez and Cohen, 2008] Pablo Arbeláez and Laurent Cohen. Constrained image segmentation from hierarchical boundaries. *CVPR*, 2008.

[Arbeláez *et al.*, 2009] Pablo Arbeláez, Michael Maire, Charless Fowlkes, and Jitendra Malik. From contours to regions: An empirical evaluation. *CVPR*, 2009.

[Arbeláez, 2006] Pablo Arbeláez. Boundary extraction in natural images using ultrametric contour maps. *POCV*, 2006.

[Bagon *et al.*, 2008] Shai Bagon, Oren Boiman, and Michal Irani. What is a good image segment? A unified approach to segment extraction. *ECCV*, 2008.

[Baker *et al.*, 2007] Simon Baker, Daniel Scharstein, J.P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 2007.

[Belongie and Malik, 1998] Serge Belongie and Jitendra Malik. Finding boundaries in natural images: A new method using point descriptors and area completion. *ECCV*, 1998.

[Belongie *et al.*, 1998] Serge Belongie, Chad Carson, Hayit Greenspan, and Jitendra Malik. Color and texture-based image segmentation using EM and its application to content-based image retrieval. *ICCV*, pages 675–682, 1998.

[Belongie *et al.*, 2002] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 2002.

[Berg and Malik, 2001] Alexander C. Berg and Jitendra Malik. Geometric blur for template matching. *CVPR*, 2001.

[Beucher and Meyer, 1992] Serge Beucher and Fernand Meyer. *Mathematical Morphology in Image Processing*, chapter 12. Marcel Dekker, 1992.

[Biederman, 1987] Irving Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.

[Black and Fleet, 2000] Michael J. Black and David J. Fleet. Probabilistic detection and tracking of motion boundaries. *IJCV*, 2000.

[Boykov and Jolly, 2001] Yuri Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. *ICCV*, 2001.

[Brox *et al.*, 2004] Thomas Brox, Andres Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. *ECCV*, 2004.

[Brox *et al.*, 2006] Thomas Brox, Andres Bruhn, and Joachim Weickert. Variational motion segmentation with level sets. *ECCV*, 2006.

[Brox *et al.*, 2009] Thomas Brox, Christoph Bregler, and Jitendra Malik. Large displacement optical flow. *CVPR*, 2009.

[Canny, 1986] John Canny. A computational approach to edge detection. *PAMI*, 1986.

[Catanzaro *et al.*, 2009] Bryan Catanzaro, Bor-Yiing Su, Narayanan Sundaram, Yunsup Lee, Mark Murphy, and Kurt Keutzer. Efficient, high-quality image contour detection. *ICCV*, 2009.

[Chung, 1997] Fan R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[Comaniciu and Meer, 2002] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 2002.

[Cour *et al.*, 2005] Timothee Cour, Florence Benezit, and Jianbo Shi. Spectral segmentation with multiscale graph decomposition. *CVPR*, 2005.

[Cremers and Soatto, 2005] Daniel Cremers and Stefano Soatto. Motion competition: A variational framework for piecewise parametric motion segmentation. *IJCV*, 2005.

[Dalal and Triggs, 2005] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. *CVPR*, 2005.

[Darrell and Pentland, 1991] Trevor Darrell and Alex Pentland. Robust estimation of a multi-layer motion representation. *IEEE Workshop on Visual Motion*, 1991.

[Dollar *et al.*, 2006] Piotr Dollar, Zhuowen Tu, and Serge Belongie. Supervised learning of edges and object boundaries. *CVPR*, 2006.

[Elder and Zucker, 1996] James Elder and Steven Zucker. Computing contour closures. *ECCV*, 1996.

[Everingham *et al.*, 2008] Mark Everingham, Luc van Gool, Chris Williams, John Winn, and Andrew Zisserman. PASCAL 2008 Results. http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html, 2008.

[Felzenszwalb and Huttenlocher, 2004] Pedro Felzenszwalb and Daniel Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 2004.

[Felzenszwalb and McAllester, 2006] Pedro Felzenszwalb and David McAllester. A min-cover approach for finding salient curves. *POCV*, 2006.

[Felzenszwalb *et al.*, 2008] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. *CVPR*, 2008.

[Fowlkes and Malik, 2004] Charless Fowlkes and Jitendra Malik. How much does globalization help segmentation? Technical Report CSD-04-1340, UC Berkeley, 2004.

[Fowlkes *et al.*, 2003] Charless Fowlkes, David Martin, and Jitendra Malik. Learning affinity functions for image segmentation: Combining patch-based and gradient-based approaches. *CVPR*, 2003.

[Freeman and Adelson, 1991] William T. Freeman and Edward H. Adelson. The design and use of steerable filters. *PAMI*, 1991.

[Gaucher and Medioni, 1999] Lionel Gaucher and Gérard Medioni. Accurate motion flow estimation with discontinuities. *ICCV*, 1999.

[Gu *et al.*, 2009] Chunhui Gu, Joseph Lim, Pablo Arbeláez, and Jitendra Malik. Recognition using regions. *CVPR*, 2009.

[Hoiem *et al.*, 2005] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Geometric context from a single image. *ICCV*, 2005.

[Hoiem *et al.*, 2007] Derek Hoiem, Andrew N. Stein, Alexei A. Efros, and Martial Hebert. Recovering occlusion boundaries from a single image. *ICCV*, 2007.

[Koffka, 1935] Kurt Koffka. *Principles of Gestalt Psychology.* Hartcourt Brace Jovanovich, 1935.

[Lampert *et al.*, 2008] Christoph H. Lampert, Matthew B. Blaschko, and Thomas Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. *CVPR*, 2008.

[Leung and Malik, 1998] Thomas Leung and Jitendra Malik. Contour continuity in region-based image segmentation. *ECCV*, 1998.

[Li *et al.*, 2004] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. *SIGGRAPH*, 2004.

[Lindeberg, 1998] Tony Lindeberg. Edge detection and ridge detection with automatic scale selection. *IJCV*, 1998.

[Liu *et al.*, 2006] Ce Liu, William T. Freeman, and Edward H. Adelson. Analysis of contour motions. *NIPS*, 2006.

[Lowe, 1999] David G. Lowe. Object recognition from local scale-invariant features. *ICCV*, 1999.

[Mahamud *et al.*, 2003] Shyjan Mahamud, Lance R. Williams, Karvel K. Thornber, and Kanglin Xu. Segmentation of multiple salient closed contours from real images. *PAMI*, 2003.

[Mairal *et al.*, 2008] Julien Mairal, Marius Leordeanu, Francis Bach, Martial Hebert, and Jean Ponce. Discriminative sparse image models for class-specific edge detection and image interpretation. *ECCV*, 2008.

[Maire *et al.*, 2008] Michael Maire, Pablo Arbeláez, Charless Fowlkes, and Jitendra Malik. Using contours to detect and localize junctions in natural images. *CVPR*, 2008.

[Malik *et al.*, 2001] Jitendra Malik, Serge Belongie, Thomas Leung, and Jianbo Shi. Contour and texture analysis for image segmentation. *IJCV*, 2001.

[Malisiewicz and Efros, 2007] Tomasz Malisiewicz and Alexei A. Efros. Improving spatial support for objects via multiple segmentations. *BMVC*, 2007.

[Malladi *et al.*, 1995] Ravikanth Malladi, James A. Sethian, and Baba C. Vemuri. Shape modeling with front propagation: A level set approach. *PAMI*, 1995.

[Martin *et al.*, 2001] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *ICCV*, 2001.

[Martin *et al.*, 2004] David Martin, Charless Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color and texture cues. *PAMI*, 2004.

[Meila, 2005] Marina Meila. Comparing clusterings: An axiomatic view. *ICML*, 2005.

[Morel and Solimini, 1995] Jean M. Morel and Sergio Solimini. *Variational Methods in Image Segmentation*. Birkhäuser, 1995.

[Morrone and Owens, 1987] M. C. Morrone and R. Owens. Feature detection from local energy. *Pattern Recognition Letters*, 1987.

[Mumford and Shah, 1989] David Mumford and Jayant Shah. Optimal approximations by piecewise smooth functions, and associated variational problems. *Communications on Pure and Applied Mathematics*, pages 577–684, 1989.

[Najman and Schmitt, 1996] Laurent Najman and Michel Schmitt. Geodesic saliency of watershed contours and hierarchical segmentation. *PAMI*, 1996.

[Parent and Zucker, 1989] Pierre Parent and Steven W. Zucker. Trace inference, curvature consistency, and curve detection. *PAMI*, 1989.

[Perona and Malik, 1990] Pietro Perona and Jitendra Malik. Detecting and localizing edges composed of steps, peaks and roofs. *ICCV*, 1990.

[Rabinovich *et al.*, 2007] Andrew Rabinovich, Andrea Vedaldi, Carolina Galleguillos, Eric Wiewiora, and Serge Belongie. Objects in context. *ICCV*, 2007.

[Rand, 1971] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846–850, 1971.

[Ren and Malik, 2003] Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. *ICCV*, 2003.

[Ren *et al.*, 2005] Xiaofeng Ren, Charless Fowlkes, and Jitendra Malik. Scale-invariant contour completion using conditional random fields. *ICCV*, 2005.

[Ren *et al.*, 2006] Xiaofeng Ren, Charless Fowlkes, and Jitendra Malik. Figure/ground assignment in natural images. *ECCV*, 2006.

[Ren *et al.*, 2008] Xiaofeng Ren, Charless Fowlkes, and Jitendra Malik. Learning probabilistic models for contour completion in natural images. *IJCV*, 2008.

[Ren, 2008] Xiaofeng Ren. Multi-scale improves boundary detection in natural images. *ECCV*, 2008.

[Rother *et al.*, 2004] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "Grabcut": Interactive foreground extraction using iterated graph cuts. *SIGGRAPH*, 2004.

[Savitzky and Golay, 1964] Abraham Savitzky and Marcel J. E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 1964.

[Saxena *et al.*, 2008] Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng. 3-d depth reconstruction from a single still image. *IJCV*, 2008.

[Sethian, 1999] James A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[Shi and Malik, 1998] Jianbo Shi and Jitendra Malik. Motion segmentation and tracking using normalized cuts. *ICCV*, 1998.

[Shi and Malik, 2000] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *PAMI*, 2000.

[Shotton *et al.*, 2006] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. *ECCV*, 2006.

[Smith *et al.*, 2004] Paul Smith, Tom Drummond, and Roberto Cipolla. Layered motion segmentation and depth ordering by tracking edges. *PAMI*, 2004.

[Stein and Hebert, 2009] Andrew N. Stein and Martial Hebert. Occlusion boundaries from motion: Low-level detection and mid-level reasoning. *IJCV*, 2009.

[Tolliver and Miller, 2006] David Tolliver and Gary L. Miller. Graph partitioning by spectral rounding: Applications in image segmentation and clustering. *CVPR*, 2006.

[Tu, 2005] Zhuowen Tu. Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering. *ICCV*, 2005.

[Unnikrishnan *et al.*, 2007] Ranjith Unnikrishnan, Caroline Pantofaru, and Martial Hebert. Toward objective evaluation of image segmentation algorithms. *PAMI*, 2007.

[Viola and Jones, 2004] Paul Viola and Michael J. Jones. Robust real-time face detection. *IJCV*, 2004.

[Wang and Adelson, 1994] John Y. A. Wang and Edward H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 1994.

[Wang *et al.*, 2005] Song Wang, Toshiro Kubota, Jeffrey M. Siskind, and Jun Wang. Salient closed boundary extraction with ratio contour. *PAMI*, 2005.

[Weiss, 1997] Yair Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. *ICCV*, 1997.

[Weiss, 2000] Yair Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 2000.

[Williams and Jacobs, 1995] Lance R. Williams and David W. Jacobs. Stochastic completion fields: A neural model of illusory contour shape and salience. *ICCV*, 1995.

[Xiao and Shah, 2005] Jiangjian Xiao and Mubarak Shah. Motion layer extraction in the presence of occlusion using graph cuts. *PAMI*, 2005.

[Yang *et al.*, 2008] Allen Y. Yang, John Wright, Yi Ma, and S. Shankar Sastry. Unsupervised segmentation of natural images via lossy data compression. *CVIU*, 2008.

[Yu, 2005] Stella X. Yu. Segmentation induced by scale invariance. *CVPR*, 2005.

[Zhu *et al.*, 2007] Qihui Zhu, Gang Song, and Jianbo Shi. Untangling cycles for contour grouping. *ICCV*, 2007.