GENERATION AND ANALYSIS OF SEGMENTATION TREES FOR NATURAL IMAGES

BY

EMRE AKBAŞ

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Doctoral Committee:

Professor Narendra Ahuja, Chair
Professor Thomas S. Huang
Professor David A. Forsyth
Associate Professor Mark A. Hasegawa-Johnson
Assistant Professor Derek W. Hoiem

ABSTRACT

This dissertation is about extracting as well as making use of the structure and hierarchy present in images. We develop a new low-level, multiscale, hierarchical image segmentation algorithm designed to detect image regions regardless of their shapes, sizes, and levels of interior homogeneity. We model a region as a connected set of pixels that is surrounded by ramp edge discontinuities where the magnitude of these discontinuities is large compared to the variation inside the region. Each region is associated with a scale depending on the magnitude of the weakest part of its boundary. Traversing through the range of all possible scales, we obtain all regions present in the image. Regions strictly merge as the scale increases; hence a tree is formed where the root node corresponds to the whole image, and nodes close to the root along a path are large, while their children nodes are smaller and capture embedded details.

To evaluate the accuracy and precision of our algorithm, as well as to compare it to the existing algorithms, we develop a new benchmark dataset for low-level image segmentation. In this benchmark, small patches of many images are hand-segmented by human subjects. We provide evaluation methods for both boundary-based and region-based performance of algorithms. We show that our proposed algorithm performs better than the existing low-level segmentation algorithms on this benchmark.

Next, we investigate the segmentation-based statistics of natural images. Such statistics capture geometric and topological properties of images, which is not possible to obtain using pixel-, patch-, or subband-based methods. We compile and use segmentation statistics from a large number of images, and propose a Markov random field based model for estimating them. Our estimates confirm some of the previous statistical properties of natural images as well as yield new ones. To demonstrate the value of the statistics, we

successfully use them as priors in image classification and semantic image segmentation.

We also investigate the importance of different visual cues to describe image regions for solving the region correspondence problem. We design and develop psychophysical experiments to learn the weights of different cues by evaluating their impact on binocular fusibility by human subjects. Using a head-mounted display, we show a set of elliptical regions to one eye and slightly different versions of the same set of regions to the other eye of human subjects. We then ask them whether the ellipses fuse or not. By systematically varying the parameters of the elliptical shapes, and testing for fusion, we learn a perceptual distance function between two elliptical regions. We evaluate this function on ground-truth stereo image pairs.

Finally, we propose a novel multiple instance learning (MIL) method. In MIL, in contrast to classical supervised learning, the entities to be classified are called bags, each of which contains an arbitrary number of elements called instances. We propose an additive model for bag classification where we exploit the idea of searching for discriminative instances, which we call prototypes. We show that our bag-classifier can be learned in a boosting framework, leading to an iterative algorithm, which learns prototype-based weak learners that are linearly combined. At each iteration of our proposed method, we search for a new prototype so as to maximally discriminate between the positive and negative bags, which are themselves weighted according to how well they were discriminated in earlier iterations. Unlike previous instance selection based MIL methods, we do not restrict the prototypes to a discrete set of training instances but allow them to take arbitrary values in the instance feature space. We also do not restrict the total number of prototypes and the number of selected-instances per bag; these quantities are completely data-driven. We show that our method outperforms state-of-the-art MIL methods on a number of benchmark datasets. We also apply our method to large-scale image classification, where we show that the automatically selected prototypes map to visually meaningful image regions.

To my late father, and to those who taught me

ACKNOWLEDGMENTS

This dissertation would not have been possible without the support and courage of my mother and sister. I thank them for allowing me to follow my ambitions, even if it meant leaving them behind in another continent. I love them very much.

I am fortunate to have my dear girlfriend and future wife, Eren, with whom I joyfully shared both the happy and tough times during my Ph.D. study. Her continuous love and support have been invaluable for this dissertation.

I would like to express my sincere gratitude to my academic and research adviser Professor Narendra Ahuja. I thank him for all the valuable guidance and support he provided throughout my graduate education. I feel indebted to him not only for the advice he gave me but also his trust and patience which made this dissertation possible.

My committee members have helped me improve this dissertation by making many insightful comments and suggestions. I would like to thank professors David Forsyth, Mark Hasegawa-Johnson, Derek Hoiem, and Thomas Huang for their time and patience.

I have learned a lot from the countless discussions I had with my friends at the Computer Vision and Robotics Lab and the Beckman Institute. In particular, I want to thank Bernard Ghanem, Sanketh Shetty, Mert Dikmen, Xianbiao Shu, Hsien Ting Cheng, Myra Nam, and Esther Resendiz.

I would like to acknowledge Professor Ranxiao Frances Wang for allowing me to use her laboratory and the head-mounted display device for the binocular fusion experiments. I also want to thank Carmen Young for sparing her valuable time to run the experiments for me.

Last, but not least, my sincere thanks go to the hard-working staff at the

ECE Publications Office, who patiently proofread and corrected many drafts of this dissertation.

Finally, the financial support of the National Science Foundation under grant IIS 08-12188 and the Office of Naval Research under grant N00014-09-1-0017 is gratefully acknowledged.

April 2011

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In 2009, Pawan Sinha and his colleagues published a very interesting experiment about visual parsing in humans [1]. They found the opportunity to investigate the development of visual skills in three adults who were blind since birth but gained sight after treatment. The question that they were interested in was how the human visual system recognizes whole objects from diverse image regions. They first studied the visual parsing ability of their subjects on static images. The subjects were asked how many objects they saw in the images in Figure 1.1.

Figure 1.1: Images used in the visual parsing experiment in [1].

The summary of the results is worth directly quoting from their paper:

> *The responses of the recently treated group*[1] *exhibited a consistent pattern. They had no difficulty in enumerating individual geometric shapes [...] that were non-overlapping (Figure (a) above). However, when the shapes overlapped, regardless of whether the shapes were presented as line drawings or as filled transparent surfaces (Figures (b) and (c)), the recently treated subjects' responses were very different from control subjects'.*[2] *They perceived*

[1] "Recently treated group" refers to their subjects who were blind from birth and gained sight recently, after treatment.

[2] Control subjects are people with normal sight.

all closed loops and regions of uniform luminance as distinct objects. [...] when viewing two overlapping squares, the recently treated subjects invariably parsed them as three objects.

The control group, which consisted of normally sighted people, consistently answered all the trials correctly; they reported seeing 2 objects in (a), (b), (c) and (d), and 1 object in (e). However, the recently treated subjects reported seeing 3 objects in (b), (c) and (e). The authors conducted similar experiments with real world images (see Figure 2 (d) in [1]), and the results were no exception: the recently treated subjects consistently tended to see the images in a fragmented manner.

The authors observed that the visual parsing skills of these subjects on static images developed over time, and at the early stages of its development, parsing was driven by low-level image attributes such as hue and luminance rather than organizational rules such as *"the cues of contour continuation, junction structure, and figural symmetry."*

The type of "visual parsing" mentioned above is an example of what is known as image segmentation in the computer vision literature. Image segmentation can be defined as partitioning an image into regions that represent image structure. An image structure is characterized as a connected group of pixels that is homogeneous with respect to certain criteria and has a contrast with its surroundings. Homogeneity and contrast can be based on different types of measurements such as color, texture, motion, depth, etc., and most of the time it is explicitly mentioned which type is used. However, the word *segmentation*, when used by itself, usually refers to color based segmentation.

This dissertation is about low-level hierarchical multiscale segmentation of images. Here, the term "low-level" refers to the fact that we are describing each pixel by its gray-level intensity (or color) which is a local and intrinsic property of that pixel. "Hierarchy" refers to the recursive containment of regions; that is, a region can contain other regions and can also be contained in others. The term "multiscale" is related to the fact that the shape, size and level of homogeneity and contrast of a region can be arbitrary. Region shapes and sizes define the geometric scales, while the levels of homogeneity and contrast define the photometric scales [2]. Therefore, segmentation is essentially automatically detecting all the regions present in an image and

identifying the scales associated with them.

In this work, we propose a new algorithm[3] for solving the low-level hierarchical multiscale image segmentation problem. We model an image region as a connected set of pixels that is surrounded by ramp discontinuities. The magnitude of these discontinuities must be large compared to the variation inside the region. We associate a photometric scale with each region depending on the discontinuity magnitudes of the weakest part of its boundary. Traversing through the range of all possible scales, we obtain all regions present in the image. As the scale increases, regions strictly merge; hence a tree is formed where the root node corresponds to the whole image, and nodes close to the root along a path are large, while their children nodes are smaller and capture embedded details.

To evaluate the accuracy and precision of our algorithm, as well as to compare it to the existing algorithms, we develop a new benchmark dataset consisting of ground-truth segmentation done by human subjects.

There are two major challenges to creating such a dataset. (1) Segmenting images by hand is a tedious process, which could adversely affect the quality of the result, and (2) most importantly, humans tend to draw the boundaries of the (semantic) objects they see, and skip many details. This high-level semantic bias towards objects must be eliminated in order to get the boundaries for all regions. We address these issues by having human subjects segment small image patches instead of whole images. This makes segmentation-by-hand a much easier process and removes the high-level knowledge bias to a large extent because a small image patch is unlikely to contain objects.

We provide evaluation methods for both boundary-based and region-based performance of algorithms. Boundary-based evaluation measures how precisely and completely the ground-truth boundaries are detected, and the region-based evaluation measures how similar the algorithm's partitioning, i.e. segmentation, is to the ground-truth partitioning. We show that our proposed algorithm performs better than the existing low-level segmentation algorithms on this benchmark.

Next, we investigate the statistics of multiscale hierarchical structures present in natural images. It is clear that a natural image is far from being

[3]Our code is available for download at `http://vision.ai.uiuc.edu/segmentation`.

a random configuration of pixels. Rather, it exhibits a high degree of organization, in its spectral and photometric properties, geometry and layout. Segmentation statistics capture the geometric and topological properties of images, which are not possible to obtain using pixel-, patch-, or subband-based methods. In order to collect such statistics, we use our multiscale hierarchical segmentation algorithm. As we mentioned above, our algorithm produces a tree, called the segmentation tree, for a given input image. Each node, or region, in the tree is described by a set of intrinsic photometric (e.g. mean and standard deviation of color) and geometric (e.g. area, compactness, histogram of oriented gradients, etc.), as well as contextual, properties (boundary contrast, properties relative to the parent, children and adjacent regions). Empirical statistics of these properties confirm some of the previous findings on statistics of natural images, e.g. dominant orientations are horizontal and vertical, and provided new findings, particular to segmentation, such as that there are more regions in the lower halves of the images than their upper halves.

Segmentation tree captures the geometrical and topological structures, along with their appearance, present in an image. Therefore, modeling the segmentation tree means learning how natural images are structured. We propose a Markov random field (MRF) model for this purpose. Using the model, we are able to compute $p(S)$, the probability of S being a segmentation tree of a natural image. We then use $p(S)$ as a prior in two applications: image categorization and semantic image segmentation. In the former, we directly use the prior to extract representative feature vectors for images to be used in classification, and obtain comparable results to the state-of-the-art on the image classification task of PASCAL VOC [3]. In the latter, we learn class-conditioned priors and use them to guide semantic segmentation. Using the priors, on the MSRC21 dataset [4], we obtain an improvement of 8% on the segmentation accuracy over the "without-prior" case.

Our work on image categorization using statistics of natural images inspired new ideas for that specific task, and we studied the problem of multiple instance learning (MIL) which resulted in a new MIL algorithm. In MIL, the entities to be classified are called bags, each of which contains an arbitrary number of elements called instances. Training labels are given at the bag level. While the labels of the instances in each bag remain unknown,

conceptually, a negative bag is assumed to contain instances that are all labeled negative, and a positive bag is assumed to contain at least one instance labeled positive. This learning problem appears in numerous fields, especially those related to computer vision applications including content-based image retrieval, image classification, object detection, etc. In such applications, an image or a video is seen as a bag of instances where only a subset of them are meaningful for the given task, e.g. class discrimination. Instances can be interest points, patches or segments, depending on the application, and the size of the meaningful subset of instances is usually small compared to the number of instances a bag typically contains. In our MIL algorithm, we exploit the idea of searching for discriminative instances, which we call prototypes. We propose an additive model for bag classification, which can be learned in a boosting framework. At each iteration of our proposed method, we search for a new prototype so as to maximally discriminate between the positive and negative bags, which are themselves weighted according to how well they were discriminated in earlier iterations. Unlike previous instance selection based MIL methods, we do not restrict the prototypes to a discrete set of training instances but allow them to take arbitrary values in the instance feature space. We also do not restrict the total number of prototypes and the number of selected-instances per bag; these quantities are completely data-driven. We show that our method outperforms state-of-the-art MIL methods on a number of benchmark datasets. We also apply our method to large-scale image classification, where we show that the automatically selected prototypes map to visually meaningful image regions.

Another part of this dissertation is about finding corresponding regions in different images. This problem arises in the context of general correspondence problem which can be defined as the problem of identifying the same physical point in images (of a scene) taken from different viewpoints. Given such images, we are interested in finding region-to-region correspondences across images. We formulate the correspondence quality as a distance function and propose to learn this function from human perception. To this end, we design and develop psychophysical experiments to investigate the importance of different visual cues to describe image regions. In particular, we learned the weights of different cues by evaluating their impact on binocular fusibility by

human subjects. Using a head-mounted display, we showed a set of elliptical shapes to one eye and slightly different versions of the same set of ellipses to the other eye of human subjects. We then asked them whether the ellipses fuse or not. By systematically varying the parameters of the elliptical shapes, and testing for fusion, we learned a perceptual distance function between two elliptical shapes. We evaluated this function on ground-truth stereo image pairs.

The contributions of this dissertation can be summarized in the following items:

- A new algorithm for low-level hierarchical multiscale segmentation of images. Our code is available for download at `http://vision.ai.uiuc.edu/segmentation`.
- A benchmark dataset for low-level segmentation.
- Presentation of segmentation based statistics of natural images, and a Markov random field based model for estimating them.
- Use of segmentation based statistics of natural images in image and scene categorization, and semantic image segmentation problems.
- A new multiple instance learning algorithm, which we call "multiple instance selection boosting," or MIS-Boost for short.
- A psychophysical experiment for evaluating the impact of different visual cues for the region correspondence problem.

1.1 Organization

This dissertation is organized as follows:

- In Chapter 2, we describe our proposed segmentation algorithm, and present and discuss its properties on a set of images. We also give sample segmentation trees.

- Chapter 3 introduces the new benchmark dataset, and methods to evaluate the performance of segmentation algorithms on this benchmark.
- We present our work on the statistics of natural images in Chapter 4. In the experiments section, we demonstrate the use of statistics in image categorization and semantic image segmentation.
- The psychophysical experiment that we developed for the region correspondence problem is given in Chapter 5.
- Finally, we conclude by describing our proposed multiple instance learning algorithm, MIS-Boost, in Chapter 6.

CHAPTER 2

A NEW ALGORITHM FOR LOW-LEVEL MULTISCALE IMAGE SEGMENTATION

In this chapter we present a new algorithm for low-level multiscale segmentation of images. The algorithm is designed to detect image regions regardless of their shapes, sizes, and levels of interior homogeneity, by doing a multiscale analysis without assuming any prior models of region geometry. We model regions as homogeneous sets of connected pixels surrounded by ramp discontinuities. A new transform, called the ramp transform, is described, which is used to detect ramp discontinuities and seeds for all regions in an image. Region seeds are grown towards the ramp discontinuity areas by utilizing a relaxation labeling procedure. Segmentation is achieved by analyzing the output of this procedure at multiple photometric scales. Finally, all detected regions are organized into a tree data structure based on their recursive containment relations. Experiments on real and synthetic images verify the desired properties of the proposed algorithm.

2.1 Introduction

Low-level image segmentation partitions a given image into regions which are characterized by some low-level properties of their constituent pixels, where the term "low-level" refers to local and intrinsic properties of pixels such as gray-level intensity (or color), contrast, gradient, etc. For a set of connected pixels to form a region, they should have a certain degree of interior homogeneity and a discontinuity with their surroundings, where the magnitude of discontinuity is large compared to the interior variation.

Our goal is to detect image regions regardless of their shapes, sizes and levels of interior homogeneity. These goals preclude the use of any prior model about region shape or geometry, and the fact that a region can have

any level of homogeneity requires us to do multiscale analysis. Furthermore, we want the algorithm to work without requiring any image-dependent parameter tuning. Achieving these goals is challenging because of the nature of discontinuities that separate regions. A sharp edge in the 3D world might be mapped to a wide ramp discontinuity in the image due to defocusing and penumbral blur in the image acquisition process. Hence, region boundaries in images, which can have arbitrary shapes, are surrounded by ramp discontinuities of various widths and heights.

2.2 Related Work

The earliest techniques proposed for segmentation were based on thresholding. In this approach, a histogram of gray-level values (or colors) of all pixels is computed and then peaks and valleys of this histogram are located. A given image is segmented by thresholding it using the values where valleys occur in the histogram. These kind of methods are extremely efficient and work well for simple figure-background images. In practice, the valleys of the histogram cannot be located easily and reliably. A survey of these methods is given in [5]. Thresholding is still being used in segmentation related tasks in recent research. For example, Matas et al. use thresholding and look for stable regions which do not change much as the threshold is changed [6].

A significant body of literature is about region-growing techniques. In this technique, a region is started from a seed pixel. Then, neighboring pixels are inspected recursively and added to the region if they pass a similarity test. Designing this similarity test is the most important part of the technique because it determines the boundary, i.e. where the region growing would end. Related to region growing are split-and-merge techniques where regions are merged or a region is split to satisfy some predefined similarity and dissimilarity rules. The reader is referred to the book by Shapiro and Stockman [7] for a recent coverage of these techniques.

Another popular technique for segmentation is clustering [8], [9], [10]. In this approach, each pixel is associated with a feature vector containing the x, y position of the pixel in the image plane and some features (such as intensity, color, texture) extracted from that pixel. Then, a clustering method is used

to find the clusters in this joint feature space. Detected clusters correspond to image regions. Although any clustering method can be used for this purpose, agglomerative methods, k-means, expectation-maximization and mean-shift clustering are widely used. The shortcoming of this approach is that one does not know which parameters are good or how to select them for the clustering algorithm, e.g. k for k-means, kernel bandwidth for mean-shift, etc. A good review of segmentation using clustering is given in the book by Forsyth and Ponce [11].

Graph based methods are also adopted for image segmentation [12], [13], [14]. In these approaches, a graph is constructed in such a way that each pixel corresponds to a node and the weighted edges between nodes encode a measure of affinity. Affinity can be expressed in terms of spatial proximity, intensity/color and texture similarity between nodes. In [12], [13], the goal is to cut the graph into a predefined number of connected components in such a way that the sum of weights of the edges that are cut should be minimal. In practice, finding these minimal cuts boils down to computing the eigenvectors of a matrix called "the affinity matrix" which contains the weights of edges in the graph. The most popular segmentation algorithm in this category is the normalized-cuts [13] (known as N-cuts). These approaches have the same shortcoming of clustering based segmentation methods: the user has to determine the number of regions in advance. Another graph based method worth noting is Felzenswalb's algorithm [14] where segmentation is formulated as finding multiple disjunct minimum-spanning-trees (MST) that cover the entire image. Each tree corresponds to a distinct region, and existence of an inter-region boundary is decided based on a heuristic predicate that compares the contrast across the boundary (minimum weight edge between two MSTs) and the interior contrast (maximum weight edge) within each region (MST). Although this formulation comes close to our own region model, the algorithm computes a single-level partitioning of the image; that is, it ignores multiscale aspect of segmentation, it is too sensitive to its input parameters (as we will show in Chapter 3) and it implicitly uses the step-edge model (hence ignores the ramp-edge model) which causes the algorithm to produce many thin regions within ramp areas between regions.

In the context of our study, we can classify the previous work as either not being multiscale, or imposing models on the geometry of edges. The ear-

liest approaches to segmentation such as thresholding [15], region-growing [15] and watersheds [16] ignore the multiscale aspect of the problem. Energy minimization based approaches such as Markov random field (MRF) modeling [17] and active contours [18] enforce constraints on the local shape of regions; therefore, they are not capable of detecting arbitrarily shaped regions. Graph based methods such as normalized cuts [13] and graph cuts [12] require the number of regions to be given as input, which does not guarantee the detection of regions at all scales. Clustering methods attempt to find regions as clusters in the joint space of pixel positions and some features, (e.g. intensity, texture, etc.) extracted from pixels. For this, they either need the number of regions or some density estimation parameters [10, 9] as input. As discussed in [19], mean-shift based segmentation cannot detect steep corners due to averaging and tends to create multiple boundaries, hence many small regions, for the blurred edges in the image.

In recent years, many segmentation algorithms have been developed by aiming to maximize performance on the Berkeley Segmentation Benchmark Dataset [20] which contains images segmented by humans. We note that these are object-level segmentations and many regions in the images are not marked (i.e. many edges are not marked even if there is strong visual evidence for an edge, or some edges are marked where there is too little or no visual evidence). It is not our goal, in this study, to segment objects out; instead we aim to detect low-level image structures, i.e. regions, as accurately and completely as possible so as to provide a reliable and usable input to higher level vision algorithms.

2.3 The Models and the Algorithm

We follow the line of research in [2] and [19], and develop a new algorithm to achieve the aforementioned goals. As in [2], we use gray-level intensity and contrast as the low-level properties to define and detect low-level image structures. We define an image region as a set of connected pixels surrounded by ramp discontinuities, as done in [19]. We model ramp discontinuities with strictly increasing (or decreasing) intensity profiles. Each ramp discontinuity has a magnitude, or contrast, which allows us to associate a photometric scale

with each boundary fragment surrounding regions. We achieve a multiscale segmentation over a range of scales, by progressively removing boundary fragments whose photometric scales are less than the current scale of analysis. Finally, all regions detected at all photometric scales are organized into a tree data structure according to their recursive containment relations.

In this work, we propose a new method, called the ramp transform, for detection of ramps and estimation of ramp parameters. At a given pixel, we analyze multiple intensity profiles passing through the pixel in different directions, and estimate the magnitude of the ramp discontinuity at that pixel by minimizing a weighted error measure. After applying the ramp transform, seeds for all image regions are detected and these seeds grow to become complete regions.

Our contributions are: (1) A new segmentation algorithm which detects image regions of all shapes, sizes and levels of intensity homogeneity. It arranges regions into a tree data structure which can be used in high level vision problems. The algorithm gives better results and is less sensitive to image-dependent parameter tuning, compared to the existing popular segmentation algorithms. (2) A new transform, called the ramp transform, which takes an image and outputs another image where the value at each pixel gives a measure of ramp discontinuity magnitude at that pixel in the original image. (3) A new ground-truth dataset for low-level image segmentation. To the best of our knowledge, this is the first dataset of its kind for such purpose. This dataset could also be used as a benchmark for edge detection algorithms.

A set of connected pixels, R, is said to form a *region* if it is surrounded by ramp discontinuities, and the magnitudes of these discontinuities are larger than both the local intensity variation and the magnitudes of discontinuities, within R. To elaborate this definition, we first describe the ramp discontinuity model and the ramp transform.

2.3.1 Ramp model

We assume that the ramp discontinuities which separate regions in an image are characterized by increasing or decreasing intensity profiles. Consider the one-dimensional image f given in Figure 2.1(a). The part of the curve

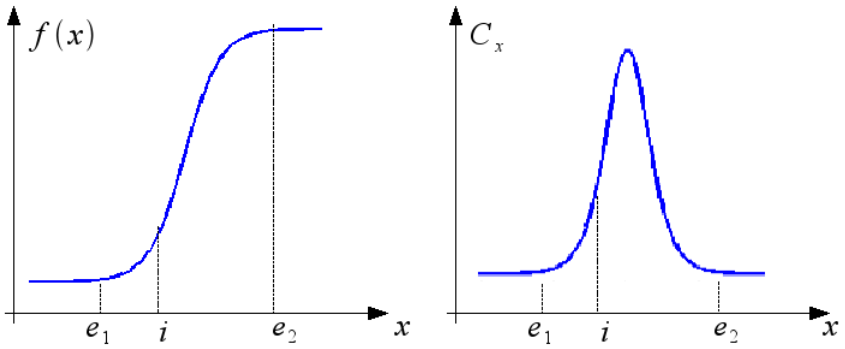


Figure 2.1: (a) Ramp model. (b) Ramp transform of $f(x)$. C_i is equal to $|f(i+a) - f(i-a)|$ where $a = \min\{|i-e_1|, |i-e_2|\}$.

between e_1 and e_2 is a ramp. The width of the ramp is $|e_1 - e_2|$, and its magnitude is $|f(e_2) - f(e_1)|$. Additionally, we define two measures, "ramp quality" and "point-magnitude," which will help us to generalize the ramp model to 2D functions.

We define the "ramp quality" as the ratio between its magnitude and width, namely: $\frac{|f(e_2)-f(e_1)|}{|e_1-e_2|}$. The "point-magnitude" at location i is defined as:

$$C_i = |f(i+a) - f(i-a)| \tag{2.1}$$

where $a = \min\{|i-e_1|, |i-e_2|\}$. Note that the ramp magnitude and point-magnitude are different measurements. Both are equal only when the point i is located at equal distances to the endpoints, e_1 and e_2, and $f(x-i)$ is an odd function.

In 2D images, computing the point-magnitude of the ramp discontinuity at $\vec{i}$, i.e. $C(\vec{i})$, is not a trivial task as it is in the 1D case. This is because an infinite number of lines, hence ramp intensity profiles, pass through the pixel $\vec{i}$. We assume that a pixel $\vec{i}$ is within a ramp discontinuity if it has at least one strictly increasing (or decreasing) intensity profile passing through it.

2.3.2 Ramp transform

The transform converts the input image I to a scalar height field C. The height at location $\vec{i}$ in C corresponds to the point-magnitude of the ramp discontinuity at $\vec{i}$ in the original image I.

If I is a 1D image, computing the ramp transform amounts to computing C_i for all i by setting $f = I$ in Eq. (2.1). The ramp transform of the ramp of Figure 2.1(a) is given in Figure 2.1(b).

If I is a 2D image, an infinite number of lines pass through $\vec{i}$, and each of these lines has its own intensity profile. To parametrize these lines – and their corresponding intensity profiles – let us define an angle θ which is the angle that the line makes with a horizontal row of the image. For a finite set of angles in $[0, 2\pi)$, we analyze the intensity profiles and measure the corresponding ramp parameters. For the ramp discontinuity at angle θ, let $q_{\vec{i}}^{\theta}$ be its ramp quality and $c_{\vec{i}}^{\theta}$ its point-magnitude at $\vec{i}$. Finally, we set $C_{\vec{i}} = \max_{\theta} c_{\vec{i}}^{\theta}$. In the following, we drop the hat of $\hat{C\vec{i}}$, and use $C_{\vec{i}}$.

2.3.3 Obtaining seeds for regions and the region model

The output, C, of the ramp transform contains point-magnitudes of the ramp discontinuities found in I. In this section, we first describe our region model and then elaborate on how seeds for all image regions are detected from C.

A set of connected pixels, R, is said to form a *region* if: (1) it is surrounded by ramp discontinuities, (2) the magnitudes of these discontinuities are greater than both the local intensity variation and the magnitudes of discontinuities, within R.

To obtain regions that conform with the above definition, we look for the basins of the height field C. To find all basins, all photometric scales, i.e. contrast levels, are traversed from lower to higher and new basins are marked as they occur. Then, the set of basin pixels, $\mathcal{S}$, contains the seeds for image regions. The remaining set of pixels, $\mathcal{D}$, correspond to the ramp discontinuity areas, and we call these pixels ramp pixels.

We find the connected-components in the set $\mathcal{S}$, and label each component,

which corresponds to a distinct basin of C, with a unique label. If there are N connected-components, then each pixel in the set S takes a label from the set $\mathcal{L} = \{1, 2, 3, \ldots, N\}$.

2.3.4 Region growing by relaxation labeling

Having obtained the regions seeds ($\mathcal{S}$), we want to grow them by propagating labels towards the ramp discontinuity areas ($\mathcal{D}$) which are unlabeled. For this purpose, we use a relaxation labeling [21] procedure. Although the classical watershed transform might be utilized here, we choose not to use it because it does not give good edge-location accuracy at corners and junctions.

Let $\vec{i} \leftarrow \ell$ denote the event of assigning label ℓ to pixel $\vec{i}$, and $P^{(t)}(\vec{i} \leftarrow \ell)$ denote the probability of this event happening at iteration t. Relaxation labeling iteratively updates these probabilities so that the labeling of pixels get more consistent as the method iterates. Next, we describe how we compute the initial probability values.

Computing Priors. For a pixel that is part of any detected region seed, we define the prior probabilities as $P^{(0)}(\vec{i} \leftarrow \ell) = 1$ if $\vec{i} \in R_\ell$ (0, else) $\forall \vec{i} \in \mathcal{S}, \forall \ell \in \mathcal{L}$.

On the other hand, the prior probabilities of the pixels which are within the ramp discontinuities, i.e. those $\vec{i} \in \mathcal{D}$, are not trivial. To compute these priors, we design a cost function for assigning label ℓ to pixel $\vec{i}$, i.e. the event $\vec{i} \leftarrow \ell$, as follows.

Consider the scenario given in Figure 2.2. Pixel $\vec{i}$ is within the ramp discontinuity area among regions R_1, R_2, and R_3. The point $\vec{j}_1$ is the closest point to $\vec{i}$, in region R_1 (similarly $\vec{j}_2$ for R_2 and $\vec{j}_3$ for R_3). Let $\overline{ij_1}$ denote the line segment connecting $\vec{i}$ and $\vec{j}_1$. We compute the cost of assigning label 1 to ramp pixel $\vec{i}$ by analyzing the intensity profile along the line segment $\overline{ij_1}$. The cost function is designed in such a way that the flatter the profile is, the lesser the cost, and vice versa. We achieve this by summing up finite differences of the intensity profile at regular intervals. Formally, the cost of assigning label ℓ to a ramp pixel $\vec{i}$ is given by:

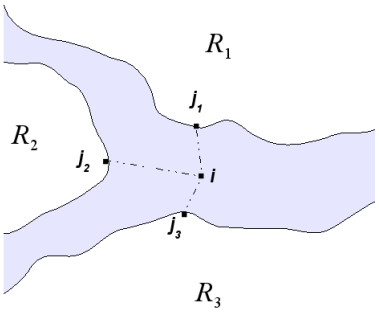


Figure 2.2: Computing the initial probabilities for relaxation labeling. R_1, R_2, R_3 represent region seeds and blue areas are unlabeled ramp areas.

$$G(\vec{i} \leftarrow \ell) = \sum_{n=1}^{\|\vec{i}-\vec{j}\|/h} |I_{\vec{i}+nh\vec{u}} - I_{\vec{i}+(n-1)h\vec{u}}| \tag{2.2}$$

where $\vec{j}$ is the closest pixel to $\vec{i}$ such that $\vec{j} \in R_\ell$, h is a small stepsize, $\|\vec{i}-\vec{j}\|$ is the distance between $\vec{i}$ and $\vec{j}$, and $\vec{u} = \frac{\vec{j}-\vec{i}}{\|\vec{i}-\vec{j}\|}$, a unit vector.

To compute prior probabilities for a ramp pixel $\vec{i}$, we use:

$$P^{(0)}(\vec{i} \leftarrow \ell) = \frac{G^{-1}(\vec{i} \leftarrow \ell)}{\sum_{k\in\mathcal{L}} G^{-1}(\vec{i} \leftarrow k)}, \text{for } \forall \vec{i} \in \mathcal{D}, \forall \ell \in \mathcal{L}. \tag{2.3}$$

Relaxation labeling. Once the probabilities are initialized by $P^{(0)}(\cdot)$, we iteratively update them by the following relaxation labeling update rule:

$$P^{(t+1)}(\vec{i} \leftarrow \ell) = \frac{P^{(t)}(\vec{i} \leftarrow \ell)(1 + Q^{(t)}(\vec{i} \leftarrow \ell))}{\sum_{k\in L} P^{(t)}(\vec{i} \leftarrow k)(1 + Q^{(t)}(\vec{i} \leftarrow k))} \tag{2.4}$$

where $Q(\cdot)$ is defined as:

$$Q^{(t)}(\vec{i} \leftarrow \ell) = \frac{1}{|\mathcal{N}_{\vec{i}}|} \sum_{\vec{j}\in\mathcal{N}_{\vec{i}}} \sum_{k\in L} R_{\vec{i}\vec{j}}(\ell, k) P^{(t)}(\vec{j} \leftarrow k). \tag{2.5}$$

Here $\mathcal{N}_{\vec{i}}$ denotes the neighbors of pixel $\vec{i}$ and $R_{\vec{i}\vec{j}}(\ell, k)$, called the compatibility function, gives a measure of how compatible the assignments "$\vec{i} \leftarrow \ell$" and "$\vec{j} \leftarrow k$" are. The constraint on $R(\cdot)$ is that it should return values in the interval $[-1, +1]$: 1 meaning that the two events are highly compatible,

-1 meaning just the opposite. We choose the following form:

$$R_{\vec{i}\vec{j}}(\ell, k) = \begin{cases} e^{-\frac{|I_{\vec{i}} - I_{\vec{j}}|}{s}} & , \ell = k \\ e^{-\frac{\mathcal{M} - |I_{\vec{i}} - I_{\vec{j}}|}{s}} & , \ell \neq k \end{cases} \tag{2.6}$$

where $\mathcal{M}$ is the maximum value that $|I_{\vec{i}} - I_{\vec{j}}|$ can take for any I, $\vec{i}$, $\vec{j}$. It is 255 for standard 8-bit images. This compatibility function forces the neighboring pixels with similar intensities to have the same labels.

Final labeling of ramp pixels. When the highest change in any $P^{(t)}(\cdot)$ becomes very small, we stop the iterations and label the ramp pixels with the labels having the maximum probabilities:

$$\vec{i} \leftarrow \arg\max_{\ell} P^{(t)}(i \leftarrow \ell). \tag{2.7}$$

2.3.5 Multiscale segmentation: Computing photometrically stable regions

After relaxation labeling, we have a full-labeling of image pixels where each unique label corresponds to a unique region. From this labeling, we produce a boundary map B where adjacent regions are separated by boundary pixels whose contrasts are already known from the output of the ramp transform, C.

Given a photometric scale σ, i.e. contrast, we want to compute the corresponding segmentation based on the following criteria: (1) all boundary pixels having contrasts larger than σ, i.e. *strong* boundary pixels, should be part of region boundaries, (2) all boundary pixels having contrasts smaller than σ, i.e. *weak* boundary pixels, should NOT be part of region boundaries.

These criteria could be satisfied by a simple thresholding of B with σ, but in practice this would lead to leakages in region boundaries which would prevent the detection of the regions. For this reason, we relax the above criteria by (1) allowing some *weak* pixels to group with *strong* pixels – so that a closed region boundary, hence a region, is formed – and (2) allowing some *strong* pixels to be dangling, i.e. not being part of any region boundary. These new criteria

suggests an optimization problem where we want to minimize the number of *weak* pixels that are part of region boundaries, and the number of *strong* pixels that are dangling. Formally, to obtain the photometric segmentation, S_σ, at a given scale σ by:

$$\min \quad \sum_{\vec{p} \in B} \mathbf{1}_{\{C(\vec{p}) < \sigma, \vec{p} \in S_\sigma\}} + \sum_{\vec{p} \in B} \mathbf{1}_{\{C(\vec{p}) > \sigma, \vec{p} \notin S_\sigma\}} \tag{2.8}$$

where $C(\vec{p})$ denotes the contrast of the boundary pixel $\vec{p}$, $\vec{p} \in S_\sigma$ means that pixel $\vec{p}$ is part of the boundary of a region in S_σ, and $\mathbf{1}_{\{.\}}$ is the indicator function. Without loss of generality, the problem above can be formulated in terms of the *boundary fragments* in B instead of the pixels in B. This is because when a boundary pixel is off, it will leave the entire fragment that it is part of in a dangling state.

Let $G = (V, E)$ be the region-adjacency graph for the full-labeling computed above (2.7). Each vertex in V corresponds to a region, and each edge in E corresponds to a boundary-fragment separating two adjacent regions. We assign weights to the edges in E using the given photometric scale σ:

$$w(e) = \sum_{\vec{p} \in e} \mathbf{1}_{\{C(\vec{p}) < \sigma\}} - \mathbf{1}_{\{C(\vec{p}) \geq \sigma\}} \qquad \forall e \in E, \tag{2.9}$$

where $w(e)$ is the weight of the boundary fragment e. The weaker this fragment (i.e. the more weak pixels it has), the less desirable it is to be part of a region boundary in S_σ. So, the problem given in (2.8) can be re-expressed as:

> **Problem:** PHOTOMETRICALLY STABLE REGIONS AT σ (PSRσ, for short)
> Given $G = (V, E)$ with edges weighted according to (2.9), remove a set of edges E' from E such that:
>
> 1. The sum of the weights of the remaining edges $(E - E')$ should be minimum.
> 2. The remaining edges $(E - E')$ should form a valid segmentation (S_σ), i.e. no dangling edges in S_σ.

In the final graph $G' = (V, E - E')$ assign vertices v_1 and v_2 the same region label if, the edge between them, $e \in (v_1, v_2)$ was removed. The solution S_σ is given by the labeled vertices in G'.

The problem stated above is similar to the MINIMUM EDGE DELETION K-PARTITION (MEDKP, for short) problem which is known to be an NP-hard optimization problem [22, 23]. In fact, with a minor modification to the definition above, these two problems become exactly the same. First, for completeness, we define the MEDKP problem below:

Problem: MINIMUM EDGE DELETION K-PARTITION (MEDKP)

Given a graph $G = (V, E)$ and a weight function $w : E \to N$, find a k-partition (coloring) of vertices such that the total weight of the monochromatic edges (i.e. those that connect same-color vertices) is minimized:

$$min \sum_{(v_1,v_2)\in E, c(v_1)=c(v_2)} w(v_1, v_2) \tag{2.10}$$

where c denotes a k-partitioning or k-coloring, $c : V \to [1 \dots k]$

In MEDKP, the purpose is to group the vertices into k groups where the total sum of the weights of the within-group edges are minimized. In PSRσ, we want to remove as many weak fragments as possible while keeping as many strong fragments as possible and not leaving leakages to prevent dangling strong fragments. If we think of removing an edge in PSRσ as coloring the vertices to which the removed edge was incident, then it is easy to see that both problems are exactly the same except for two things: (1) in MEDKP only positive weights are allowed, (2) in MEDKP the sum of the monochromatic edges, i.e. edges to be removed, is minimized, whereas in PSRσ we want to maximize the sum of the removed edges. Therefore, with the following two modifications, MEDKP and PSRσ become equivalent: (1) Modify (2.9) by multiplying the right-hand side by -1, (2) add a big-enough constant to the weights of all the edges in PSRσ's G so that all weights become positive.

Since PSRσ is NP-hard, we consent to a greedy optimization for its solution. In particular, we use agglomerative region merging to obtain S_σ from B and C as follows. Let $S_\sigma = B$, and repeatedly remove fragments from S_σ, which are weak (i.e. it has positive weight, see Eq. (2.9)), until there is no weak fragment left. This procedure guarantees a local minimum solution for PSRσ.

Regions at all photometric scales are obtained by starting from the lowest level photometric scale and removing boundary fragments having contrast less than σ, progressively as σ increases. This process, which is an agglomerative clustering of regions according to the inequality above, ensures that remaining regions always conform with the region model and successive merges create a strict hierarchy.

Algorithm 1 Segmentation algorithm

Input: Image I.
Output: Segmentation tree, T, of I.

1: $C \leftarrow$ RampTransform(I)
2: Find basins of C. These are seeds for regions.
3: Grow seeds to ramp areas by relaxation labeling which gives a full-labeling of all pixel, equivalently a boundary map B.
4: **for** $\sigma = \sigma_{min}$ to σ_{max} with $\Delta\sigma$ increments **do**
5: $S_\sigma \leftarrow$ PSR$\sigma(B)$
6: **end for**
7: Collect and make a list, L, of all regions from all S_σ.
8: Sort L by region area.
9: Initialize segmentation tree, T, to the whole image.
10: **for** each region r in L **do**
11: Insert r into T such that r becomes a leaf node in T and r is entirely contained in its parent node.
12: **end for**

2.3.6 Constructing the segmentation tree

Due to the nature of the multiscale segmentation process described above, regions merge as the scale of analysis, σ, is increased. This allows us to arrange the regions into a tree data structure. Suppose that regions R_1 and R_2 at photometric scale σ_n have merged and become R_3 at scale σ_{n+1}. Then, in the segmentation tree R_1 and R_2 should be the children of R_3. Applying

this rule recursively for all regions, we obtain a tree of regions, called the segmentation tree, where the root node corresponds to the entire image itself. We give the overall segmentation algorithm, where input is an image I, and the output is the segmentation tree of I, in Algorithm 1. We illustrate all steps of the algorithm on a simple synthetic image, in Figure 2.3.

2.3.7 An efficient alternative to region growing

In this section, we propose an efficient alternative to region growing by relaxation labeling. This alternative also includes a slight modification to the ramp transform as well.

We found out that the ramp transform (Section 2.3.2) failed to locate the ramp endpoints correctly in some cases and this failure caused errors in the localization of boundaries. Examples are given in Figures 2.4 and 2.5.

Figure 2.4(c) shows the image intensity profile sampled at the green dots – which lie on the same straight line – shown in 2.4(b). These dots are numbered from 1 to 20 (in red) for illustrative purposes. Observe that from the first green dot (1^{st}) to the last one (20^{th}), we perceive two boundaries which are marked by arrows: one at 7^{th}, and the other at 14^{th}; i.e. there is a single region between these dots. However, the ramp transform cannot find any ramp endpoints at these positions, since the sign of the derivative along the whole profile does not change. The problem is that the region between the 7^{th} and the 14^{th} dots neither has a flat intensity profile nor is surrounded by ramps having opposite signs of slope. Nonetheless, the region is not completely missed because ramps have a predefined maximum width. This maximum width guarantees to mark two points as endpoints (e.g. for the 8^{th} green dot, the ramp endpoints are the 5^{th} and the 11^{th}, if the maximum ramp width is chosen to be 6 pixels) even if none are found by looking for a change of sign of the derivative. The output of the ramp transform and the overall segmentation result are given in Figure 2.5. The boundary localization error is clearly visible in Figure 2.5(c).

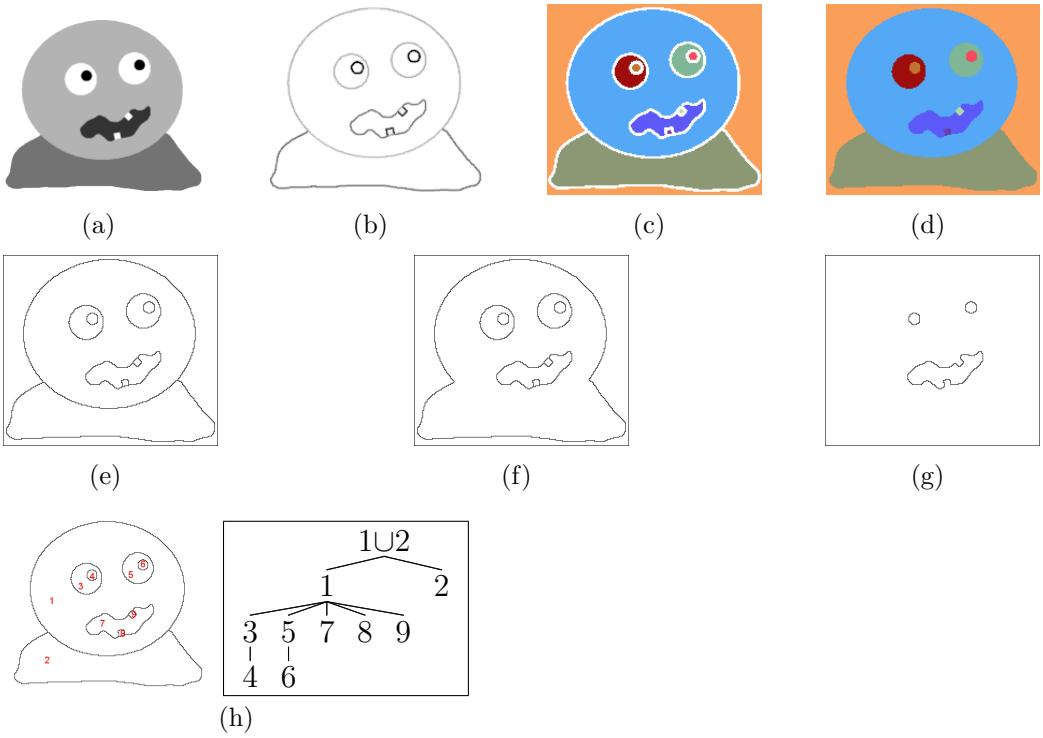
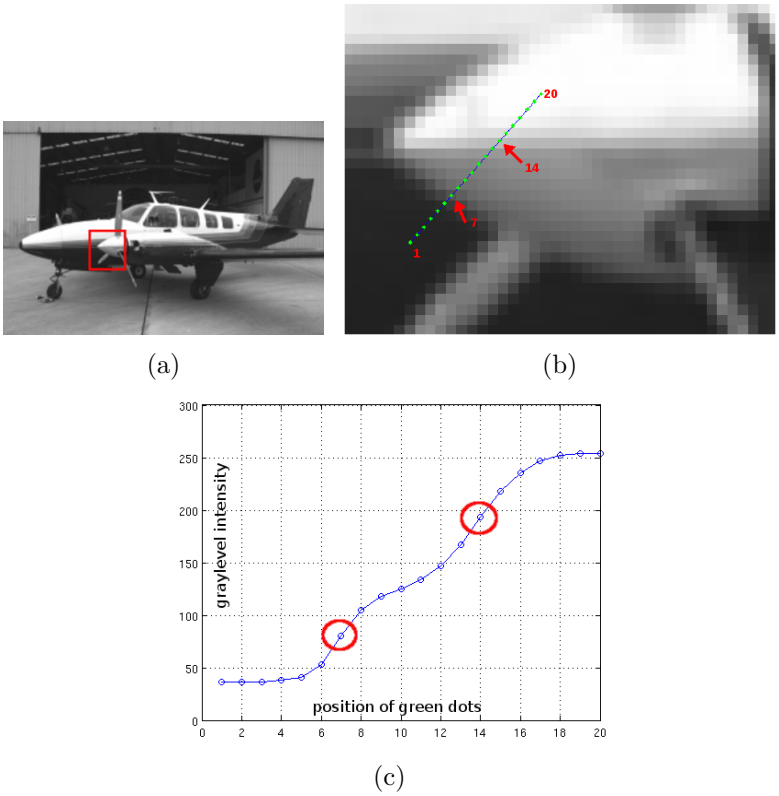


Figure 2.3: Illustration of steps in the algorithm. (a) Input image I. (b) Output of ramp transform, C, applied to I. Here, the darker the pixel, the higher the contrast of the underlying ramp. (c) Basins of C. Each basin is represented with a different color. These basins correspond to region seeds and the remaining pixels are ramp pixels (white color). (d) Final labeling obtained by growing the region seeds towards the ramp pixels using relaxation labeling. (e, f, g) Results of multiscale segmentation. (e) Segmentation result for photometric scale $\sigma = 5$. All regions are included. (f) Segmentation for $\sigma = 65$. Two regions (head and the body) merged. This means that the photometric scale of the boundary fragment in between the two merging regions is less than 65. (g) Segmentation for $\sigma = 80$. More regions have disappeared. The remaining regions are of photometric scale larger than $\sigma = 80$, ensured by the region model and the algorithm. (h) Segmentation tree. On the left, each region is labeled by a number. Using the containment relations of regions, our algorithm computes the tree given on the right-hand side.



(a) (b)

(c)

Figure 2.4: An example for the ramp transform's failure to estimate the ramp endpoints correctly. **(a)** An input image. The red rectangle shows the rectangular patch of interest for this example. **(b)** Zoomed-in version of the red rectangle in (a). Image intensity is sampled along the green dots which lie on the blue straight line. Green dots are marked with sequential integers in red fonts. **(c)** Intensity profile along the green dots in (b). Note that we perceive a region between the 7^{th} and the 14^{th} dots but the intensity profile is not flat between these points and the sign of the derivative of the whole profile is the same, hence no ramp endpoints are found by the ramp transform.

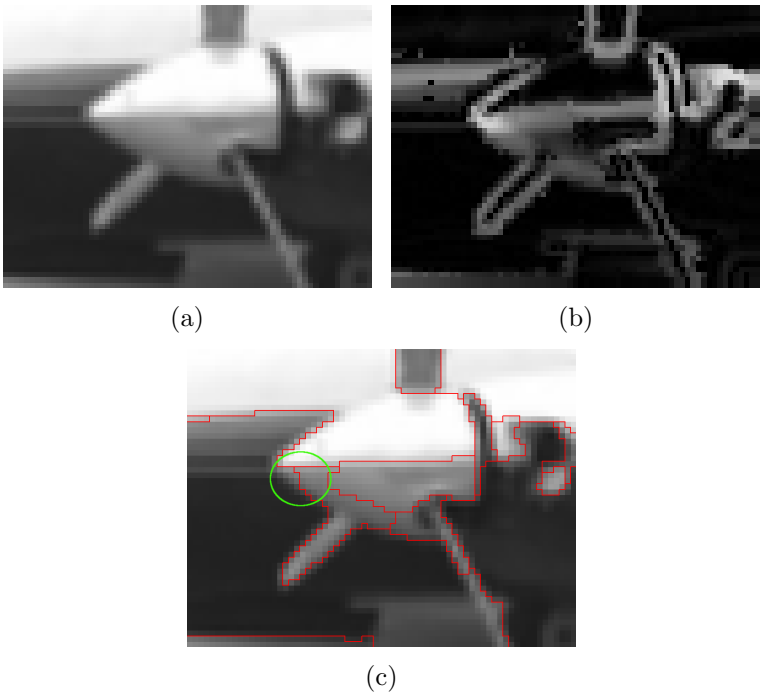
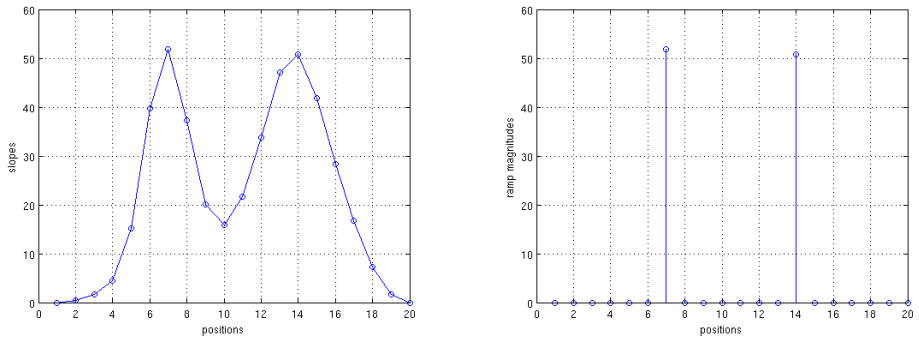


Figure 2.5: Ramp transform's boundary localization error. **(a)** An image patch. **(b)** Output of ramp transform on (a). **(c)** Segmentation result (a) at scale $\sigma = 20$. Red lines show region boundaries. Observe the boundary localization error marked by the green circle in (c).



(a) (b)

Figure 2.6: **(a)** Slopes of the profile in Figure 2.4(c) at each pixel. **(b)** Non-maxima suppression applied to (a).

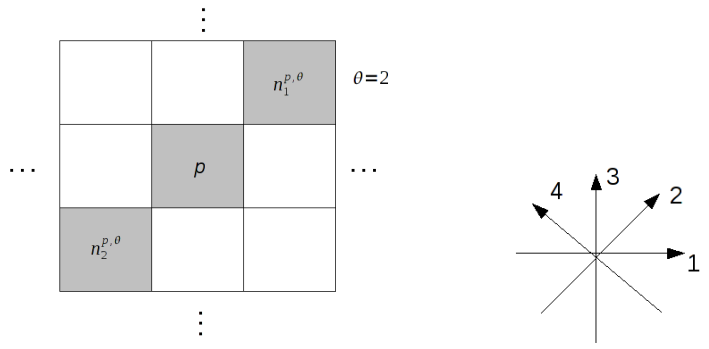
2.3.8 Ramp detection by non-maxima suppression

Because of the problems mentioned above, we propose to change the way ramp discontinuities are detected. Let us consider the 1D intensity profile given in Figure 2.4(c), and estimate its slope at each pixel (using the filter $[-1 \;\; 0 \;\; +1]$). The result is given in Figure 2.6(a) where two peaks are clearly observed at locations 7 and 14. (Note that these are the locations where we expect the region boundaries.) Each peak corresponds to a unique ramp, and each ramp should be represented at the location of its peak slope with a magnitude equal to this peak value itself. These can be achieved by applying non-maxima suppression to the curve in 2.6(a). The final result, shown in Figure 2.6(b), gives the location of the ramps detected and their magnitudes. Next, we describe how we applied the same procedure to 2D images.

Given a 2D image I, we want to compute the ramp magnitude and direction at a pixel $\boldsymbol{p}$. We first analyze the intensity profile passing through $\boldsymbol{p}$ at a certain direction θ. Let $\boldsymbol{n}_1^{p,\theta}$, $\boldsymbol{n}_2^{p,\theta}$ denote the pixels which are immediate neighbors to $\boldsymbol{p}$ and are on both sides of $\boldsymbol{p}$ through direction θ (see Figure 2.7). We define the ramp slope at pixel $\boldsymbol{p}$ at direction θ as:

$$D(\boldsymbol{p}, \theta) = I(\boldsymbol{n}_1^{p,\theta}) - I(\boldsymbol{n}_2^{p,\theta}). \tag{2.11}$$

As in the 1D case, if this ramp slope (or magnitude) is smaller than the slope computed at the neighboring pixels, i.e. if it is not a maxima, then it



(a) (b)

Figure 2.7: Illustration of how ramp magnitude is computed. **(a)** The pixel of interest $\boldsymbol{p}$ and its 8-neighborhood. Direction $\theta = 2$ is shown as an example. $\boldsymbol{p}$ and its immediate neighboring pixels at direction $\theta = 2$ are marked with gray. Ramp magnitude at pixel $\boldsymbol{p}$ is given by the expression (2.13). **(b)** 4 directions are used.

is suppressed:

$$C(\boldsymbol{p},\theta) = \mathbf{1}_{\{D(\boldsymbol{p},\theta)>D(\boldsymbol{n}_1^{p,\theta})\}}\mathbf{1}_{\{D(\boldsymbol{p},\theta)>D(\boldsymbol{n}_2^{p,\theta})\}}D(\boldsymbol{p},\theta). \tag{2.12}$$

Here $C(\boldsymbol{p},\theta)$ denotes the ramp magnitude at $\boldsymbol{p}$ along direction θ. Note that if $\boldsymbol{p}$ is suppressed by any of its neighbor pixels, then $C(\boldsymbol{p},\theta) = 0$.

Finally, we define the ramp magnitude at $\boldsymbol{p}$ as:

$$C(\boldsymbol{p}) = \max_{\theta} C(\boldsymbol{p},\theta) \tag{2.13}$$

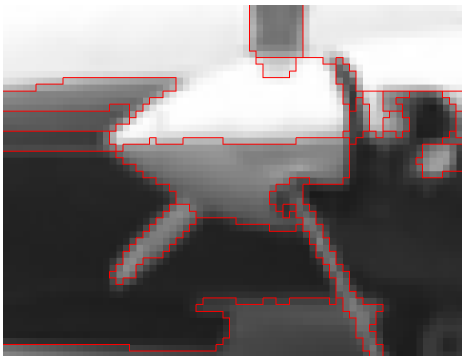
and the ramp direction as:

$$T(\boldsymbol{p}) = \arg\max_{\theta} C(\boldsymbol{p},\theta) \tag{2.14}$$

In our implementation, we used only 4 directions, and considered only the 8-neighborhood of pixels to find $\boldsymbol{n}_1^{p,\theta}$ and $\boldsymbol{n}_1^{p,\theta}$. See Figure 2.7 for an illustration.

Ramps detected by the procedure described above for the input image given in Figure 2.5(a) are given in Figure 2.8(a). The segmentation obtained using this ramp map is shown in Figure 2.8(b). The boundary localization

(a) The output of the new ramp detection method on the image in Figure 2.5(a). Compare this to the output of the ramp transform in Figure 2.5(b).



(b) Segmentation result obtained using the ramp map in (a). Compare this to the previous result given in Figure 2.5(c). Boundary localization error is corrected.

Figure 2.8: Example output for the new ramp detection method.

error seen in Figure 2.5(c) is corrected.

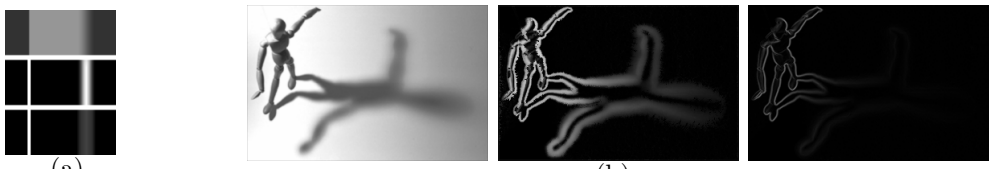
We provide a MATLAB implementation of our segmentation algorithm online at `http://vision.ai.uiuc.edu/segmentation`.

2.4 Experiments and Results

We first demonstrate the properties of ramp transform. To show that it can correctly measure the contrast of the underlying ramp edge, we created a synthetic image containing two edges of the same intensity contrast (Figure 2.9(a)). One of the edges is a step-edge, and the other is a ramp edge. Although they have different widths, we expect that the ramp transform gives similar values for these two edges. Figure 2.9(a) illustrates this property.

Figure 2.9(b) illustrates the ramp transform on a real image (taken from [24] with permission). This image contains ramp edges of various widths. A fixed-length gradient filter is incapable of measuring the ramp magnitudes correctly (see Figure 2.9(b), bottom). The ramp transform successfully estimates the pointwise ramp magnitudes as expected. Next, we describe how we quantitatively compared our segmentation algorithm with available algorithms.

Next, we give some ramp detection results in Figure 2.10 for both the orig-



(a) (b)

Figure 2.9: Illustration of the ramp transform. (a) Top: A synthetic image containing a sharp step edge, on the left, and a wide ramp edge on the right, which was obtained by blurring a step edge by a Gaussian kernel of $\sigma = 4$. The contrasts of both edges are 100. Middle: Ramp transform of the synthetic image. For both edges the peak value of the response of the transform is 100, which is the contrast of the edges. Bottom: Gradient magnitude, obtained by horizontal and vertical $[-1\ +1]$ filters. The responses for two edges are not the same. (b) Ramp transform of a real image. Left: An image containing ramp discontinuities of varying widths. Middle: Ramp transform of the image. Right: Gradient magnitude of the image.

inal ramp transform (Section 2.3.2) and the non-maxima-suppressed version of it.

Next, we give photometric segmentation results on real images. Figures 2.11-2.15 show input images and their segmentations at four selected photometric scales. In the following, we demonstrate the performance of the segmentation algorithm for several challenging cases.

Thin regions The bicycle image in Figure 2.12 has many thin regions (see the wheels) which are correctly segmented. Other examples for thin regions can be found in the house image (Figure 2.11, see the roof of the house), in the airplane image (Figure 2.14, see the propeller, the lights on top, and the thin strip on the ground), in the building image (Figure 2.15, see the frames of the windows), and more examples can be found in the "man with his bicycle image" (Figure 2.13, the microphone on man's shirt, the pole in the background, etc.).

High-curvature boundaries Steep corners and jagged boundaries can be included under this title. Examples for steep corners can be observed in the house (Figure 2.11) and the bicycle (Figure 2.12) images. Examples for junctions can be found in all of the images given (Figures 2.11-2.15). In our

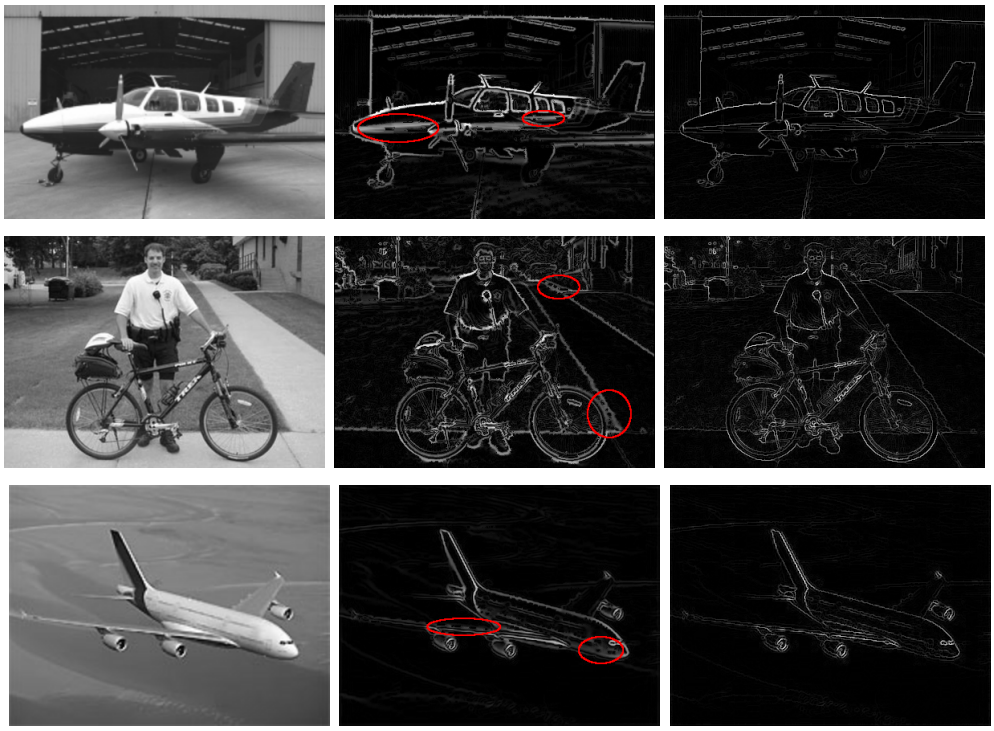


Figure 2.10: Three real images and their ramp maps. The results in the second column are obtained by the ramp transform (RT), and those in the last column are obtained by the non-maxima suppression (NMS) algorithm described in Section 2.3.8. In general, RT gives thicker responses than NMS and this causes two problems: (1) failure to detect thin and/or small regions (because they do not have any seeds due to the thick responses), and (2) thick responses give rise to false boundaries. We marked some examples of these problems with red circles above. Compare the circled parts with their counterparts in the last column. NMS does not suffer from the thick response problem.

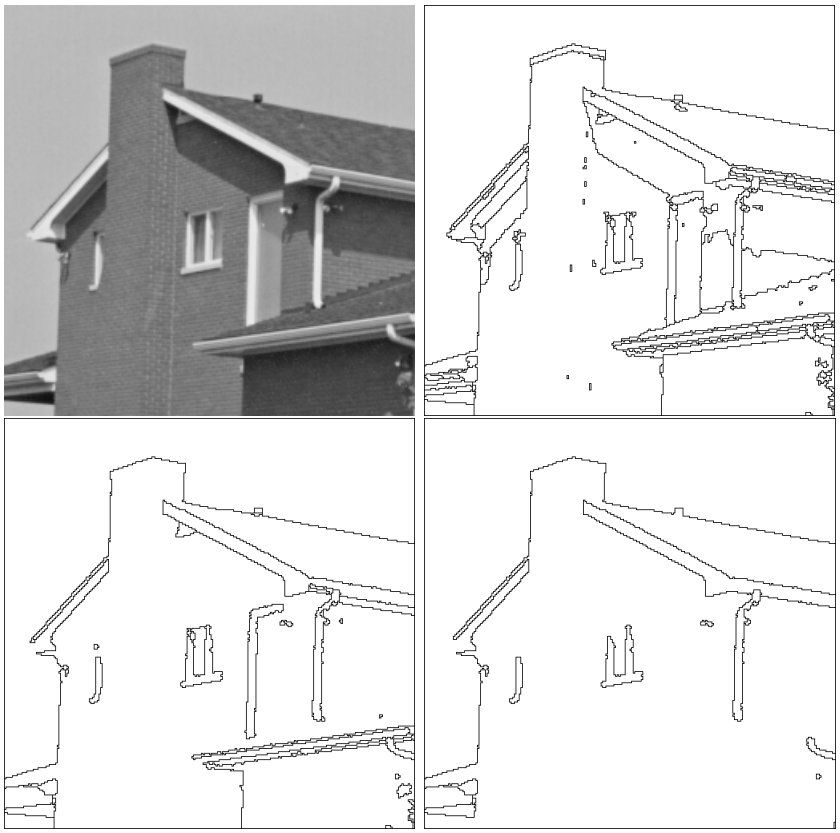


Figure 2.11: House image. Segmentation results shown at scales 25, 45, and 70.

experiments, we did not encounter any cases where a region boundary is lost due to a leakage in junctions.

Wide ramp (diffuse) edges The mannequin image in Figure 2.16 is a good example where ramp edges of various widths coexist. There are very sharp edges (on the mannequin) and wide, blurred edges (on the shadow) as well. The segmentation algorithm detects the region boundaries correctly despite this variety of edge widths.

We finally give two sample segmentation trees in Figures 2.17 and 2.18. As we noted earlier in Section 2.3.6, the hierarchy is completely determined by containment relations of regions (not by their photometric scales). In a segmentation tree, the root node corresponds to the whole image, and nodes close to the root along a path are large, while their children nodes are smaller and capture embedded details.

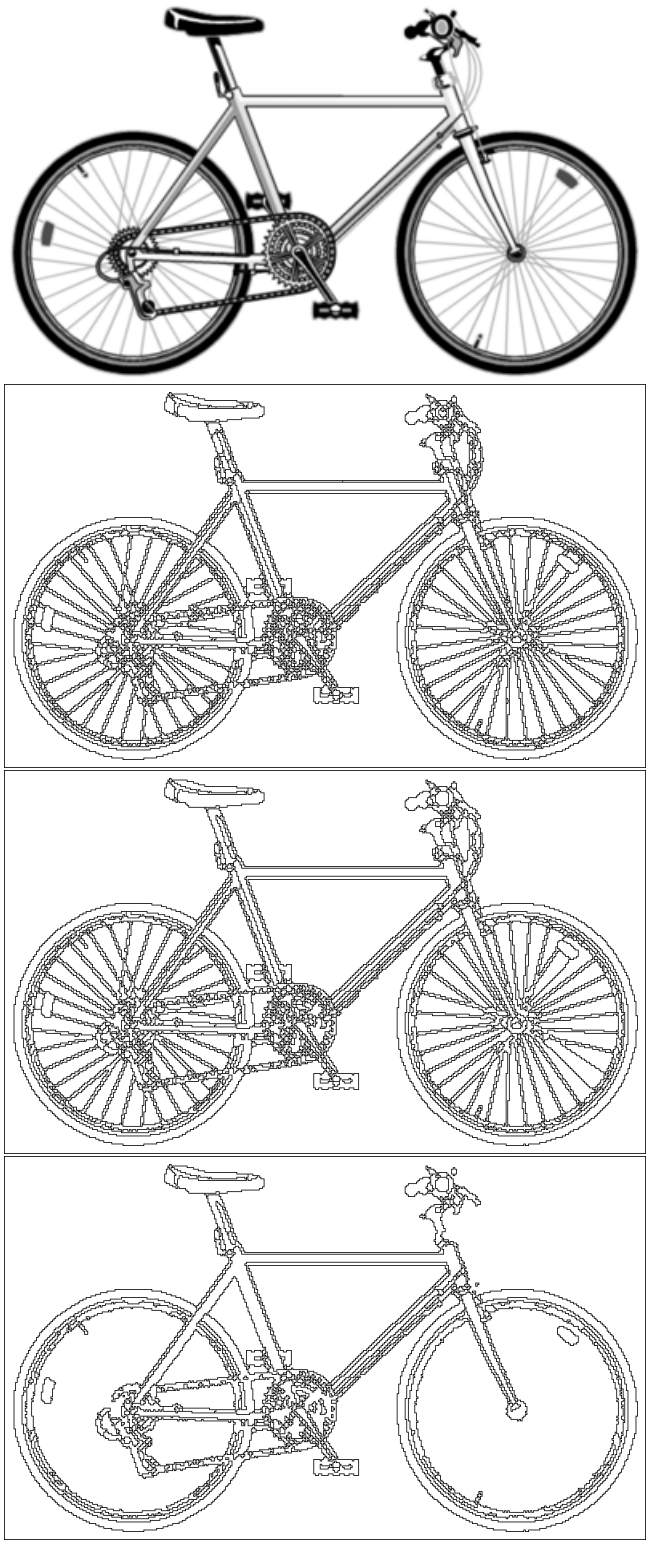


Figure 2.12: Bicycle image. Segmentation results shown at scales 20, 45, and 80.

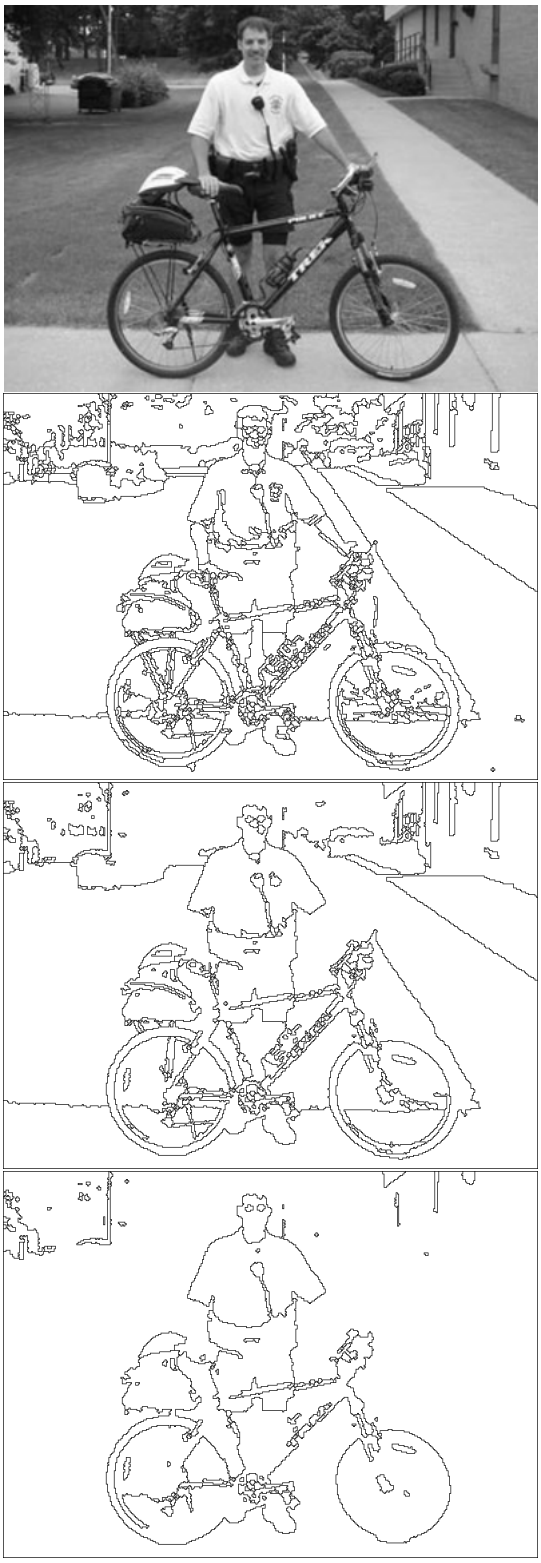


Figure 2.13: Image of a man with his bicycle. Segmentation results shown at scales 25, 40, and 60.

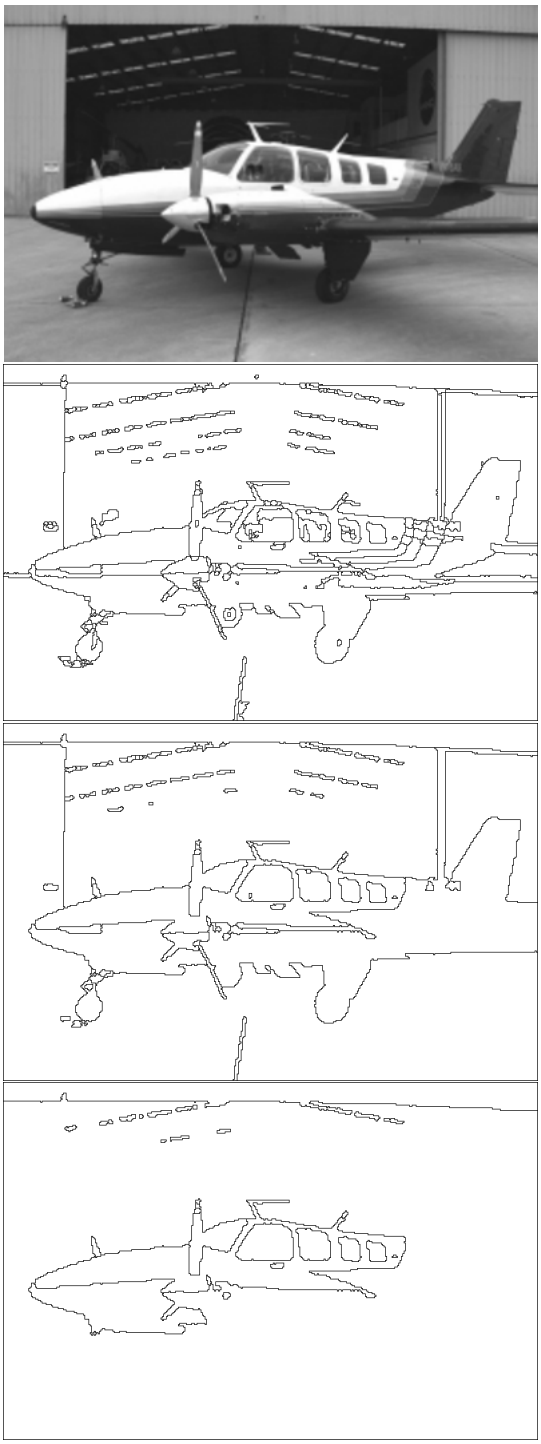


Figure 2.14: Airplane image. Segmentation results shown at scales 20, 40, and 65.

Figure 2.15: Building image. Segmentation results shown at scales 30, 45, and 80.

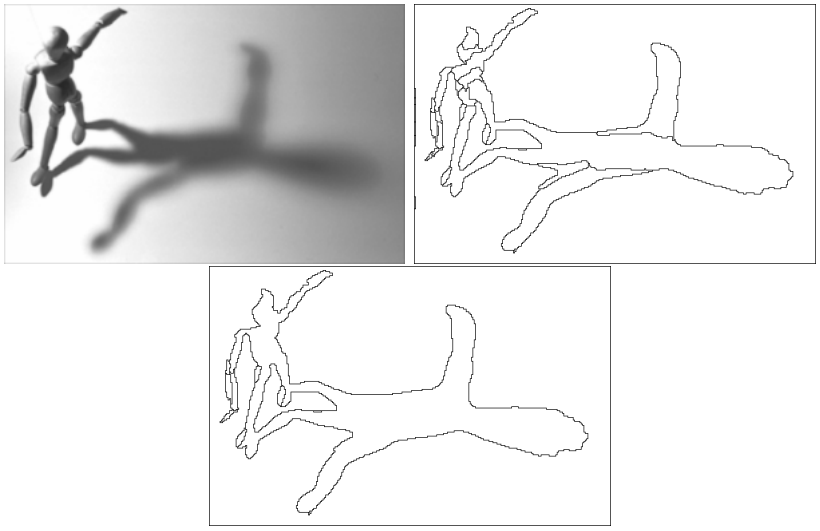


Figure 2.16: Mannequin image. Segmentation results shown at scales 20 and 35.

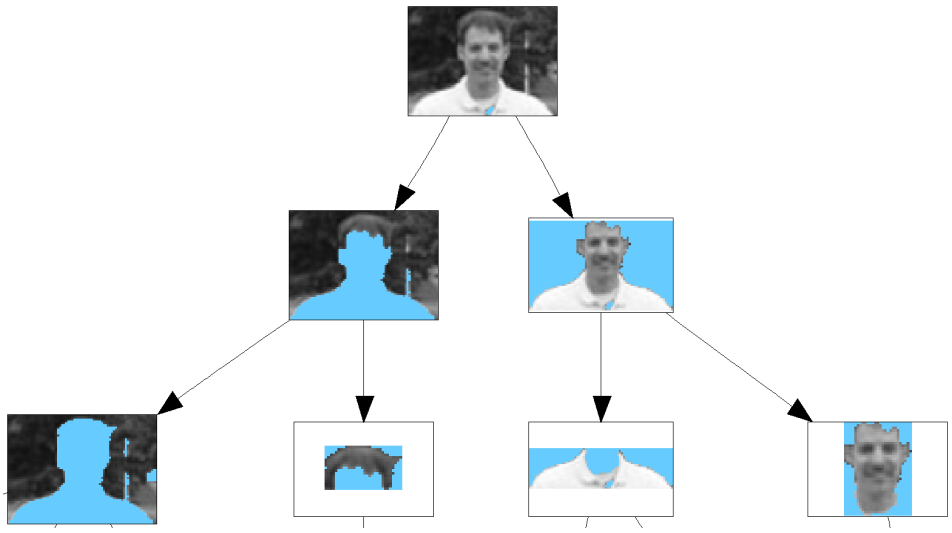


Figure 2.17: Part of an automatically generated segmentation tree. The root node corresponds to the whole image, and nodes close to the root along a path are large, while their children nodes are smaller and capture embedded details. Each region is drawn on a light-blue background for better visualization of its boundaries.

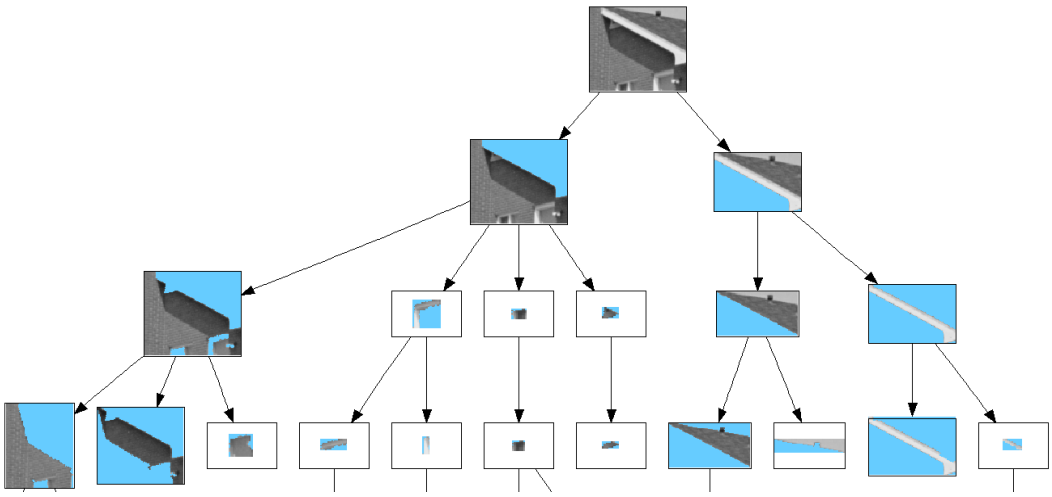


Figure 2.18: Part of an automatically generated segmentation tree. See the caption of Figure 2.17 for an explanation.

2.5 Summary

In this chapter, we presented a new algorithm for low-level multiscale image segmentation. The algorithm is capable of detecting low-level image structures having arbitrary shapes, at arbitrary homogeneity levels. A low-level image structure, or region, is defined as a connected set of pixels surrounded by ramp discontinuities. To detect regions, the image is converted to a ramp magnitude map. We call this conversion the ramp transform. Then we find the basins of the ramp magnitude map, and consequently obtain region seeds. These seeds are grown by a relaxation labeling procedure to get the final segmentation. After this, we obtain multi-photometric scale segmentation by doing a multiscale analysis over a range of scales where, at each scale, boundary fragments having less contrast than the current scale are removed. This process guarantees a strict hierarchy. Using this property, we arrange the regions into a tree data structure which we call the segmentation tree. We provide a MATLAB implementation of our algorithm online at `http://vision.ai.uiuc.edu/segmentation`. The code takes an image as input, and produces the segmentation tree.

CHAPTER 3

A BENCHMARK DATASET FOR LOW-LEVEL SEGMENTATION

3.1 Introduction

This chapter is about evaluating the performance of the segmentation algorithm we presented in the previous chapter. We design and develop a benchmark dataset for low-level segmentation, consisting of ground-truth segmentations done by human subjects. We provide evaluation methods for both boundary-based and region-based performance of algorithms. Boundary-based evaluation measures how precisely and completely the ground-truth boundaries are detected, and the region-based evaluation measures how similar the algorithm's partitioning, i.e. segmentation, is to the ground-truth partitioning. We use the benchmark to quantitatively compare the performance of our algorithm to those of existing, popular approaches.

3.2 The Dataset

Creating a ground truth dataset for low-level image segmentation is a challenging task because (1) segmenting images by hand is a laborious and tedious process, which could adversely affect the quality of the result, and (2) most importantly, humans tend to draw the boundaries of the (semantic) objects they see, and skip many details. This high-level semantic bias towards objects must be eliminated in order to get the boundaries for all regions.

We address these challenges by having human subjects segment small image patches instead of whole images. This makes segmentation-by-hand a much easier process and removes the high-level knowledge bias to a large extent because a small image patch is unlikely to contain objects. We also

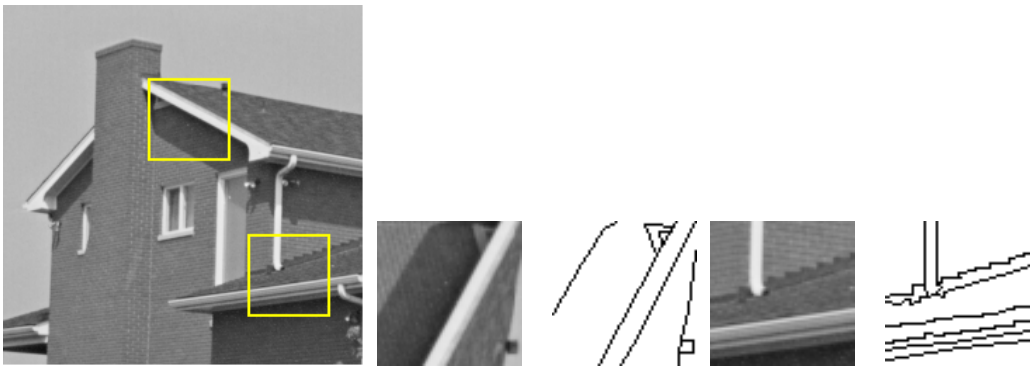
Figure 3.1: The set of 15 images used to create the benchmark dataset.

randomly rotate the image patch (at multiples of 90 degrees) to further reduce this bias.

We used a set of 15 images to create the benchmark dataset. Figure 3.1 shows a mosaic of these images. We fixed the size of the patches at 50×50 pixels and randomly extracted 25 patches per image, thus obtaining 375 patches in total. Each patch was segmented by 3 to 5 different human subjects. We developed a graphical user interface which let the subjects segment patches by hand using a stylus pen on a tablet-pc.

Figure 3.2 shows an image from our dataset and two example patches randomly extracted from it. Human segmentations for these patches are also given.

One might ask why we needed to develop a new benchmark dataset while there is already one, the Berkeley Segmentation Benchmark Dataset (BSDS) [20]. The short answer is that it is not suitable for evaluating low-level segmentation. In BSDS, human subjects were asked to segment out objects, not regions, and that is why many details in terms of low-level segmentation are missing. An example is given in Figure 3.3. There are many such examples. Had we used the ground-truth given in Figure 3.3, then we would have been penalized for detecting the high-contrast (black and white) re-



(a) (b) (c)

Figure 3.2: **(a)** An image from the dataset. **(b)** The patch represented by the upper yellow square on (a) and its ground-truth segmentation. Note that the patch is rotated 90 degrees clockwise. **(c)** The other patch and its ground-truth segmentation.

Figure 3.3: An example image and its ground-truth segmentation from the Berkeley Segmentation Benchmark Dataset [20]. Many regions in the lower-right quadrant of the image are not marked at all in the ground-truth.

gions in the lower-right corner of the image. We do not consider BSDS as a suitable benchmark for low-level image segmentation. BSDS is good for benchmarking high-level or object-level segmentation. Our goal in this study is not object-level segmentation; instead, we want to detect all regions as accurately and completely as possible.

3.3 Performance Measures

We evaluate the performance of a segmentation algorithm by looking at its segmentation accuracy over the patches where ground-truth segmentations are provided by human subjects. We provide two different methods to evaluate the accuracy of segmentation: boundary-based method, and region-based method. Boundary-based evaluation measures how precisely and completely the ground-truth boundaries are detected, and the region-based evaluation measures how similar the algorithm's segmentation is to the ground-truth segmentation.

3.3.1 Boundary-based evaluation

The segmentation performance for a single patch is measured by precision-recall values obtained by comparing the ground-truth and machine's output. We illustrate this evaluation method with an example. Consider the image in Figure 3.4(a) and its patch in Figure 3.4(b) with its ground-truth in Figure 3.4(c) (call this as G). Suppose that our segmentation algorithm produced the result given in Figure 3.4(d); thus the segmentation corresponding to the patch is Figure 3.4(f) (let this be R). Now, we compare G and R and find correspondences between the boundary pixels in R and G. Each boundary pixel in G either matches with a single boundary pixel in R, or does not match with anything at all. To find the optimal matching between G and R, we use the method described in Appendix B of [25], which casts the problem as a minimum cost bipartite assignment problem. The result of matching is given in Figure 3.4(g) where red pixels denote the boundary pixels of G, and blue pixels denote the boundary pixels of R. The white line between a red pixel and a blue pixel indicates a match.

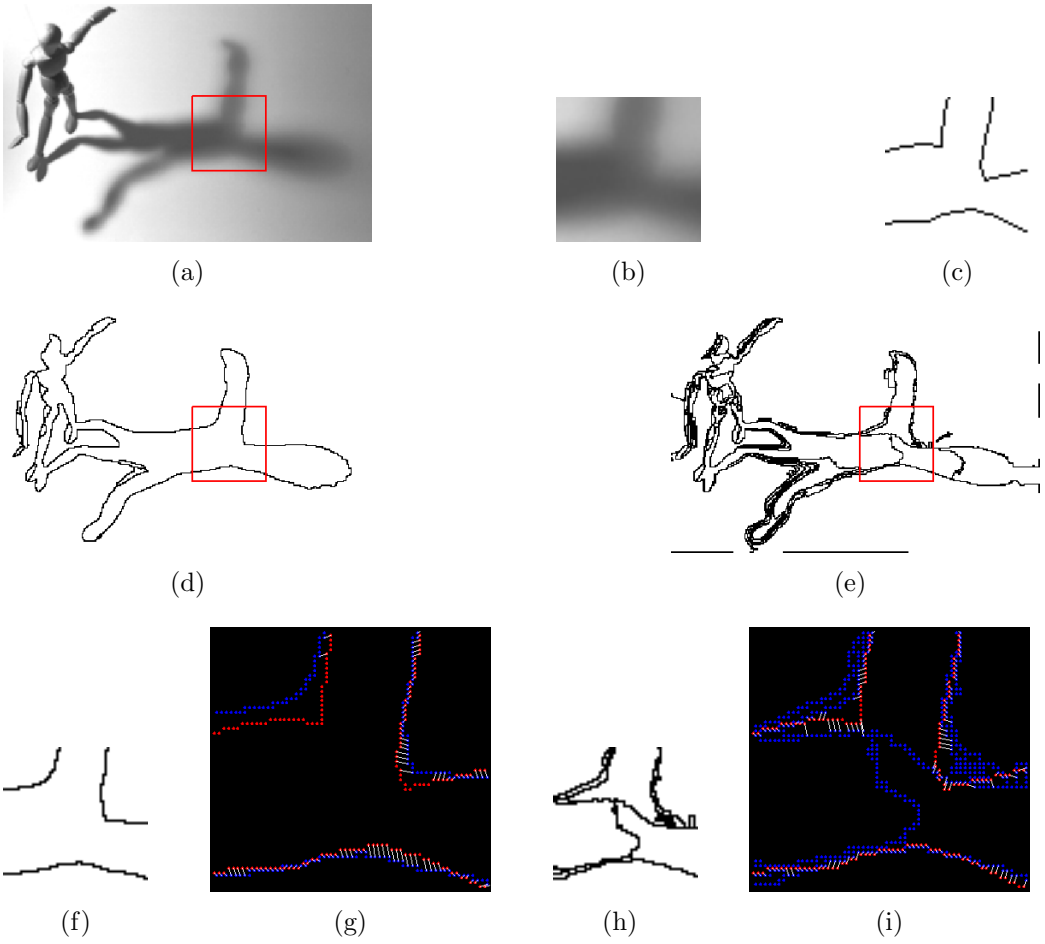


Figure 3.4: Illustration of the performance measure. **(a)** An image from our dataset. A patch is marked by the red square. **(b)** Magnified version of the patch. **(c)** Provided human segmentation for the patch. **(d)** Segmentation of (a) obtained by our algorithm. **(e)** Segmentation of (a) obtained by the mean-shift based algorithm. **(f)** Our result at the location of the patch (red square in (d)). **(g)** Matching result between the ground-truth (c) and our result (f). Red pixels represent the ground-truth, blue pixel represent algorithm's output (our result, in this case). White lines denote matching pixels. **(h)** Mean-shift based method's result at the location of the patch (red square in (e)). **(i)** Matching result between (c) and (h). See (g) for explanation.

As done in [25], we measure the *goodness* of the match by precision-recall. If the image patch is considered as a query, G as its relevant (ground-truth) result, and R as the retrieved result (by the algorithm), we could compute precision-recall as:

$$\text{Recall } r = \frac{|\text{relevant } \cap \text{ retrieved}|}{|\text{relevant}|}, \tag{3.1}$$

$$\text{Precision } p = \frac{|\text{relevant } \cap \text{ retrieved}|}{|\text{retrieved}|}. \tag{3.2}$$

We combine precision and recall using the F-measure defined as: $f = \frac{2pr}{p+r}$. For the matching result of Figure 3.4(g), precision-recall and F-measure values are $p = 0.61$, $r = 0.66$, $f = 0.63$, whereas for Figure 3.4(i) they are $p = 0.31$, $r = 0.95$, $f = 0.47$. Note that for the latter case, the recall is very high since only a few red pixels are unmatched, but the precision is low because there are plenty of blue pixels that are unmatched.

3.3.2 Region-based evaluation

The boundary-based method falls short of penalizing a serious segmentation error: leakage. When there is a leakage on a boundary, the whole segment actually gets lost, but the boundary-based method could still give a good result – if there is a partial boundary left – depending on the amount of leakage (Figure 3.5). The region-based method, on the other hand, directly compares two different partitionings, and not the recovered boundaries.

We use the variation of information (VI)[26] as our region-based evaluation measure. VI is commonly used in the clustering literature and it measures the "distance" between two partitionings of the same set.

Given two segmentations X, Y of the same patch, VI is defined as:

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

where H is the entropy, and I is the mutual information function:

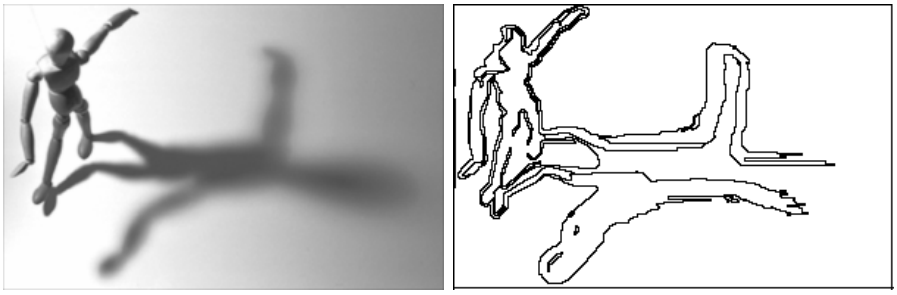


Figure 3.5: Illustration of why a region-based evaluation method is needed. On the right is the segmentation result obtained by Felzenszwalb's algorithm [14] (with parameters $\sigma = 0.5$, $k = 750$, and $a = 10$; see Section 3.4 for an explanation of parameters) for the mannequin image on the left. The region that corresponds to the shadow of the mannequin is not correctly segmented: it is merging with the background at the head's shadow (compare this result with the one given in Figure 2.16). Boundary-based method does not penalize this error much because most of the shadow's boundary is there, whereas the region-based method would give a high penalty because a large region (shadow) is completely missing.

$$H(X) = -\sum_{x} p(x)\log(p(x)) \tag{3.4}$$

$$I(X,Y) = \sum_{y}\sum_{x} p(x,y)\log\left(\frac{p(x,y)}{p_X(x)p_Y(y)}\right). \tag{3.5}$$

Above, lower-case letters represent regions; that is, x is a region in X. $p(x)$ and $p(x,y)$ are computed as:

$$p(x) = \frac{\#\text{ pixels in } x}{\#\text{ pixels in } X}\,, \qquad p(x,y) = \frac{\#\text{ pixels in } (x \cap y)}{\#\text{ pixels in } X}. \tag{3.6}$$

3.4 Algorithms

We compared our algorithm with three available algorithms which are widely used in the literature: Felzenszwalb's graph-based algorithm [14], Multiscale NCuts [27], and the mean-shift algorithm [10].

Both Felzenszwalb's algorithm and the mean-shift algorithm require three input parameters from the user and we do not know which values to use. So,

we sample a large number of input parameters for both algorithms. Mean-shift based segmentation method [10] requires a spatial bandwidth σ_s, a range bandwidth σ_r, and a minimum region area a. We selected the following input parameter space: $\{\sigma_s, \sigma_r, a\} \in \{5, 7, 9, 11, 15, 20, 25\} \times \{3, 6, 9, 12, 15, 18, 21, 24, 27\} \times \{5, 10\}$. The graph-based algorithm [14] expects a smoothing scale σ, a threshold k which is the scale of observation (equation (5) in [14]), and a minimum region size a, as input. We used the following parameter space: $\{\sigma, k, a\} \in \{0.5, 1, 1.5, 2\} \times \{250, 500, 750, 1000, 1250, 1500, 1750, 2000, 2250, 2500\} \times \{5, 10\}$.

Our algorithm outputs a hierarchy of regions. However, in order to compute its segmentation accuracy and compare it with other algorithms, we need single-layer segmentation results. For this purpose, we flatten the segmentation tree by projecting all detected regions onto a plane and use the result on that plane as our single-layer segmentation output. We vary the lowest photometric scale (σ) as our variable: $\sigma = \{10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38\}$. Regions with lower photometric scales than σ are ignored, and not included in the segmentation tree.

For Multiscale NCuts, the variable is the number of regions. We used $\{20, 30, 40, \ldots, 290, 300\}$.

3.5 Comparison

For each algorithm, we vary the input parameters and compute average recall and precision values over all patches of all images. These values are given in Figure 3.6. There is one point per input parameter set of each algorithm. In this plot, one ideally wants to be as close as possible to the top-right corner which denotes the perfect recall and precision point. We see that our algorithm performs better, in both precision and recall, than the other three algorithms.

Since each patch is segmented by multiple different subjects, we can compute a "human performance" by cross-validation. Specifically, for each patch, we pick one of the human segmentations, and evaluate its performance by comparing it to the other human segmentations of the same patch. Then,

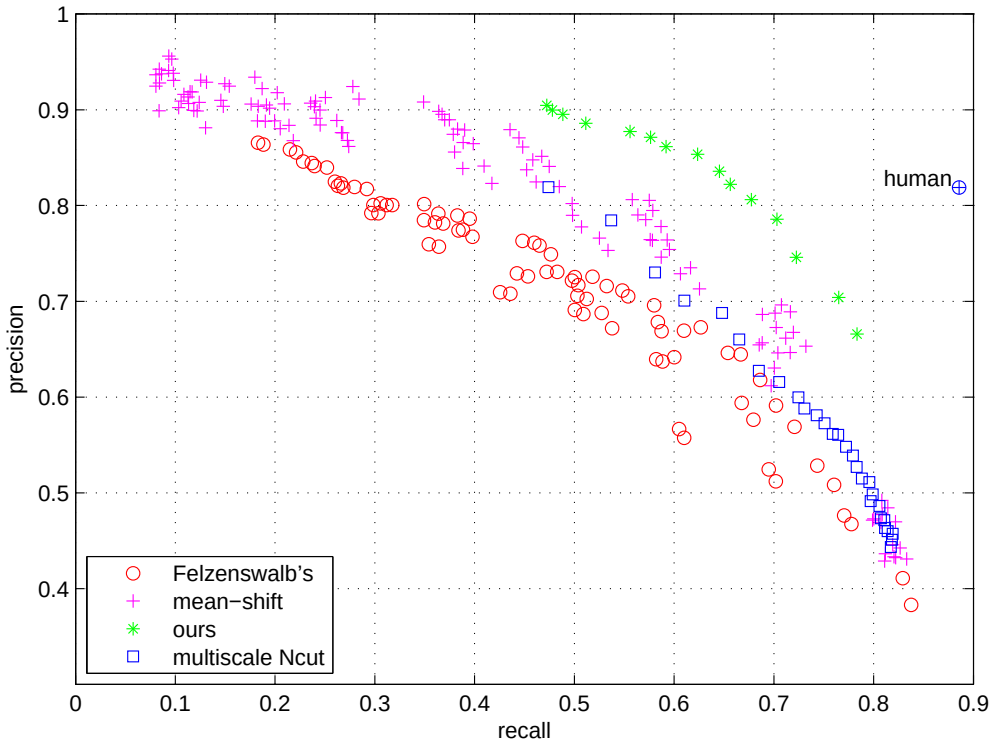


Figure 3.6: Recall-precision plot for all algorithms over all patches of all images. Each point in the plot corresponds to a different input parameter set for its corresponding algorithm. Cross-validated human performance is also given (as a single point) in the plot. Our proposed algorithm has better recall-precision than the others.

we repeat the same for all human segmentations, and take the average of the results. This method is commonly known as leave-one-out cross-validation. "Human performance" is given in Figure 3.6 as a single point because there are no input parameters to vary for human segmentation. Although not perfect, it has pretty high recall (0.89) and precision (0.82), which suggest that segmentations done by different human subjects are pretty consistent.

An interesting observation about the performances of the mean-shift and the graph-based algorithm is that they both have highly varying precision values for a very narrow range of recall values. This means that changing the input parameters, while keeping the number of recalled boundary pixels (almost) the same, causes large changes in the precision. This is indeed the case: both algorithms produce many spurious regions when the input parameters are changed, for the same number of recalled boundary pixels. On the other hand, NCuts' performance have relatively less variance. However, it

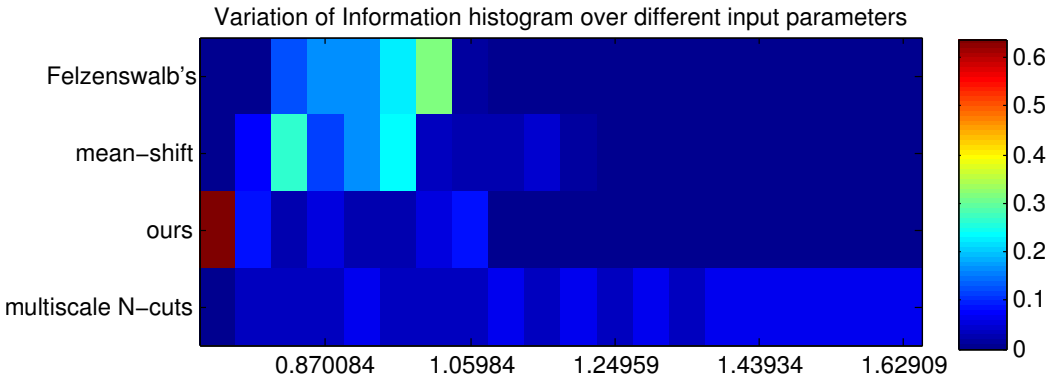


Figure 3.7: Histograms of average variation of information scores for each algorithm. Each row is a histogram of VI scores over the different input parameters of the respective algorithm. Our algorithm achieves the least (i.e. the best) VI, outperforming others. Cross-validated human performance is 0.58.

is concentrated in the high-recall low-precision regime, and pretty far away from the human performance. Our algorithm is the least sensitive to its input parameters (its set of precision-recall points form a relatively compact set compared to others'), and it performs better than the others.

For the region-based evaluation, we computed the average variation of information (VI) over all patches of all images per input parameter set of each algorithm – as we did in the boundary-based evaluation above. We give the color-coded histograms of these VI scores in Figure 3.7. Each row represents a histogram of VI scores over different input parameters. It is easily observed that our algorithm has the lowest (around 0.86), hence the best, VI scores. Our comments above about the variability of results when the input parameters are changed seem to be valid – even more so – for this evaluation measure, too; our algorithm is the least sensitive to its input parameters.

"Human performance" is again computed by leave-one-out cross-validation, and turns out to be 0.58.

3.6 Summary

In this chapter, we presented a new benchmark dataset for low-level segmentation. The dataset consists of a number of small image patches along with their ground-truth segmentations done by human subjects. We provided evaluation methods for both boundary-based and region-based performance of algorithms. Boundary-based evaluation measures how precisely and completely the ground-truth boundaries are detected, and the region-based evaluation measures how similar the algorithm's partitioning, i.e. segmentation, is to the ground-truth partitioning. We used the benchmark to quantitatively compare the performance of our algorithm to those of existing, popular approaches, and showed that our algorithm outperformed others in both measures.

CHAPTER 4

SEGMENTATION-BASED STATISTICS OF NATURAL IMAGES

4.1 Introduction

This chapter is about investigating the statistics of low-level multiscale segmentation of natural images. We compile and use the segmentation statistics from a large number of images, and we present a Markov random field based model for estimating them. Our statistics confirm some of the previous findings included in the past work, and they also provide some new findings, particular to segmentation. Such statistics capture geometric and topological properties of images. To demonstrate the value of the statistics, we use them as priors in two applications: image classification and semantic image segmentation.

A natural image is far from being a random configuration of pixels. Rather, it exhibits a high degree of organization, in its spectral and photometric properties, geometry and layout. The work on natural image statistics attempts to derive probabilistic models of image characteristics. Interestingly, the question of what makes an image natural does not have a trivial answer. Perhaps the most widely accepted definition is that an image is natural if it has a similar statistical structure to which the human visual system is adapted [28].

Natural image statistics have received significant attention in recent years. Models developed for it have been used in numerous applications such as denoising [29, 30], inpainting [29], scene categorization [31], contextual priming for object detection [32], sensory processing in biology [33], saliency detection [34] and texture synthesis [35] among many others [36].

4.1.1 Related work

Previous work on natural image statistics can be grouped in two broad categories. The first group analyze natural images for the purpose of obtaining some characteristic properties, laws and/or invariance properties. Some example findings of these works are the following: (1) the image spectra obey a power law (see [33, 31, 36] for a list of references), (2) scale invariance of statistics [33, 37, 36], (3) responses of image patches to certain filters follow non-Gaussian, highly kurtotic, heavy-tailed distributions [38, 29, 36], (4) edges are dominantly oriented either horizontally or vertically [39].

The second group is targeted at developing models of natural image statistics for use in various image processing and analysis applications. Due to space limitations, we only mention some of the recent work here. A widely adopted model for natural image statistics is the Field of Experts (FoE) model [40, 29] proposed by Roth and Black. FoE learns a set of linear filters whose responses are modeled by heavy-tailed, kurtotic distributions within a high-order Markov random field model. They successfully demonstrate the capability of FoE in denoising and inpainting applications. Weiss and Freeman proposed a more efficient learning algorithm for the FoE model in [30]. FoE has been further improved by Heess et al. to allow for bimodal potentials and they used their model for texture synthesis. In [41], Cho et al. questioned the heavy-tailed distribution assumption and they proposed a new image prior which adapts itself to the content, i.e. the type of underlying texture. Torralba exploited statistics of natural images for scene categorization [31] and for contextual priming for object detection, i.e. learning priors on possible locations and sizes of objects. Finally, we refer the reader to [36] for a list of other applications of natural image statistics.

The two categories above have one thing in common: all the works perform analyses that are pixel, patch or subband-based. To the best of our knowledge, none of the methods has investigated statistics of properties that capture geometric and topological structure of images, e.g., as defined by image segments present in an image segmentation. This work is aimed at computing such statistics of natural images.

It has been shown that low level segmentation serves as a useful seed for simultaneous discovery, modeling, recognition and segmentation of objects

in images [42]. Therefore, the performance of the segmentation based algorithms for recognition, etc., may be improved by using segmentation related statistics of natural images, e.g., as priors in Bayesian framework. For a variety of reasons, including the dependence of the available segmentation algorithms on user provided parameters, segmentation statistics do not appear to have been obtained for natural images. We use the segmentation algorithm we presented in Chapter 2 (also in [43]), which does not require major prompting by the user. To the best of our knowledge, this is the first attempt to obtain and model the statistics of natural images based on low-level image segmentation.

4.1.2 Overview of our approach

In this chapter, we compile as well as use the segmentation statistics from a large number of natural images. First, we present the statistics we choose to estimate and discuss why we do so. Next, we present a statistical model for learning these statistics. Since our low level segmentation is represented by a graph, our estimation procedure involves estimating the statistics of selected properties of this graph. We use Markov random field (MRF) based modeling for this purpose. The statistics we have estimated confirm some of the previous findings, included in the past work (e.g. dominant orientations are horizontal and vertical). They also provide new findings, particular to segmentation, and therefore not included in the previous work (e.g. number of regions versus photometric scale follows an exponential distribution, and there are more regions in the bottom halves of the images than there are in the upper halves).

To demonstrate the value of the statistics, we next make their use as priors in two applications: image classification and semantic image segmentation. It is expected that, as usual, the use of priors would improve the performance of the algorithms. This expectation is borne out by our experimental results in both cases.

We use the low-level multi-scale segmentation algorithm of Chapter 2 in our experiments. The reason we specifically chose this algorithm is that it (i) does not require any major user supplied parameters while it provides all

segmentations organized in a hierarchy, and (ii) it was shown to outperform other available algorithms in a low-level segmentation benchmark (Chapter 3).

There are two main contributions of this work: (1) we present statistics of low-level segmentation of natural images for the first time, (2) we present a statistical model to learn these statistics, and demonstrate the use of this model in two applications.

4.2 Models and Statistics of Natural Images

In this section, we first give a brief description of the segmentation graph and the properties that we use to describe it. Next, we provide some statistics of these properties over a large number of natural images. Finally, we present the proposed Markov random field (MRF) based model, and describe how to learn it from a set of given images.

4.2.1 Segmentation graph and region properties

Our segmentation algorithm partitions a given image into homogeneous regions of a priori unknown shape, size and degree of photometric homogeneity. The algorithm organizes all detected regions into a hierarchical structure called the segmentation tree which captures the low-level, spatial, and photometric image structure in a hierarchical manner. Nodes at upper levels correspond to larger segments, while their children nodes capture smaller embedded details. An example segmentation tree is given in Figure 4.1. Note that the dashed edges are not originally part of the tree. We introduce these lateral edges between sibling pairs and between the regions that are adjacent (i.e. they share a common boundary fragment) to facilitate the description of each node in the tree. We call this new structure the segmentation graph. This graph is different from the connected segmentation tree of [44] where lateral edges are introduced only between siblings using a Voronoi neighborhood system.

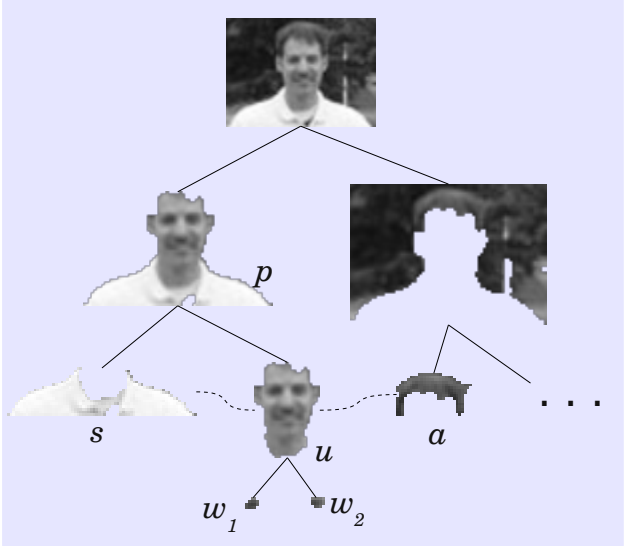


Figure 4.1: A sample segmentation tree. If the region of interest is u, then p is the parent, s is the sibling, w_1 and w_2 are the children and a is an adjacent neighbor. Dashed edges between siblings and adjacent nodes are not originally part of the segmentation tree. We add them to obtain the segmentation graph.

We describe each node, i.e. region, in the segmentation graph by a set of its intrinsic photometric and geometric, as well as relative properties. A property is intrinsic if it can be computed using only the region itself. Relative properties, on the other hand, describe how the region is related to its hierarchical and lateral neighbors in the segmentation graph.

Intrinsic properties The intrinsic geometric properties we use are the following: (1) area (normalized by the image area), (2) squared perimeter over area (indicates how compact the region is, the most compact shape is a disk), (3) eccentricity, i.e. scalar that specifies the eccentricity of the ellipse that has the same second-moments as the region, (4) the first four Hu moment invariants, (5) orientation, (6) location of the center of mass w.r.t. the image coordinates, (7) perimeter, (8) solidity, i.e. the proportion of the pixels in the convex hull that are also in the region, (9) extent, i.e. the ratio of pixels in the region to pixels in the bounding box of the region, (10) major and minor axes lengths normalized by image perimeter. (11) We further describe the shape of the region by pyramid histogram of oriented gradients (PHOG) where we evenly divide the bounding box of the region into 1, 4, and 16 cells,

thereby obtaining 3 levels, and extract an 8-dimensional gradient orientation histogram from each cell. Apart from these properties, each region is also associated with a photometric scale which is the contrast level at which it was detected.

Intrinsic photometric properties of a region are the mean and standard deviation of its grayscale intensities. If the image is colored, then we use mean and standard deviation of each of the RGB channels.

Relative properties (1) Number of children, (2) outerring area, i.e. area of the region divided by its total combined area with its children, (3) mean and standard deviation of the contrast along its boundary, (4) normalized area, i.e. area divided by the parent's area, (5) location of the center of mass w.r.t. the parent's coordinate system, (6) area dispersion of the children, (7) sibling-context vector, i.e. a vector which records the general direction in which the region sees its siblings. We refer the reader to [42] for the details of the last four properties (4-7).

Each region in the segmentation graph is represented by a vector which is the concatenation of the properties listed above.

We justify the selection of the properties above by prior empirical results in the literature. Different combinations of these properties have been shown to be effective in a variety of problems [45, 46, 42].

4.2.2 Statistics of selected properties

We randomly collected a set of 2000 images.[1] We segmented each image and obtained its segmentation graph along with the property vectors described in the previous section. Below we provide sample statistics (in the form of histograms) of the following selected properties: (1) location of center of mass, (2) orientation, (3) number of regions versus the photometric scale, (4) area, and (5) depth and average branching factor of the segmentation tree. We chose these properties because we believe that their statistics reveal some interesting information about the images.

[1] 600 images from the "Flickr random image generator" website (http://beesbuzz.biz/crap/flrig.cgi) and 1400 images from PASCAL VOC 2010 dataset.

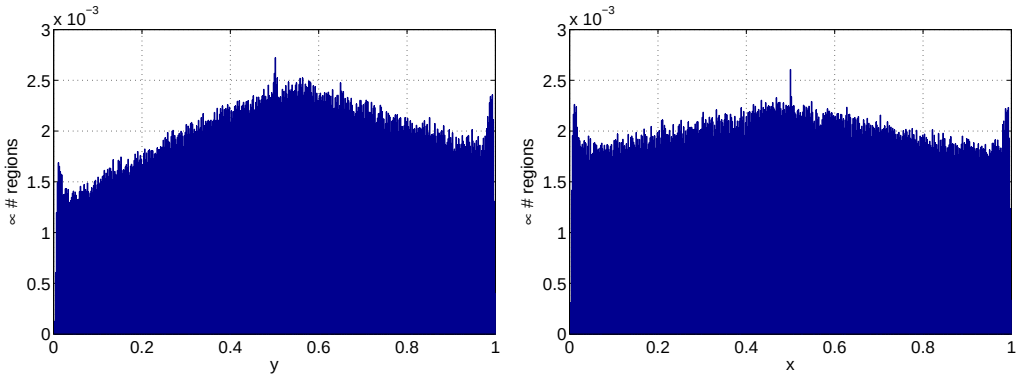


Figure 4.2: Histograms of xy-coordinates of center-of-mass of regions.

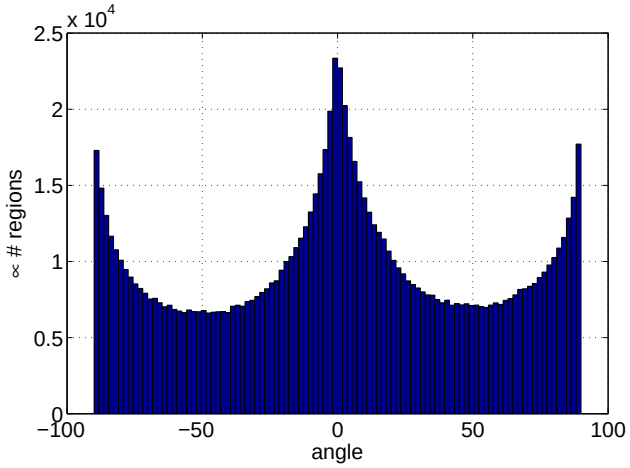


Figure 4.3: Histograms of orientations.

Location of center of mass Figure 4.2 gives the histograms of row (y) and column (x) coordinates of the center of mass of regions. Row and column values were normalized by the height and the width of images, respectively. Top left corner of the image is assumed to be $(0, 0)$ and bottom-right is $(1, 1)$. The histograms suggest that there are more regions around the center of the image than the surround. The column distribution is close to uniform and there seems to be no bias for either the left or the right half of the image. On the other hand, the row values seems to be biased towards the $[0.5, 1]$ interval, which means there are more regions in the bottom halves of the images than there are in the upper halves.

Orientation Figure 4.3 gives the histogram of orientation angles (in the range of $[-90°, 90°]$ degrees) of regions. The shape of the histogram suggests

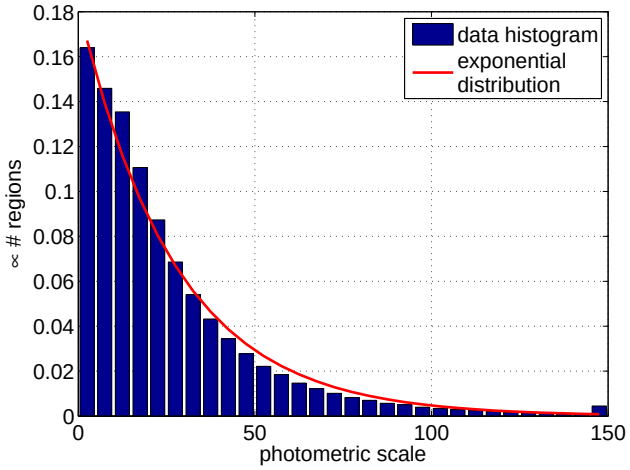


Figure 4.4: Histogram of the photometric scales of regions.

that there are two dominant orientations: horizontal (0°), and vertical (−90° and 90°). This is a confirmation of the findings in [39] where Sobel filters were used to obtain orientation statistics of edges. Another interesting observation is that the number of horizontal regions is slightly more than the number of vertical regions.

Photometric scale Figure 4.4 shows the histogram of the photometric scale of the regions. We know that, by design, the segmentation algorithm produces fewer and fewer regions as the photometric scale is increased. However, the histogram in Figure 4.4 gives us more information: the number of regions versus the photometric scale follow an exponential distribution (with $\lambda = 27.29$).

Area Figure 4.5 shows the histogram of region areas.[2] As the histogram suggests, most of the regions are small compared to the whole image area. In fact, the percentage of regions which have area less than 5% of the image is 98.81%. Region areas seem to follow a generalized extreme value distribution (shown in red), with parameters $\mu = 0.12, \sigma = 0.02, K = 0.5992$.

Depth and avg. branching factor In Figure 4.6, we give the histograms of the depth and the average branching factor (ABF) of the segmentation

[2]For better visualization, we exponentially scaled the area values by an exponent of 0.25.

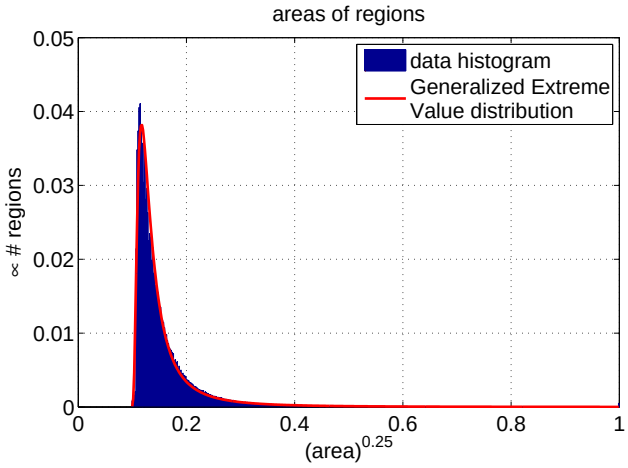


Figure 4.5: Histogram of areas of regions.

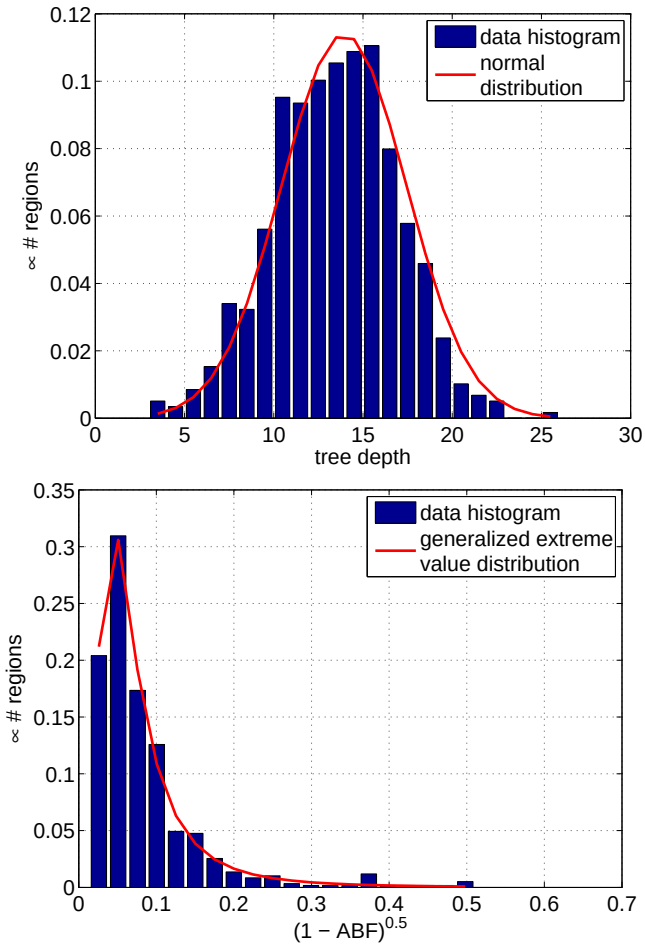


Figure 4.6: Histograms of depth and average branching factor (ABF) of the segmentation trees.

trees. The depth values follow a normal distribution with $\mu = 13.94$ and $\sigma = 3.5$. To better visualize the ABF values, we transformed them using $\sqrt{1 - ABF}$. In this transformed space, ABF values follow a generalized extreme value distribution with parameters $\mu = 0.05$, $\sigma = 0.03$, $K = 0.36$. When the leaf nodes, which have no children, are not counted, ABF is 2.1. When the leaves are also considered ABF slightly drops down under 1.

4.2.3 MRF modeling of the segmentation graph

Let S be the segmentation graph of an image I. The central question in modeling natural image statistics is to develop a model for $p(I)$, i.e. the probability of being an image. In this work, we propose to develop this model based on the segmentation graph properties of the image; that is we assume the probability I is equivalent to the probability of observing its segmentation graph S, i.e. $p(S) \simeq p(I)$.

We consider S as an undirected graph whose nodes correspond to image regions and the edges between the nodes represent parent-child, sibling or adjacency relations. Each node is associated with a vector of properties which depends only on that node and its immediate (i.e. first order) neighbors. Therefore, S is Markovian.

We assume that a node together with its immediate neighbors (parent, children, siblings, adjacent neighbors) form a maximal clique of the segmentation graph.[3] Suppose S has K nodes and let $\boldsymbol{r}_k$ denote the property vector extracted from the clique centered at the k^{th} region. Then, our proposed model is:

$$P(S;\Theta) = \frac{1}{Z(\Theta)} \prod_{k=1}^{K} \psi(r_k;\Theta) \tag{4.1}$$

where we parametrized $p(S)$ with our model parameters Θ and $\psi(\cdot)$ is the clique-potential function, and $Z(\cdot)$ is a normalization factor called the partition function. Using the Markov-Gibbs equivalence, we can write (4.1) as:

[3]Similarly, in the Fields-of-Experts model, a square patch centered on a pixel is assumed to define a maximal clique.

$$p(S;\Theta) = \frac{1}{Z(\Theta)} e^{-E(S;\Theta)} \tag{4.2}$$

where

$$Z(\Theta) = \int_V e^{-E(s;\Theta)} ds \tag{4.3}$$

and V is the space of all possible segmentations and $E(\cdot)$ is the energy function which is defined as:

$$E(S;\Theta) = \sum_{\boldsymbol{r} \in S} U(\boldsymbol{r};\Theta). \tag{4.4}$$

$U(\cdot)$ is called the log-potential function; in fact, we have $U(\cdot) = ln\ \psi(\cdot)$. We discuss the form $U(\cdot)$ in Section 4.2.3 below.

Estimating the parameters, Θ, of the model

It is well known that maximum likelihood (ML) estimation of Θ is intractable because of the partition function $Z(\Theta)$. A simple approximate scheme to ML estimation is the pseudo-likelihood estimation which is an asymptotically consistent estimator of ML [47].

Pseudo-likelihood ($\mathcal{PL}$) approximation is based on the conditional probability of a node:

$$p(\boldsymbol{r}_k | \boldsymbol{r}_j \in \mathcal{N}_{\boldsymbol{r}_k}) = \frac{e^{-U(r_k;\Theta)}}{\int_\Omega e^{-U(r;\Theta)} dr} \tag{4.5}$$

where $\mathcal{N}_{\boldsymbol{r}_k}$ denotes the set of immediate neighbors of $\boldsymbol{r}_k$. $\mathcal{PL}$ is defined as:

$$\mathcal{PL}(\Theta) = \prod_{k=1}^{K} p(\boldsymbol{r}_k | \boldsymbol{r}_j \in \mathcal{N}_{\boldsymbol{r}_k}) = \prod_{k=1}^{K} \frac{e^{-U(r_k;\Theta)}}{\int_\Omega e^{-U(r;\Theta)} dr}. \tag{4.6}$$

One way to maximize $\mathcal{PL}$ is by gradient ascent where the gradient is:

$$\frac{\partial ln(\mathcal{PL}(\Theta))}{\partial \Theta} = K \frac{\int_\Omega e^{-U(r;\Theta)} \frac{\partial U(\boldsymbol{r};\Theta)}{\partial \Theta} dr}{\int_\Omega e^{-U(r;\Theta)} dr} - \sum_{k=1}^{K} \frac{\partial U(\boldsymbol{r}_k;\Theta)}{\partial \Theta}. \tag{4.7}$$

Note the integrals in the first term above. Depending on the dimensionality of the property vectors $\boldsymbol{r}$, numerical integration techniques ranging from simple quadrature methods to Monte Carlo methods could be used.

Other than gradient ascent, a recently popular method to optimize Eq. (4.2) is the contrastive-divergence method which has been used in the FoE models. We do not use this method because it would require us to draw samples from V, the space of all possible segmentations.

Potential functions

The criterion for a good potential function, $U(\cdot)$, is such that it should be minimized when it describes a clique, i.e. a node and its immediate neighbors as completely as possible. That is, $U(\boldsymbol{r};\Theta)$ should be minimum, if the property vector $\boldsymbol{r}$ is describing a valid region together with its related nodes. A natural choice would be to use a negated probability density function (pdf). We note that when $U(\boldsymbol{r};\Theta)$ is in the form

$$U(\boldsymbol{r}) = -\ln(q(\boldsymbol{r})) \tag{4.8}$$

where $q(\cdot)$ is a pdf, the maximization $\mathcal{PL}$ becomes much easier due to the fact that the denominator in (4.5) always evaluates to 1 since $q(\cdot)$ is a pdf. This saves a lot of computation because we no longer need the first term and its (possibly) high-dimensional integrals in the gradient (4.7).

Following the observation above, we choose $q(\cdot)$ to be a Gaussian mixture model (GMM) due to its generality and some empirical evidence that GMMs model region properties well in certain tasks [46].

4.3 Experiments

In this section, we demonstrate the use of our proposed model in two different applications: image classification, and semantic segmentation. In the first application, we use our model as a generative prior from which we extract representative vectors to be used in classification. We present image classification results on the PASCAL VOC 2007 dataset. In the second application, we show how our model can be used as class-conditioned image priors to improve the performance of a simple semantic segmentation algorithm.

4.3.1 Image classification: PASCAL VOC 2007

We make use of our proposed model for image classification using the Fisher kernel approach [48, 49]. Let S_1 and S_2 be two different segmentation graphs which are generated by the prior $p(S;\Theta)$. Then, a representative feature vector for S_1 is:

$$\boldsymbol{f}_1 = \nabla_\Theta \log p(S_1;\Theta). \tag{4.9}$$

Feature vector $\boldsymbol{f}_1$ represents S_1 because the gradient of the prior model evaluated at S_1 describes the directions in which the model parameters should be changed to best fit S_1. In fact, $\boldsymbol{f}_1$ is a sufficient statistics for S_1 [48]. Similarity between between S_1 and S_2 is given by the kernel:

$$K(S_1, S_2) = \boldsymbol{f}_1^T F^{-1} \boldsymbol{f}_2, \tag{4.10}$$

where F is the Fisher information matrix of $p(S;\Theta)$. Appropriately scaled $\boldsymbol{f}_1$ (i.e. $F^{-\frac{1}{2}}\boldsymbol{f}_1$) is called the Fisher vector representing S_1 (see [49, 50] for further details). Note that the size of the Fisher vector depends only on the number of parameters of the model $p(S;\Theta)$ and not on the size, i.e. the number of nodes, of S_1. Efficient linear classifiers can be trained on Fisher vectors.

We used the PASCAL VOC 2007 dataset which contains about ten thousand images of 20 object classes. We followed the standard practice and used

the "trainval" set to train (and tune) our system, and tested on the "test" set. The peformance is measured by average precision (AP) over 20 classes.

In our experiments, we used all the features described in Section 4.2.1 with the exception that we compressed PHOG features using PCA due to the high dimensionality of the PHOG vectors (21 cells, each 8 dimensional, hence 168 dimension in total). Instead of fixing the reduced number of PHOG dimensions, we allowed it to vary as a parameter of the system. In addition to this parameter, the only parameter of our model is the number of Gaussian components in the potential function (Eq (4.8)). We tuned these two parameters by cross-validation, i.e. training on the "train" set and validating on the "val" set.

First, we trained a single universal prior (as described in Section 4.2.3) using all the images of all classes. Then, we extracted the Fisher vectors from this prior and trained support vector machine (SVM) classifiers [51] with linear kernels. We also tried the sigmoid kernel and it performed better, so all the following results we report are obtained using the sigmoid kernel.[4] We call this universal prior model as SS-U, short for "universal segmentation statistics."

While training SS-U, we observed that the optimal parameters, i.e. # of Gaussian components, and the reduced dimensionality of PHOG, are different for different classes. An example for this situation is given in Figure 4.7 where, on one hand, the "bus" class enjoys high-dimensional PHOG vectors (around 100) with a small number of Gaussian components (around 40), the "dog" class gives better AP when the PHOG-dimensionality is low (around 40) and the number of Gaussian components is high (around 90). Following this observation, we trained another system where we learned a separate prior model for each class. We call this system SS-PC, short for "segmentation statistics per class."

The average APs obtained by SS-U and SS-PC are shown in Figure 4.8, together with the best 5 and the worst 2 methods competed in PASCAL

[4]Note that by using the sigmoid-kernel, we are not invalidating Eq. (4.10). The sigmoid kernel, $tanh(gu'v + r)$, still uses the dot product to compute the similarity between u and v but then scales the result using a sigmoid function with parameters g, and r.

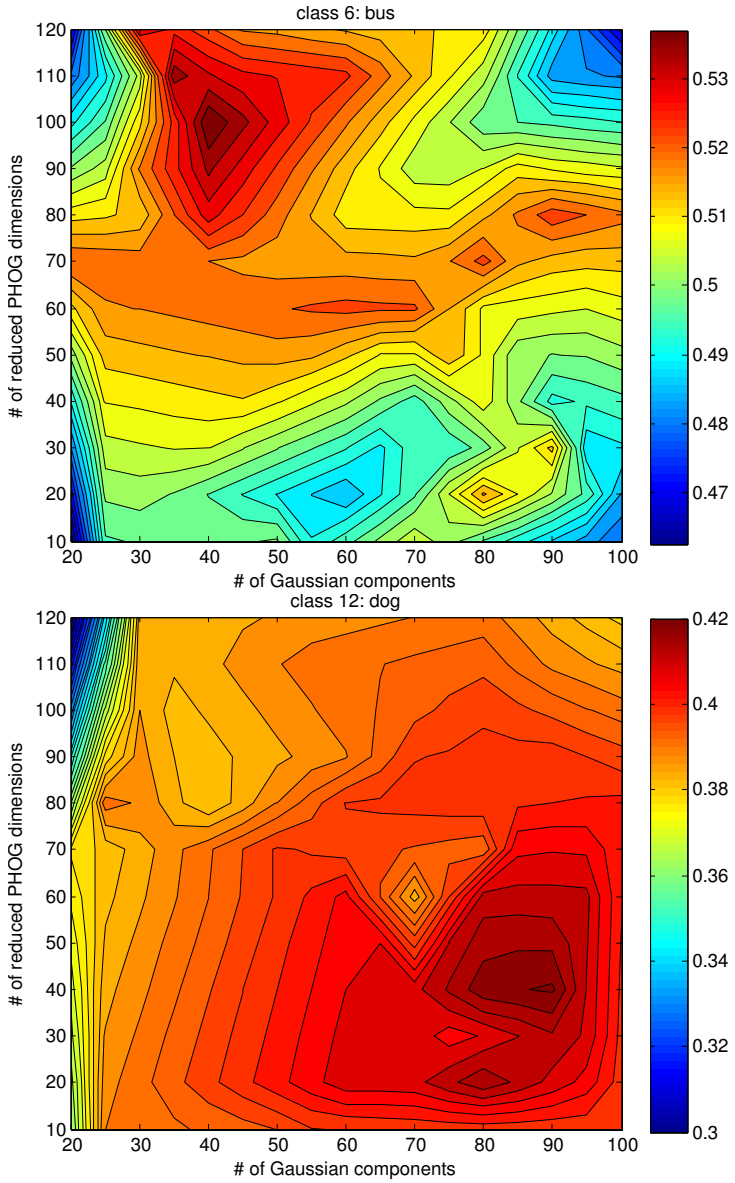


Figure 4.7: Cross-validation results for "bus" and "dog" classes. These classes prefer different quantities for the # of Gaussian components parameter and the PHOG-dimensionality parameter. The color represents the AP score obtained.

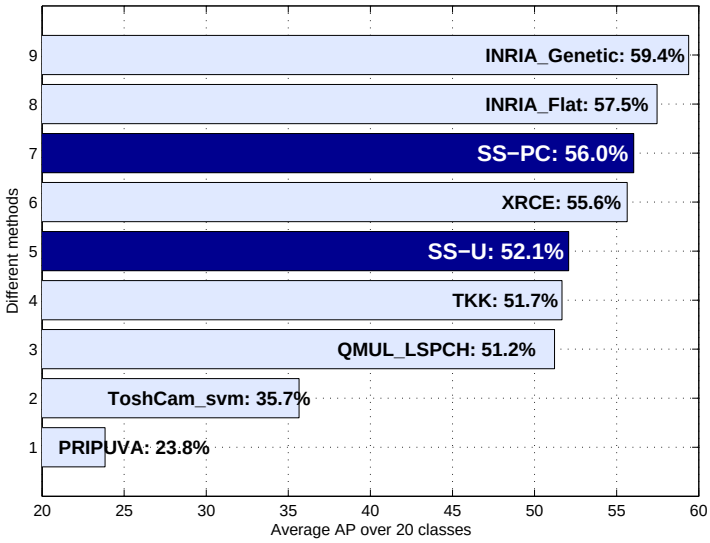


Figure 4.8: The performances (as average AP over 20 classes) of SS-U, SS-PC (dark blue bars), together with the best 5 and the worst 2 PASCAL VOC 2007 competitors (light blue bars).

VOC 2007.[5] There have been others results reported on this dataset. We present top state-of-the-art results in Table 4.1.

Although our proposed models do not perform as well as the best method available, we want to emphasize some of the advantages of our model. All the methods above either use sophisticated and costly learning algorithms or need thousands of features per image, or both. The best system, "Cls + Loc" [52] combines the best method of PASCAL VOC 2007 with a costly sliding window-based object detector. Multiple kernel learning (MKL) [53] also trains a costly learning system and uses thousands of features. The "kernel codebook" [54] and the improved Fisher kernel (IFK) [50] are comparable to our proposed model in simplicity, but we are working with fewer features (per image) than they require. On average, a typical 500x500 image gives us only a few hundred regions (the actual average over the 2000 image dataset is about 350). However, [50] extracts features on regular grids (every 16 pixels) at five scales, i.e. around 5000 features. Given that our learning machinery (Fisher kernel + SVM) is very similar to that of [50], the fact that we are getting comparable results suggests that image regions are richer, more descriptive features than the rectangular patches extracted from regular grids.

[5]Refer to the challenge's website at http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/results/index.shtml for the abbreviations of method names.

Table 4.1: Comparison of the proposed models SS-U and SS-PC with the state-of-the-art on PASCAL VOC 2007.

Method	AP (in %)
Cls + Loc [52]	63.5
MKL [53]	62.2
Kernel Codebook [54]	60.5
Best of VOC07 [3]	59.4
IFK (SIFT) [50]	58.3
SS_PC	56.0
SS_U	52.1
Standard FK [50]	47.9

Another difference worth mentioning is that we do not make use of the "spatial pyramid" which has become a general trend in image classification (e.g. [50]). We store the location information within the feature vector as the location of the center of mass of regions. Note that this is a simpler approach than the spatial pyramid where one needs to train separate learners per "spatial cell."

4.3.2 Semantic segmentation: MSRC-21 dataset

In our second experiment, we show how to use learned image priors to help improve semantic segmentation of images. To this end, we use the MSRC-21 dataset which contains 591 images of 21 classes. The task is to label every pixel of the test images with one of these 21 labels. We follow the practice of [4] and randomly split the set into 276 training and 315 testing images.

Our semantic segmentation model is as follows. Suppose that we have a classifier which can predict the semantic label of a region. To label a pixel, we first classify the regions that contain it (note that a certain pixel might be included in more than one region, because of the segmentation hierarchy). The classifier returns its predictions along with confidence scores, or probabilities. Then, we choose the label with the maximum confidence and assign it to the pixel in question. We use SVM with RBF kernel as our region classifier which is trained on all the regions of the training images.

Now, let us look at how image priors help this process. The region classifier

Table 4.2: Results and comparison on the MSRC-21 dataset.

Method	building	grass	tree	cow	sheep	sky	aeroplane	water	face	car	bicycle	flower
no prior	64.3	85.6	72.3	53.7	51.0	69.8	46.0	65.8	68.6	47.7	70.0	68.6
with prior	73.8	92.6	83.5	61.0	59.5	79.4	55.0	74.5	75.4	62.3	74.6	81.4

Method	sign	bird	book	chair	road	cat	dog	body	boat	**average**
no prior	62.6	25.4	64.9	16.0	67.2	44.3	31.8	38.9	7.3	53.4
with prior	70.9	35.8	71.8	25.0	74.5	51.6	39.1	49.0	10.3	62.0

is actually trained to compute the probability $p(c|r)$ where c is a class variable and r represents a region, i.e. its properties. However, we are also given the images that contain these training regions. In fact, we can learn $p(c|r, I)$[6] instead of just $p(c|r)$, where I is the image that contains region r. If we write out

$$p(c|r, I) = \frac{p(r, I|c)p(c)}{p(r, I)} = \frac{p(r|c)p(I|c)p(c)}{p(r, I)} \tag{4.11}$$

where we assumed conditional independence of r and I given the class c, then

$$p(c|r, I) \propto p(c|r)p(I|c). \tag{4.12}$$

The first term on the right-hand side above is the region classifier and the second term is the class-conditional image prior. We train these priors as described in Section 4.2.3. A separate prior is trained per class using the images that contain objects of that class.

We give per-class and average segmentation accuracies on the MSRC-21 dataset in Table 4.2. Note that the model with the prior (Eq. (4.12)) improves the average result by 8.6%.

[6]This must actually be $p(c|r, I\backslash r)$ but we can assume $I\backslash r \approx I$ as the average region size is small compared to the size of the image.

4.3.3 Scene classification

In this section, we propose another way of using the segmentation statistics of images. Although fundemantally the same as the Fisher kernel idea in the sense that the prior is used to generate representative feature vectors, the method described here is sufficiently different in the techniques it uses to deserve its own section.

To classify an image using segmentation based features, we model the image in terms of a probability density function, a Gaussian mixture model (GMM) to be specific, of its region features. This GMM is fit to the image by adapting a universal GMM which is estimated so it fits all images. Adaptation is done using a maximum-aposteriori criterion. We use kernelized versions of Bhattacharyya distance to measure the similarity between two GMMs and support vector machines to perform classification. We outperform previously reported results on a publicly available scene classification dataset. These results suggest further experimentation in evaluating the promise of low level segmentation in scene classification.

Introduction The most commonly used image representations for scene classification are in terms of properties of image patches. Rectangular or circular patches are sampled densely along a regular grid overlaid onto the image or at points detected by an interest point detector. These patches are described in various ways, including in terms of normalized intensity values, SIFT features, color and texture histograms, or filter responses [55, 56, 57, 58, 59]. These descriptions then serve as the basis for class representations. They are either directly fed into discriminative algorithms [55] such as support vector machines (SVM), or first, generatively modeled by bag-of-words models [56, 59], probabilistic graphical models [56], or latent topic models [58, 59], followed by learning of some discriminative aspects of these generative models using a classifier.

As an alternative to the patch-, grid-, and filter-based representations above, in this work we use features derived from a low-level segmentation of the image. We use the multiscale segmentation algorithm described in Chapter 2, which is designed to detect image regions regardless of their shape and size, spatial distribution, and contrast. The algorithm organizes all detected

regions hierarchically into a tree data structure where the root node corresponds to the whole image. Nodes closer to the root correspond to larger regions, while their children nodes capture embedded details. Our representation consists of intrinsic properties of the image regions (capturing region geometry and its photometric appearance), as well as properties of their mutual embedding properties which are captured in the tree. Together the two sets of properties constitute our feature space whose capabilities we wish to evaluate via semantic scene classification.

Overview of Our Approach Our approach consists of the following major steps. We first obtain parametric models of the aforementioned two sets of properties of image segmentation, to represent them concisely. We achieve this by fitting a Gaussian mixture model (GMM) to the region properties observed within an image. This avoids the need for choosing a specific vocabulary per image and selecting the number of histogram bins or sizes associated with the widely used bag-of-words model or histogramming methods. Another advantage is that by utilizing kernel functions which measure the similarity between GMMs we can let the SVMs directly exploit the generative models in a discriminative setting.

However, the usual problem with such estimation of high-dimensional probability density functions (pdf), in this case a GMM, is the scarcity of the training samples. One approach to overcoming this problem is to first capture the gross distributional characteristics of the entire set of samples, e.g., via a universal GMM. This approach is popular in speech processing [60, 61], and is also now becoming popular in computer vision [62, 63]. We adapt the universal GMM to each image using a maximum-aposteriori (MAP) criterion. The goal of this adaptation is to maximize the posterior probability of the parameters of the image-generative model (GMM) given the universal GMM and the new data, i.e. regions of the image to be modeled. This is accomplished using an expectation-maximization (EM) procedure [60].

Once each image is modeled by a GMM, we use SVMs for classification. For SVMs to work on these GMMs, we need a kernel function which measures the similarity between two GMMs. We use kernelized versions of the Bhattacharyya distance [64] for this purpose.

Image Representation We represent an image by the list of its regions obtained by a low-level multiscale segmentation algorithm [43]. Each region is described by the following 20 features: (1) area, (2) mean intensity, (3) standard deviation of the intensity, (4) (perimeter)^2 / area, (5) outer-ring area, i.e. the area of the region except its children, (6) orientation, (7) eccentricity, (8) solidity, i.e. the proportion of the pixels in the convex hull that are also in the region, (9-12) the first four central moments, (13) mean contrast of the boundary, (14) standard deviation of the boundary contrast, (15-16) x-y coordinates of the center of mass expressed in the image coordinate system, (17) perimeter, (18) extent, i.e. the ratio of pixels in the region to pixels in the total bounding box, (19) major axis length, (20) minor axis length. As mentioned earlier, these features capture different aspects of image segmentation, including intrinsic geometric (1,4,6-12, 15-20) and photometric properties (2,3) of the regions, their relative properties (13, 14), and a topological property (5). More diversity in the selection of these features is possible, and will be a part of our future work which will be guided by the results obtained in this study.

Images as adapted GMMs We want to model an image by a GMM of its region properties. However, as mentioned above, robustly fitting a GMM to a small number of regions (some images have only 30-40 regions) is a problem. To overcome this, we employ the MAP adaptation (or Bayesian adaptation). A previously trained *universal GMM* is used as a prior mixture and adapted to the image that we want to model. Then, the image is represented by this adapted GMM. For this purpose, we train a universal GMM using all the regions of all training images by using EM. We denote this universal GMM by its parameters $\Theta^u = \{c_i^u, \mu_i^u, \Sigma_i^u\}_{i=1}^{N^u}$, where c_i^u is the mixture coefficient, μ_i^u is the mean, and Σ_i^u is the covariance matrix of the i^{th} Gaussian. N is the number of Gaussian components in the mixture.

MAP adaptation Once the universal GMM is trained, we adapt it to the regions of each image that we want to model. Let this image be I and let it have R regions: $I = \{r_i | i = 1, 2, \ldots, R\}$. Recall that r_i is a 20-dimensional vector. For completeness, we provide the equations for the MAP adaption here (see [60] or [62] for details). The adaptation is achieved by an EM

procedure.

In the E-step, occupancy probabilities, i.e. the probability that an observed region r is generated by the i^{th} Gaussian component, are estimated:

$$w_i(r) = \frac{c_i p_i(r|\Theta)}{\sum_{j=1}^{N} c_j p_j(r|\Theta)} \tag{4.13}$$

where

$$p_i(r|\Theta) = \frac{exp\{-\frac{1}{2}(r-\mu_i)'\Sigma_i^{-1}(r-\mu_i)\}}{(2\pi)^{(D/2)}\sqrt{|\Sigma_i|}}. \tag{4.14}$$

In the M-step, the parameters of the GMM are re-estimated (adapted):

$$\hat{c}_i = \frac{\sum_{j=1}^{R} w_i(r_j) + \tau}{R + N \cdot \tau} \tag{4.15}$$

$$\hat{\mu}_i = \frac{\sum_{j=1}^{R} w_i(r_j) r_j + \tau \mu_i^u}{\sum_{j=1}^{R} w_i(r_j) + \tau} \tag{4.16}$$

$$\hat{\Sigma}_i = \frac{\sum_{j=1}^{R} w_i(r_j) r_j r_j' + \tau(\Sigma_i^u + \mu_i^u \mu_i'^u)}{\sum_{j=1}^{R} w_i(r_j) + \tau} - \hat{\mu}_i \hat{\mu}_i' \tag{4.17}$$

Here the parameter τ is called *the relevance factor* and it regulates the balance between the universal GMM and the new data, i.e. the image that we want to model. In our experiments, we choose the value of τ using cross-validation.

Classification We use SVMs for classification. As each image is represented by a GMM (adapted from the universal GMM), we need a similarity function between two given GMMs. For the special case of $N = 1$, we use the Bhattacharyya kernel [63] which has the following closed form solution for two Gaussians p and q:

$$\begin{aligned} &K(p,q) = |\Sigma|^{1/2}|\Sigma_p|^{-1/4}|\Sigma_q|^{-1/4} \\ &\exp\Big(-\tfrac{1}{4}\mu_p'\Sigma_p^{-1}\mu_p - \tfrac{1}{4}\mu_q'\Sigma_q^{-1}\mu_q + \tfrac{1}{2}\mu'\Sigma\mu \Big) \end{aligned} \tag{4.18}$$

where $\mu = \frac{1}{2}(\Sigma_p^{-1}\mu_p + \Sigma_q^{-1})^{-1}\mu_q)$ and $\Sigma = (\frac{1}{2}\Sigma_p^{-1} + \frac{1}{2}\Sigma_q^{-1})^{-1}$.

For the case of $N > 1$, there is no exact solution. For this, various approximations have been proposed [63] in the literature. We use the following approximation from [65] which uses the fact that there is one-to-one correspondence between the Gaussians of the universal model and the adapted model. For two GMMs p and q:

$$\begin{aligned} K(p,q) \approx \sum_{i=1}^{N} \Bigg\{ \Bigg[\frac{1}{2} \Bigg(\frac{\Sigma_i^p + \Sigma_i^u}{2} \Bigg)^{-\frac{1}{2}} (\mu_i^p - \mu_i^u) \Bigg]^T \\ \Bigg[\frac{1}{2} \Bigg(\frac{\Sigma_i^q + \Sigma_i^u}{2} \Bigg)^{-\frac{1}{2}} (\mu_i^q - \mu_i^u) \Bigg] \Bigg\} + \\ \sum_{i=1}^{N} tr \Bigg[\Bigg(\frac{\Sigma_i^p + \Sigma_i^u}{2} \Bigg)^{\frac{1}{2}} (\Sigma_i^p)^{-\frac{1}{2}} \Bigg(\frac{\Sigma_i^q + \Sigma_i^u}{2} \Bigg)^{\frac{1}{2}} (\Sigma_i^q)^{-\frac{1}{2}} \Bigg]. \end{aligned} \tag{4.19}$$

Note that the second term on the right-hand side of the equation can be written as an inner product of two vectors – since the covariance matrices are diagonal – which makes its implementation easy and its evaluation efficient.

Experiments For experimental validation, we used the publicly available dataset of Oliva and Torralba[7] [55]. This dataset contains 2688 color images organized in 8 classes: coast, forest, mountain, open country, highway, inside city, tall building, and street (see Figure 4.9 for some example). Each class has a different number of images, ranging between 292 to 410.

We compare our results with two previous methods: one that uses gist features and SVM [55] and another that uses SIFT features and a hybrid generative/discriminative method [57, 59]. In [55], the dataset is split into training and testing subsets by randomly choosing 100 images for training from each class, and using the rest for testing. Although the images are in color, the authors use the grayscale versions since gist features cannot utilize color information. In [57, 59], the authors split the dataset into two by randomly choosing half of the images per class for training, and they use the rest for testing. Results on both grayscale and color versions of the datasets are reported. In all experiments of [55, 57, 59], the classification accuracy is reported as the mean of the diagonal of the confusion matrix. However, the authors do not mention whether these reported numbers are the best results

[7] `http://people.csail.mit.edu/torralba/code/spatialenvelope/`

coast

highway

forest

street

Figure 4.9: Example images from the scene classification dataset [55].

they get over different random splits of the dataset, or the average results over a number of trials. We present our best results as well as the average results we get over 10 random splits.

For each random split of the dataset, we ran our GMM system for four different choices of the number of components N: 1, 50, 75, and 100. The relevance factor, τ, was fixed to 60, 5, 2, and 1, respectively. These values were found in our preliminary experiments by 5-fold cross-validation on the training sets.

When there was only a single component in our GMM ($N = 1$), i.e. the model was a single multidimensional Gaussian, we used a full-covariance matrix. When $N > 1$, we used diagonal-covariance matrices. There are two reasons for using diagonal-covariance matrices, both motivated by computational considerations: (1) It is easier to avoid singularities (in the covariance matrices) during the GMM training and/or adaptation, and (2) it is much more efficient in terms of time and memory.

The classification performances of the previous methods and our method

Table 4.3: Comparison of classification performances of our method and others.

version	# train per class	[55]	[57],[59]	Our best
gray	100	83.70%	-	**85.07**% ($N = 1$)
gray	50%	-	84.39% [57]	**87.38**% ($N = 1$)
color	50%	-	86.65% [57], 87.80% [59]	**87.87**% ($N = 1$)

version	# train per class	Our average ($N = 1$)	Our average ($N = 75$)
gray	100	$83.89 \pm 0.49\%$	$83.11 \pm 0.75\%$
gray	50%	$86.10 \pm 0.82\%$	$84.70 \pm 0.73\%$
color	50%	$87.13 \pm 0.53\%$	$86.01 \pm 0.72\%$

are given in Table 4.3. We outperform the previously reported results in all cases. Interestingly, in almost all experiments, the single full-covariance Gaussian model ($N = 1$) outperformed the GMM with multiple components. Very rarely was the GMM better than the single full-covariance model. We believe that this situation might be due to three reasons: (1) Since GMM ($N > 1$) uses only diagonal-covariance matrices, it only takes into account the variances of the features, and this will tend to increase the required number of components, to compensate for the covariances of features. On the other hand, all the covariance information is captured in the single Gaussian case. (2) There is no exact similarity measure between two GMMs. The approximation we used might be degrading the classification performance. (3) Although we have not tested it statistically, the region properties of a given image might be truly distributed as a single multidimensional Gaussian. Other researches also reported that single, full-covariance Gaussian models give consistently good results, and are sometimes better than a GMM [66, 67].

Figure 4.10 shows a typical confusion matrix for the "color, 50%" version of the dataset. The most confused two classes are "open country" and "coast", which is also the case in previous work [55, 57, 59]. We give a couple of misclassified images from these classes in Figure 4.11.

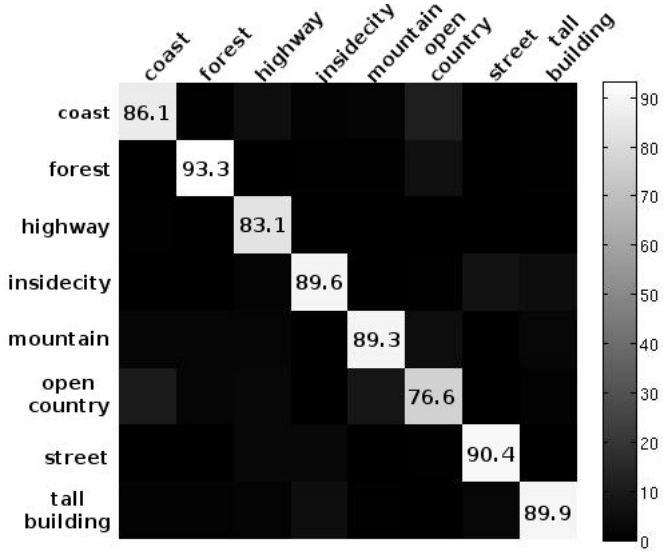


Figure 4.10: Class confusion matrix of our classifier for 8 classes.

Figure 4.11: Misclassified image examples. Images in the upper row belong to the "coast" class but they were labeled as "opencountry." Images in the lower row belong to the "opencountry" class but they were labeled as "coast."

4.4 Discussion and Summary

We presented a set of statistics based on low-level segmentation of natural images. Based on these statistics, we were able to confirm that dominant orientations in natural images are horizontal and vertical. We also provided new findings such as that the number of regions versus photometric scale follows an exponential distribution, and that there are more regions in the bottom halves of the images than there are in the upper halves. We proposed a MRF based model to learn the segmentation statistics and successfully used this model in image categorization and semantic image segmentation.

Perhaps the main limitation of our model is that it is not a truly generative model for images; i.e. the model is not able to reconstruct the image. For this reason, for a class of problems where the output itself is a natural image, e.g. denoising, inpainting, etc., it is not trivial how to use our model. On the other hand, we believe that many high-level vision problems might benefit from the model proposed in this study. We tried to demonstrate this in two different applications.

We also proposed another method of using the segmentation-based statistics of images for scene classification. To classify a scene image using segmentation based features, we model it in terms of a probability density function, a Gaussian mixture model (GMM) to be specific, of its region features. This GMM is fit to the image by adapting a universal GMM which is estimated so it fits all images. Adaptation is done using a maximum-aposteriori criterion. We use kernelized versions of Bhattacharyya distance to measure the similarity between two GMMs and support vector machines to perform classification. Our method outperformed previously reported results on a publicly available scene classification dataset.

CHAPTER 5

LEARNING TO MATCH REGIONS FROM HUMAN PERCEPTION

5.1 Introduction

The correspondence problem, that is, identifying the same physical point in images taken from different viewpoints, is still a challenging problem in computational vision. Despite the considerable effort devoted to its solution [68, 11], available correspondence algorithms fail to deal with a variety of common real world inputs effectively [69], yet it is quickly and seamlessly being solved by the human visual system without any conscious effort. The ability of visual correspondence is central to many cognitive abilities including depth perception, tracking moving objects, perceiving the 3D structure of objects, and recognizing objects and scenes among others.

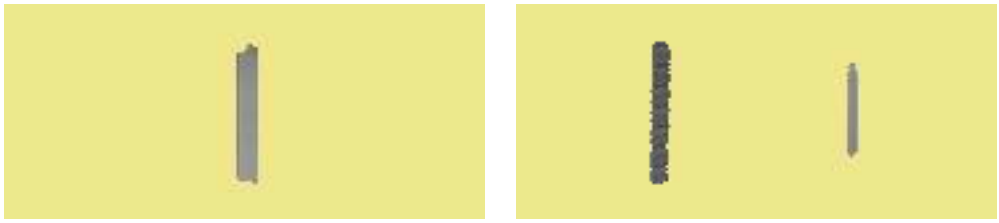
Figure 5.1: A sample stereo image pair from the Middlebury benchmark dataset [70]. Stereo correspondence is a special case of general correspondence problem where the corresponding pixels lie in the same row in both images.

The correspondence problem is an integral part of many computational vision tasks such as stereo correspondence (see Figure 5.1 for an example),

N-view correspondence, object tracking, structure from motion, 3D reconstruction and image stitching. Previous related work can roughly be categorized into two main groups: (a) correlation-based [68, 70, 71], and (b) token (i.e. structure) based methods [68, 11]. Correlation-based methods, which are the most popular correspondence algorithms today, match image sub-windows using correlation, while token-based methods rely on matches between extracted image features such as edges, corners, curves, and regions [68]. Correlation-based methods work well for images of dense texture, which are taken from relatively close viewpoints under similar illumination conditions. On the other hand, token-based methods do not rely on dense texture and are quite robust over illumination changes and large viewpoint differences. However, these methods are limited by the reliability and robustness of the token extraction algorithms, and this is the primary reason why these methods have been less popular than the correlation-based methods [68].

Due to the recent progress in automatic image segmentation [2, 72, 43], we believe that it is worth revisiting the token-based methods where tokens are regions obtained by segmentation. The segmentation algorithm we proposed in Chapter 2 is suitable for this purpose because (i) it detects all image regions regardless of their shapes, sizes and levels of interior homogeneity, and (ii) we experimentally showed that our algorithm achieves a better and more reliable segmentation than the other available algorithms (Chapter 3).

At the core of all token-based methods lies an affinity function which either measures how well two given tokens match, or decides if they match at all. In this study, tokens refer to image regions obtained by our segmentation algorithm. Regions can be described by their geometric (shape, size) and photometric (appearance) cues. In previous token-based methods, weights were assigned to different cues in an ad-hoc manner [73, 74]. From a psychological point of view, there is abundant evidence that these cues indeed affect the perceived quality of token correspondence. However, the results on how significant each cue is to the correspondence, when they are put into competition, are "far less conclusive" [75].



Frame 1 Frame 2

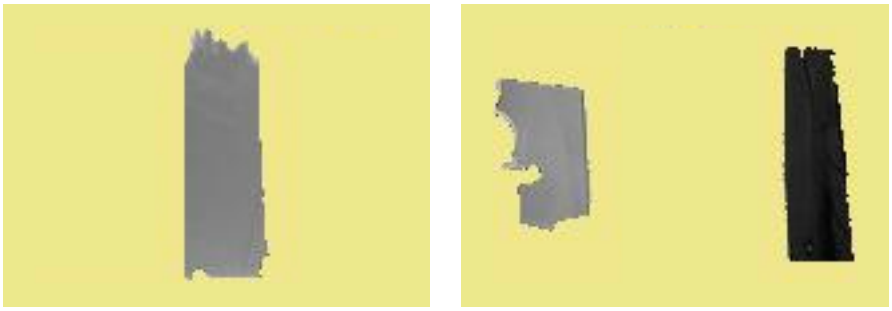
Figure 5.2: An example pair of frames for the apparent motion experiment. The region in Frame 1 is called the *target*, and the regions in Frame 2 are called the *distracters*. Each distracter is dissimilar to the target only in one cue: the left distracter is similar to the target in shape and size but not in color; the right distracter is similar to the target in shape and color but not in size.

5.2 Preliminary Experiments

For the purpose of learning how significant different visual cues are – for the region correspondence problem – from the human visual system, we designed and experimented with three different methods: (i) apparent motion, (ii) equating just-noticeable differences, and (iii) binocular fusion. In the following, we describe these methods, discuss their advantages and disadvantages, and explain why we opted to use the binocular fusion method over others.

5.2.1 Apparent motion

In this method, we let two different types of visual cue compete with each other in an apparent motion experiment. Each trial consists of two frames. In the first, we place a region, called the *target*, at the center of the frame, and in the second one we place two other regions, called the *distracters*, one to the left of the center, and the other to the right of the center. Example frames can be found in Figures 5.2 and 5.3. We show these two frames sequentially to a human subject for a very short amount of time per frame. We then ask the following question to the subject: "In which direction did the target region move, left or right?" This question puts the distracters in competition, and depending on the subject's answer, we would know that the target is visually more similar to one of the distracters than the other



Frame 1 Frame 2

Figure 5.3: Another example pair of frames for the apparent motion experiment. See the caption of Figure 5.2 for an explanation. In this particular example, the left distracter is similar to the target in color and shape but not in size; the right distracter is similar to the target in shape and size but not in color.

one.

We pick the target region randomly from a large pool of image regions[1] obtained from the segmentations of a large number of natural images. After the target is selected, we search for a distracter which is similar to the target in all cues but one. For example, the left distracter in Frame 2 of Figure 5.2 is similar to the target (Figure 5.2, Frame 1) in shape and size but different in color. We pick the other distracter so that it is dissimilar to the target only in one cue and that cue is not the same as the one for the first distracter. Continuing the example of Figure 5.2, the distracter on the right (in Frame 2) is similar to the target in color and shape but not in size. In this case, if the human subject says "left," then we would have reason to believe that the perceptual similarity between the target and the left distracter is larger than that between the target and the right distracter, which can be interpreted to mean that the magnitude of cue differences between the target and the left distracter – only color in this case – is smaller than the the magnitude of cue differences between the target and the right distracter – only size in this case.

To learn an affinity function on pairs of regions, one needs to collect a large amount of human subject answers by systematically varying the following three things:

[1]In our preliminary experiments, we used the PASCAL VOC 2010 dataset and obtained around 3 million regions.

Frame 1 Frame 2

Figure 5.4: An example pair of frames for the apparent motion experiment with synthetic regions. The left distracter has the same size and shape as the target, but their colors are different. The right distracter differs from the target only in size.

1. the target,
2. the cues in which the target and their distracters are different,
3. the magnitude of these cue differences.

Working with real image regions presents a challenge on how to choose the distracters systematically once a target is fixed. We want the distracter to be slightly different from the target only in one cue, and since we cannot produce a real image region synthetically, coming up with a desired distracter becomes a search problem in the large pool of regions we already have. We find the closest region to the desired distracter, but this region might still not be very close to the ideal distracter. For example, the left distracter in Frame 2 of Figure 5.3 is the best region we could find, which is very similar to the target in color and shape cues, but slightly different in the color cue. Since we are searching for real regions within a pool, we do not have complete control on the magnitude of cue differences between the target and its distracters.

This fact was easily noticeable in our preliminary experiments: in almost all the trials, different human subjects said that they did not see any apparent motion but three different regions. This observation led us to using synthetic elliptical regions. The advantage of using elliptical regions is that we have total control of their cues, hence we can synthesize any target or distracter. An example target-distracter frame pair is given in Figure 5.4. Instead of using synthetic regions, another option would be to apply certain transformations to real regions in order to vary its cues to obtain the desired distracters, but we have not explored this option in this study.

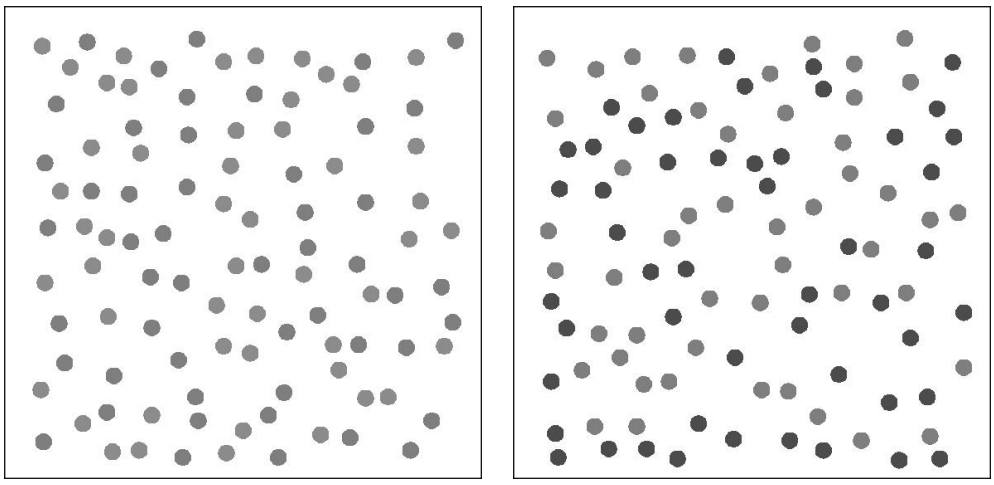


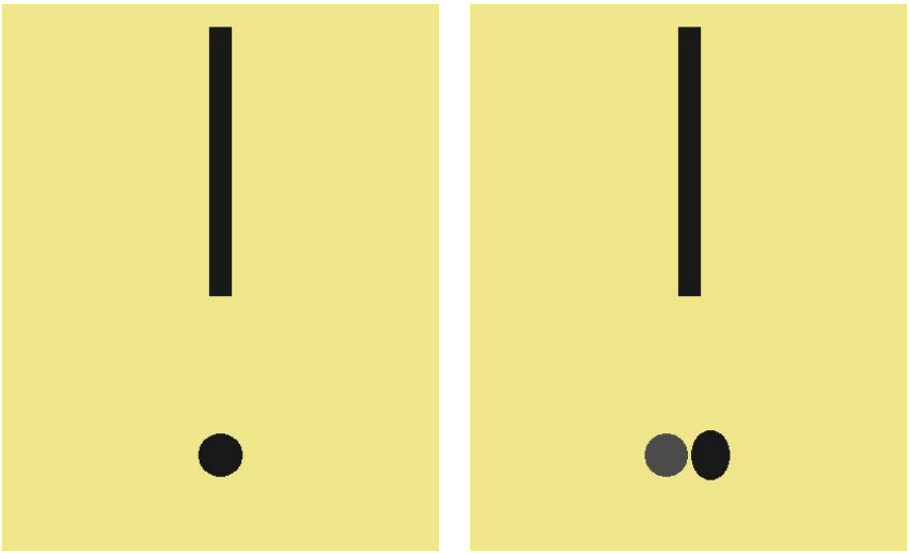
Figure 5.5: Two sample trials for the just-noticeable differences experiment. In each picture, there are regions sampled from two different region populations which differ only in their color cues. The magnitude of the difference between the populations in the left picture is less than that of the right picture.

5.2.2 Equating just-noticeable differences

In this method, we detect just-noticeable difference thresholds for each cue by showing a number of randomly placed elliptical regions sampled from two different populations. The populations differ statistically by a certain amount in a particular cue. After displaying the regions to the human subject for a short time, we ask whether he/she perceived a single population or two. If one sees two different populations, this means that the cue difference between the two populations is noticeable. Consequently, we assume that just-noticeable differences across different cues are perceptually equivalent. Two example trials for this method can be seen in Figure 5.5.

5.2.3 Binocular fusion

The human visual system is able to sense depth from two slightly different projections of the physical world onto the retinas of two eyes. If two retinal images are very different from each other, binocular rivalry occurs where perception alternates between the left and the right retinal image. When the retinal images are relatively less different, unstable composites of the



Left eye's view Right eye's view

Figure 5.6: A sample trial for the binocular fusion method. Left eye sees a target region and a vertical reference bar; right eye sees two distracters and the same reference bar as in the left view. See text for detailed explanation.

two images might be perceived. In neither case can we sense depth or see a proper 3D image.

We utilize this binocular fusion ability of the human visual system in our third method. We put different cues into competition by showing different images to the left and right eyes. As in the apparent motion method, a target region is shown to one eye and two distracters are shown to the other eye. If there is a "match" between the target token and any of the distracters, then the subject sees a 3D image. The perceived 3D depth depends on which of the distracters is matched to the target, and therefore the observer's depth judgment response tells us which cue is preferred.

Figure 5.6 shows an example trial. The left eye sees a target region, and a vertical bar which is used as a reference for depth perception. The right eye sees two distracters and the same reference bar as in the left view. As in the apparent motion experiment, the two distracters are dissimilar to the target in different visual cues. The distracter on the left has the same shape and size as the target but slightly different color, whereas the distracter on the right has the same color and size as the target but a slightly different shape, i.e. aspect ratio.

The vertical reference bar is at the same location in both views to ensure perfect fusion in the subject's visual system. However, the distracters are located in such a way that they have opposing signs of disparity values. Depending on which of the two distracters fuse with the target, the subject would see the target either in front of the bar or behind it. For the particular example of Figure 5.6, the left distracter is on the left of the reference bar, so it has a negative disparity. If the left distracter fuses with the target, then the subject would see that the target region is behind the bar, whereas if the right distracter fuses with the target, it would be seen as it is in front of the bar.

Nice as it may sound, this idea did not work as well in practice. We did preliminary experiments with different human subjects using many target regions, and the consensus among subjects was that they did not see a stable fused region; however, they saw the reference bar as stable and having clear, sharp boundaries. The target and one of the competing distracters did not "fuse" as we expected; rather, an unstable composite image with unclear boundaries was seen by the subjects. We believe that this is due to the following two reasons: (i) the distracters are too close to each other, and (ii) both distracters are very similar to the target. One can think that placing the distracters farther away from each other and making them less similar to the target would solve these problems, but doing so would actually make fusion even more difficult because of the increased spatial and visual disparity.

We explored another binocular fusion based method, which is similar to the just-noticeable differences idea presented in Section 5.2.2. In this method, we show a target to one eye, and a distracter to the other eye. Then we ask whether the subject sees a stable region with clear and sharp boundaries. By systematically varying the difference between the target and the distracter, we are able to quantify the boundary between "fusion" and "non-fusion."

After a few iterations of designing the method, we converged on the following. We display many copies of the target region to one of the eyes, and a succession of distracters, where the amount of visual difference increases as you go from left to right, is displayed to the other eye. Then we ask the subject to move a vertical bar between the regions, and put it in such a place so that the regions to the left of the bar fuse and the ones to the right of it do not.

A sample pair of views is given in Figure 5.7. In the left eye's view, there are 10 exact copies of the target region. In the right eye's view, there are 10 distracters having the same color and size as the target but becoming increasingly dissimilar to the target in the shape cue, from left to right. There is a vertical bar at the bottom of the display, which can be moved to the left and right by the observing subject. There are also on-screen instructions which are placed on top of the regions. The task for the subject is to place the bar in between two of the 10 regions so that the regions to the left of the bar fuse; that is, they have stable images with clear boundaries, and the regions to the right do not fuse. By analogy with the just-noticeable differences method, we call this particular method as "just-fusable-differences."

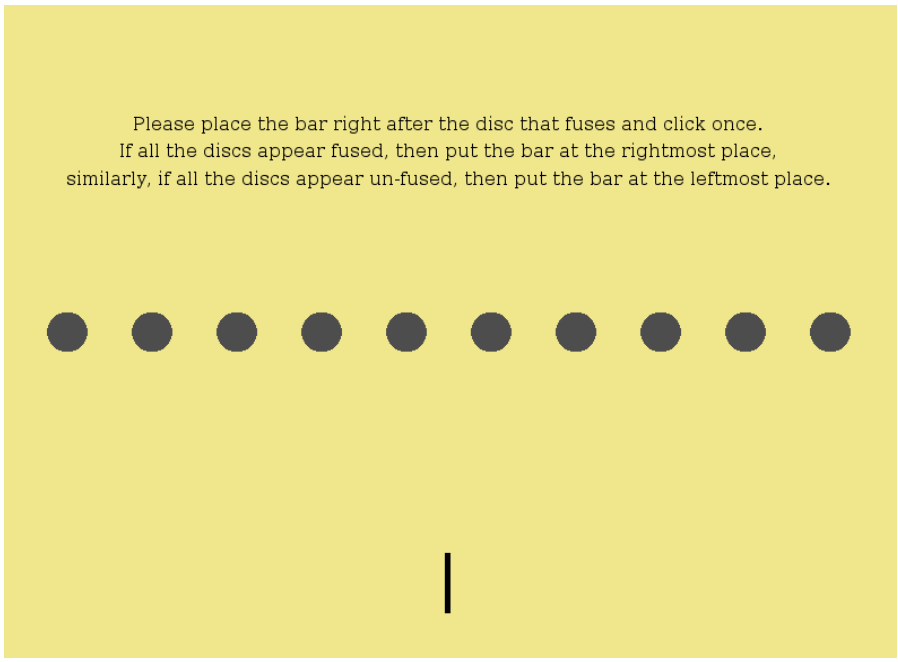
For all the binocular fusion related experiments, we used a head-mounted display device[2] as seen in Figure 5.8.

While prototyping interfaces for the head-mounted display, we worked with the anaglyphic versions of the same stereo image pairs for quick development. Figure 5.9 presents the red-cyan anaglyphic version of the stereo pair given in Figure 5.6. We include a quick tutorial about creating anaglyphic images from stereo pairs in Appendix B.

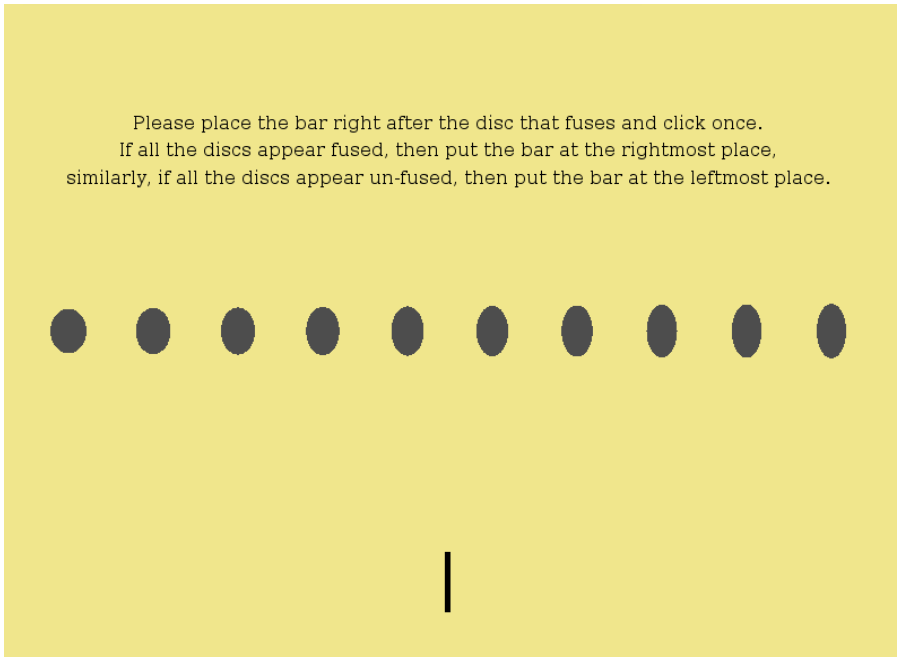
5.2.4 Discussion of the preliminary experiments

Apparent motion method, when compared to the other two methods, has the advantage of letting us learn an affinity function directly from the subject responses without any further assumption (see Appendix A). This is possible because different cues are directly put into competition, whereas in the other two methods, we need to make further assumptions such as that just-noticeable differences for different cues are perceptually equivalent. Despite this advantage, the apparent motion method was the least consistent one when we look at the variance of different human subject answers for the same trial. For both the apparent motion method and the just-noticeable differences method, the common feedback from the human subjects was that most of the time they needed to think before answering, and some even said they tried to remember how they replied in a similar previous trial in order

[2]Model "5DT HMD 800" from Virtual Realities (http://www.vrealities.com/5dt.html).



(a) Left eye's view



(b) Right eye's view

Figure 5.7: A sample stereo pair for the binocular fusion method. Left eye sees multiple copies of the target region, while the right eye sees corresponding distracters which are increasingly dissimiar in shape to the target. The subject is expected to place the vertical bar in between the fused and non-fused regions. A stereo pair of regions is said to be fused if they are perceived as a stable image with clear and sharp boundaries.

Figure 5.8: The head-mounted display device we used in our experiments.

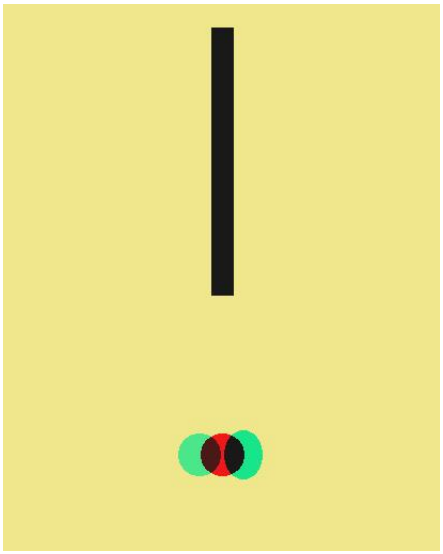


Figure 5.9: Red-cyan anaglyphic version of the pair given in Figure 5.6.

to be consistent in their answers. This feedback suggests that the two methods are cognitively more demanding than just establishing correspondence since correspondence occurs automatically without any conscious effort in our visual systems.

Among all three methods, subject responses were the most consistent (i.e. least variance) for the binocular fusion method. Furthermore, subjects said that this method did not require any "thinking" or "remembering" as did the first two methods; hence they were more confident in their responses, which explains the relatively less variance. However, they mentioned that some attention is needed to decide if the region boundaries are clear and sharp enough.

Completing the whole experiment involves systematically varying the targets and their distracters, and collecting responses from a pool of human

subjects. Due to the time and human costs, we could run only one of the three methods into completion, and based on the observations above, we opted to continue with the binocular fusion method.

5.3 Binocular Fusion Experiment

In this section we describe the complete binocular fusion experiment in detail. As mentioned in the previous sections, we used the "just-fusable-differences" method with synthetic elliptical regions. An elliptical region can be fully described by its three properties, or cues: size, color (grayscale), and shape. Size is expressed in the number of pixels. Color is a scalar between 0 (for black) and 1 (for white) since we restrict ourselves to homogeneous-color grayscale regions for simplicity. Shape is described in terms of aspect ratio which is the ratio of the length of the major axis to the length of the minor axis.

In the following sections, we describe how we systematically varied the targets, and their corresponding distracters.

5.3.1 Choosing the target regions

We do not know a priori whether the region affinity function that we want to estimate depends on the absolute locations of its input regions in the three dimensional cue space, and we do not want to make such assumptions without having a reason to believe so. Therefore, we sample target points from the cue space uniformly. The more points we use, the better. However, each target point adds to the time and human cost of the whole experiment. We sampled 3 points at equal intervals from each dimension: color $\in \{0.3, 0.5, 0.7\}$, shape $\in \{1, 4, 7\}$, and size $\in \{4000, 6000, 8000\}$, thereby obtaining a total of 27 target regions from the resulting 3x3 grid.

5.3.2 Creating distracters from target regions

For each target point, we create a set of distracters by varying its cues in the following way. Let $p = (p_{color}, p_{size}, p_{shape})$ be a target region, and let us create its positive color-distracters, which is the set of distracters whose size and shape are the same as p but whose color values are larger than p:

$$Q^{+}_{color} = \{(q_{color}(c), q_{size}, q_{shape}) \mid q_{size} = p_{size}, q_{shape} = p_{shape}, c = 0, 1, \ldots, N-1\}, \tag{5.1}$$

where

$$q_{color}(c) = p_{color} + \Delta^{min,+}_{color} + c\frac{\Delta^{max,+}_{color} - \Delta^{min,+}_{color}}{N}. \tag{5.2}$$

N is the number of distracters that are simultaneously displayed on a trial (see Figure 5.7 for a sample trial screen), which is 10 in our experiments. The values $\Delta^{min,+}_{color}$ and $\Delta^{max,+}_{color}$ are determined in such a way that a difference of $\Delta^{min,+}_{color}$ does not prevent p and q from fusing, and a difference of $\Delta^{max,+}_{color}$ is large enough to make sure that p and q do not fuse. These minimum and maximum delta values are determined per target per cue by doing preliminary experiments with two human subjects.

For each target point, there are 6 distracter sets in total (3 different cues, and 2 signs) which makes 27x6 = 162 trial screens in total.

5.3.3 The experiment

Each human subject was required to give responses to trials for approximately 50 minutes. We divided this session into 3 rounds where each round included distracters of only a single cue. Before each round, there was a training phase where the subject had to make 5 correct decisions in a row. In each round, the order of trials was random, and each trial was asked 3 times to the subject. The set of instructions we gave to the subjects before a session started can be found in Appendix C.

5.3.4 Collected data

As explained in Section 5.2.3, human subjects respond to trials by placing the vertical bar in between the fusing and non-fusing regions. Therefore, each response is actually an evidence of a boundary between fusion and non-fusion for a certain cue.

We collected data from 28 human subjects, and made histograms of their responses. These histograms are given in Figures 5.10, 5.11, 5.12. In each figure, there are two color-coded matrices where columns are for Δ_{cue} value-bins and each row is a histogram for a certain group of targets. We organized the responses in these rows to see whether the fusion versus non-fusion boundaries depend on where the targets are actually located in the visual-cue space.

We see that the responses concentrate on a range of values. For example, the normalized[3] Δ_{size} values appear to be accumulated at and around 0.35 (Figure 5.10) regardless of target's cues. Similarly, normalized Δ_{shape} is concentrated around 0.28 (Figure 5.11). While color responses for the set of positive color-distracters, Q^{+}_{color}, group around 0.3 (Figure 5.12 top image), the Δ^{-}_{color} values (Figure 5.12 bottom image) do not seem to be following this pattern.

5.3.5 Learning a similarity function between two regions

We assume that fusibility thresholds, that is the boundary between the fusion versus non-fusion, across different cues are perceptually equivalent. To be specific, for a target ellipse $p = (p_{color}, p_{size}, p_{shape})$, a distracter q with cues $(p_{color} + \Delta_{color}, p_{size}, p_{shape})$ and another distracter r with cues $(p_{color}, p_{size} + \Delta_{size}, p_{shape})$ are at perceptually equal distances to p. Here, Δ_{color} and Δ_{size} represent the just-fusable-difference thresholds, or fusion versus non-fusion boundary values, for color and size cues, respectively.

We further assume that the perceptual similarity is a linear combination of perceptual similarities across different features. That is, for two regions p_1 and p_2 with cues (c_1, a_1, s_1) and (c_2, a_2, s_2) where c is for color, a is for size (area), and s is for shape, a dissimilarity function F can be defined as:

[3]Actual delta value is divided by the cue value of the target.

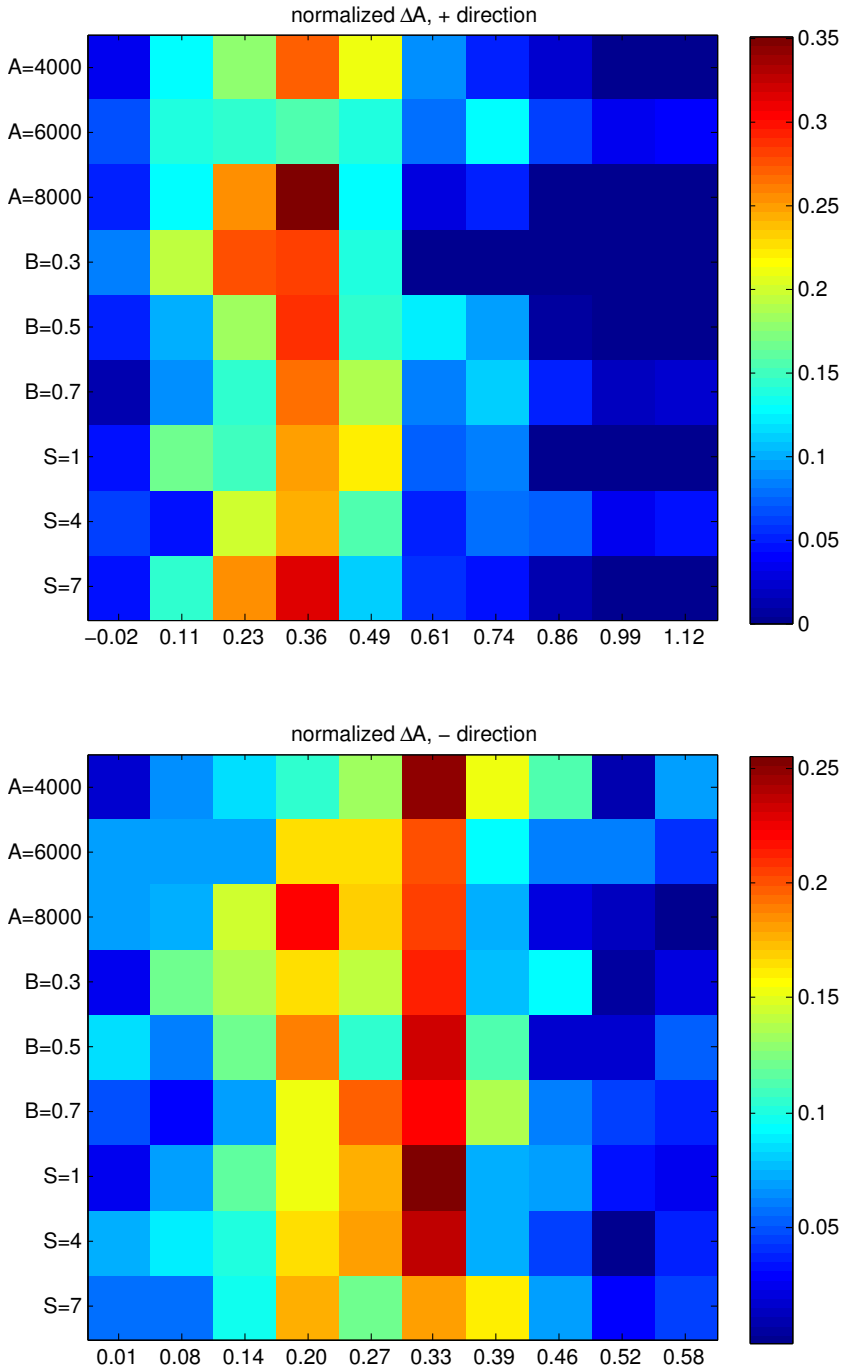


Figure 5.10: Histograms of subject responses to the size distracter trials. Each column corresponds to a Δ_{size} value, and each row is a histogram of responses corresponding to a group of targets. For example, the first row denoted by "A=4000" corresponds to all the targets having a size of 4000 pixels (A stands for area, B is brightness, and S is shape).

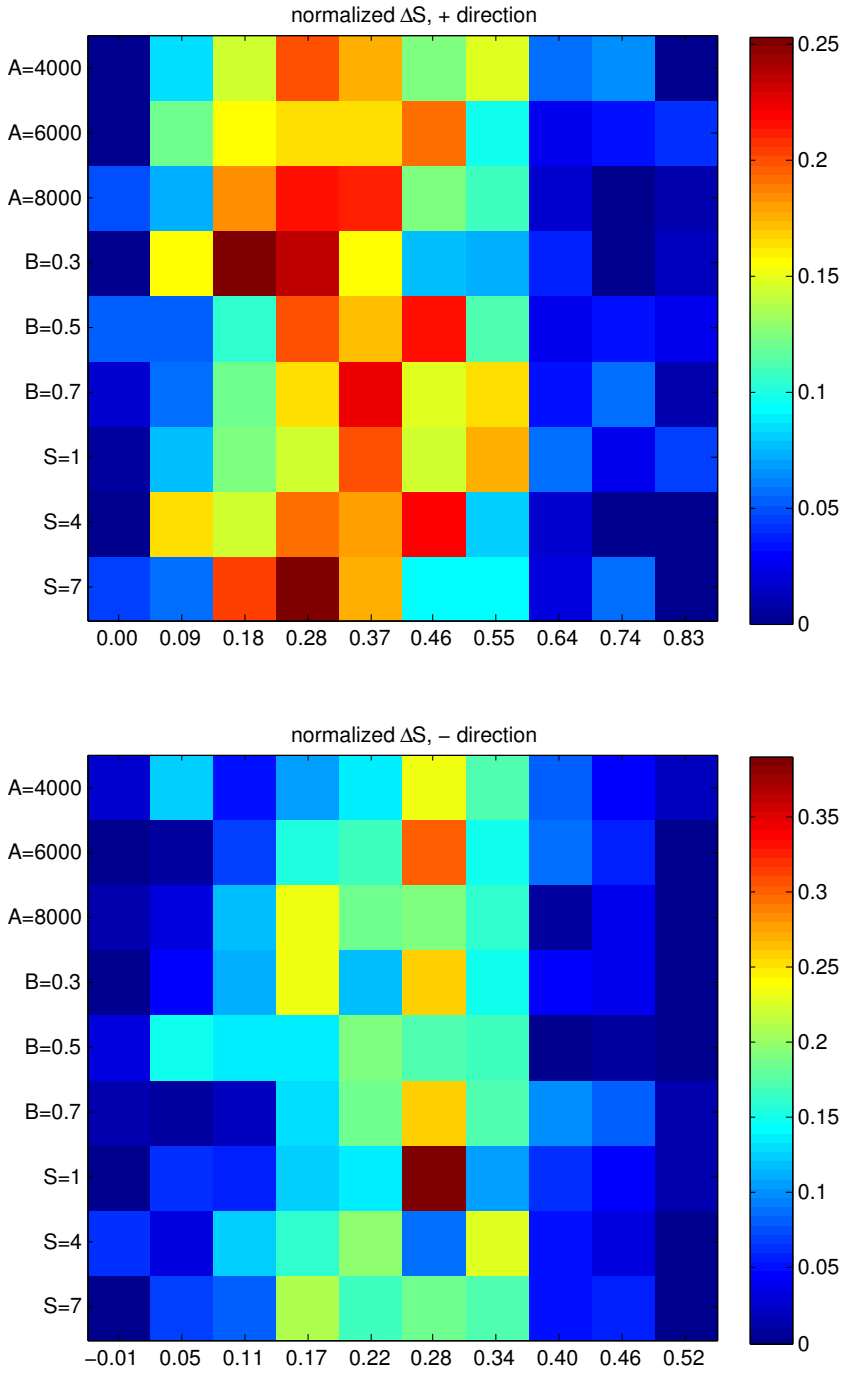


Figure 5.11: Histograms of subject responses to the size distracter trials. Each column corresponds to a Δ_{shape} value, and each row is a histogram of responses corresponding to a group of targets. See Figure 5.10 caption for more detail.

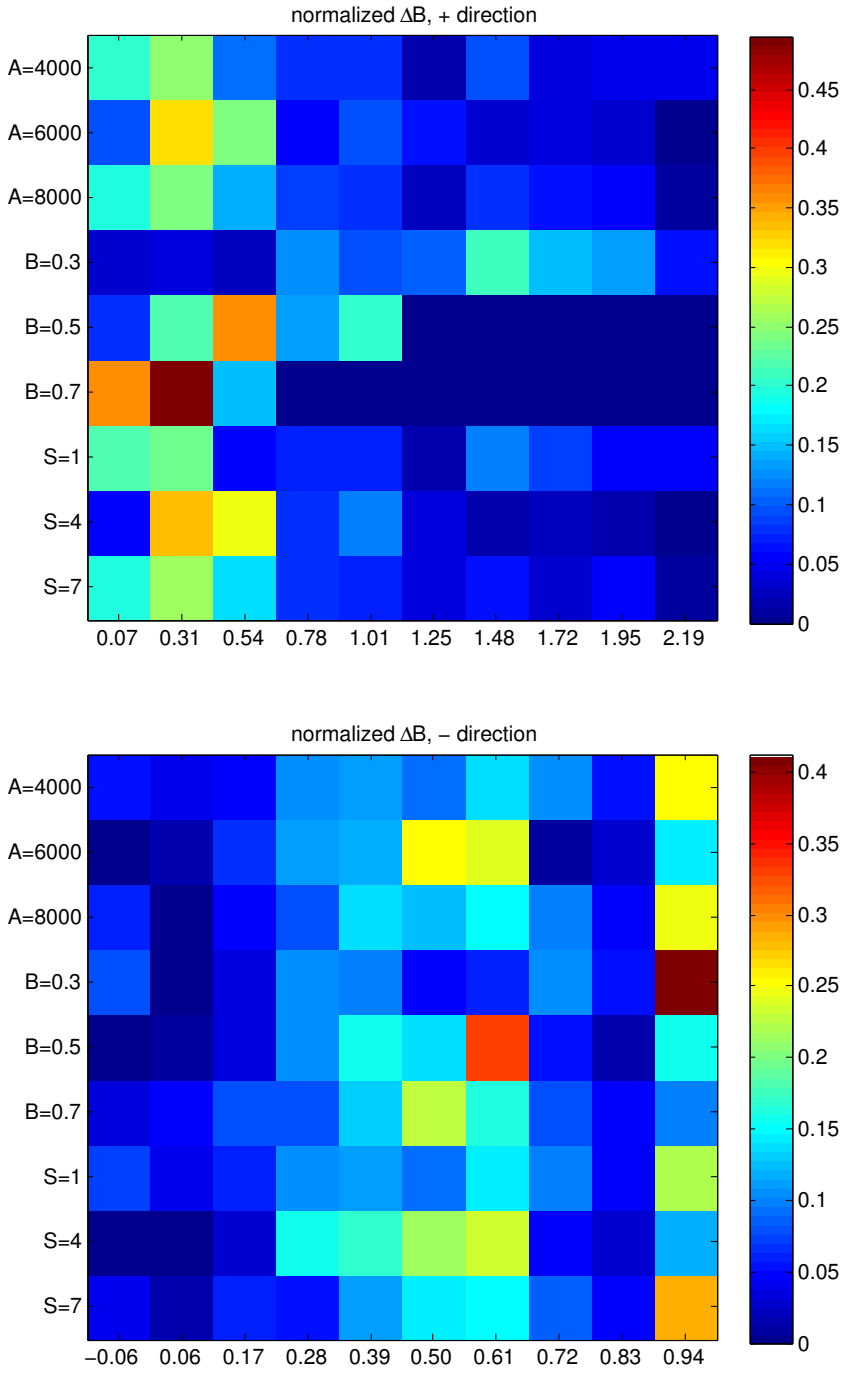


Figure 5.12: Histograms of subject responses to the size distracter trials. Each column corresponds to a Δ_{color} value, and each row is a histogram of responses corresponding to a group of targets. See Figure 5.10 caption for more detail.

$$F(p_1, p_2) = |c_1 - c_2|\Delta_{color} + |a_1 - a_2|\Delta_{area} + |s_1 - s_2|\Delta_{shape}. \quad (5.3)$$

The fusibility thresholds above are learned from the collected data. A straightforward way to do it is to assume that these threshold values are constant and do not depend on the cues of either p_1 or p_2. A more accurate thing to do is to assume a model for the dependency of fusibility thresholds to the cues of p_1. Then function F becomes:

$$\begin{aligned} F(p_1 \to p_2) &= |c_1 - c_2|\Delta_{color}(c_1, a_1, s_1) + |a_1 - a_2|\Delta_{area}(c_1, a_1, s_1) \\ &\quad +|s_1 - s_2|\Delta_{shape}(c_1, a_1, s_1). \end{aligned} \quad (5.4)$$

Note that function F is no longer a symmetric function. We assumed a linear model for the fusibility thresholds and obtained the following functions by fitting the model to the collected data:

$$\Delta_{color}(c, a, s) = 1.404 + 0.000a - 1.627c + 0.006s, \quad (5.5)$$

$$\Delta_{area}(c, a, s) = 0.298 - 0.000a + 0.305c - 0.005s, \quad (5.6)$$

$$\Delta_{shape}(c, a, s) = 0.309 - 0.000a + 0.198c - 0.008s. \quad (5.7)$$

5.4 Results and Discussion

We evaluate the learned dissimilarity function F in a region matching task where we detect regions by our segmentation algorithm (presented in Chapter 2). Region are described by three cues: size, average grayscale color and the aspect ratio of the ellipse that has the same second moments as the region.

We utilize readily available stereo datasets as ground-truth for region matching. In a stereo dataset, a pair of images (L, R) is given along with

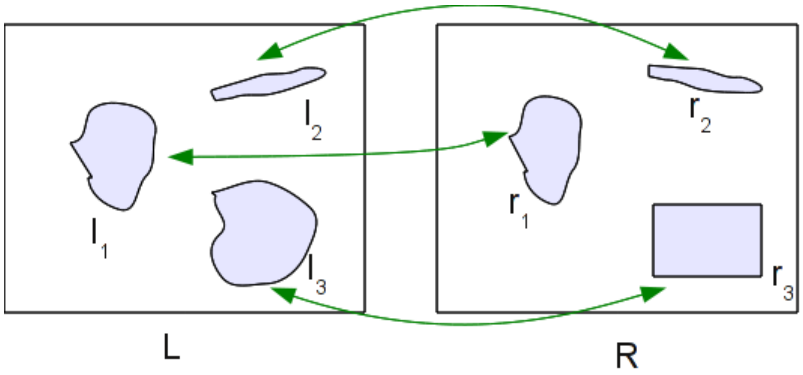


Figure 5.13: Matched regions for a given threshold T.

its disparity map (D). We first segment the two images L and R separately. Then, for each region l_1 in L, using the learned dissimilarity function, we search for the best matching region r_1 among the regions of R within a certain bounding-box centered around l_1. Using the disparity map D, we find the ground-truth match of l_1, $g(l_1)$, in R. The quality of match between l_1 and r_1 could be quantified with the intersection-over-union of r_1 and $g(l_1)$.

To compute overall matching quality, we threshold the dissimilarity values returned by the learned dissimilarity function. Given a threshold T, dissimilarities less than T are considered to be matches and we compute the precision and recall corresponding to T as follows:

$$\text{recall} = \frac{\text{\# of matches pixels}}{\text{\# of pixels in L}} \tag{5.8}$$

$$\text{precision} = \frac{\text{\# of matches pixels}}{\text{\# of retrieved pixels in R}}. \tag{5.9}$$

This matching process is illustrated in Figures 5.13 and 5.14.

We use the stereo pairs and their provided ground-truth disparity maps from the Middlebury dataset [70] to evaluate the performance of the dissimilarity function we learned in the previous section. Sample region matches for the "Middlebury1" pair are given in Figure 5.15. Figure 5.16 gives sample matches for the "monopoly" pair.

We give the precision-recall plots for Middlebury1, monopoly, baby1, baby2, and rocks2 stereo pairs in Figures 5.17, 5.18, and 5.19. Within those plots,

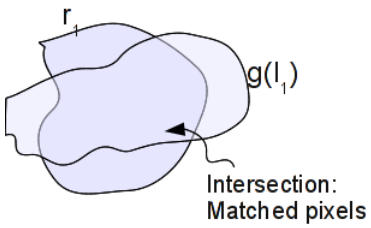


Figure 5.14: "Matched pixels" are the pixels that are at the intersection of r_1, which is l_1's best match with respect to the learned dissimilarity function, and $g(l_1)$, l_1's ground-truth match which is computed by using the disparity map.

we also report the performance of uniform fusibility weights, that is $\Delta_{color} = \Delta_{area} = \Delta_{shape} =$ some positive constant, when used in Equation (5.4), instead of the learned ones (Equations (5.5), (5.6), (5.7)).

From the precision-recall plots in Figures 5.17, 5.18, 5.19, one can conclude that using the learned weights, instead of uniform weights, does not make a significant difference. This is probably because there are not many correspondence contenders within the search space of a region, which is something to be expected from a structure-based method. The fact that region matching quality does not seem to be sensitive to the values of the weights is actually good news because we are using very simple cue descriptors (brightness for appearance, aspect ratio for shape, and size), and the number of correspondence contenders should decrease as the region descriptors get more complicated. Finally, it should be noted that describing a real image region by three simple descriptors – as we did in our experiments – is a very naïve thing to do, and how to utilize the learned perceptual distance function for (and generalize it to) real image regions, or how to re-design the experiment (especially the data collection part), for real image regions, remain to be investigated.

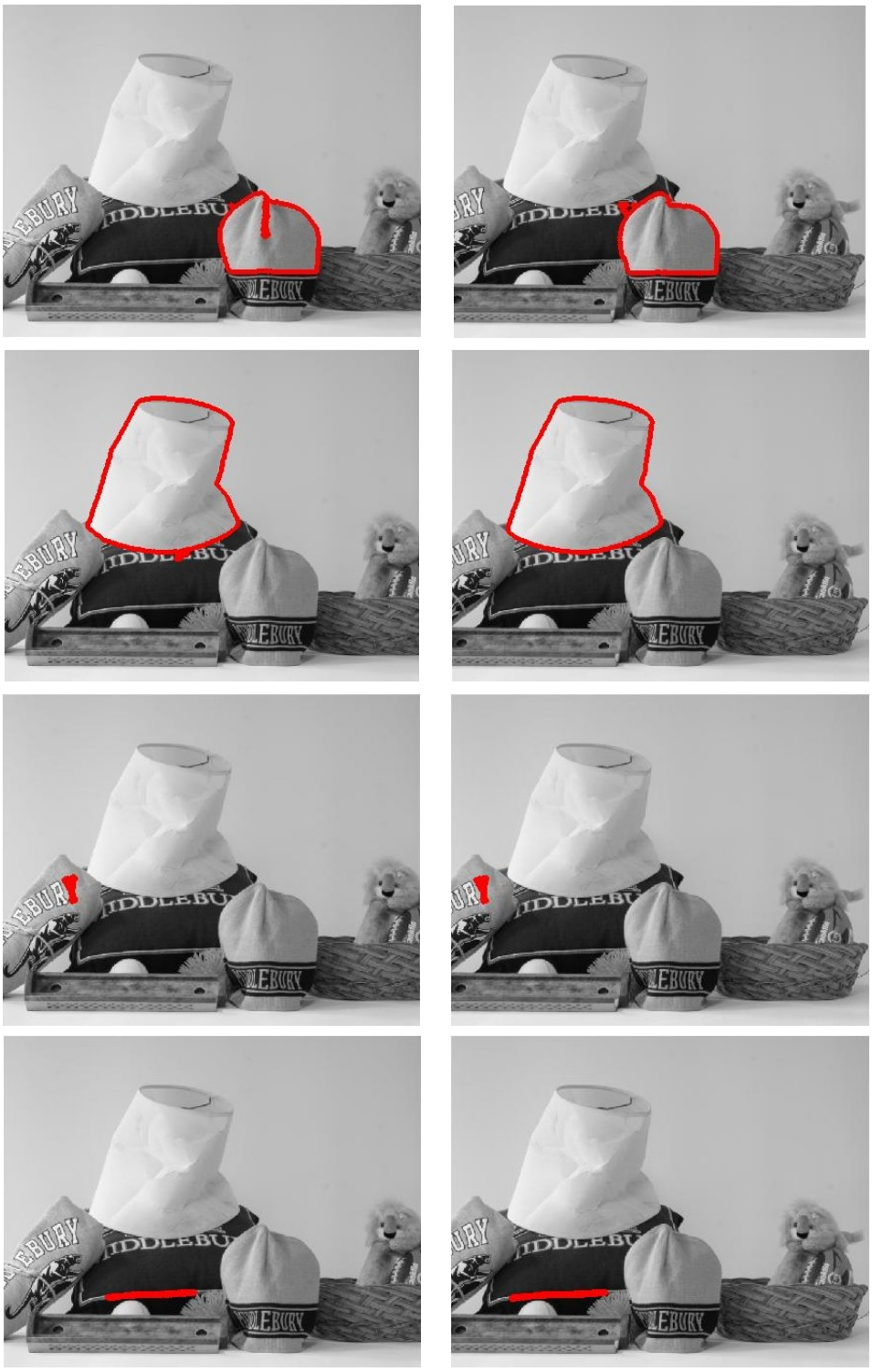


Figure 5.15: Sample region matches from the Middlebury1 stereo pair.

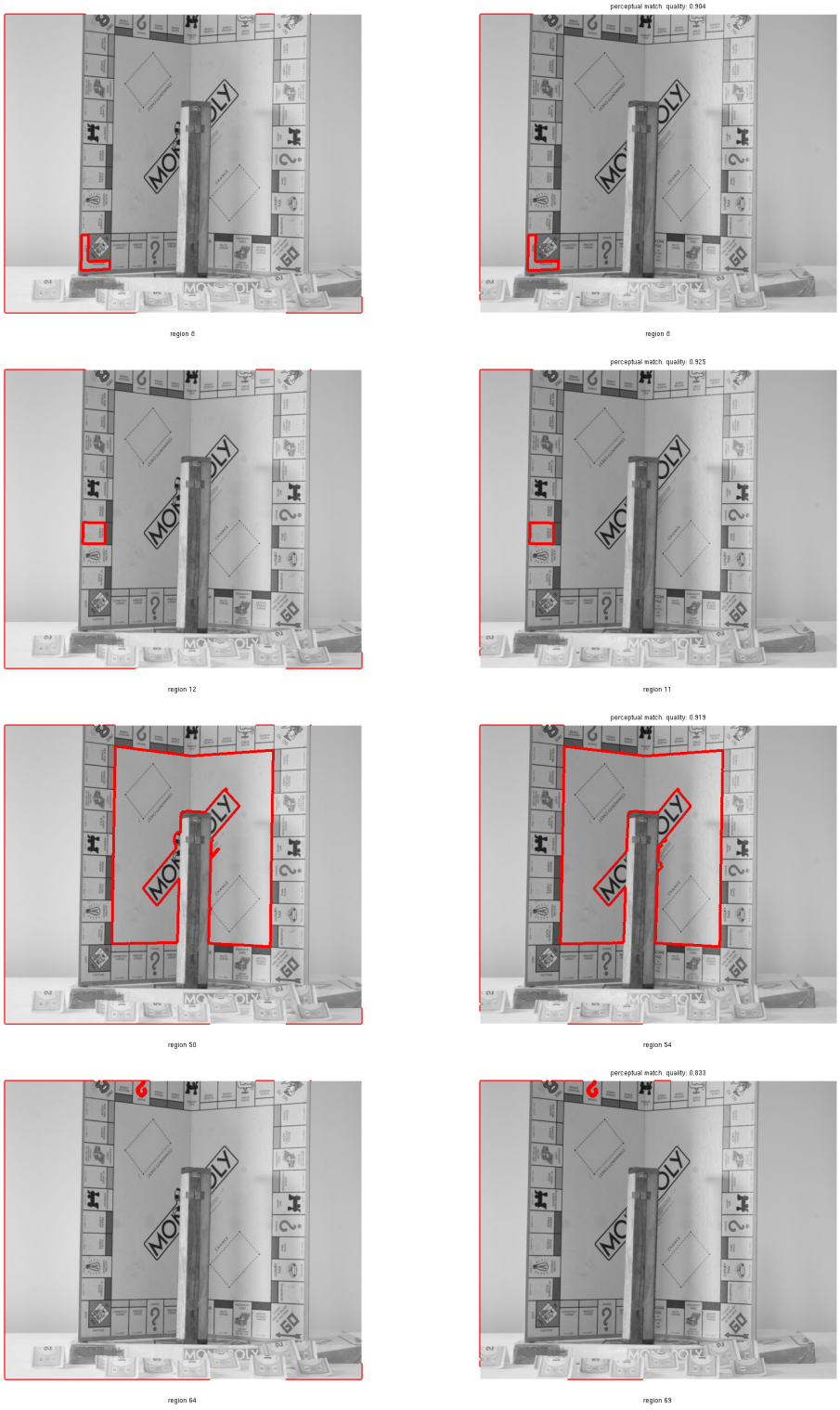


Figure 5.16: Sample region matches from the monopoly stereo pair.

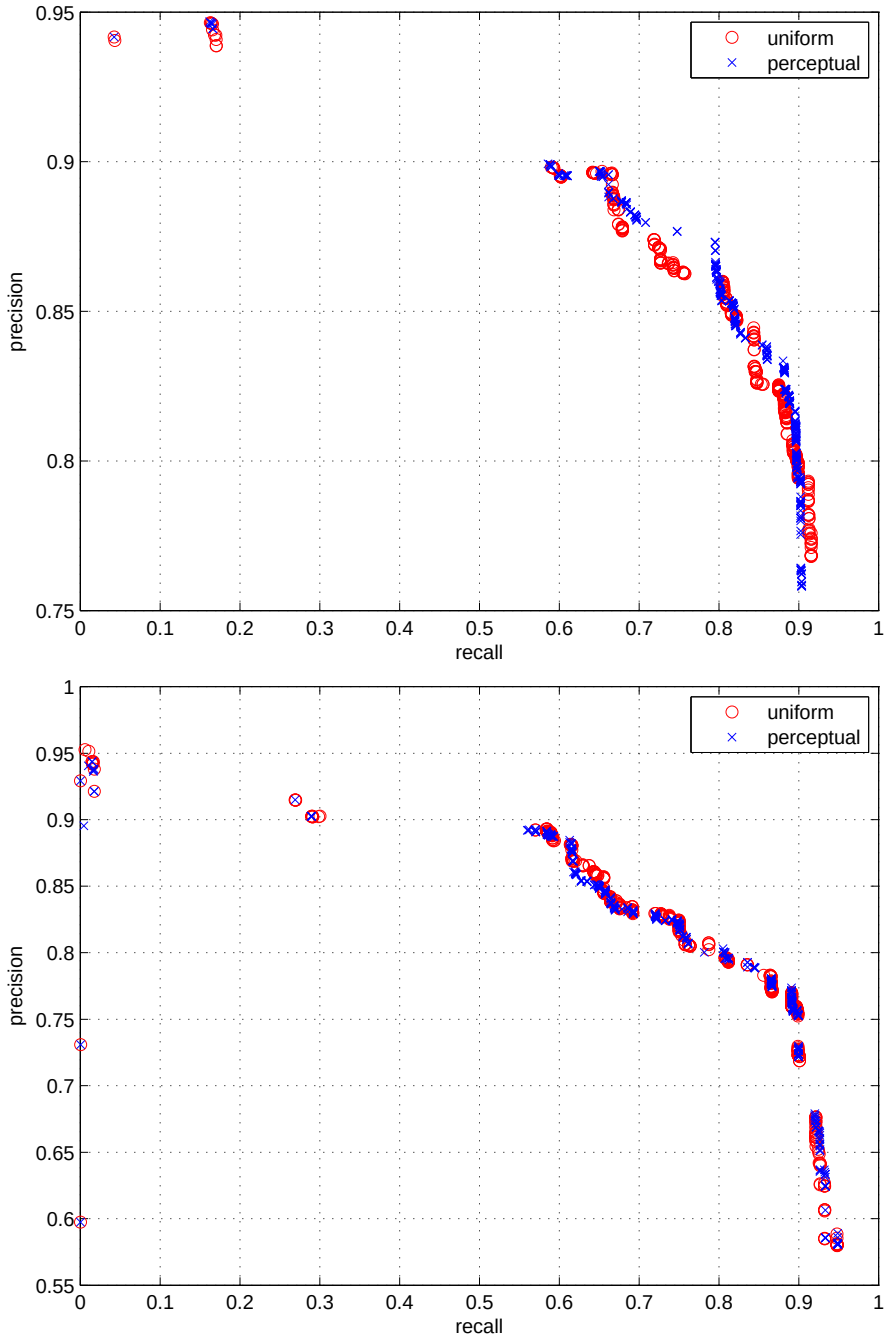


Figure 5.17: Precision-recall plot of region-match quality by the learned cue weights (denoted as "perceptual") and uniform weights, for the Middlebury1 (top) and the baby1 (bottom) stereo pairs.

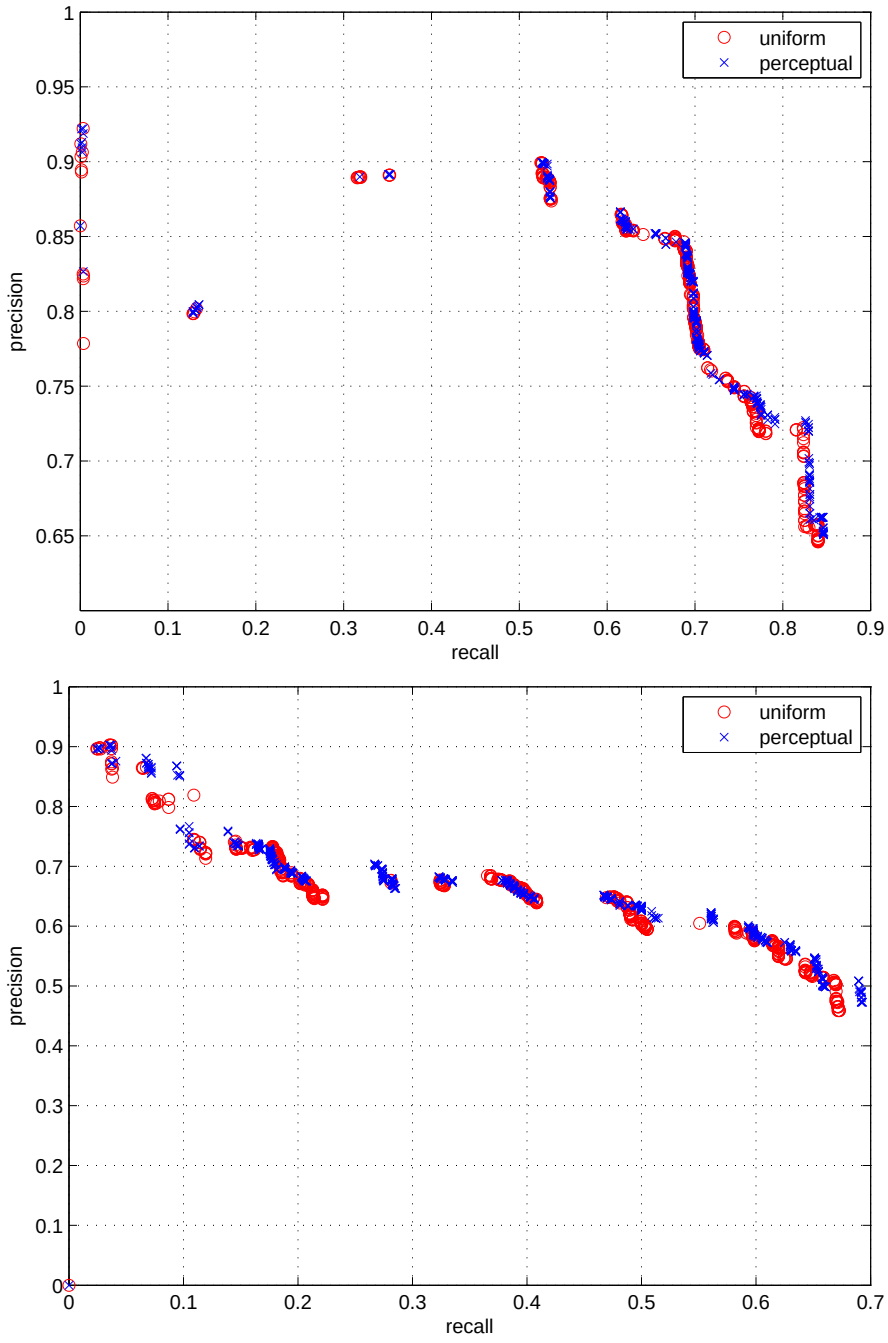


Figure 5.18: Precision-recall plot of region-match quality by the learned cue weights (denoted as "perceptual") and uniform weights, for the baby2(top) and rocks2 (bottom) stereo pairs.

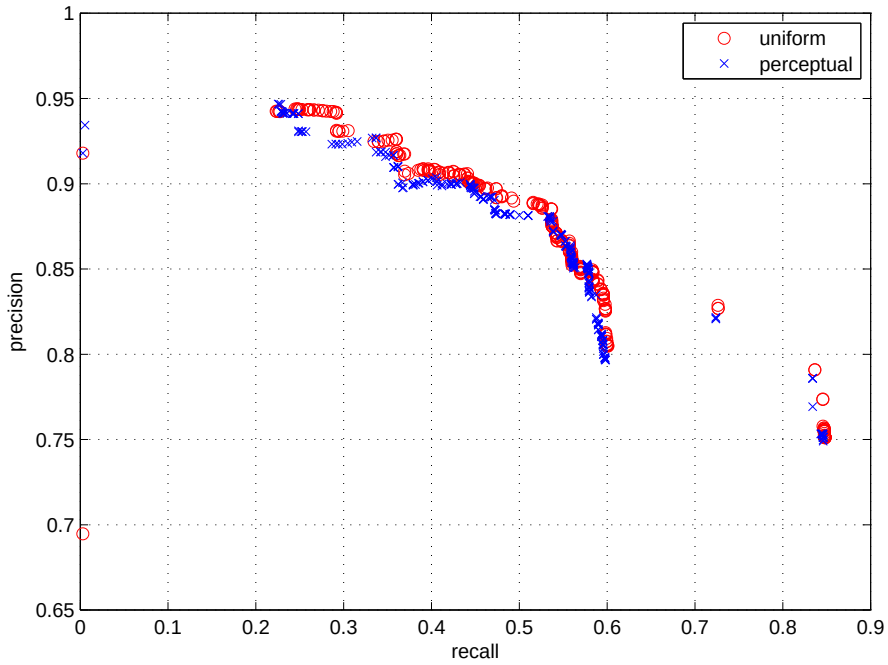


Figure 5.19: Precision-recall plot of region-match quality by the learned cue weights (denoted as "perceptual") and uniform weights, for the monopoly stereo pairs.

CHAPTER 6

MULTIPLE INSTANCE SELECTION BOOSTING

6.1 Introduction

Multiple instance learning (MIL) can be viewed as a weakly supervised learning problem where a set of bags, each containing an arbitrary number of instances, are labeled while the labels of the instances in each bag remain unknown. In a typical two-class MIL setting, a negative bag contains instances that are all labeled negative, while a positive bag contains at least one instance labeled positive. This learning problem manifests itself in numerous fields, especially those related to computer vision applications including content-based image retrieval, image classification, object detection, etc. In such applications, an image or a video is seen as a bag of instances where only a subset of them are meaningful for the given task, e.g. class discrimination. Instances can be interest points, patches or segments, depending on the application, and the size of the meaningful subset of instances is usually small compared to the number of instances a bag typically contains. While the main goal of MIL algorithms is to produce bag-level labels, there are methods that predict labels for the instances as well.

MIL algorithms are useful when labeling each and every instance is expensive. A typical example problem is image categorization where the category of the image is determined based on whether it contains an object (or multiple objects) of a certain object class. Labeling the objects would mean marking all occurrences of objects, which is time-consuming and difficult compared to just saying that an image belongs to a class. Another example would be action recognition from videos. It is very easy to label a whole sequence compared to labeling the spatio-temporal objects that are relevant to the action. In this sense, many visual recognition problems are inherently

MIL problems.

In this chapter, we first review the related MIL work in the literature. Then, we propose a new MIL algorithm, which we call "multiple instance selection boosting," or MIS-Boost for short. We conclude the chapter after presenting our experiments to evaluate the performance of MIS-Boost on several standard datasets, and compare it to the other MIL methods as well.

6.2 Related Work

During the past two decades, many MIL methods have been proposed with a significant interest in MIL emerging in recent years especially within the machine learning and vision communities. The work in [76] is one of the earliest papers that address the MIL problem, whereby it was cast in the framework of recognizing hand-written numerals. Over the next 20 years, MIL literature has abounded with algorithms that differ in two main respects: **(1)** the level at which the labeling is determined: instance-level (bottom-up) or bag-level (top-down) and **(2)** the type of data modeling assumed, i.e. generative vs. discriminative.

(1). Bottom-up vs. Top-down: While most MIL approaches address the problem of predicting the class of a bag directly without inferring the labels of the instances that belong to this bag, some approaches use max margin techniques to do this inference [77, 78, 79]. The latter approaches exploit the fact that the instances of negative bags have negative labels (-1) and at least one instance in each positive bag has a positive label ($+1$). For example, a MIL version of SVM is proposed in [77], where the traditional SVM optimization problem is transformed into a mixed-integer program and subsequently solved by alternating between solving a traditional SVM problem and heuristically choosing the positive instances for each positive bag.

(2). Generative vs. Discriminative: Some MIL approaches are generative in nature, since they assume that the underlying instances conform to a certain structure. An early generative approach is based on finding an optimal hyper-rectangle discriminant in the instance space [80]. Other prominent

generative MIL approaches are based on the notion of diverse density (DD) [81]. These approaches seek a "concept" instance[1] that is *close* to at least one instance of each positive bag and *far* away from all the instances in the negative bags. In other words, a concept instance is a vector in instance space that best describes the positive bags and discriminates them from the negative bags. The existence of such a concept assumes that positive instances are compactly clustered and well separated from negative instances. Such an assumption is strict and does not always hold in natural data, which tends to be multi-modal. DD-based MIL approaches compute this optimal concept by formulating the problem in a maximum likelihood framework using a noisy-OR model of the likelihood. Improvements on the original DD formulation have been made, where the EM algorithm is used to find the concept instance in [82] and multiple concepts are estimated in [83]. Moreover, standard supervised learning techniques, such as kNN, linear and kernel SVM, AdaBoost, and Random Forests, have been adapted to the MIL problem, thus leading to citation kNN [84], MI-kernel [85], MIGraph/miGraph [86], mi/MI/DD-SVM [77, 87], MI-Boost [88], MI-Winnow [89], MI logistic regression [90], and most recently MIForests [91]. Furthermore, some MIL approaches actively seek instances in the training set (denoted prototypes) that carry discriminative power between the positive and negative classes. In what follows, we will denote these as instance selection MIL methods. Such approaches transform the original feature space into another space defined by the selected prototypes (e.g. using bag-to-instance similarities) and subsequently apply standard supervised learning techniques in the new space. In [92], all training instances are selected to be prototypes, while only one instance per bag is systematically initialized, greedily updated, and selected in [93, 94].

6.2.1 Overview of our approach

Our proposed instance selection method (dubbed Multiple Instance Selection Boost or MIS-Boost) is inspired by the instance selection MIL methods mentioned above (e.g. MILES [92, 95] and MILIS [94]) and the MI-Boost algorithm in [88]. It was hinted earlier that instance selection MIL methods

[1]This instance does not have to be one of the instances in the training set.

comprise two fundamental stages. **(i)** In the representation stage, the original training instances are represented in a new feature space determined by the selected prototypes. **(ii)** In the classification stage, a supervised learning technique is used to build a classifier in the new feature space to optimize a given classification cost. Most of these methods treat the two stages independently and sequentially, in such a way that representation (i.e. prototype selection) is unaffected by class label distribution. Here, MILIS is an exception, since it iteratively selects prototypes from the training set to minimize classification cost. This selection is further restricted, since only one prototype is selected per training bag. We consider this to be a restriction because we believe that the prototype selection process should be data dependent. For example, in the case of image classification, some "simple" object classes (e.g. airplane) may yield a smaller number of prototypes than other more "complex" classes (e.g. bicycle).

As compared to previous methods, prototypes selected by MIS-Boost do not necessarily belong to the given training set and the number of these prototypes is not predefined, since they are determined in a data-driven fashion. Since the search space for prototype instances is no longer limited, more discriminative and possibly fewer prototypes can be learned. This learning process directly involves minimization of the final classification cost. As such, MIS-Boost learns a new representation based on the estimated prototypes, in a boosting framework. This leads to an iterative algorithm, which learns prototype-based base classifiers that are linearly combined. At each iteration of MIS-Boost, a prototype is learned so that it maximally discriminates between the positive and negative bags, which are themselves weighted according to how well they were discriminated in earlier iterations. The number of prototypes is determined in a data-driven way by cross-validation. Experiments on benchmark datasets show that MIS-Boost achieves state-of-the-art performance. When applied to image classification, MIS-Boost selects prototypes that map to meaningful image segments (e.g. class specific object parts).

6.3 Proposed Algorithm

Given a training set $T = \{(B_1, y_1), (B_2, y_2), \ldots, (B_N, y_N)\}$ where $y_i \in \{-1, +1\}$ $\forall i$, B_i represents the i^{th} bag and y_i its label, our goal is to learn a bag-classifier $F : B \to \{-1, +1\}$. Each bag consists of an arbitrary number of instances. The number of instances in the i^{th} bag is denoted by n_i, so we have $B_i = \{\vec{x}_{i1}, \vec{x}_{i2}, \ldots, \vec{x}_{in_i}\}$. Each instance $\vec{x}_{ij}$ lives in a D-dimensional feature space, $\vec{x}_{ij} \in \mathbb{R}^D \ \ \forall i, j$.

Although the algorithm we propose below is a general purpose multiple instance learning algorithm, for illustration purposes, we will describe it in an image classification context where a bag correspond to an image, and instances within a bag corresponds to the regions obtained from the corresponding image. We assume that ground-truth bag labels are given based on the existence of instance(s) of a certain object class. A good example for such a scenario is the image categorization track of PASCAL VOC datasets where, for example, an image is labeled with "airplane" if that image contains an airplane.

We propose the following additive model as our bag classification function:

$$F(B) = \text{sign}\left(\sum_{m=1}^{M} f_m(B)\right), \tag{6.1}$$

where each $f_m(B)$, called a base classifier, is associated with a prototype instance $\vec{p}_m \in \mathbb{R}^D$. The function $f_m : B \to [-1, 1]$ is also a bag classifier, like F, but it returns a score in between -1 and 1 which quantifies the "existence" of the prototype instance, $\vec{p}_m$, within the bag B. The "existence" of $\vec{p}_m$ within bag B_i is the distance from $\vec{p}_m$ to the closest region to $\vec{p}_m$ within B_i, that is:

$$D(\vec{p}_m, B) = \min_j \left(d(\vec{p}_m, \vec{x}_{ij})\right), \tag{6.2}$$

where $d(\cdot, \cdot)$ is a distance function, for which we use the Euclidean distance, between two instances, i.e. region feature vectors. We call $D(\cdot, \cdot)$ the instance-to-bag distance function. Here, we note that this instance-to-bag distance is used in other instance selection MIL methods (e.g. MILES and MILIS); however, the prototype $\vec{p}_m$ in these methods is restricted to a

discrete subset of the training samples. By removing this restriction on $\vec{p}_m$ and allowing it to take arbitrary values in $\mathbb{R}^D$, more discriminative and possibly fewer prototypes need to be *learned*. The function $f_m(\cdot)$ computes the instance-to-bag distances first, and classifies the bags using these distances. Although $f_m(\cdot)$ can take any suitable form, we opt to use the simple scaled and shifted sigmoid function, parameterized by β_0, β_1, and $\vec{p}_m$:

$$f_m(B) = \frac{2}{1 + e^{-(\beta_1 D(\vec{p}_m, B) + \beta_0)}} - 1. \tag{6.3}$$

6.3.1 Learning F

Since F is an additive model, boosting can be used to learn it [96]. Among the many variants of boosting, we choose to use Gentle-AdaBoost for its numerical stability properties [96]. We give the Gentle AdaBoost algorithm in Table 6.1 for completeness.

Table 6.1: Pseudo-code for Gentle-AdaBoost algorithm.

Gentle AdaBoost

Input: Training set $\{(B_1, y_1), (B_2, y_2), \ldots, (B_N, y_N)\}$, number of weak learners M

Output: Classifier $F(x)$

Weights $w_i \leftarrow 1/N$ for $i = 1, 2, \ldots, N$, and $F(x) = 0$.

for $m = 1, 2, \ldots, M$ **do**

Learn a weak classifier f_m that minimizes:

$$\varepsilon_{\mathrm{m}} = \sum_{i=1}^{N} w_i (y_i - f_m(B_i))^2, \tag{6.4}$$

Update $F(x) \leftarrow F(x) + f_m(x)$,

Update $w_i \leftarrow w_i e^{-y_i f_m(B_i)}$ and normalize weights so that $\sum w_i = 1$.

end for

Output $F(x) = \text{sign}\left(\sum_{i=1}^{M} f_m(x)\right)$

6.3.2 Learning base classifier $f_m(\cdot)$

The interesting bit of the algorithm given in Table 6.1 is the step where the weak classifier is learned. We essentially need to solve the following optimization problem:

$$\underset{\vec{p}_m,\beta_0,\beta_1}{\arg\min}\, \varepsilon_m \quad \text{where} \quad \varepsilon_m = \sum_{i=1}^{N} w_i \left(y_i - \frac{2}{1 + e^{-(\beta_1 D(\vec{p}_m, B) + \beta_0)}} + 1 \right)^2. \tag{6.5}$$

Here, w_i is the weight of the i^{th} bag at the current iteration. The main difficulty in optimizing the cost function above is the fact that the instance-to-bag distance term $D(\vec{p}_m, B)$ involves the non-differentiable "min" function. It is this same function that forces other instance selection methods (e.g. MILES and MILIS) to restrict the prototype search space to a subset of the training samples. For example, MILES considers all training samples as valid prototypes, thus, making learning the classifier (ℓ_1 SVM) significantly computationally expensive. On the other hand, MILIS takes a brute-force approach to prototype selection by greedily choosing one instance from each training bag as a valid prototype. Although selection is done so that an overall classification cost is iteratively reduced, this selection strategy highly restricts the feasible prototype space. To alleviate the problem of non-differentiability in our formulation, we replace "min" in $D(\vec{p}_m, B_i)$ with a differentiable approximation (known as "soft-min") to form the soft-instance-to-bag distance $\tilde{D}(\vec{p}_m, B_i)$:

$$D(\vec{p}_m, B_i) \approx \tilde{D}(\vec{p}_m, B_i) = \sum_{j=1}^{n_i} \pi_j d(\vec{p}_m, \vec{x}_{ij}), \tag{6.6}$$

where

$$\pi_j = \frac{e^{-\alpha d(\vec{p}_m, \vec{x}_{ij})}}{\sum_{k=1}^{n_i} e^{-\alpha d(\vec{p}_m, \vec{x}_{ik})}}, \tag{6.7}$$

α is a large positive constant. We give a definition of the soft-minimum function in Appendix D. The new cost function is:

$$\tilde{\varepsilon}_m = \sum_{i=1}^{N} w_i \left(y_i - \frac{2}{1 + e^{-(\beta_1 \tilde{D}(\vec{p}_m, B) + \beta_0)}} + 1 \right)^2 . \tag{6.8}$$

Replacing $D(\vec{p}_m, B)$ with $\tilde{D}(\vec{p}_m, B)$ makes the cost (6.8) differentiable, allowing for gradient descent optimization. However, it is not a convex cost function, so we will suffer from getting stuck into local minima. As a remedy for this, we propose to do multiple starts, each time initializing $\vec{p}_m$ with a different value. Preferably, these initialization points should be sampled from entire the instance feature. For this purpose, we cluster all the instances using the k-means algorithm, and use the cluster centers as initialization points for $\vec{p}_m$.

We minimize the cost in (6.8) using coordinate-descent. We start by initializing $\vec{p}_m$ to a cluster center, and optimize over the β_0, β_1 parameters. Then, we fix these β's and optimize over $\vec{p}_m$. We iterate this procedure until convergence; that is, the difference between successive errors becomes smaller than a given threshold. We give the algorithm to learn a weak classifier in Table 6.2, and gradient of the cost function in Appendix E.

6.3.3 Determining the number of weak learners

As the number of base classifiers increases, Gentle-AdaBoost tends to overfit on the training data. In order to prevent this and determine the number of base classifiers automatically, we perform 4-fold cross validation, whereby we randomly split the training data into 4 equal-size pieces and use 3 pieces for training and the rest for validation. We run the algorithm for a large number of base classifiers and pick the number which gives the least classification error on the validation set. We give the pseudo-code of MIS-Boost in Table 6.3.

6.4 Experiments

In this section, we evaluate the performance of MIS-Boost on five different MIL benchmark datasets and two COREL image classification datasets. We

Table 6.2: Pseudo-code for learning a weak classifier.

Learning f_m

Input: Training set $\{(B_1, y_1), (B_2, y_2), \ldots, (B_N, y_N)\}$, Weights w_i, $i = 1, 2, \ldots, N$, cluster centers $\{\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_K\}$, and error tolerance Tol.
Output: Weak classifier $f_m(x)$
// Initialize $\vec{p}_m$ to each cluster center
for $\vec{p}_m^0 = \vec{c}_1, \vec{c}_2, \ldots, \vec{c}_K$ **do**
error(-1) $\leftarrow \infty$
$(\beta_0^0, \beta_1^0) \leftarrow \arg\min \tilde{\varepsilon}_m|_{(\vec{p}_m = \vec{p}_m^0)}$ {Fix $\vec{p}_m$ and minimize over β's}
error(0) $\leftarrow \tilde{\varepsilon}_m(\vec{p}_m^0, \beta_0^0, \beta_1^0)$
$t \leftarrow 0$
while $|\text{error}(t) - \text{error}(t-1)| \geq$ Tol **do**
$t \leftarrow t + 1$
$\vec{p}_m^t \leftarrow \arg\min_{\vec{p}_m} \tilde{\varepsilon}_m|_{(\beta_0 = \beta_0^{t-1}, \beta_1 = \beta_1^{t-1})}$ {Fix β's and minimize over $\vec{p}_m$}
$(\beta_0^t, \beta_1^t) \leftarrow \arg\min \tilde{\varepsilon}_m|_{(\vec{p}_m = \vec{p}_m^{t-1})}$
error(t) $\leftarrow \tilde{\varepsilon}_m(\vec{p}_m^t, \beta_0^t, \beta_1^t)$
end while
Keep $(\vec{p}_m^*, \beta_0^*, \beta_1^*)$ with the least error so far
end for
Set $(\vec{p}_m, \beta_0, \beta_1) \leftarrow (\vec{p}_m^*, \beta_0^*, \beta_1^*)$, and output f_m

Table 6.3: Pseudo-code for the MIS-Boost algorithm.

MIS-Boost

Input: Training set $\{(B_1, y_1), (B_2, y_2), \ldots, (B_N, y_N)\}$, maximum number of weak learners M, number of clusters K
Output: Classifier $F(x)$
Cluster all instances, $\vec{x}_{ij},\ i = 1, 2, \ldots, N; j = n_i$, into K clusters.
Cluster centers are $\{\vec{c}_1, \vec{c}_2, \ldots, \vec{c}_K\}$.
Split the training set into train-set and validation-set.
Weights $w_i \leftarrow 1/N$ for $i = 1, 2, \ldots, N$, and $F(x) = 0$.
for $m = 1, 2, \ldots, M$ **do**
Learn a weak classifier f_m that using the algorithm given in Table 6.2.
Update $F(x) \leftarrow F(x) + f_m(x)$,
Update $w_i \leftarrow w_i e^{-y_i f_m(B_i)}$ and normalize weights so that $\sum w_i = 1$.
Evaluate $F(x)$ on the validation-set, compute validation-error(m).
end for
$M \leftarrow \arg\min_m$ (validation-error)
Output $F(x) = \text{sign}\left(\sum_{i=1}^{M} f_m(x)\right)$

compare our performance to those of the most recent and state-of-the-art MIL methods available for each dataset. In another experiment, we use MIS-Boost in a large-scale image classification task and visualize samples of the instances that are closest to the learned prototype(s). The results of this experiment suggest that the learned prototypes are not only discriminative but also visually meaningful; that is, they are similar to the parts of image that are relevant for classification.

6.4.1 Benchmark MIL datasets

The drug activity prediction datasets, "Musk1" and "Musk2" described in [80], and the image datasets, "Elephant", "Fox", "Tiger" introduced in [77] have been widely used and have become standard benchmark datasets for MIL methods. For each dataset, we perform 10-fold cross validation and report the average per-fold test classification accuracy. This is the standard way of reporting results on these datasets. In all our experiments in this section and the two following sections, we set the number of clusters $K = 100$, and the maximum number of base learners, or prototypes, to $M = 100$.

We report our results in Table 6.4, where we list the results of the most recent and state-of-the-art MIL methods. To the best of our knowledge, this table gives the most comprehensive comparison between MIL methods on the benchmark datasets. Clearly, MIS-Boost outperforms other methods in all datasets except the "Tiger" class. There, we have the second best accuracy with only a 0.5% difference with the top performing method, miGraph [86]. Among all the methods in Table 6.4, MIS-Boost is the most similar to MILES and MILIS, as they are also instance-selection-based methods. Except on "Musk1", our algorithm significantly outperforms these two methods. We believe that this improvement is largely due to the fact that our method, in contrast to MILES and MILIS, does *not* restrict the prototypes to a subset of the training instances, as we discussed in Section 6.3.2.

Classification accuracies on train, test, and validation sets are given for a sample run of MIS-Boost in Figure 6.1. The best performance on the validation set is obtained at 4 weak learners, which shows the efficiency of our instance selection approach. In fact, using only 2 weak learners we get

Table 6.4: Percent classification accuracies of MIL algorithms on benchmark MIL datasets. Best results are marked in bold fonts.

Method	Musk1	Musk2	Elephant	Fox	Tiger
MIS-Boost	**90.3**	**94.4**	**89.0**	**80.0**	85.5
MIForest[91]	85	82	84	64	82
MIGraph[86]	90.0	90.0	85.1	61.2	81.9
miGraph[86]	88.9	90.3	86.8	61.6	**86.0**
MILBoost[88]	71	61	73	58	58
EM-DD[82]	84.8	84.9	78.3	56.1	72.1
DD[87]	88.0	84.0	N/A	N/A	N/A
MI-SVM[77]	77.9	84.3	81.4	59.4	84.0
mi-SVM[77]	87.4	83.6	82.0	58.2	78.9
MILES[92]	88	83	81	62	80
MILIS[94]	88	83	81	62	80
MI-Kernel[85]	88	89	84	60	84
AW-SVM[97]	86	84	82	64	83
AL-SVM[97]	86	83	79	63	78
MissSVM[79]	87.6	80.0	N/A	N/A	N/A

a test-set performance of 100%. The classification accuracy on the test-set using 4 weak learners is 88.9%. As more weak learners are added, the performance on the train set reaches up to 100%, while the performance on the validation and test sets does not improve.

6.4.2 COREL dataset

The COREL-2000 image classification dataset [92] contains 2000 images in 20 classes. COREL-1000 is just a subset of this dataset, which contains the first 10 classes. We use the same features and experimental settings as in [92], and train one-vs.-all MIS-Boost classifiers to deal with the multiclass case. The results of MIS-Boost and three most recent, state-of-the-art methods are given in Table 6.5. Our method outperforms the other methods on both datasets. To illustrate the data-driven nature of our algorithm, we give the number of prototypes learned per class in Figure 6.2.

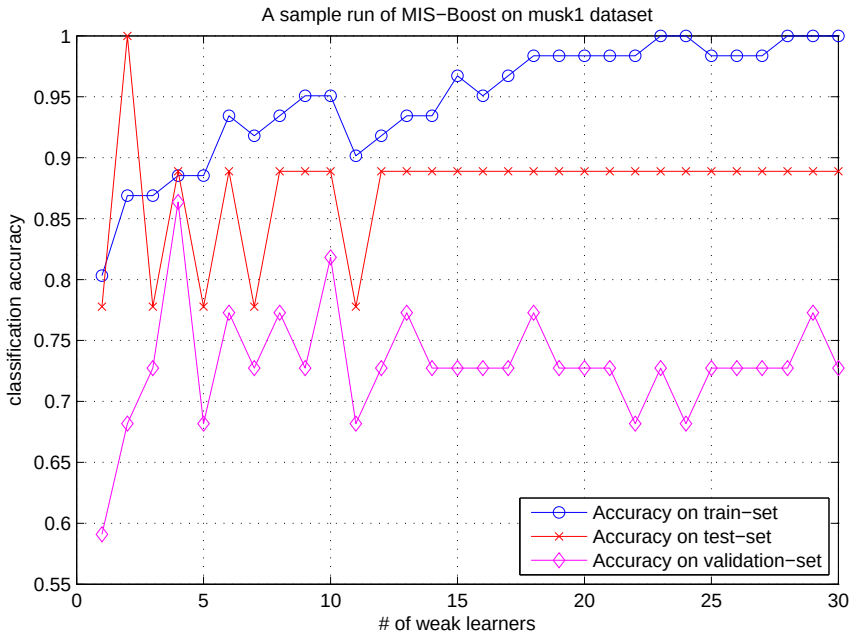


Figure 6.1: A sample run of MIS-Boost on the *musk1* dataset. Note that the best validation accuracy is obtained at 4 weak learners where the test-set classification accuracy is 88.9%. Adding further weak learners does not improve the performance on either the validation-set nor the test-set.

Table 6.5: Percent classification accuracies on the COREL-1000 and COREL-2000 datasets.

Method	COREL-1000	COREL-2000
MIS-Boost	**84.2**	**70.8**
MILIS[94]	83.8	70.1
MILES[92]	82.3	68.7
MIForest[91]	82	69

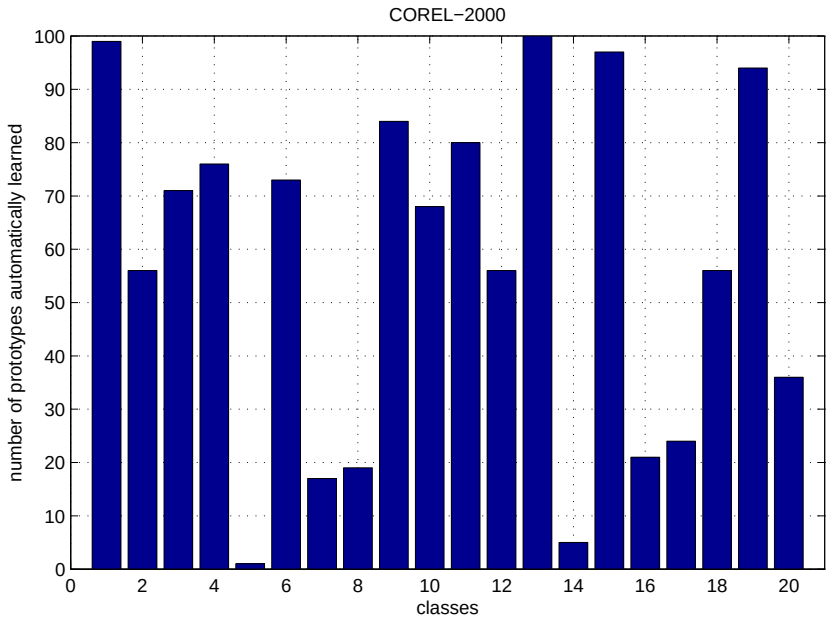


Figure 6.2: Number of base learners, or prototypes, per class as determined by MIS-Boost on the COREL-2000 dataset.

6.4.3 PASCAL VOC 2007

The image categorization task of PASCAL VOC is inherently a MIL problem since the label of an image indicates the existence of at least one object of that label class within the image. However, to the best of our knowledge, no MIL results have been reported on this dataset. This is probably because of the large number of images (10^4) it contains. If we assume that each image has a few hundred instances, then the total number of instances is in the order of millions. Instance selection based methods like MILES would easily run into memory problems. In this section, we evaluate the performance of MIS-Boost on this large-scale image classification dataset, and visualize the selected instances, i.e. those instances that are closest to the learned prototypes, to see if they overlap with the object(s) of interest. To this end, we run MIS-Boost on three selected classes from the dataset, "aeroplane", "bicycle", and "tvmonitor".

To decide what type of instances (or features) to use, we did preliminary experiments with SIFT keypoint/descriptors [98] and regions obtained by the segmentation algorithm [43]. Regions gave better results (0.55 average-

precision (AP))[2] than the SIFT descriptors (0.38 AP) on the "aeroplane" class, so we decided to use regions.

This dataset is not only huge but also unbalanced. The "aeroplane" class has 442 positive vs. 9518 negative; the "bicycle" class has 482 positive vs. 9458 negative; and "tvmonitor" has 485 positive vs. 9429 negative images (bags). This imbalance makes finding discriminative instances in the positive bags, among the clutter from the relatively large number of negative bags, quite challenging.

MIS-Boost yields an average-precision (AP) score of 0.55 for the "aeroplane" class. This score is significantly below the state-of-the-art (e.g. 0.76 in [99]) on that dataset. We believe that this discrepancy is largely due to the fact that MIL methods do not model the context (or the background), while [99] and other similar approaches do so. Although the MIL approach seems to be the most appropriate one for the PASCAL dataset (given how the ground-truth is formed), these results suggest that the context/background information is highly discriminative. Another reason for the discrepancy might be the instance features we use; namely, the segmentation might fail to capture the object or its parts. MIS-Boost yields an AP of 0.28 on "bicycle" (compared to 0.65 in [99]) and 0.36 on "tvmonitor" (compared to 0.52 in [99]).

Next, we visualize the selected instances in each image that are closest to the learned prototypes. Figure 6.3 gives examples of true positives from the aeroplane class. On each image, we show the top three instances, i.e. regions, that are closest to the top 3 prototypes learned by MIS-Boost. These regions make the highest contribution to the correct classification of their images. Similarly, Figure 6.4 and Figure 6.5 present the same for the "bicycle" and "tvmonitor" classes, respectively. As one can observe from the images, the most discriminative regions usually overlap with the object of interest. Occasionally, some wrong instances are selected as shown in the lower-left and lower-right images of Figure 6.5. Apparently, the "tvmonitor" classifier learned square-shaped or frame-shaped prototypes, and the window in the lower-left image and the door in the lower-right image are good selections for this prototype.

[2]This is the standard performance measure used in PASCAL VOC.

Figure 6.3: Example true positives from the "aeroplane" class (i.e. these images contain at least one instance of aeroplane). On each image, the three regions that are most similar to the top three prototypes learned by MIS-Boost are shown with yellow boundaries. (Best viewed in color.)

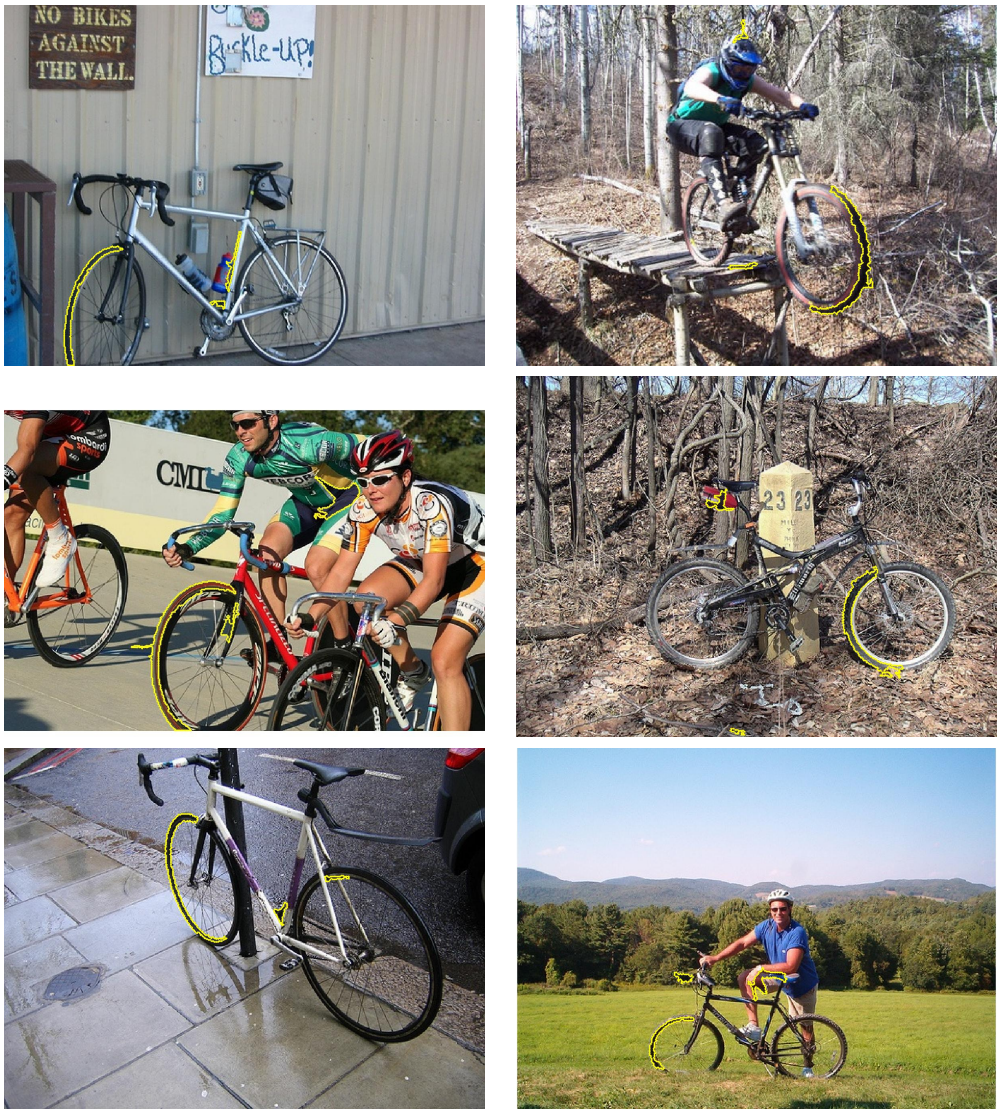


Figure 6.4: Example true positives from the "bicycle" class. See caption of Figure 6.3 for an explanation of the yellow boundaries. (Best viewed in color.)

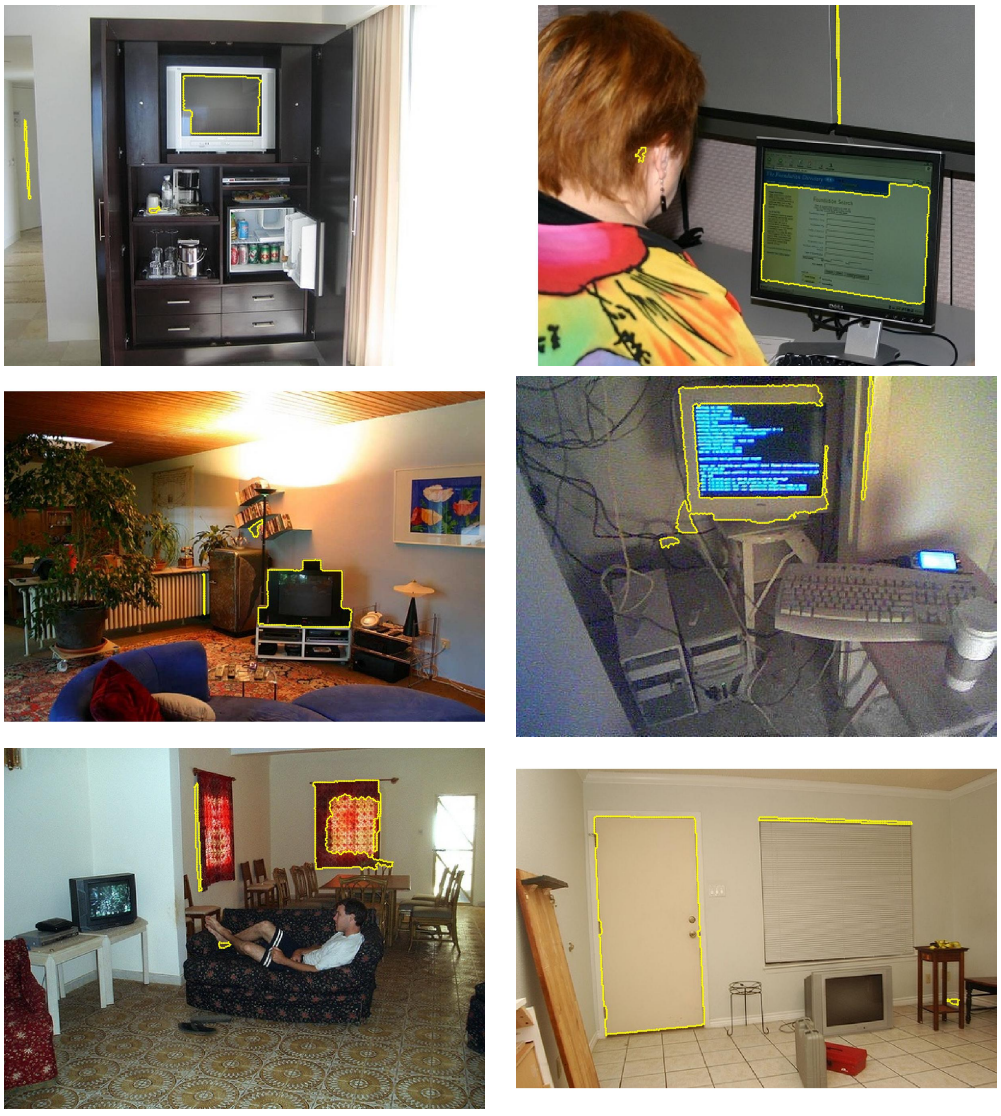


Figure 6.5: Example true positives from the "tvmonitor" class. See caption of Figure 6.3 for an explanation of the yellow boundaries. (Best viewed in color.)

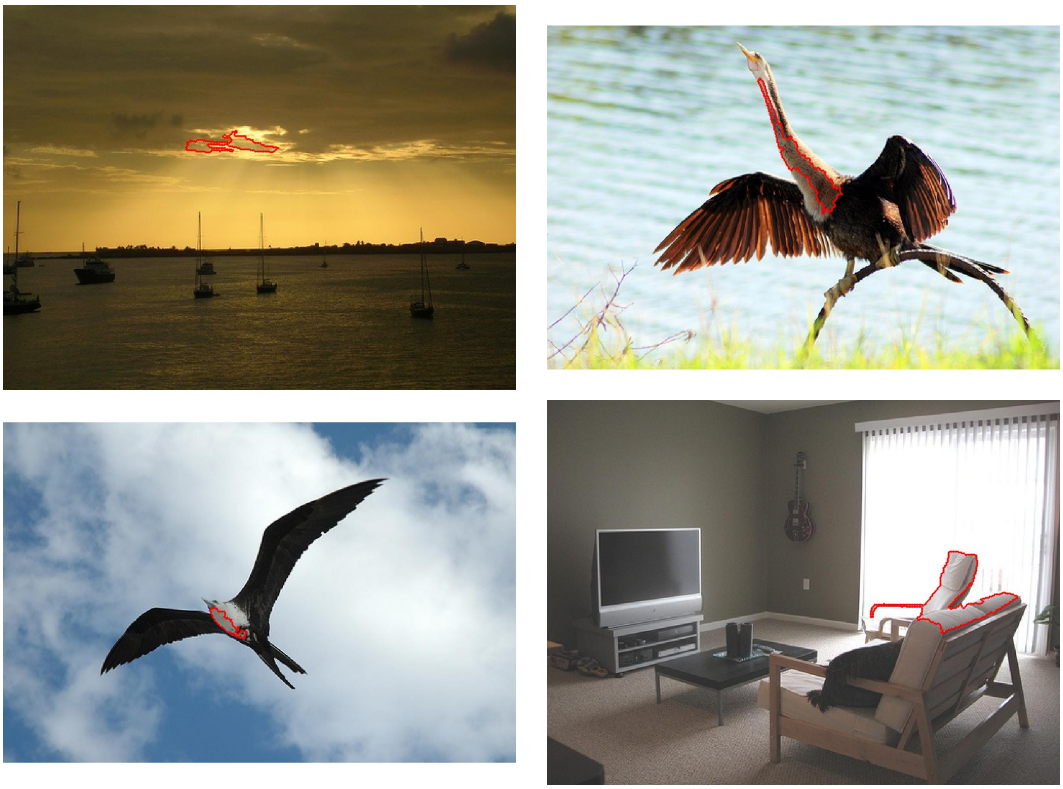


Figure 6.6: Example false positive predictions by MIS-Boost or the "aeroplane" class. On each image, we show with red boundaries the region closest to the first prototype learned by MIS-Boost.

We also give some false-positive prediction examples for the "aeroplane" class in Figure 6.6. The regions that are most similar to the first prototype learned by MIS-Boost are visually similar to the regions marked for true positive predictions given in Figure 6.3.

6.5 Summary

We presented a new multiple instance learning (MIL) method that learns discriminative instance prototypes by explicit instance selection in a boosting framework. We argued that the following three design choices and/or assumptions restrict the capacity of a MIL method: **(i)** treating the prototype learning/choosing step and learning the final bag classifier independently, **(ii)** restricting the prototypes to a discrete set of instances from the training set, and **(iii)** restricting the number of selected-instances per bag. Our method,

MIS-Boost, overcomes all three restrictions by learning prototype-based base classifiers that are linearly boosted. At each iteration of MIS-Boost, a prototype is learned so that it maximally discriminates between the positive and negative bags, which are themselves weighted according to how well they were discriminated in earlier iterations. The number of total prototypes and the number of selected-instances per bag are determined in a completely data-driven way. We showed that our method outperforms state-of-the-art MIL methods on a number of benchmark datasets. We also applied MIS-Boost to large-scale image classification, where we showed that the automatically selected prototypes map to visually meaningful image regions.

CHAPTER 7

CONCLUSIONS

We presented our work about extracting as well as making use of the structure and hierarchy present in images. We started with developing a new low-level, multiscale, hierarchical image segmentation algorithm which is designed to detect image regions regardless of their shapes, sizes, and levels of interior homogeneity, with minimal user input. Our region model is a connected set of pixels that is surrounded by ramp edge discontinuities where the magnitude of discontinues is large compared to the variation inside the region. Each region is associated with a scale depending on the magnitude of the weakest part of its boundary. Traversing through the range of all possible scales, we obtain all regions present in the image. Regions strictly merge as the scale increases, so a tree is formed where the root node corresponds to the whole image; nodes close to the root along a path are large, while their children nodes capture embedded details.

To evaluate the accuracy and precision of our algorithm, as well as to compare it to the existing algorithms, we proposed and developed a new benchmark dataset for low-level image segmentation. In this benchmark, small patches of many images are hand-segmented by human subjects. We provided evaluation methods for both boundary-based and region-based performance of algorithms. We showed that our proposed algorithm performs better than the existing low-level segmentation algorithms on this benchmark. To the best of our knowledge, this is the first benchmark dataset with human segmentation based ground truth for the evaluation of low-level segmentation algorithms.

We used our segmentation algorithm to obtain statistics of natural images and demonstrated how to use these statistics as prior knowledge in two vision tasks: image/scene categorization and semantic image segmentation. First, we compiled segmentation statistics from a large number of natural images.

These statistics confirmed some of the previous findings on statistics of natural images, e.g. dominant orientations are horizontal and vertical, and provided new findings, particular to segmentation, such as that there are more regions in the lower halves of the images than their upper halves. Next, we proposed a Markov random field (MRF) for estimating segmentation-based statistics on unseen images, and we successfully used this model in image categorization and semantic image segmentation.

We next looked at the problem of region correspondence and investigated the importance of different visual cues to describe image regions for the correspondence problem. We designed and developed psychophysical experiments to learn the weights of different cues by evaluating their impact on binocular fusibility by human subjects. Using a head-mounted display, we showed a set of elliptical shapes to one eye and slightly different versions of the same set of ellipses to the other eye of human subjects. We then asked them whether the ellipses fuse or not. By systematically varying the parameters of the elliptical shapes, and testing for fusion, we learned a perceptual distance function between two elliptical shapes. We evaluated this function on ground-truth stereo image pairs.

Finally, we proposed a novel multiple instance learning (MIL) algorithm. The crux of our algorithm is the idea of selecting discriminative instances, which we call prototypes, in a boosting framework. Unlike previous instance selection based MIL methods, we do not restrict the prototypes to a discrete set of training instances but allow them to take arbitrary values in the instance feature space. We also do not restrict the total number of prototypes and the number of selected-instances per bag; these quantities are completely data-driven. We showed that our method outperforms state-of-the-art MIL methods on a number of benchmark datasets. We also applied our method to large-scale image classification, where we showed that the automatically selected prototypes map to visually meaningful image regions.

APPENDIX A

MAXIMUM-MARGIN DISTANCE LEARNING FROM THE APPARENT MOTION METHOD

We define a weighted distance between two regions r_1 and r_2 as:

$$d_w(r_1, r_2) = \sum_{i=1}^{F} w_i^2 (r_{1i} - r_{2i})^2. \tag{A.1}$$

We learn the weight vector w by using the responses of human subjects as follows. For a triple of regions (p, q, r), let p be the target region, and q and r be the left and right distracters, respectively. A human subject response comes in the form:

"Region p is closer to region q than it is to region r."

So, any feasible w should satisfy the above answer:

$$d_w(p, q) < d_w(q, r). \tag{A.2}$$

Let us expand (A.2):

$$\sum_{i=1}^{F} w_i^2 (p_i - q_i)^2 - \sum_{i=1}^{F} w_i^2 (p_i - r_i)^2 > 0. \tag{A.3}$$

Let $x_i^{(p,q,r)} = ((p_i - q_i)^2 - (p_i - r_i)^2)$ and $\omega_i = w_i^2$. Then, (A.3) can be written as:

$$\omega^T x^{(p,q,r)} > 0 \tag{A.4}$$

which is a formal way of saying that p is close to q than it is to r.

For a given region triplet (p, q, r), we can see (A.4) as a decision function on $x^{(p,q,r)}$:

$$f(x) = sign(y(x)) = sign(\omega^T x) = \begin{cases} +1 & \text{if } \omega, \text{x satisfies (A.2)} \\ -1 & \text{else} \end{cases} \tag{A.5}$$

where $y(x) = 0$ is the decision hyperplane.

Let us assume that for the set of given human subject answers $A = \{x^{(p,q,r)} |\ p \text{ is closer to } q \text{ than to } r\}$, there exists a ω which satisfies (A.4) for all $x \in A$.

The margin of x, i.e. the distance of x to the decision hyperplane $y(x) = \omega^T x = 0$, is:

$$\frac{\omega^T x}{||\omega||}. \tag{A.6}$$

Maximum margin solution is the one which maximizes the minimum margin over all $x \in A$:

$$\arg\max_{\omega} \{\frac{1}{||w||} \ \min_{x} \ \omega^T x\}. \tag{A.7}$$

Note that scaling of $\omega \rightarrow k\omega$ does not change the margin (A.6) of any point x. This fact is used to simplify the optimization in (A.7) by setting $\omega^T x = 1$ for the point x that is closest to the decision hyperplane. Then, (A.7) becomes:

$$\arg\min_{\omega} ||w|| \ \text{ s.t. } \ \omega^T x \geq 1 \quad \forall x \in A. \tag{A.8}$$

which is equivalent to:

$$\arg\min_{\omega} \{||w|| + c \sum \xi_i\} \quad \text{s.t.} \quad \omega^T x_i \geq 1 - \xi_i \ \ , \xi_i \geq 0 \ \forall x \in A \tag{A.9}$$

This is a convex function and it is in the same form as the cost of a soft-margin support vector machine problem. Gradient descent methods can be

used to minimize it. If speed is a problem, then stochastic methods can be used as in the Pegasos algorithm [100].

APPENDIX B

CREATING ANAGLYPH IMAGES FROM STEREO IMAGE PAIRS

Anaglyph images are specially designed to create stereoscopic 3D effects. When an anaglyph image is seen appropriately, that is using a pair of anaglyph filters, a sense of 3D depth is perceived in the image. A common and widely used example for anaglyph filters is a pair of red-cyan glasses (see Figure B.1).

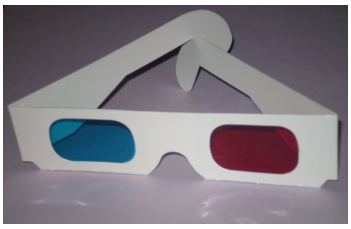


Figure B.1: Red-cyan glasses, a widely used anaglyph filter.

Suppose we have two separate images for the left eye's view and right eye's view. To create an anaglyph image, we need to set the green and the blue channels of the right image, and the red channel of the left image to zeros. The sum of the resulting two images gives the anaglyph image.

Here is a MATLAB function to create an anaglyph given a stereo pair of images:

```
function img3D = make_anaglyph(left_image, right_image)
left_image(:,:,1) = 0;
right_image(:,:,2:3) = 0;
img3D = left_image + right_image;
```

In the following we demonstrate its use in a simple example:

```
str = 'Anaglyph_3D_text';

disparity_1st_row = -2;
disparity_3rd_row = +2;

%% left image
x1 = 10 + disparity_1st_row/2;
x2 = 10;
x3 = 10 + disparity_3rd_row/2;

h=figure;
rectangle('position',[0 0 85 50],'linewidth',10);        % frame
text(x1,40,str,'Fontsize',30,'fontweight','bold');       % 1st row
text(x2,25,str,'Fontsize',30,'fontweight','bold');       % 2nd row
text(x3,10,str,'Fontsize',30,'fontweight','bold');       % 3rd row
axis off
f = getframe(h);
left_image = f.cdata;
close(h);

%% right image
x1 = 10 - disparity_1st_row/2;
x2 = 10;
x3 = 10 - disparity_3rd_row/2;

h=figure;
rectangle('position',[0 0 85 50],'linewidth',10);        % frame
text(x1,40,str,'Fontsize',30,'fontweight','bold');       % 1st row
text(x2,25,str,'Fontsize',30,'fontweight','bold');       % 2nd row
text(x3,10,str,'Fontsize',30,'fontweight','bold');       % 3rd row
axis off
f = getframe(h);
right_image = f.cdata;
close(h);

%% create anaglyph image and display it
img = make_anaglyph(left_image, right_image);
figure, imshow(img)
```

which creates the image in Figure B.2.

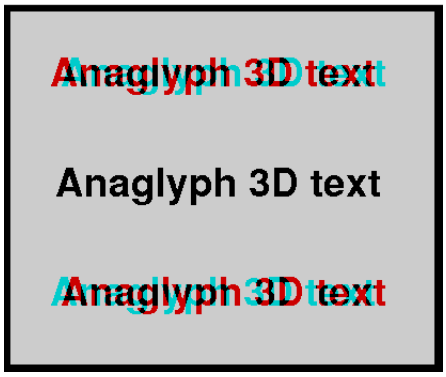


Figure B.2: An anaglyph image produced by the script given on page 125. When viewed using red-cyan glasses, the text at the first row seems closer to the viewer than the frame, whereas the the third row seems to be behind the frame. The frame and the middle row are seen to be at the same level.

APPENDIX C

INSTRUCTIONS FOR SUBJECTS IN THE BINOCULAR FUSION EXPERIMENT

"The images formed in our two eyes are often different. Our visual system can usually overcome this discrepancy and form a coherent percept (Binocular Fusion). In this experiment, we will study WHEN binocular fusion occurs.

You will be wearing a head-mounted display (HMD) device, which presents different images to your left and right eyes. Some of the objects in the image should look fused (stable and coherent), while the others should not fuse (double image or unstable). Your task is to indicate the boundary between the fused and un-fused objects. The fused objects will always be on the left side.

The experiment is divided into 3 parts. Each part starts with a practice session, followed by the real trials. There will be feedback in the practice session to help you distinguish between what fused object and un-fused object look like in that particular block. The un-fused objects may look very different in different parts of the experiment.

You can use your hand to hold the HMD to offset its weight, and fine-adjust the position of the HMD to make sure you can see the on-screen text clearly. You can take a break any time during the experiment."

APPENDIX D

THE SOFT-MINIMUM FUNCTION AND ITS GRADIENT

Let "min" be the function that returns the smallest element of its input vector, $y = [y_1\ y_2\ \ldots\ y_n]^T \in \mathbb{R}^n$:

$$\min(y) = y_k \ \text{ where } \ y_k \leq y_i \ \ \forall i = 1, 2, \ldots, n. \tag{D.1}$$

The term "min(y)" can be approximated by a linear combination of the elements of y:

$$\min(y) \approx \text{soft-min}(y) = \sum_{i=1}^{n} \pi_i y_i, \tag{D.2}$$

where the coefficients are defined as:

$$\pi_i = \frac{e^{-\alpha y_i}}{\sum_{j=1}^{n} e^{-\alpha y_j}}, \ \text{ and } \ \alpha \ \text{ is a large positive constant.} \tag{D.3}$$

In fact, we have:

$$\begin{aligned} \alpha \to \infty \ &\Rightarrow \ \textstyle\sum \pi_i y_i \to \min(y) \\ \alpha \to -\infty \ &\Rightarrow \ \textstyle\sum \pi_i y_i \to \max(y) \\ \alpha = 0 \ &\Rightarrow \ \textstyle\sum \pi_i y_i = \text{mean}(y). \end{aligned} \tag{D.4}$$

D.1 Gradient of Soft-Min

$$\frac{d\text{soft-min}(y)}{dx} = \sum_{i=1}^{n} \left(\frac{d\pi_i}{dx} y_i + \pi_i \frac{dy_i}{dx} \right) \tag{D.5}$$

$$
\begin{aligned}
\frac{d\pi_i}{dx} &= \frac{-\left(\alpha e^{-\alpha y_i}\frac{dy_i}{dx}\right)\left(\sum_j e^{-\alpha y_i}\right) + e^{-\alpha y_i}\left(\sum_j \alpha e^{-\alpha y_j}\frac{dy_i}{dx}\right)}{\left(\sum_j e^{-\alpha y_j}\right)^2} \\
&= -\alpha\pi_i\frac{dy_i}{dx} + \pi_i\frac{\sum_j \alpha e^{-\alpha y_j}\frac{dy_i}{dx}}{\sum_j e^{-\alpha y_j}} \qquad \text{(D.6)} \\
&= \alpha\pi_i\left(\sum_j \pi_j\frac{dy_j}{dx} - \frac{dy_i}{dx}\right).
\end{aligned}
$$

APPENDIX E

GRADIENT OF WEAK CLASSIFIER COST

Partial derivative of $\tilde{\varepsilon}_m$ (Equation (6.8) on page 107) with respect to β_1:

$$\begin{aligned}\frac{\partial\tilde{\varepsilon}_m}{\partial\beta_1} &= 2\sum_{i=1}^{N} w_i \left(f_m(B_i) - y_i\right) \frac{\partial f_m(B_i)}{\partial\beta_1} \\ &= \sum_{i=1}^{N} w_i \left(f_m(B_i) - y_i\right) \left(f_m(B_i) + 1\right)\left(1 - f_m(B_i)\right)\tilde{D}(p_m, B_i).\end{aligned} \tag{E.1}$$

Partial derivative w.r.t. β_0 is similar.

Partial derivative w.r.t. p_m is as follows.

$$\begin{aligned}\frac{\partial\tilde{\varepsilon}_m}{\partial p_m} &= 2\sum_{i=1}^{N} w_i \left(f_m(B_i) - y_i\right) \frac{\partial f_m(B_i)}{\partial p_m} \\ &= \sum_{i=1}^{N} w_i \left(f_m(B_i) - y_i\right) \left(f_m(B_i) + 1\right)\left(1 - f_m(B_i)\right)\beta_1 \frac{\partial\tilde{D}(p_m, B_i)}{\partial p_m},\end{aligned} \tag{E.2}$$

where the last term, $\frac{\partial\tilde{D}(p_m,B_i)}{\partial p_m}$, can be written using the gradient given in Appendix D.

REFERENCES

[1] Y. Ostrovsky, E. Meyers, S. Ganesh, U. Mathur, and P. Sinha, "Visual parsing after recovery from blindness," *Psychological Science: A Journal of the American Psychological Society / APS*, vol. 20, no. 12, pp. 1484–91, Dec. 2009. [Online]. Available: http://pss.sagepub.com/content/20/12/1484.full

[2] N. Ahuja, "A transform for multiscale image segmentation by integrated edge and region detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 18, no. 12, pp. 1211–1235, 1996.

[3] M. Everingham, L. van Gool, C. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes challenge 2007 (VOC 2007) results," 2007, (Last accessed March 2010). [Online]. Available: http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/

[4] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context," *International Journal of Computer Vision (IJCV)*, vol. 81, no. 1, pp. 2–23, Dec. 2009. [Online]. Available: http://www.springerlink.com/content/71556x2n0809ql11/

[5] P. K. Sahoo, S. Soltani, A. K. Wong, and Y. C. Chen, "A survey of thresholding techniques," *Comput. Vision Graph. Image Process.*, vol. 41, no. 2, pp. 233–260, 1988.

[6] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions," in *British Machine Vision Conference (BMVC)*, P. L. Rosin and A. D. Marshall, Eds. British Machine Vision Association, 2002.

[7] G. Stockman and L. G. Shapiro, *Computer Vision*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[8] C. Carson, S. Belongie, H. Greenspan, and J. Malik, "Blobworld: Image segmentation using expectation-maximization and its application to

image querying," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 24, no. 8, pp. 1026–1038, 2002.

[9] D. Comaniciu, V. Ramesh, and P. Meer, "The variable bandwidth mean shift and data-driven scale selection," in *International Conference on Computer Vision (ICCV)*, 2001, pp. 438–445.

[10] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 24, no. 5, pp. 603–619, 2002.

[11] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[12] R. Zabih and V. Kolmogorov, "Spatially coherent clustering using graph cuts," in *Computer Vision and Pattern Recognition*, 2004, pp. 437–444.

[13] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 22, no. 8, pp. 888–905, 2000.

[14] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision (IJCV)*, vol. 59, no. 2, pp. 167–181, 2004.

[15] R. M. Haralick and L. G. Shapiro, "Survey- image segmentation techniques," *Computer Vision Graphics and Image Processing*, vol. 29, pp. 100–132, 1985.

[16] F. Meyer and S. Beucher, "Morphological segmentation," *Journal of Visual Communication and Image Representation*, vol. 1, no. 1, pp. 21–46, 1990.

[17] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 6, no. 11, pp. 721–741, 1984.

[18] C. Xu and J. L. Prince, "Snakes, shapes, and gradient vector flow," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 359–369, 1998.

[19] H. Arora and N. Ahuja, "Analysis of ramp discontinuity model for multiscale image segmentation," in *International Conference on Pattern Recognition (ICPR)*, 2006, pp. 99–103.

[20] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *International Conference on Computer Vision (ICCV)*, vol. 2, July 2001, pp. 416–423.

[21] A. Rosenfeld, R. A. Hummel, and S. W. Zucker, "Scene labeling by relaxation operations," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 6, no. 6, pp. 420–433, 1976. [Online]. Available: http://dx.doi.org/10.1109/TSMC.1976.4309519

[22] V. Kann, S. Khanna, J. Lagergren, and A. Panconesi, "On the hardness of approximating Max k-Cut and its dual," *Chicago Journal of Theoretical Computer Science*, vol. 1997, no. 2, June 1997, [Note to the ECE Pubs. Office: there isn't any page numbers given for this article]. [Online]. Available: http://cjtcs.cs.uchicago.edu/articles/1997/2/contents.html

[23] P. Crescenzi and V. Kann, "Minimum edge deletion k-partition," March 2000, *A compendium of NP optimization problems*, (Last accessed April 2010). [Online]. Available: http://www.csc.kth.se/~viggo/wwwcompendium/node43.html

[24] J. H. Elder and S. W. Zucker, "Local scale control for edge detection and blur estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 20, no. 7, pp. 699–716, 1998.

[25] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 26, no. 5, pp. 530–549, 2004.

[26] M. Meila, "Comparing clusterings by the variation of information," in *Proc. Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory (COLT) and 7th Kernel Workshop*, 2003, pp. 173–187.

[27] T. Cour, F. Benezit, and J. Shi, "Spectral segmentation with multiscale graph decomposition," in *Conf. on Computer Vision and Pattern Recognition (CVPR)*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 1124–1131. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2005.332

[28] A. Hyvärinen, J. Hurri, and P. O. Hoyer, *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision.* Springer Computational Imaging and Vision, Vol. 39, 2009. [Online].

Available: http://www.springer.com/computer/image+processing/book/978-1-84882-490-4

[29] S. Roth and M. J. Black, "Fields of Experts," *International Journal of Computer Vision (IJCV)*, vol. 82, no. 2, pp. 205–229, Jan. 2009. [Online]. Available: http://www.springerlink.com/content/l185w1g412788vj1/

[30] Y. Weiss and W. T. Freeman, "What makes a good model of natural images?" in *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2007, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4270117

[31] A. Torralba and A. Oliva, "Statistics of natural image categories." *Network: Computation in Neural Systems*, vol. 14, no. 3, pp. 391–412, Aug. 2003. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/12938764

[32] A. Torralba, "Contextual Priming for Object Detection," *International Journal of Computer Vision (IJCV)*, vol. 53, no. 2, pp. 169–191–191, 2003. [Online]. Available: http://www.springerlink.com/content/l81xhl05464u1473/

[33] D. Ruderman, "The statistics of natural images," *Network: Computation in Neural Systems*, vol. 5, pp. 517–548, 1994.

[34] L. Zhang, M. H. Tong, T. K. Marks, H. Shan, and G. W. Cottrell, "SUN: A Bayesian framework for saliency using natural statistics." *Journal of vision*, vol. 8, no. 7, pp. 32.1–20, Jan. 2008. [Online]. Available: http://www.journalofvision.org/content/8/7/32.full

[35] N. Hees, C. K. I. Williams, and G. E. Hinton, "Learning generative texture models with extended Fields-of-Experts," in *British Machine Vision Conference (BMVC)*, 2009.

[36] A. Srivastava, A. Lee, E. Simoncelli, and S.-C. Zhu, "On Advances in Statistical Modeling of Natural Images," *Journal of Mathematical Imaging and Vision*, vol. 18, no. 1, pp. 17–33–33, 2003. [Online]. Available: http://www.springerlink.com/content/jt354188q4685l29/

[37] D. Zoran and Y. Weiss, "Scale invariance and noise in natural images," in *Conf. on Computer Vision and Pattern Recognition (CVPR)*, Sep. 2009, pp. 2209–2216. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5459476

[38] J. Huang and D. Mumford, "Statistics of Natural Images and Models," in *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1999, pp. 637–663.

[39] D. M. Coppola, H. R. Purves, A. N. McCoy, and D. Purves, "The distribution of oriented contours in the real world," *Proceedings of the National Acedemy of Sciences (PNAS)*, vol. 95, no. 7, pp. 4002–4006, 1998. [Online]. Available: http://www.pnas.org/cgi/content/abstract/95/7/4002

[40] S. Roth and M. J. Black, "Fields of Experts: A Framework for Learning Image Priors," in *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 860–867. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1467533

[41] T. S. Cho, N. Joshi, C. L. Zitnick, S. B. Kang, R. Szeliski, and W. T. Freeman, "A content-aware image prior," in *Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2010, pp. 169–176. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5540214

[42] S. Todorovic and N. Ahuja, "Unsupervised category modeling, recognition, and segmentation in images." *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 30, no. 12, pp. 2158–74, Dec. 2008. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4441718

[43] E. Akbas and N. Ahuja, "From Ramp Discontinuities to Segmentation Tree," in *Asian Conference on Computer Vision (ACCV)*. Xi'an, China: Springer, 2009, pp. 123 –134.

[44] N. Ahuja and S. Todorovic, "Connected Segmentation Tree: A joint representation of region layout and hierarchy," in *Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2008, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4587626

[45] B. Ghanem, E. Resendiz, and N. Ahuja, "Segmentation-based Perceptual Image Quality Assessment (SPIQA)," in *International Conference on Image Processing (ICIP)*, Oct. 2008, pp. 393–396. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4711774

[46] E. Akbas and N. Ahuja, "Low-Level Image Segmentation Based Scene Classification," in *International Conference on Pattern Recognition (ICPR)*, Aug. 2010, pp. 3623–3626. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5597902

[47] S. Z. Li, *Markov Random Field Modeling in Image Analysis*, 3rd ed. Springer-Verlag, 2009. [Online]. Available: http://www.cbsr.ia.ac.cn/users/szli/mrf_book/book.html

[48] T. S. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Advances on Neural Information Processing (NIPS)*, 1999, pp. 470–476. [Online]. Available: http://portal.acm.org/citation.cfm?id=340715

[49] F. Perronnin and C. Dance, "Fisher Kernels on Visual Vocabularies for Image Categorization," in *Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2007, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4270291

[50] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *European Conference on Computer Vision (ECCV)*, 2010, pp. 143–156.

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

[52] H. Harzallah, F. Jurie, and C. Schmid, "Combining efficient object localization and image classification," in *International Conference on Computer Vision (ICCV)*, 2009, pp. 237–244.

[53] J. Yang, Y. Li, Y. Tian, L. Duan, and W. Gao, "Group-sensitive multiple kernel learning for object categorization," in *International Conference on Computer Vision (ICCV)*, 2009, pp. 436–443.

[54] J. C. van Gemert, C. J. Veenman, A. W. M. Smeulders, and J. M. Geusebroek, "Visual word ambiguity," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 32, no. 7, pp. 1271–1283, 2010. [Online]. Available: http://www.science.uva.nl/research/publications/2010/vanGemertTPAMI2010

[55] A. Oliva and A. Torralba, "Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope," *International Journal of Computer Vision (IJCV)*, vol. 42, no. 3, pp. 145–175, May 2001. [Online]. Available: http://www.springerlink.com/content/k62tg81w8352g71h

[56] L. Fei-Fei and P. Perona, "A Bayesian Hierarchical Model for Learning Natural Scene Categories," in *Conf. on Computer Vision and Pattern Recognition (CVPR)*, vol. 2. IEEE, 2005, pp. 524–531. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1467486

[57] A. Bosch, A. Zisserman, and X. Muñoz, "Scene Classification Via pLSA," in *European Conference on Computer Vision (ECCV)*, ser. LNCS, 2006, pp. 517–530. [Online]. Available: http://www.springerlink.com/index/10.1007/11744085

[58] J. Vogel and B. Schiele, "Semantic Modeling of Natural Scenes for Content-Based Image Retrieval," *International Journal of Computer Vision (IJCV)*, vol. 72, no. 2, pp. 133–157, 2007. [Online]. Available: http://www.springerlink.com/index/10.1007/s11263-006-8614-1

[59] A. Bosch, X. Muñoz, and A. Zisserman, "Scene classification using a hybrid generative/discriminative approach." *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 30, no. 4, pp. 712–27, 2008. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/18276975

[60] J.-L. Gauvain and L. Chin-Hui, "Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains," *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 2, pp. 291–298, 1994. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=279278

[61] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1-3, pp. 19–41, 2000. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1051200499903615

[62] F. Perronnin, "Universal and adapted vocabularies for generic visual categorization," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 30, no. 7, pp. 1243–56, 2008. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/18550906

[63] L. Yan and F. Perronnin, "A similarity measure between unordered vector sets with application to image categorization," in *Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2008, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4587600

[64] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions," *Bulletin of the Calcutta Mathematical Society*, vol. 35, pp. 99 – 109, 1943.

[65] Y. Chang Huai, L. Kong Aik, and L. Haizhou, "A GMM supervector Kernel with the Bhattacharyya distance for SVM based speaker recognition," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2009, pp. 4221–4224. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4960560

[66] J. D. H. Farquhar, S. Szedmak, H. Meng, and J. Shawe-Taylor, "Improving "bag-of-keypoints" image categorisation: Generative models and pdf-kernels," University of Southampton, Tech. Rep., 2005.

[67] P. J. Moreno, P. Ho, and N. Vasconcelos, "A kullback-leibler divergence based kernel for svm classification in multimedia applications," in *NIPS*, S. Thrun, L. K. Saul, and B. Schölkopf, Eds. MIT Press, 2003.

[68] M. Z. Brown, D. Burschka, and G. D. Hager, "Advances in computational stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 25, pp. 993–1008, 2003.

[69] A. S. Ogale, "The compositional character of visual correspondence," Ph.D. dissertation, University of Maryland at College Park, College Park, MD, USA, 2004, aAI3139114.

[70] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision (IJCV)*, vol. 47, pp. 7–42, April 2002. [Online]. Available: http://portal.acm.org/citation.cfm?id=598429.598475

[71] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *Conf. on Computer Vision and Pattern Recognition (CVPR)*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 519–528. [Online]. Available: http://portal.acm.org/citation.cfm?id=1153170.1153518

[72] M. Tabb and N. Ahuja, "Multiscale image segmentation by integrated edge and region detection," *IEEE Transactions on Image Processing*, vol. 6, no. 5, pp. 642–655, 1997.

[73] A. Al-Hamadi, R. Niese, and B. Michaelis, "Feature-based correspondence analysis in color image sequences," in *Computer Vision and Graphics*, ser. Computational Imaging and Vision, K. Wojciechowski, B. Smolka, H. Palus, R. Kozera, W. Skarbek, and L. Noakes, Eds. Springer Netherlands, 2006, vol. 32, pp. 179–186, 10.1007/1-4020-4179-9_26. [Online]. Available: http://dx.doi.org/10.1007/1-4020-4179-9_26

[74] I. K. Sethi, N. V. Patel, and J. H. Yoo, "A general approach for token correspondence," *Pattern Recognition*, vol. 27, no. 12, pp. 1775 – 1786, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/B6V14-48MXN18-3W/2/436af01332b594f5037694faf0c95c1a

[75] N. Prins, "Correspondence matching in long-range apparent motion precedes featural analysis," *Perception*, vol. 37, no. 7, pp. 1022 – 1036, 2008. [Online]. Available: http://www.perceptionweb.com/abstract.cgi?id=p5945

[76] J. Keeler, D. Rumelhart, and W. Leow, "Integrated segmentation and recognition of hand-printed numerals," in *Advances on Neural Information Processing (NIPS)*, 1990, pp. 557–563.

[77] S. Andrews and I. Tsochantaridis, "Support vector machines for multiple-instance learning," in *Advances on Neural Information Processing (NIPS)*, 2002, pp. 561–568.

[78] P.-M. Cheung and J. T. Kwok, "A regularization framework for multiple-instance learning," *International Conference on Machine Learning (ICML)*, no. 1, pp. 193–200, 2006.

[79] Z.-H. Zhou and J.-M. Xu, "On the relation between multi-instance learning and semi-supervised learning," *International Conference on Machine Learning (ICML)*, no. 1997, pp. 1167–1174, 2007.

[80] T. G. Dietterich, R. H. Lathrop, and T. Lozano-perez, "Solving the Multiple-Instance Problem with Axis-Parallel Rectangles," *Artificial Intelligence*, vol. 89, pp. 31–71, 1997.

[81] O. Maron and T. Lozano-Pérez, "A framework for multiple-instance learning," in *Advances on Neural Information Processing (NIPS)*, M. I. Jordan, M. J. Kearns, and S. A. Solla, Eds. The MIT Press, 1997.

[82] Q. Zhang and S. Goldman, "EM-DD: An improved multiple-instance learning technique," *Advances on Neural Information Processing (NIPS)*, vol. 2, pp. 1073–1080, 2002.

[83] R. Rahmani, S. Goldman, H. Zhang, S. R. Cholleti, and J. E. Fritts, "Localized content-based image retrieval." *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 30, no. 11, pp. 1902–12, 2008.

[84] J. Wang and J.-D. Zucker, "Solving the multiple-instance problem: A lazy learning approach," in *International Conference on Machine Learning (ICML)*, P. Langley, Ed. Morgan Kaufmann, 2000, pp. 1119–1126.

[85] T. Gartner, P. Flach, A. Kowalczyk, and A. Smola, "Multi-instance kernels," in *International Conference on Machine Learning (ICML)*, 2002, pp. 179–186.

[86] Z.-H. Zhou, Y.-Y. Sun, and Y.-F. Li, "Multi-instance learning by treating instances as non-I.I.D. samples," *International Conference on Machine Learning (ICML)*, pp. 1–8, 2009.

[87] Y. Chen and J. Wang, "Image categorization by learning and reasoning with regions," *Journal of Machine Learning Research (JMLR)*, vol. 5, pp. 913–939, 2004.

[88] P. A. Viola, J. C. Platt, and C. Zhang, "Multiple instance boosting for object detection," in *Advances on Neural Information Processing (NIPS)*, 2005, pp. 1–8.

[89] S. Cholleti, S. Goldman, and R. Rahmani, "MI-Winnow: A New Multiple-Instance Learning Algorithm," *International Conference on Tools with Artificial Intelligence*, pp. 336–346, 2006.

[90] Z. Fu, "Fast multiple instance learning via l1,2 logistic regression," *International Conference on Pattern Recognition (ICPR)*, pp. 1–4, 2008.

[91] C. Leistner, A. Saffari, and H. Bischof, "MIForests: Multiple-Instance Learning with Randomized Trees," *European Conference on Computer Vision (ECCV)*, pp. 29–42, 2010.

[92] Y. Chen, J. Bi, and J. Z. Wang, "MILES: multiple-instance learning via embedded instance selection." *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 28, no. 12, pp. 1931–47, 2006.

[93] Z. Fu and A. Robles-Kelly, "An instance selection approach to Multiple Instance Learning," in *Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2009, pp. 911–918.

[94] Z. Fu, A. Robles-Kelly, and J. Zhou, "MILIS: Multiple Instance Learning with Instance Selection." *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 33, no. 10, pp. 1–20, 2010.

[95] J. Foulds and E. Frank, "Revisiting multiple-instance learning via embedded instance selection," in *Proceedings of the 21st Australasian Joint Conference on Artificial Intelligence: Advances in Artificial Intelligence*, ser. AI '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 300–310. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-89378-3_29

[96] J. Friedman, T. Hastie, and R. Tibshirani, "Additive Logistic Regression: A Statistical View of Boosting," *The Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000. [Online]. Available: http://projecteuclid.org/DPubS?service=UI\&version=1.0\&verb=Display\&handle=euclid.aos/1016218223

[97] P. V. Gehler and O. Chapelle, "Deterministic annealing for multiple-instance learning," in *11th International Conference on Artificial Intelligence and Statistics*, M. Meila and X. Shen, Eds., 03 2007, pp. 123–130.

[98] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision (IJCV)*, no. 2, pp. 91 – 110, 2004.

[99] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the Fisher Kernel for Large-Scale Image Classification," in *European Conference on Computer Vision (ECCV)*, vol. 6314. Springer Berlin / Heidelberg, 2010, pp. 143–156–156. [Online]. Available: http://www.springerlink.com/content/151r875374318254/

[100] S. Shalev-Shwartz, Y. Singer, and N. Srebro, "Pegasos : Primal Estimated sub-GrAdient SOlver for SVM," in *International Conference on Machine Learning (ICML)*. ACM, 2007, pp. 807–814. [Online]. Available: http://portal.acm.org/citation.cfm?id=1273598