PROBABILISTIC MODELS FOR REGION-BASED SCENE UNDERSTANDING

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Stephen Gould
June 2010

Abstract

One of the long-term goals of computer vision is to be able to understand the world through visual images. This daunting task involves reasoning simultaneously about objects, regions and the 3D geometric relationship between them. Traditionally, computer vision research has tackled this reasoning in isolation: independent detectors for finding objects, image segmentation algorithms for defining regions, and specialized monocular depth perception methods for reconstructing geometry. This method of isolated reasoning, however, can lead to inconsistent interpretations of the scene. For example, one algorithm may reason that a particular bounding box contains an object while another algorithm reasons that those same pixels be labeled something completely different, e.g., sky or grass.

In this thesis we present a unified probabilistic model that avoids these inconsistencies. We develop our model in stages: First, we introduce a region-based representation of the scene in which pixels are grouped together into consistent regions, and each region is annotated with a semantic and geometric class label. Our semantic labels consist of a number of background classes (e.g., sky, grass, road) and a single generic foreground object class. Importantly, all pixels are uniquely explained by this representation. Our model defines an *energy function*, or factored scoring function, over this representation which allows us to evaluate the quality of any given decomposition of the scene. We present an efficient move-making algorithm that allows us to perform approximate maximum a posteriori (MAP) inference in this model given a novel image.

Next, we extend our model by introducing the concept of objects. Here, we build on the region-based representation by allowing foreground regions to be grouped together into coherent objects. For example, grouping separate head, torso and legs regions into a single person object. We add corresponding terms to our energy function that allow us to recognize these objects and model contextual relationships between objects and adjacent regions. For example, we would expect to find cars on road and boats on water. We also extend our inference algorithm to allow for larger top-down moves to be evaluated.

Last, we show how our region-based approach can be used to interpret the 3D structure of the scene and estimate depth, resulting in a complete understanding of regions, objects, and geometry.

For Lesley.

Acknowledgments

I feel truly privileged to have had the opportunity to work with Daphne Koller, who always puts the needs of her students above anything else. Thank you, Daphne, for your encouragement and support, but most importantly the inspiration that you have provided me over the past five years. Your technical brilliance, research insight, and professional standards make you an ideal role model and mentor.

I am also profoundly grateful to Andrew Ng, with whom I worked during the first few years of my Ph.D., and who has continued to offer guidance and show interest in my work. Thank you, Andrew, for your generous advice over the years.

I would like to thank Stephen Boyd, who was my Masters academic advisor (way back in 1997) and has been an inspiration to me ever since. His casual style combined with technical clarity have been a great example to me. Thank you to Kalanit Grill-Spector for chairing my defense at short notice and for providing a different perspective on my research.

I am indebted to Gal Elidan, Geremy Heitz, and Ben Packer who helped induct me into the DAGS research group and, almost as important, the morning coffee ritual. I will always cherish the memories of our excursions to Peet's Coffee. Thanks also to Pawan Kumar for his guidance during the later stages of my thesis (and for introducing a bit of British tea into our morning coffee).

I also owe a great deal to the many friends and colleagues that I have met at Stanford—Pieter Abbeel, Fernando Amat, Alexis Battle, Vasco Chatalbashev, Gal Chechik, Adam Coates, David Cohen, James Diebel, Yoni Donner, John Duchi, Varun Ganapathi, Tianshi Gao, Vladimir Jojic, Adrian Kaehler, Joni Laserson, Quoc Le, Honglak Lee, Su-in Lee, Beyang Liu, Farshid Moussavi, Suchi Saria, Christina Pop, Morgan Quigley, Jim Rodgers, Olga Russakovsky, Ashutosh Saxena, David Vickrey, Haidong Wang, Huayan Wang—and to the numerous undergraduate and masters students with whom I've worked—Ravi Parikh, Rick Fulton, Evan Rosen, Ben Sapp, Marius Meissner, Ian Goodfellow, Sid Batra, Paul Baumstarck, and many others. Thank you all for the ideas, advice, and contributions you have made to my research. I am also indebted to the many academics and visiting researchers with whom I've interacted over the past several years. In particular, I give warm thanks to Jana Kosecka, Richard Zemel, Gary Bradski and Fei-Fei Li for providing external viewpoints.

Most importantly, I would like to thank my wonderful family. First, to my dad, Arnold, my sister, Hayley, and all my family back in Australia, I thank you for your continued support and encouragement from abroad. Last, to my wife Naomi: the decision to move from our comfortable professional lives in Australia to the life of a "graduate student couple" at Stanford has been an adventure. Thank you, Naomi, for plunging into this adventure with me, for your continued love and support, and for keeping our lives normal. While all I had to do was write some scientific papers and a couple of lines of code, you had the formidable task of raising a family—a task at which you have excelled. To my two gorgeous daughters, Ariella and Bronte, I thank you for your smiles, your laughs, your 4am wake-ups, your tantrums, and your questions—as a scientist there is no better question to hear than "Why, daddy?". When you get older, I hope you appreciate how much all of these mean to me.

Contents

Abstract **v**

Acknowledgments **ix**

1 Introduction **1**

1.1 Contributions 5

1.2 Thesis Outline 7

1.3 Related Work 9

1.4 Previously Published Work 17

2 Background **19**

2.1 Machine Learning and Graphical Models 19

2.1.1 Classification and Regression 20

2.1.2 Probabilistic Graphical Models 25

2.2 Computer Vision 32

2.2.1 Image Formation 32

2.2.2 Low Level Image Features 35

2.2.3 Sliding-Window Object Detection 38

2.2.4 Image Segmentation 46

2.3 Chapter Summary 51

3 Image Datasets and Dataset Construction **53**

3.1 Standard Image Datasets 54

3.2 Stanford Background Dataset 58

3.2.1 Amazon Mechanical Turk 58
3.3 Chapter Summary 61

4 Scene Decomposition 67

4.1 Scene Decomposition Model 70
4.1.1 Predicting Horizon Location 71
4.1.2 Characterizing Individual Region Appearance 72
4.1.3 Individual Region Potentials 74
4.1.4 Inter-Region Potentials 76
4.2 Inference and Learning 79
4.2.1 Inference Algorithm 79
4.2.2 Learning Algorithm 81
4.3 Experiments 83
4.3.1 Baselines 84
4.3.2 Region-based Approach 84
4.3.3 Comparison with Other Methods 86
4.4 Chapter Summary 87

5 Object Detection and Outlining 93

5.1 Region-based Model for Object Detection 96
5.1.1 Energy Function 97
5.1.2 Object Features 102
5.2 Inference and Learning 107
5.2.1 Inference 107
5.2.2 Proposal Moves 108
5.2.3 Learning 112
5.3 Experiments 113
5.3.1 Street Scenes Dataset 113
5.3.2 Cascaded Classification Models Dataset 116
5.4 Chapter Summary 120

6 Scene Reconstruction **125**
6.1 Geometric Reconstruction 128
6.2 Depth Estimation 131
6.2.1 Encoding Geometric Constraints 133
6.2.2 Features and Pointwise Depth Estimation 134
6.2.3 MRF Model for Depth Reconstruction 137
6.3 Experiments 139
6.4 Chapter Summary 143

7 Conclusions and Future Directions **149**
7.1 Summary 149
7.2 Open Problems and Future Directions 151
7.2.1 Learning from Multiple Sources 151
7.2.2 Structure and Occlusion 152
7.2.3 Expanding the Object Classes 153
7.2.4 Hierarchical Decomposition 154
7.2.5 Activity Recognition 155
7.3 Conclusion 155

A Implementation Details **157**
A.1 Region Appearance Features 158
A.2 Checking for Connected Components 161

Bibliography **181**

List of Tables

4.1 Local pixel appearance features, α_p. 73
4.2 Summary of the features $\phi_r(A_r, \mathcal{P}_r)$ used to describe regions. 75
4.3 Summary of the features $\phi_{rs}(A_r, \mathcal{P}_r, A_s, \mathcal{P}_s)$ used to describe the pairwise relationship between two adjacent regions. 78
4.4 Multi-class image segmentation and surface orientation (geometry) accuracy on the Stanford Background Dataset. 84
4.5 Comparison with state-of-the-art 21-class MSRC results against our restricted model. 87
4.6 Comparison with state-of-the-art Geometric Context results against our restricted model. 87

5.1 Summary of the features $\phi_{or}(\mathcal{P}_o, \mathcal{P}_r)$ used to describe the contextual pairwise relationship between an object o and adjacent region r. . . . 102
5.2 Summary of the features $\phi_o(\mathcal{P}_o, v^{\text{hz}}, \mathcal{I})$ used to describe an object in our model. 106
5.3 Object detections results for the Street Scenes dataset. 114
5.4 Precision and recall for each object category calculated on the MAP assignment. 118
5.5 Object detections results for the CCM dataset. 119

6.1 Quantitative results comparing variants of our “semantic-aware” approach with strong baselines and other state-of-the-art methods. . . . 140
6.2 Quantitative results by predicted semantic class ($\log_{10}$ error). 142
6.3 Quantitative results by predicted semantic class (relative error). . . . 142

List of Figures

1.1 Inconsistent results in attempting to understanding a scene using separate computer vision tasks. 3
1.2 Illustration of the aim of this thesis to understanding outdoor scenes in terms of regions, objects and geometry. 4
1.3 Hierarchical model of a scene. 6

2.1 Example of a simple pairwise CRF. 27
2.2 The geometry of image formation. 33
2.3 Convolution of an image with a filter. 35
2.4 The SIFT feature descriptor. 37
2.5 Illustration of the 17-dimensional "texton" filters used in multi-class image segmentation. 38
2.6 Application of 17-dimensional "texton" filters to an image. 39
2.7 Sliding-window object detection. 40
2.8 Feature construction for HOG-based object detector. 41
2.9 Feature construction for patch-based object detector. 42
2.10 Measuring object detection performance. 44
2.11 Comparison of different unsupervised image segmentation algorithms. 47
2.12 Illustration of a conditional Markov random field for multi-class image labeling. 51

3.1 Screenshot of an Amazon Mechanical Turk HIT for labeling regions in an image. 62
3.2 Typical label quality obtained from AMT jobs. 63

3.3 Comparison of the label quality obtained from our AMT jobs versus the 21-class MSRC dataset. 64
3.4 Two examples of poorly labeled images obtained from a semantic labeling job on Amazon Mechanical Turk (AMT). 65

4.1 Example of a typical urban scene. 68
4.2 Example over-segmentation dictionary, Ω, for a given image. 80
4.3 Examples of typical scene decompositions produced by our method. . 86
4.4 Representative results when our scene decomposition model does well. 90
4.5 Examples of where our scene decomposition algorithm makes mistakes. 91

5.1 Contextual information is critical for recognizing small objects. 96
5.2 An illustration of random variables defined by our joint object detection and scene decomposition model. 98
5.3 Illustration of soft masks for proposed object regions. 105
5.4 Illustration of a preprocessed top-down object proposal move. 111
5.5 Illustration of eroding a top-down bounding box move. 112
5.6 Object detection precision-recall curves for patch-based and HOG-based detectors on the Street Scenes dataset. 115
5.7 Example output from our joint scene decomposition and object detection algorithm. 116
5.8 Qualitative object detection results for the Street Scenes dataset. . . 117
5.9 Object detection precision-recall curves for patch-based and HOG-based detectors on the CCM dataset. 120
5.10 Qualitative object detection results for the CCM dataset. 123
5.11 Failure modes of our scene understanding algorithm. 124

6.1 Example output from our semantically-aware depth estimation model. 126
6.2 Schematic illustrating geometric reconstruction of a simple scene. . . 129
6.3 Novel views of a scene with foreground objects generated by geometric reconstruction. 131
6.4 Illustration of semantically derived geometric constraints. 133

6.5 Smoothed per-pixel log-depth prior for each semantic class. 135
6.6 Graph showing the effect of using the Huber penalty for constraining depth range. 138
6.7 Plot of log_{10} error metric versus relative error metric comparing algorithms from Table 6.1. 141
6.8 Quantitative depth estimation results on the Make3d dataset. 146
6.9 Novel 3D views of different scenes generated from our semantically-aware depth reconstruction model. 147

A.1 Feature matrices for storing pixel feature vectors. 158
A.2 Integral images and row integrals. 160
A.3 4-connected and 8-connected neighborhoods. 162

List of Algorithms

2.1 Discrete AdaBoost 24
2.2 Iterated Conditional Modes (ICM) 28
2.3 Factor Addition 30
2.4 Min-Sum Message Passing 31
2.5 Mean-shift Segmentation 48
4.1 Scene Decomposition Inference. 81
5.1 Scene Decomposition with Objects Inference. 109
A.1 Compute Region Features. 164
A.2 Find Connected Components. 165

Chapter 1

Introduction

One of the long-term goals of Artificial Intelligence (AI) is to develop autonomous agents (i.e., machines) that can reason about their environments from visual inputs [Russell and Norvig, 2002]. This ambitious goal manifests itself in many applications including robot perception for navigation, inventory taking and manipulation (e.g., [Newman et al., 2006, Klingbeil et al., 2010, Quigley et al., 2009, Saxena et al., 2007]); surveillance for physical security and environmental monitoring (e.g., [Thirde et al., 2006]); interpretation of medical images for diagnosis and computer-assisted surgery (e.g., [Mirota et al., 2009]); and understanding of Internet images for search and knowledge acquisition (e.g., [Lew et al., 2006, Li et al., 2007]). Each of these diverse applications requires complex reasoning about the interactions between the real-world entities that give rise to the visual inputs that we, or our machines, perceive. For example, to understand a photograph of a visual scene we need to reason about objects (some of which may be partially occluded), background regions, surface color and reflectance properties, 3D structure, image scale, lighting, the complex interactions between all of them.

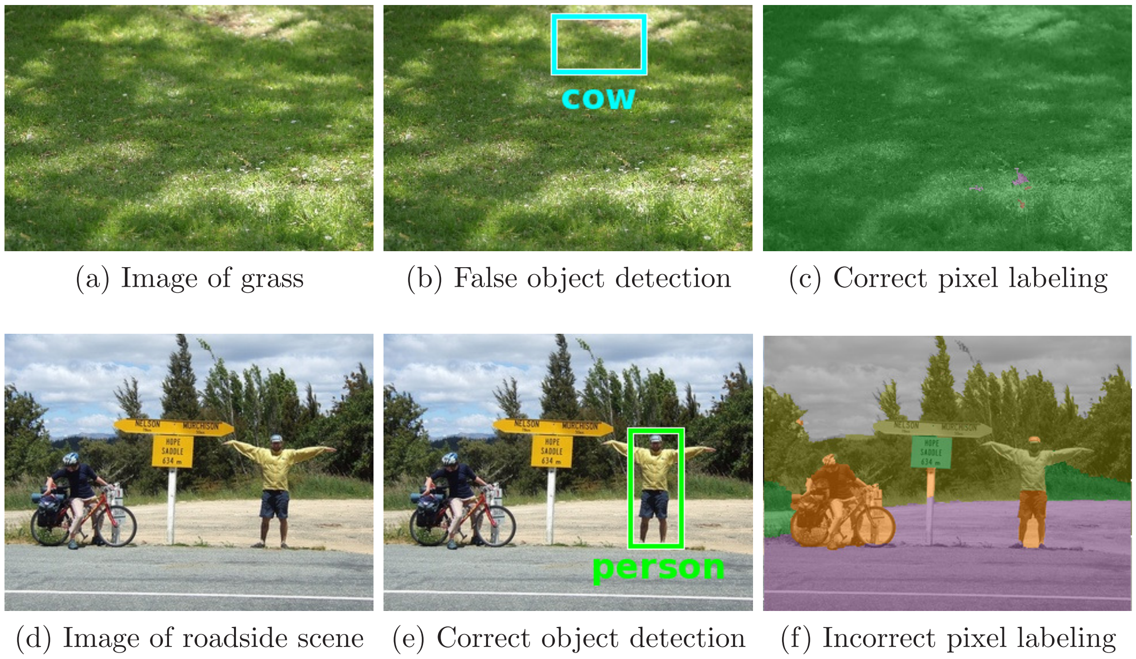
To further appreciate the challenges of automatic scene understanding, consider that a machine is presented with a description of the scene as a rectangular array of numbers. (We will discuss image formation in more detail in Section 2.2.1.) It is from this very plain representation that we need to bring the scene back to life by discovering the underlying structure and objects within it.

Many early computer vision researchers undertook the challenge of understanding an image of a scene and some impressive systems were built [Waltz, 1975, Ohta et al., 1978, Hanson and Riseman, 1978, Brooks, 1983]. These early systems, however, were brittle and failed to scale to large real-world problems. It is not surprising that, to make progress towards AI's long-term goal, researchers turned to tackling individual aspects of the scene understanding problem in isolation.

Coupled with advances in machine learning and computational resources, much progress has been made in the last 30 years on isolated vision tasks such as object detection [Viola and Jones, 2004, Dalal and Triggs, 2005], multi-class image segmentation and labeling [He et al., 2004, Shotton et al., 2006, Gould et al., 2008, Ladicky et al., 2009], and 3D perception [Delage et al., 2006, Hoiem et al., 2007a, Saxena et al., 2008, Hedau et al., 2009]. In fact, some tasks, such as face detection which aims to place a bounding box around all faces in an image, can now perform robustly enough to be employed in commercial applications (e.g., digital cameras). Unfortunately, as we shall see, this "divide and conquer" approach to scene understanding can only take us so far, and leads to inconsistent conclusions about the scene.

Consider two standard (isolated) tasks in computer vision. The first, *object detection*, aims to place a bounding box around all objects of a particular class within the scene. The second, *multi-class image labeling*, aims to assign each pixel with a class label from a predetermined set. Figure 1.1 shows the output from state-of-the-art algorithms developed for these two tasks run on two different images. In the first row, the object detection algorithm has incorrectly classified a patch of grass to be a cow, while the pixel labeling algorithm correctly classifies the same patch. The second row shows the opposite situation. Here, the object detection algorithm does well at identifying the man, whereas the pixel labeling algorithm performs poorly and mislabels the torso of the man as tree.

These simple examples highlight one of the key problems with treating computer vision tasks in isolation—they produce incoherent results. When treating scene understanding tasks separately, there is no guarantee that the algorithm will produce consistent output. We, along with many other contemporary researchers [Hoiem et al., 2008a, Heitz et al., 2008b, Li et al., 2009], believe that the time has come to



(a) Image of grass (b) False object detection (c) Correct pixel labeling

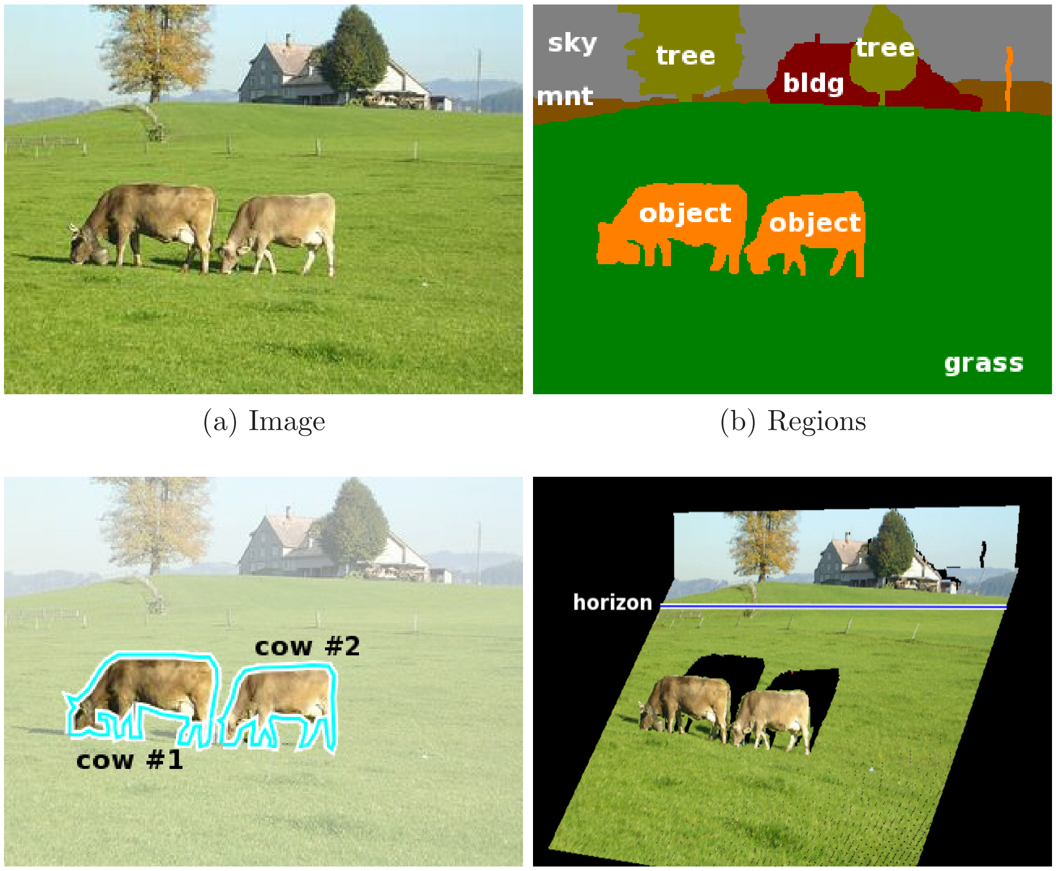
(d) Image of roadside scene (e) Correct object detection (f) Incorrect pixel labeling

Figure 1.1: Inconsistent results in attempting to understanding a scene using separate computer vision tasks. In the first row, an object detector, looking at local image features, incorrectly detects a cow. However, a separate pixel labeling algorithm correctly determines that the image is all grass. In the second row, the pixel labeling algorithm incorrectly labels the man's shirt as tree, while an object detector run on the same image correctly finds the man. Resolving these inconsistencies requires joint reasoning over both tasks.

unite separate scene understanding tasks into a single *holistic* model. By moving to such a model, we enforce coherence between related tasks and thereby resolve any inconsistency.

A second reason for combining computer vision tasks is that sharing information between tasks can only help improve results [Heitz et al., 2008b]. Consider the misclassification of the patch of grass as a cow in Figure 1.1(b). Although this error seems ridiculous to us, it is less absurd when we consider the limited information (i.e., local features) available to the object detector's classification engine. By utilizing information from the pixel labeling task, we hope that, these sorts of errors can be avoided.

Figure 1.2 illustrates the three different tasks that we aim to unite in this thesis. The first task (in Figure 1.2(b)) aims to divide the scene into regions and labels each



(a) Image (b) Regions

(c) Objects (d) Geometry

Figure 1.2: Illustration of the aim of this thesis to understanding outdoor scenes in terms of regions, objects and geometry.

one with a semantic class label. We call this task *scene decomposition*. It is similar to pixel labeling, which assigns a label to each pixel, but in addition groups pixels into coherent regions so that two adjacent pixels can have the same class label but be grouped into different regions, thereby separating adjacent objects of the same class. As we shall discuss in Chapter 4, there are many different ways in which a scene can be decomposed. For example, we could naively treat each pixel as its own region. For now, it is sufficient for us to loosely define scene decomposition to be the segmentation of an image into semantically and geometrically consistent regions with respect to a predefined set of possible labels. In our work, this label set includes a generic object label for modeling the wide variety of foreground object classes that we are likely to encounter.

The second task, shown in Figure 1.2(c), is in some sense an enhancement of the first. Here, we would like to refine generic class labels, such as *object*, into specific categories, and identify individual instances of these categories. The last task (shown in Figure 1.2(d)) involves understanding the 3D structure of the scene, including the location of the horizon and the relative placement of objects and dominant background regions. It is only, by understanding all three aspects—regions, objects and geometry—that we can really claim to have understood the scene. Of course, there are other tasks that can build on these (e.g., activity recognition), and we will discuss some of these as future research directions in Chapter 7.

This thesis contributes to bridging the gap between low-level image representation (i.e., pixels) and high-level scene understanding in terms of regions, objects and geometry. As discussed above, our aim is to decompose a scene into semantic and geometrically consistent regions. Each region is annotated with a semantic and geometric class labels. From this region-level description of the scene we will progress to identifying the objects in the scene and its geometry. This hierarchical model, illustrated in Figure 1.3, shares much in common with early computer vision systems which attempted to fully interpret the scene (e.g., [Ohta et al., 1978]). However, unlike these early feed-forward systems, our framework allows feedback from high-level scene entities to the low-level decomposition. Specifically, our model is holistic and we allow probabilistic influence to flow in both directions. Furthermore, the parameters of our models are learned from data allowing us to consider many more features and making our models more robust.

The following section summarizes our main contributions. We then provide a brief outline for the remainder of this dissertation and contrast our research with related work in the field.

1.1 Contributions

This thesis makes five main contributions to the goal of holistic scene understanding:

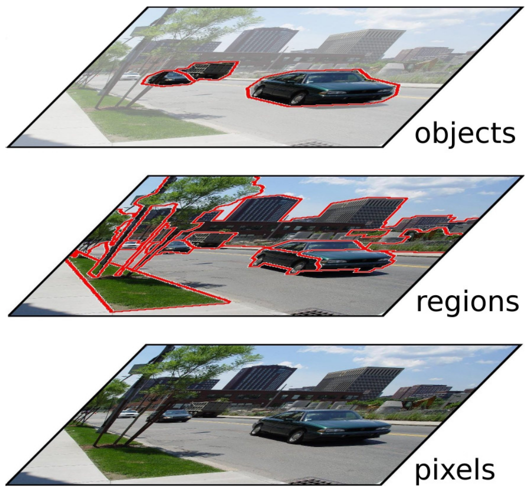


Figure 1.3: Hierarchical model of a scene.

1. Unified energy model for image segmentation, multi-class labeling, and object detection and outlining. We propose an energy function that links three fundamental tasks for scene understanding. The first task, image segmentation, aims to divide the scene into salient regions that are a consistent part of an object or background surface. The second task, multi-class labeling, aims to annotate pixels (or regions) with appropriate semantic and geometric class labels. The last task, object detection and outlining, addresses the problem of recognizing and identifying the boundary of complete objects, many of which will be composed of multiple foreground regions (e.g., the face, torso and legs of a person). While previous works have considered propagation of information from one task to another, our is the first to incorporate these tasks into a unified model.

2. Introduction of regions as variables in our model. A key contribution of our work is to allow the underlying region-based representation of a scene to change as we reason about the scene. Many previous works apply an initial preprocessing step where pixels in an image are grouped into superpixels, thereby reducing computational complexity as well as providing a stable base on which to compute features. However, ours is the first approach to integrate this segmentation component into the model by introducing regions as variables which are to be inferred.

3. Efficient inference. One of the major difficulties with holistic models of vision is the significant burden placed on the inference procedure to deal with the interactions between model components. We address this problem by developing an efficient inference procedure that makes large moves in energy space. This approach allows our algorithm to find good approximate solutions of the energy function.

4. Incorporation of semantics as an aid for 3D scene reconstruction. We propose an approach to recovering the 3D layout of a scene that makes use of semantic content. Specifically, we use predicted semantic labels and region boundaries as context for depth reconstruction. Unlike previous approaches, this allows us to exploit class-related geometric priors and better model the relationship between region appearance and depth.

5. Development of scene dataset and annotation tools. As part of the development of the algorithms described in this thesis, we collected a large dataset of outdoor scenes and developed software tools for annotating these scenes with semantic and geometric information. The software tools allow us to leverage the Amazon Mechanical Turk (AMT) online workforce to obtain high-quality labeled images cheaply and rapidly.

1.2 Thesis Outline

The remaining chapters of this thesis are summarized below:

Chapter 2: Background. This chapter includes the basic technical background and notation needed to understand the remainder of the thesis. The chapter is divided into two main sections. In the first section, we discuss the machine learning and probabilistic modeling techniques used in our work. The second section provides a review of image formation, standard image features, and state-of-the-art techniques for high-level computer vision, in particular, object detection and multi-class image labeling.

Chapter 3: Image datasets and dataset construction. In this chapter we discuss the various publicly available datasets used in the computer vision community and highlight the ones used for evaluating our work. Many of the datasets are developed with a particular task in mind, and we discuss their suitability to the scene understanding task. We then introduce a new dataset that includes both semantic and geometric labels and describe an online annotation tool that we developed for constructing this dataset.

Chapter 4: Scene decomposition: from pixels to regions. In this chapter we present our model for decomposing a scene into semantic and geometrically consistent regions. We describe our representation for the entities of the scene—pixels, regions, and geometry—over which we wish to reason. We then present our unified energy function for scoring different decompositions of a scene and describe methods for learning and performing approximate inference on this energy function.

Chapter 5: Object detection and outlining: from regions to objects. Building on the scene decomposition framework of Chapter 4, we introduce the notion of objects into our model. Here, our objective is to identify and outline known objects, each of which can be composed of one or more foreground regions. We show how our unified energy function can be extended to support this objective and discuss a modification to our inference procedure that improves efficiency.

Chapter 6: Scene reconstruction: from semantics to geometry. In this chapter, we discuss models for 3D reconstruction of a scene given a decomposition into semantically labeled regions. We first show how a basic geometric reconstruction can be produced by "popping up" vertical regions from a supporting ground plane. Next, we discuss a model for metric reconstruction in which we perform depth perception aided by our semantic understanding.

Chapter 7: Conclusions and future directions. We conclude the thesis with a summary of our main contributions and discussion of future directions for improving our work.

Appendix A: Implementation details. Here we discuss some implementation details necessary for efficient feature computation and inference.

1.3 Related Work

Our work builds on recent advances in many areas of computer vision and machine learning, for example, multi-class image labeling [He et al., 2004, Kumar and Hebert, 2005, Shotton et al., 2006], object detection and recognition [Torralba et al., 2004a, Dalal and Triggs, 2005, Opelt et al., 2006a, Viola and Jones, 2004], and geometric reasoning [Criminisi et al., 2000, Hoiem et al., 2007a, Saxena et al., 2008]. While these works only attempt to interpret part of the scene, they all provide insights and contribute results which we leverage. As we will discuss, we also share common features with other works that attempt a more complete interpretation of the scene [Heitz et al., 2008b, Hoiem et al., 2008a, Li et al., 2009].

Early Work in Scene Understanding. We begin with a discussion of some early work in scene understanding. Researchers, inspired by the study of biological systems, developed a computational theory of vision [Marr, 1982, Barrow and Tenenbaum, 1981] that supported a feed-forward model where low-level inputs are combined into higher and higher level structures. One approach was to move from a *primal sketch* of the scene, in which an image is decomposed into edges and regions, to a full 3D interpretation (including recognition of objects and their geometric arrangement within the scene).

This ambitious full-scene approach to computer vision was demonstrated by a number of early systems that attempted to combine bottom-up processing with higher-level modeling. Such full scene interpretation systems included the segment merging approach of Ohta et al. [1978], the VISIONS system of Hanson and Riseman [1978], and the geometric primitives approach (ACRONYM) of Brooks [1983]. Unfortunately, these early systems were brittle due to many issues, including hand-coding of parameters and limited computational resources. Attempts at holistic scene understanding were soon abandoned in favor of more tractable tasks such as shape from shading [Horn, 1989], edge detection [Canny, 1986], and recognition of specific object categories, e.g., faces [Viola and Jones, 2001].

There were also many early attempts to solve subtasks in computer vision by narrowing the range of inputs to the problem. Perhaps the most famous of these is the work of Waltz [1975], which aims to automatically elicit the 3D structure of a scene from line drawings. Waltz's algorithm begins by extracting the corners and junctions from the image, and then attempts to label these and the edges between them to indicate geometry (e.g., concave, convex, occluding, etc.). From this labeling, a 3D reconstruction of the scene could be produced. A key contribution of Waltz [1975] was to enforce consistency in the labeling of edges and junctions, and the work became the cornerstone for a large and important body of algorithms that solve so-called constraint satisfaction problems (CSPs).[1] However, clean line drawings are a significant simplification of real-world scenes, and unfortunately, Waltz's algorithm was unable to meet to the challenges of natural scenes.

Many of the ideas developed in these early works are being revisited in contemporary research. What is remarkable is that, with improved machine learning techniques, computational power, and availability of large amounts of data, the ideas are producing fruitful results. That is, the early computer vision researchers had the right insights, but not the right tools to make their ideas successful.

Multi-class Image Segmentation. Turning to more recent times, the problem of multi-class image segmentation (or pixel labeling) has been successfully addressed by a number of works [He et al., 2004, Shotton et al., 2006, Ladicky et al., 2009]. The goal of these works is to label every pixel in the image with a single class label from a fixed predefined set. Typically, these algorithms construct conditional Markov random fields (CRFs) over the pixels (or small coherent regions called *superpixels*) with local class-predictors, based on pixel appearance, and a pairwise smoothness term to encourage neighboring pixels to take the same label. The works differ in the details of the energy functions and the inference algorithms used. For example, Ladicky et al. [2009], in addition to using very sophisticated features, take the innovative step

[1]Constraint satisfaction problems are closely related to the Markov random fields (MRFs) that we use in our work to enforce soft, rather than hard, constraints.

of encouraging label consistency over large contiguous regions (known as *superpixel*) through the use of high-order energy terms. However, none of the works aim to separate instances of the same object class. In these approaches a crowd of people will be labeled as a single contiguous "person" region rather than delineating the individual people.

Some novel works introduce 2D layout consistency between objects [Winn and Shotton, 2006] or object shape [Winn and Jojic, 2005, Kumar et al., 2005, Yang et al., 2007] in an attempt to segment out foreground objects. These, and other attempts at *figure-ground* segmentation [Rother et al., 2004, Boykov and Jolly, 2001, Mortensen and Barrett, 1998, Borenstein and Ullman, 2002, Leibe et al., 2004, Levin and Weiss, 2006, Winn et al., 2005], work well when objects have a distinct shape (or appearance) and are well-framed (i.e., large and centered) within the image. However, many of the scenes that we consider contain small objects—a domain not suited to these approaches. Furthermore, these approaches do not attempt to label the background regions which we require for context and geometry.

In addition to layout consistency and shape, some researchers have explored relative location and co-occurrence between regions [Rabinovich et al., 2007, Gould et al., 2008, Lim et al., 2009]. For example, Rabinovich et al. [2007] learn the number of times different object classes appear together in the same image, and use that information to constrain pixel labelings in novel images. Gould et al. [2008] explore context-dependent relative location priors as a means of enforcing spatial arrangement of classes. For example, the relative location priors model that cows tend to be surrounded by grass (in 2D images). None of these works, however, take into account the 3D arrangement of a scene and do not learn to enforce global consistency, such as that "sky" needs to be above "ground".

Liu et al. [2009] use a non-parametric approach to image labeling by warping a novel image onto a large set of labeled images with similar appearance. The method then copies the (warped) labels and combines them using a CRF model to obtain a labeling for the novel image. This is a very effective approach since it scales easily to a large number of classes. However, the method does not attempt to understand the scene semantics. In particular, their method is unable to break the scene into separate objects (e.g., a row of cars will be parsed as a single region) and cannot capture combinations of classes not present in the training set. As a result, the approach performs poorly on most foreground object classes.

The use of multiple different over-segmented images (i.e., superpixels) as a pre-processing step is not new to computer vision. Russell et al. [2006], for example, use multiple over-segmentations for finding objects in images, and many of the depth reconstruction methods described below (e.g., [Hoiem et al., 2007a]) make use of over-segmentations for computing feature statistics. In the context of multi-class image segmentation, Kohli et al. [2008] specify a global objective which rewards solutions in which an entire superpixel is labeled consistently. However, their energy function is very restricted and does not, for example, capture the interaction between region appearance and class label, nor does their energy function allow for label-dependent pairwise preferences, such as foreground objects above road. Their approach was extended in Ladicky et al. [2009] to handle the first of these deficiencies, but not the second. Unlike all of these methods that use multiple over-segmented images, our method uses multiple over-segmentations to build a dictionary of *proposal moves* for optimizing a global energy function—the segments themselves are not used for computing features nor do they appear explicitly in our objective.

Image Parsing. As an extension to multi-class image segmentation, there has also been some recent work on *image parsing* [Tu et al., 2003, 2005, Han and Zhu, 2009]. Here, the aim is to decompose the image into a single hierarchical segmentation describing the scene. Such a representation would allow, for example, a segmentation that understands both car objects as a whole and wheels as parts of cars. These methods point to promising future results. However, they are currently still in their infancy and are, as yet, unable to explain the whole scene. For example, the work of Tu et al. [2003, 2005] demonstrates effective segmentation of a scene into face and text regions, but does not attempt to understand the remaining areas in the image.

Han and Zhu [2009] define a *parse grammar* for understanding man-made scenes (e.g., buildings) and generate a top-down parse of the scene into primitives defined by this grammar. Their approach shares many ideas with some early approaches to scene understanding (e.g., [Ohta et al., 1978, Brooks, 1983]), but using modern machine learning techniques. Currently, the system only works on highly structured scenes consisting of mainly rectangular elements. Furthermore, Han and Zhu [2009] focus on extracting scene geometry and make no attempt to understand the scene's semantics.

Object Detection. Object detection is one of the most heavily studied topics in computer vision and much progress has been made in recent years [Viola and Jones, 2004, Brubaker et al., 2007, Fergus et al., 2003, Torralba et al., 2004a, Dalal and Triggs, 2005, Opelt et al., 2006a, Fei-Fei et al., 2006, Felzenszwalb et al., 2010]. The vast majority of works use localized cues extracted from within candidate object bounding boxes to find and classify objects. The highly successful approach of Dalal and Triggs [2005], for example, uses the orientation of edges extracted from overlapping cells within the candidate bounding box to build up a descriptor for classifying the object. The state-of-the-art work of Felzenszwalb et al. [2010] builds on this approach by extracting these same features at two resolutions, one for the whole object and one for object parts (which are free to deviate slightly from their nominal positions within the bounding box for the whole object). All of these methods attempt to understand the scene as a collection of independent objects and do not consider other aspects of the scene (e.g., background regions). They do form incredible powerful building blocks for holistic scene understanding, however, and we make extensive use of the tools developed along this line of research in Chapter 5.

A number of researchers have suggested the importance of context in object detection [Divvala et al., 2009, Galleguillos and Belongie, 2008, Torralba et al., 2006]. The empirical study of Divvala et al. [2009] showed that many different sources of context can be leveraged to improve object detection performance, including scene level context (i.e., certain objects are more likely to occur within certain scene types), object size, object location, and spatial support. Some previous works have also made attempts at incorporating contextual information for improving object detection [Fink and Perona, 2003, Torralba et al., 2004b, Murphy et al., 2003, Heitz and Koller, 2008, Tu, 2008]. However, the majority of these works incorporate feature-level context (e.g., neighborhood appearance) rather than semantic or geometric context.

A notable exception is the work of Hoiem et al. [2006], which relates global scene geometry (i.e., camera pose) to object size. This enforces consistency between the height of detected objects, allowing some false positives to be rejected. The approach requires a number of well detected objects to be present in the scene in order to get a good estimate of camera pose. Moreover, in cluttered environments (e.g., urban scenes) many "in-context" false positives will remain.

Other works attempt to integrate tasks such as object detection and multi-class image segmentation into a single CRF model. However, these models either use a different representation for object and non-object regions [Wojek and Schiele, 2008] or rely on a pixel-level representation [Shotton et al., 2006]. The former does not enforce label consistency between object bounding boxes and the underlying pixels while the latter does not distinguish between adjacent objects of the same class. Recent work by Yang et al. [2010] performs joint reasoning over objects and their segmentations in a unified probabilistic model. However, they are only concerned with foreground object segmentations and do not explain the full scene or its geometry.

Recent work by Gu et al. [2009] uses regions for object detection instead of the traditional sliding-window approach. However, unlike the method that we develop in this thesis, they use a single over-segmentation of the image and make the strong assumption that each segment represents a probabilistically recognizable object part. Our method, on the other hand, assembles objects and background regions using segments from multiple different over-segmentations. The multiple over-segmentations avoids errors made by any one segmentation. Furthermore, we incorporate background regions which allows us to eliminate large portions of the image thereby reducing the number of component regions that need to be considered for each object.

Geometric Reasoning and 3D Structure. We can roughly partition methods that attempt to explain the 3D structure of an entire scene into two groups: geometric models and depth perception models. For indoor environments, Delage et al. [2006] use an MRF for reconstructing the location of walls, ceilings, and floors using geometric cues (such as long straight lines) derived from the scene. More recently, Hedau et al. [2009] recover the spatial layout of cluttered rooms using similar geometric cues. Both models make strong assumptions about the structure of indoor environments (such as the "box" model of a room [Hedau et al., 2009]) and are not suitable to the less structured outdoor scenes that we consider.

An early approach to outdoor scene reconstruction is the innovative work of Hoiem et al. [2005b] who cast the problem as a multinomial classification problem similar in spirit to the multi-class image labeling works discussed above. In their work, pixels are classified as either ground, sky, or vertical. A simple 3D model can then be constructed by "popping up" vertical regions. The model was later improved [Hoiem et al., 2007a] to incorporate a broader range of geometric subclasses (porous, solid, left, center, right). These models make no attempt to estimate absolute depth. Furthermore, many objects commonly found in everyday scenes (e.g., cars, trees, and people) do not neatly fit into the broad classes they define. A car, for example, consists of many angled surfaces that cannot be modeled as vertical.

It is interesting to note the connection between background semantic classes and the geometric subclasses defined by Hoiem et al. [2007a]: trees are generally porous; buildings are vertical; and road, grass and water are horizontal. Indeed, our own scene decomposition model (discussed in Chapter 4) exploits the strong correlation between geometry and semantics in outdoor scenes.

In related work, Hoiem et al. [2007b], attempt to infer occlusion boundaries within images. The success of their approach hinges on the quality of the geometric cues obtained from the pixel labeling model just discussed [Hoiem et al., 2007a]. By similar construction to the pixel labeling work, a "pop up" of the scene can be generated. An interesting feature of this work is that it makes use of the constraint satisfaction algorithm of Waltz [1975] discussed previously.

A more semantically motivated approach to depth reconstruction was recently adopted by Russell and Torralba [2009] who utilize detailed human-labeled segmentations to infer the geometric class of regions (ground, standing, attached) and region edges (support, occlusion, attachment). In their model, depth inference is done by modeling support and attachment relationships relative to a ground plane. Currently, their model relies on detailed human annotation of regions, and in particular, their polygonal boundaries, within the scene.

Our depth estimation work (discussed in Chapter 6) is most heavily influenced by the work of Saxena and colleagues [Saxena et al., 2008, 2005] who take a very different approach to the task of 3D reconstruction. Instead of inferring geometric class labels, they infer the absolute depth of the pixels in the image. However, unlike their approach, which completely ignores semantic context, our work makes use of semantic information (i.e., region boundaries and class labels) to guide depth perception. This has a number of advantages: First, we can use simpler features since depth perception in our model is conditioned on semantic class and thus avoids the need for features that correlate with depth across all classes. Second, we avoid the need for modeling occlusions and folds since these can be easily obtained from the region boundaries and associated semantic labels (sky is always occluded; ground plane classes "fold" into foreground classes). Last, co-planarity and connectivity constraints can be imposed differently within each semantic class. For example, a building is more likely to be planar than a tree.

Holistic Scene Understanding. None of the above methods directly tackle the problem of describing the entire scene and all restrict themselves to annotating either semantic components or geometric components, but not both. A few recent works have begun to re-explore the ideas of the early computer vision researchers and develop holistic models for scene understanding [Heitz et al., 2008b, Hoiem et al., 2008a, Li et al., 2009]. These works aim to leverage correlations between different aspects of a scene to improve performance across the different tasks.

Heitz et al. [2008b] made one of the first attempts by combining scene categorization, object detection, image segmentation, and depth information into a single model. Significant improvements was demonstrated by propagating information from one task to another through a cascade of repeated models. However, no consistency was enforced in the final output generated by the cascade, and the lower layers simply act to provide strong additional features to subsequent layers in the cascade.

In a similar attempt, Hoiem et al. [2008a] propose a system for integrating the tasks of object recognition, surface orientation estimation, and occlusion boundary detection. Like the model of Heitz et al. [2008b], their system is modular and leverages state-of-the-art components. And similarly, consistency is not enforced in the final output.

Li et al. [2009] also develop a holistic model of a scene that enforces label consistency between tasks. In their work, classification, segmentation, and textual annotation, are combined within a single model. Their model explains the whole scene, but in order to make inference tractable, they do not model spatial constraints. Furthermore, their work makes no attempt to distinguish between multiple instances of the same object category in the image.

The success of these contemporary holistic scene understanding models has been a key motivation for our work. However, unlike these approaches, we desire a model that produces a coherent labeling of the scene, identifies distinct object instances, and interprets the scene geometry.

1.4 Previously Published Work

Much of the work described in this thesis has been previously published in conference proceedings. In particular, our scene decomposition model and inference procedure (described in Chapter 4), in which we decompose a scene into semantic and geometrically consistent regions, was published in Gould et al. [2009b]. A brief discussion of the dataset and image annotation pipeline using Amazon Mechanical Turk (AMT) appears in this work. We provide a more complete description in Chapter 3.

An early version of the extension to our scene decomposition model for recognizing and outlining foreground objects (described in Chapter 5) was published in Gould et al. [2009c]. However, experiments on the CCM dataset have not appeared previously. Finally, our model for reconstructing the 3D geometry of a scene from a semantic decomposition (described in Chapter 6) first appeared in Gould et al. [2009b], and was later expanded to include depth perception in Liu et al. [2010]. This thesis further extends that work.

Chapter 2

Background

The work in this thesis is built on the foundations of two large fields of study—computer vision (CV) and machine learning (ML). Indeed, these fields are now very much integrated: Much of modern computer vision (at least, high-level computer vision) incorporates machine learning techniques so that algorithms can make use of data for training parameters rather than setting them manually, thereby accelerating development and hopefully making the algorithms more. Likewise, many machine learning techniques are motivated by problems arising in computer vision and then applied more broadly. The purpose of this chapter is to provide a very brief introduction to these two fields. The chapter is not intended to be a comprehensive treatment of either computer vision or machine learning. For an in depth coverage of the topics covered in this chapter, the reader should consult one of the excellent textbooks on machine learning (e.g., [Koller and Friedman, 2009, Duda et al., 2000, Bishop, 1996, 2007, Hastie et al., 2009]) or computer vision (e.g., [Forsyth and Ponce, 2002, Ma et al., 2005, Hartley and Zisserman, 2004, Trucco and Verri, 1998]).

2.1 Machine Learning and Graphical Models

Machine learning is a large and diverse field concerned with the development of algorithms that adapt themselves based on observed data. In this thesis we will be concerned with *supervised learning*, which deals with the problem of learning a mapping from features to outputs (i.e., labels or distributions over labels) given a set of training exemplars. In our work, the features are represented by n-dimensional real-valued vectors $\boldsymbol{x} \in \mathbb{R}^n$. The labels y can either be continuous, discrete, or structured (i.e., vector-valued with dependencies between the outputs). We will use the notation $y \in \mathcal{Y}$ to denote that y can take its value from the discrete set $\mathcal{Y}$. The machine learning framework learns a classification (or regression) function $f : \mathbb{R}^n \rightarrow \mathcal{Y}$ or probability distribution $\mathrm{P}(Y \mid \boldsymbol{X})$.

The supervised learning paradigm, typically, contains the following steps:

- **Data collection and partitioning.** Data instances are acquired and labeled. The data is usually then split into training and testing sets.
- **Feature extraction.** Feature vectors $\boldsymbol{x} \in \mathbb{R}^n$ are extracted from the data instances.
- **Parameter learning.** A parametric model is selected and the parameters θ learned with respect to an objective over the training set $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$.
- **Evaluation.** The performance of the model is evaluated on the set of test instances. The performance metric need not match the training objective (usually due to the intractability of training with the true objective).

The above procedure is usually repeated with different *folds* of the data, that is, different partitioning into training and testing sets to get a measure of the algorithm's sensitivity to changes in training/testing sets.

In the following section we will provide an overview of machine learning algorithms for classification and regression. In Section 2.1.2 we describe probabilistic graphical models that allow us to efficiently model probability distributions over structured data (i.e., multi-valued output).

2.1.1 Classification and Regression

In this section we discuss *regression* (continuous-valued output spaces) and *classification* (discrete-valued output spaces) models that we use in this thesis.

Linear Regression

Regression aims to predict (or model) a continuous-valued target variable. The most simple type of regression is *linear regression*. Here, we assume that our target variable $y \in \mathbb{R}$ can be modeled as a linear combination of features, that is, $y = \theta^T \boldsymbol{x} + \eta$ where η is some unknown random noise (usually assumed to be zero-mean Gaussian). The parameters $\theta \in \mathbb{R}^n$ are estimated by minimizing a cost function over the training set $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^N$. A very common cost function is the sum-of-squares residual, giving rise to the optimization problem,

$$\text{minimize (over } \theta) \qquad \sum_{i=1}^{N} \left(y^{(i)} - \theta^T \boldsymbol{x}^{(i)}\right)^2 \tag{2.1}$$

which can be solved in closed-form to give $\theta^\star = \left(\sum_{i=1}^N \boldsymbol{x}^{(i)} \left(\boldsymbol{x}^{(i)}\right)^T\right)^{-1} \left(\sum_{i=1}^N y^{(i)} \boldsymbol{x}^{(i)}\right)$.

In our work on depth estimation, we will use a more robust cost function that is less sensitive to outliers. Specifically, we solve the optimization problem,

$$\text{minimize (over } \theta) \qquad \sum_{i=1}^{N} h\left(y^{(i)} - \theta^T \boldsymbol{x}^{(i)}; \beta\right) \tag{2.2}$$

where $h(x; \beta)$ is the *Huber penalty* defined as

$$h(x;\beta) = \begin{cases} x^2 & \text{for } -\beta \leq x \leq \beta \\ \beta(2|x| - \beta) & \text{otherwise.} \end{cases} \tag{2.3}$$

The optimization problem in Equation 2.2 has no closed-form solution, but the objective function is convex and usually solved using a quasi-newton technique [Boyd and Vandenberghe, 2004]. To prevent over-fitting, a regularization penalty on the parameters, $\lambda r(\boldsymbol{\theta})$, can be added to the objective, where $\lambda \geq 0$ controls the strength of the regularization, and $r(\boldsymbol{\theta})$ is the regularization function (or prior). A typical choice for the regularization function is the ℓ_2-regularizer, $r(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2^2$, which keeps the problem convex.

Logistic Regression Models

Multi-class logistic regression (sometimes called *softmax* regression) is a parametric probability distribution often used for discriminative classification. The model defines a probability distribution over a discrete random variable Y and observed features $\boldsymbol{x} \in \mathbb{R}^n$ as

$$\mathrm{P}\left(Y = y \mid \boldsymbol{X} = \boldsymbol{x}; \boldsymbol{\theta}\right) = \frac{\exp\left(\theta_y^T \boldsymbol{x}\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(\theta_{y'}^T \boldsymbol{x}\right)} \tag{2.4}$$

where $\boldsymbol{\theta} \in \mathbb{R}^{n \times |\mathcal{Y}|}$ are the parameters of the model.[1] In our work, we will often use a quadratic kernel function on the raw features by augmenting $\boldsymbol{x}$ with the product of all pairs of features, i.e., $\boldsymbol{x}' = (x_1, \ldots, x_n, x_1^2, \ldots, x_i x_j, \ldots, x_n^2)$, where for all pairwise terms $x_i x_j$ we have $j \geq i$. This allows more expressive models to be learned by considering pairwise correlations between the features.

In general, we can replace the observed features $\boldsymbol{x}$ in Equation 2.4 with a joint feature function over both features and labels, i.e., $\phi(y, \boldsymbol{x}) \in \mathbb{R}^m$ where $\phi : \mathcal{Y} \times \mathbb{R}^n \to \mathbb{R}^m$. In this case the multi-class logistic model can be rewritten as

$$\mathrm{P}\left(Y = y \mid \boldsymbol{X} = \boldsymbol{x}; \theta\right) = \frac{\exp\left(\theta^T \phi(y, \boldsymbol{x})\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(\theta^T \phi(y, \boldsymbol{x})\right)} \tag{2.5}$$

where $\theta \in \mathbb{R}^m$ are the parameters of the model. In this form, the standard multi-class logistic model (Equation 2.4) has joint feature function

$$\phi(y, \boldsymbol{x}) = \left(\mathbf{1}\{y = 1\}\, \boldsymbol{x}, \ldots, \mathbf{1}\{y = K\}\, \boldsymbol{x}\right) \tag{2.6}$$

for label set $\mathcal{Y} = \{1, \ldots, K\}$. Here, $\mathbf{1}\{P\}$ is the *indicator function* (or *Iverson bracket*) taking the value one if P is true, and zero otherwise. This joint feature representation allows more general interactions between features and labels to be encoded, and will be useful when we introduce conditional Markov random fields (CRFs) in Section 2.1.2.

The multi-class logistic regression model can be used as a discriminative classifier

[1] Note that this is an over-complete parameterization. Often the parameters for the last class label are fixed to zero, i.e., $\theta_{|\mathcal{Y}|} = \mathbf{0}_n$.

by taking the most likely assignment to the output according to the model, i.e., $\hat{y} = \text{argmax}_y \mathrm{P}\left(y \mid \boldsymbol{x}\right)$. In our work, however, we use the multi-class logistic as a component within our CRF models for scoring local variable configurations.

Most often, the parameters of the multi-class logistic are learned according to a maximum-likelihood objective (on the training set) with regularization on the parameters to control overfitting, i.e.,

$$\text{maximize (over } \boldsymbol{\theta}) \quad \sum_{i=1}^{N} \log \mathrm{P}\left(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta}\right) - \lambda r(\boldsymbol{\theta}) \tag{2.7}$$

where $\lambda \in \mathbb{R}_+$ is the regularization constant and $r(\boldsymbol{\theta})$ is the regularization function (or prior). As with linear regression, a typical choice for the regularization function is the ℓ_2-regularizer, $r(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_2^2$. The objective in Equation 2.7 is concave (as long as $r(\boldsymbol{\theta})$ is convex) admitting efficient computation of the optimal parameters.

Boosted Classifiers

Boosting [Freund and Schapire, 1995] is another popular approach to learning discriminative classifiers. Here, many *weak learners* are combined into a single strong classifier. The algorithm has many variants and we only describe the most basic variant in this section. We will assume that our problem is a binary classification problem with label space $\mathcal{Y} = \{-1, 1\}$. Each weak learner constructs a classifier from features to labels, i.e., $h_t : \mathbb{R}^n \rightarrow \{-1, 1\}$. In our work, we will assume that the weak learners are decision trees [Hastie et al., 2009]. The algorithm for learning the strong classifier $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is given in Algorithm 2.1. Classification is performed as $\hat{y} = \mathbf{sign}\left(f(\boldsymbol{x})\right)$. This algorithm can be shown to minimize the exponential training loss [Friedman et al., 1998].

As mentioned, there are many variants of the boosting, including extensions to multi-valued and continuous label spaces. In this thesis we use the GentleBoost [Friedman et al., 1998] variant of boosting (which sets $\alpha_t = 1$ in line 12 of Algorithm 2.1) for learning pixel-wise semantic features.

Algorithm 2.1: Discrete AdaBoost [Freund and Schapire, 1995]

```
1  Input  : $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$      // dataset with $\boldsymbol{x}^{(i)} \in \mathbb{R}^n$, $y^{(i)} \in \{-1, 1\}$
2           $T$                               // number of boosting rounds
3           $\mathcal{H}$                               // (implicit) family of weak learners
4  Output: $f(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$   // classifier function
5  initialize weights: $w_i \leftarrow \frac{1}{N}$ for all $i = 1, \ldots, N$
6  foreach $t \in \{1, \ldots, T\}$ do
7      find best weak learner, $h_t = \text{argmin}_{h \in \mathcal{H}} \sum_{i=1}^{N} w_i \mathbf{1}\{y^{(i)} \neq h(\boldsymbol{x}^{(i)})\}$
8      compute weighted error rate, $\epsilon_t = \sum_{i=1}^{N} w_i \mathbf{1}\{y^{(i)} \neq h_t(\boldsymbol{x}^{(i)})\}$
9      if $\epsilon_t > \frac{1}{2}$ then
10         break
11     end
12     set $\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
13     update $w_i \leftarrow w_i \exp\left\{-\alpha_t y^{(i)} h_t(\boldsymbol{x}^{(i)})\right\}$ for all $i = 1, \ldots, N$
14     normalize so that $w_i$ sum to one
15 end foreach
16 return $f(\boldsymbol{x}) = \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$
```

Support Vector Machines

Support vector machines (SVMs) are a third discriminative classification framework that are used extensively in computer vision, and in particular, object detection (see Section 2.2.3). The aim of an SVM is to learn decision boundaries in feature space that maximally separate the different classes. One of the nice properties of SVMs is that they can efficiently learn non-linear decision boundaries by mapping features to high dimensional spaces using kernel functions using the so-called *kernel trick*. However, in computer vision, due to the large number of training examples and runtime requirements, linear SVMs are most often used.

The objective for a linear SVM (with binary label space $\mathcal{Y} = \{-1, 1\}$) is to learn a function $f(\boldsymbol{x}; \boldsymbol{w}, b) = \boldsymbol{w}^T \boldsymbol{x} + b$ that maximally separates positive and negative training examples. Specifically, the training objective is

$$\begin{array}{lll} \text{minimize (over } \boldsymbol{w}, b, \xi) & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i & \\ \text{subject to} & y^{(i)}\left(\boldsymbol{w}^T\boldsymbol{x}^{(i)} + b\right) \geq 1 - \xi_i, & i = 1, \ldots, N \\ & \xi_i \geq 0, & i = 1, \ldots, N \end{array} \tag{2.8}$$

where C is a parameter that controls the trade-off between maximizing the margin and separating the positive and negative examples (which may not always be linearly separable).

The basic SVM framework can be extended for the structured prediction task, where $\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_n$ is an output space over many interacting variables, but we do not consider this extension when learning models in this thesis.

2.1.2 Probabilistic Graphical Models

Probabilistic graphical models [Koller and Friedman, 2009] are a powerful framework that combines probability theory and graph theory for modeling probability distributions over large structured output spaces. Two key benefits of graphical models (over an explicit representation of the joint probability distribution) is that they can be represented compactly and induce efficient algorithms for (approximate) inference and learning. This is achieved by representing the probability distribution over a set of random variables $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$ in a factored form:

$$\mathrm{P}(Y_1, \ldots, Y_n) = \frac{1}{Z}\prod_{c \in \mathcal{C}} \Psi_c(\boldsymbol{Y}_c) = \frac{1}{Z}\exp\left\{-\sum_{c \in \mathcal{C}} \psi_c(\boldsymbol{Y}_c)\right\} \tag{2.9}$$

where $\mathcal{C}$ are a set of cliques in the graph and Z is the so-called *partition function* that ensures that the probability distribution sums to one. The functions $\Psi_c(\boldsymbol{Y}_c)$ are called *potential functions* and define a local model over the subset of the variables $\boldsymbol{Y}_c \subseteq \boldsymbol{Y}$ in the c-th clique.

There are many different flavors of graphical models and in this thesis we will focus on models known as conditional Markov random fields.

Conditional Markov Random Fields

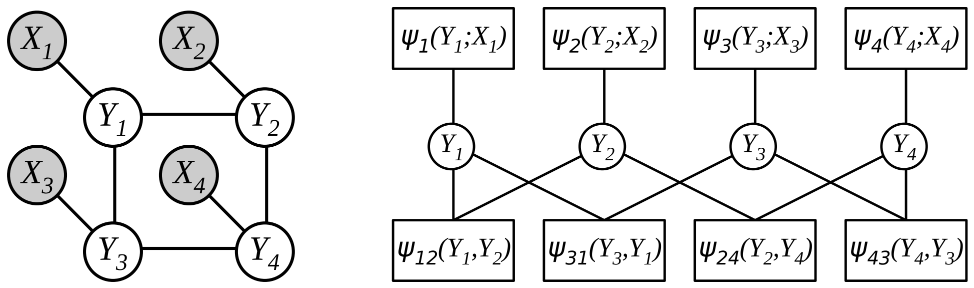
Conditional Markov random fields (CRFs) are a type of probabilistic graphical model originally introduced by Lafferty et al. [2001] in the context of natural language processing, but now commonly used for many other problems, including computer vision and computational biology. Concretely, a CRF defines a model over a finite set of random variables $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$ and observed features $\boldsymbol{X}$. In the context of scene understanding, the observations $\boldsymbol{X}$ represent the image features extracted from the image and the random variables $\boldsymbol{Y}$ represent the entities over which we are reasoning (e.g., semantic labels for the pixels). Each random variable Y_i can be assigned a value from some (discrete) value space $\mathcal{Y}_i$, e.g., the semantic label for a region can be *sky*, *road*, *grass*, etc. The joint assignment to all random variables is denoted by $\boldsymbol{y} \in \mathcal{Y}$ where $\mathcal{Y} \subseteq \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_n$ is the space of all possible joint assignments. The general form of a CRF is then

$$\mathrm{P}\left(\boldsymbol{Y} = \boldsymbol{y} \mid \boldsymbol{X} = \boldsymbol{x}\right) = \frac{1}{Z(\boldsymbol{x})} \exp\left\{ -\sum_c \psi_c(\boldsymbol{y}_c; \boldsymbol{x}) \right\} \tag{2.10}$$

where each term $\psi_c(\boldsymbol{y}_c; \boldsymbol{x})$ is defined over a subset, or *clique*, of random variables $\boldsymbol{Y}_c \subseteq \boldsymbol{Y}$ and $\boldsymbol{y}_c \in \mathcal{Y}_c$ represents the corresponding assignment to these variables. The terms are known as (log-space) *clique potential* or *factors*. Formally, we have the potential $\psi_c : \mathcal{Y}_c \times \mathcal{X} \to \mathbb{R}$ defining a preference for assignments to the subset of random variables $\boldsymbol{Y}_c$ given the observations $\boldsymbol{X}$. The sum over potential functions is called the *energy function* and is denoted $E\left(\boldsymbol{y}; \boldsymbol{x}\right) = \sum_c \psi_c(\boldsymbol{y}_c; \boldsymbol{x})$. The term $Z(\boldsymbol{x})$ is a normalization constant (partition function) that ensures the probability distribution sums to one, i.e.,

$$Z(\boldsymbol{x}) = \sum_{\boldsymbol{y} \in \mathcal{Y}} \exp\left\{-E\left(\boldsymbol{y}; \boldsymbol{x}\right)\right\}. \tag{2.11}$$

A simple example of a pairwise CRF over four random variables is shown in Figure 2.1(a). The graphical structure denotes direct dependencies between variables by connecting them with an edge. In the example, random variable Y_1 is dependent



(a) Pairwise CRF (b) Factor graph representation

Figure 2.1: Example of a simple pairwise CRF over variables $\boldsymbol{Y} = (Y_1, \ldots, Y_4)$. The model includes unary potentials for each variable $\psi_i(Y_i; X_i)$ and pairwise terms between adjacent variables $\psi_{ij}(Y_i, Y_j)$ for $(i, j) \in \{(1, 2), (2, 4), (4, 3), (3, 1)\}$. The factor graph representation shows which variables are involved in which terms. In general, the pairwise terms can also be conditioned on observed features.

on Y_2 but independent of Y_4 (given Y_2 and Y_3). Because the variables $\boldsymbol{X}$ are always observed, the dependencies between them do not need to be modeled (giving rise to a conditional distribution). The energy function for this example is

$$E(Y_1, \ldots, Y_4; X_1, \ldots, X_4) = \sum_{i=1}^{4} \psi_i(Y_i; X_i) + \sum_{(i,j)\in\mathcal{E}} \psi_{ij}(Y_i, Y_j) \tag{2.12}$$

where $\mathcal{E} = \{(1, 2), (2, 4), (4, 3), (3, 1)\}$ is the set of edges. Note that, in general, the pairwise terms can also be functions of observed features.

Another convenient graphical representation is the *factor graph* shown in Figure 2.1(b) for the same example CRF. This representation makes explicit the potential functions by defining a bipartite graph over variables and potentials. We will use this representation when we describe inference below.

In this thesis, we will parameterize our potential functions as log-linear models. That is, we will define $\psi_c(\boldsymbol{y}_c; \boldsymbol{x}, \theta_c) = \theta_c^T \phi(\boldsymbol{y}_c, \boldsymbol{x})$ where $\phi(\boldsymbol{y}_c, \boldsymbol{x}) \in \mathbb{R}^m$ is a fixed joint feature function and $\theta_c \in \mathbb{R}^m$ are learned parameters. Learning the parameters of large CRF models is a difficult problem and many solutions have been suggested

Algorithm 2.2: Iterated Conditional Modes (ICM) [Besag, 1986].

```
Input  : $\mathcal{C} = \{\psi_c(\boldsymbol{Y}_c; \boldsymbol{X})\}_{c=1}^{C}$    // set of clique potentials
         $\boldsymbol{x}$                                  // observed features
Output: $\hat{\boldsymbol{y}}$                                  // approximate MAP solution
initialize $\hat{\boldsymbol{y}}$ to an arbitrary assignment
repeat
    foreach $Y_i \in \boldsymbol{Y}$ do
        set $\hat{y}_i = \operatorname{argmin}_{y_i} \sum_{c:Y_i \in \boldsymbol{Y}_c} \psi_c(\hat{\boldsymbol{y}}_{c-i}, y_i; \boldsymbol{x})$
    end foreach
until no variables change
return $\hat{\boldsymbol{y}}$
```

(e.g., [Besag, 1975, Sutton and McCallum, 2005, Ganapathi et al., 2008, Tsochantaridis et al., 2004, Taskar et al., 2005]). We defer discussion of learning to the appropriate place in later chapters.

Having learned the models, our main interest will be to infer the *maximum a posteriori* (MAP) assignment to the random variables $\boldsymbol{Y}$. This is equivalent to *energy minimization*, i.e.,

$$\operatorname*{argmax}_{\boldsymbol{y}} \ \mathrm{P}\left(\boldsymbol{y} \mid \boldsymbol{x}\right) = \operatorname*{argmin}_{\boldsymbol{y}} E(\boldsymbol{y}; \boldsymbol{x}). \tag{2.13}$$

Note that an important property of MAP inference is that it does not require computation of the partition function $Z(\boldsymbol{x})$, which would involve summing over all possible joint assignments to $\boldsymbol{y}$ and is generally intractable. Nevertheless, for general graphs MAP inference is still intractable and approximate approaches must be used.

The most simple approach to approximate MAP inference is to perform single variable coordinate-descent in energy space. Specifically, the approach starts with an arbitrary assignment to the random variables. Then, iterating until convergence, each variable Y_i is set, in turn, to the minimizing energy assignment conditioned on the current assignment of all other variables $\boldsymbol{y}_{-i}$. This algorithm is known as iterated conditional modes (ICM) [Besag, 1986], and is shown in Algorithm 2.2. Generalization to larger steps where multiple variables are considered jointly is possible.

A more sophisticated inference procedure is known as max-product belief propagation and is based on the idea of *message passing.* Since we will be operating on energy functions (i.e., in log-space), we present the equivalent *min-sum* variant of the max-product algorithm. The algorithm proceeds by sending messages between nodes (subsets of random variables, usually the cliques of the energy function) that tell the receiving node how much it should modify its belief based on the current belief of the sending node. The scope of a message is the set of variables in common between the two nodes. We will describe the algorithm with respect to a factor graph representation, but note that it generalizes to many other graphical representations.

As discussed, the nodes in a factor graph representation are variables Y_i and factors $\psi_c(\boldsymbol{Y}_c; \boldsymbol{X})$. Messages are sent between variable nodes and factor nodes in both directions. We denote these messages by $m_{c\to i}(y_i)$ and $m_{i\to c}(y_i)$ for the message from factor c to variable i and from variable i to factor c, respectively. The *belief* of a variable is the sum of all messages coming into that variable's node, i.e.,

$$b_i(y_i) = \sum_{c:Y_i \in \boldsymbol{Y}_c} m_{c\to i}(y_i). \tag{2.14}$$

Likewise, the belief for a factor is $b_c(\boldsymbol{y}_c) = \psi_c(\boldsymbol{y}_c; \boldsymbol{x}) + \sum_{i:Y_i \in \boldsymbol{Y}_c} m_{i\to c}(y_i)$, where we have also added the clique potential into the belief. Note that the beliefs and messages are multi-dimensional (i.e., $b_i : \mathcal{Y}_i \to \mathbb{R}$, etc.) and the scope of the messages are a subset of the scope of the beliefs. Summation is defined in a coordinate-wise fashion where the smaller factors are expanded to the larger space. Algorithm 2.3 shows the procedure for general factor addition.

The min-sum algorithm iteratively sends messages between nodes, updating each node's belief. Formally, we define a min-sum message from factor node c to variable node i as

$$m_{c\to i}(y_i) = \min_{\boldsymbol{y}_c \setminus y_i} \left\{ \psi_c(\boldsymbol{y}_c; \boldsymbol{x}) + \sum_{j:Y_j \in \boldsymbol{Y}_c \setminus Y_i} m_{j\to c}(y_j) \right\} \tag{2.15}$$

where the summation is over all incoming messages other than the one from variable

Algorithm 2.3: Factor Addition

```
1 Input  : $\psi_a(\boldsymbol{Y}_a)$          // first addend factor over $\boldsymbol{Y}_a$
2          $\psi_b(\boldsymbol{Y}_b)$          // second addend factor over $\boldsymbol{Y}_b$
3 Output: $\psi_c(\boldsymbol{Y}_c)$          // output factor over $\boldsymbol{Y}_c = \boldsymbol{Y}_a \cup \boldsymbol{Y}_b$
4 set $\boldsymbol{Y}_c = \boldsymbol{Y}_a \cup \boldsymbol{Y}_b$
5 foreach $\boldsymbol{y}_c \in \mathcal{Y}_c$ do
6     set $\boldsymbol{y}_a$ and $\boldsymbol{y}_b$ to be the projection of $\boldsymbol{y}_c$ onto $\boldsymbol{Y}_a$ and $\boldsymbol{Y}_b$, respectively
7     set $\psi_c(\boldsymbol{y}_c) \leftarrow \psi_a(\boldsymbol{y}_a) + \psi_b(\boldsymbol{y}_b)$
8 end foreach
9 return $\psi_c(\boldsymbol{Y}_c)$
```

node i. The message from variable node i to factor node c is similarly defined.

The min-sum algorithm is shown in Algorithm 2.4 for log-space potential functions. The normalization in lines 8 and 12 are included for numerical stability and do not affect the MAP assignment computed. For tree-structured graphs the max-product algorithm can be shown to converge to the correct MAP solution. For arbitrary graphs, the algorithm may not converge and if it does can give incorrect results [Koller and Friedman, 2009]. Nevertheless, the algorithm is popular and empirical results seem to suggest that it works well in practice.

There are many variants on message passing algorithms [Wainwright et al., 2005, Kolmogorov, 2006, Heskes, 2006, Elidan et al., 2006, Globerson and Jaakkola, 2007], all of which result in approximate solutions for large graphical models. In addition, there are numerous other approaches to approximate energy minimization [Greig et al., 1989, Boykov et al., 1999, Kolmogorov and Zabin, 2004, Kolmogorov and Rother, 2007, Rother et al., 2007, Sontag et al., 2008, Komodakis et al., 2007, Gould et al., 2009a] some of which provide strong guarantees. However, these either require a restricted form of energy function (e.g., submodular) or are computationally intractable for the size of models that we consider in this thesis.

Algorithm 2.4: Min-Sum Message Passing

```
1  Input  : $\mathcal{C} = \{\psi_c(\boldsymbol{Y}_c; \boldsymbol{X})\}_{c=1}^{C}$    // set of clique potentials
2           $\boldsymbol{x}$                                  // observed features
3  Output: $\hat{\boldsymbol{y}}$                                  // approximate MAP solution
4  initialize messages: $m_{c \to i}(y_i) = \mathbf{0}$ and $m_{i \to c}(y_i) = \mathbf{0}$ for all $c, Y_i \in \boldsymbol{Y}_c$
5  repeat
6      foreach $c = 1, \ldots, C$ and $i$ such that $Y_i \in \boldsymbol{Y}_c$ do
7          send message: $m_{c \to i}(y_i) = \min_{\boldsymbol{y}_c \setminus y_i} \left\{ \psi_c(\boldsymbol{y}_c; \boldsymbol{x}) + \sum_{j \neq i} m_{j \to c}(y_j) \right\}$
8          normalize: $m_{c \to i}(y_i) = m_{c \to i}(y_i) - \max_y \{m_{c \to i}(y)\}$
9      end foreach
10     foreach $i = 1, \ldots, N$ and $c$ such that $\boldsymbol{Y}_c \ni Y_i$ do
11         send message: $m_{i \to c}(y_i) = \sum_{c' \neq c} m_{c' \to i}(y_i)$
12         normalize: $m_{i \to c}(y_i) = m_{i \to c}(y_i) - \max_y \{m_{i \to c}(y)\}$
13     end foreach
14 until no messages change or maximum iterations reached
15 foreach $Y_i \in \boldsymbol{Y}$ do
16     set $\hat{y}_i = \operatorname{argmin}_y \sum_{c: Y_i \in \boldsymbol{Y}_c} m_{c \to i}(y)$
17 end foreach
18 return $\hat{\boldsymbol{y}}$
```

2.2 Computer Vision

Computer vision is an incredibly vast subject with many subfields. In this section we provide a brief overview of the material needed for this thesis. We begin by reviewing the process of image formation in which a 3D scene is transformed into the discrete 2D image representation. Next, we discuss the computation of low-level image features which provide the basis for high-level scene understanding. Last, we describe two of the fundamental tasks in high-level computer vision—object detection and image segmentation. These are the two main tasks that we would like to unite in this thesis.

2.2.1 Image Formation

An image $\mathcal{I}$ is represented in a computer as a rectangular array of intensity values or color tuples (typically, red, blue and green). Each entry in the array, or *pixel*, represents the color of some point (or small patch) in the real-world. Image formation describes the (geometrical) process by which a real-world scene is projected onto the 2D imaging plane, and is based on the theory of *projective geometry* [Hartley and Zisserman, 2004, Ma et al., 2005].

In the following, we will use $(u, v) \in \mathbb{Z}^2$ to denote 2D pixel coordinates in the camera plane, and $(x, y, z) \in \mathbb{R}^3$ to denote real-world 3D coordinates. Typically, the camera coordinate system will treat the top-left pixel in the image as $(0, 0)$, and we adopt this convention. While on the topic of coordinate systems, we distinguish between two different 3D reference frames: the *world coordinate system* and the *camera coordinate system*. The transformation from a point $(x_w, y_w, z_w) \in \mathbb{R}^3$ in the world coordinate system to a point $(x_c, y_c, z_c) \in \mathbb{R}^3$ the camera coordinate system involves a *rigid body* transform:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \boldsymbol{R} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \boldsymbol{t} \tag{2.16}$$

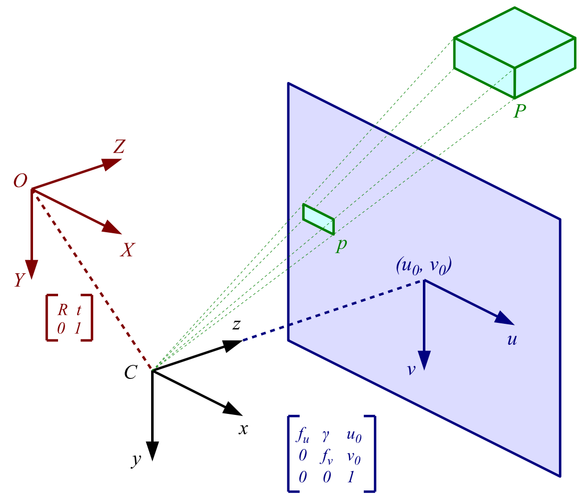

P
O
Z
X
Y
p
(u_0, v_0)
$\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$
z
u
C
v
x
y
$\begin{bmatrix} f_u & \gamma & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$

Figure 2.2: The geometry of image formation.

where $\boldsymbol{R} \in SO(3)$ defines the rotation from world coordinate system to camera coordinate system, and $\boldsymbol{t} \in \mathbb{R}^3$ is the translation between the world origin and camera origin. This is illustrated in Figure 2.2. For the purpose of this thesis, we will assume that there is no translation between the world and camera coordinate systems, i.e., $\boldsymbol{t} = \mathbf{0}$. For the remainder of this section, unless explicitly stated, we will treat all 3D vectors (x, y, z) as being in the world coordinate system.

Now, consider an ideal camera model, i.e., one with no lens distortion.[2] Then, a pixel p with coordinates (u_p, v_p) (in the camera plane) is the image of a point in 3D space that lies on the ray extending from the camera origin through (u_p, v_p) in the camera plane. The ray $\mathbf{r}_p \in \mathbb{R}^3$ in the world coordinate system is given by

$$\mathbf{r}_p \propto \boldsymbol{R}^{-1}\boldsymbol{K}^{-1} \begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix} \tag{2.17}$$

[2]Lens distortion is usually only a problem for cheap cameras (e.g., webcams or cell-phones) or cameras with very wide-angle lenses. The distortion effects can be calibrated out using well-known techniques [Tsai, 1986]. However, for the purpose of this thesis, we will assume that our images have minimal lens distortion.

where $\boldsymbol{K} \in \mathbb{R}^{3\times3}$ is the camera matrix [Ma et al., 2005], which projects 3D points onto the camera plane (see Figure 2.2). Unless otherwise specified, we will assume that $\mathbf{r}_p$ has been normalized (i.e., $\|\mathbf{r}_p\|_2 = 1$). Note, that any point that lies along this ray will be projected onto the same image pixel and the pixel's color or intensity value will depend on the closest such point. For an ideal camera, the camera matrix has the form

$$\boldsymbol{K} = \begin{bmatrix} f_u & \gamma & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.18}$$

where f_u and f_v are the (u- and v-scaled) focal lengths[3] of the camera, and the principal point (u_0, v_0) is the center pixel in the image. The parameter γ models skewness between the x- and y-imaging axes and can be considered zero for most practical purposes.

In this thesis, we will generally assume that the image was take with the camera's horizontal (x) axis parallel to the ground, and thus only model rotations about this axis. A tile angle of θ then yields the rotation matrix

$$\boldsymbol{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}. \tag{2.19}$$

Since pixels are discretized, and the recording of light intensity involves noisy physical processes, the actual generation of an image of a real-world scene is significantly more complicated than described here. Nevertheless, the description above provides us with a sufficiently accurate model for 3D reconstruction (as we shall see in Chapter 6).

[3]The focal length can be understood geometrically as the distance between the camera center and the imaging plane.

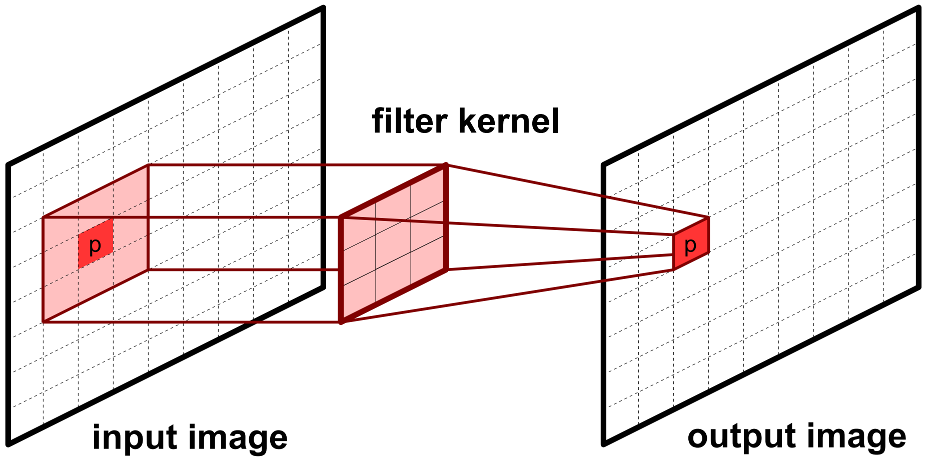


Figure 2.3: Convolution of an image with a filter. The value of an output pixel is determined by placing the filter kernel over the input image at the corresponding pixel location. The value is then the dot-product between the reflected filter kernel and the input image.

2.2.2 Low Level Image Features

Representing an image by an array of three color components is useful for viewing and for storage, but not very helpful for automatic scene understanding. For that we need to extract image features that represent more salient features, such as edges and textures. One of the first steps towards this higher-level representation is to convolve the image with a number of different filters. These filters can be applied to the image's color channels or, more commonly, to a grayscale version of the image.

Convolution of an arbitrary size input image $\mathcal{I}_i$ with a $(2w+1) \times (2h+1)$ kernel filter K to produce an output (response) image $\mathcal{I}_o$, denoted $\mathcal{I}_o = \mathcal{I}_i * K$, is given by

$$\mathcal{I}_o(u,v) = \sum_{i=-w}^{w} \sum_{j=-h}^{h} \mathcal{I}_i(u-i, v-j) K(i,j) \tag{2.20}$$

This operation is depicted graphically in Figure 2.3. There are several strategies for dealing with positions where the filter kernel extends beyond the boundary of the image, including truncating the output image, zero-padding the input image, or replicating boundary pixel values. In our work, we adopt the latter approach.

One of the most simple filters is aimed at finding edges in an image by computing discrete derivatives. The filter is often combined with a Gaussian smoothing filter to reduce noise. The resulting Sobel filter (for computing vertical gradients) is

$$K = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \tag{2.21}$$

which, as can be seen, separates into a vertical smoothing filter and horizontal difference operator. Horizontal edges can be found by filtering with K^T. The gradient magnitude (or strength) is $\sqrt{\mathcal{I}_x^2 + \mathcal{I}_y^2}$ where $\mathcal{I}_y$ and $\mathcal{I}_x$ are the vertically and horizontally filtered images, respectively. Similarly, the orientation of the gradient at each pixel is $\tan^{-1}(\mathcal{I}_x(u, v)/\mathcal{I}_y(u, v))$. Naively, thresholding the gradient magnitude is a simple way to find edges but is likely to produce noisy results. Usually, to find better edges, these Sobel operators are used as a preprocessing stage for a more sophisticated algorithm, for example, the famous Canny edge detector [Canny, 1986].

Edges and small edge fragments are important features for finding region boundaries and identifying objects. However, committing to hard edges as a preprocessing step often leads to poor performance. A better approach is to robustly describe the local edge gradients around each pixel (or a sparse subset of pixels known as *keypoints*). One of the most popular methods for describing such local gradient information is the SIFT descriptor of Lowe [2004]. Briefly, the descriptor is constructed by considering gradients in a neighborhood region around each pixel. The neighborhood is divided into a 4×4 array and each cell further subdivided into a 4×4 subarray. Smoothed gradient magnitudes for each cell in the subarrays are accumulated into a histogram consisting of eight bins according to gradient orientation. These histograms are then combined (by weighted-average) into histograms for the coarser 4×4 array cells. The descriptor is rotated by the dominant orientation within the local neighborhood to provide rotation invariance. The resulting descriptor contains 128 elements (16 histograms, one for each coarse cell, and each histogram containing eight bins). Finally, the descriptor is normalized to unit length to make it robust to affine changes to

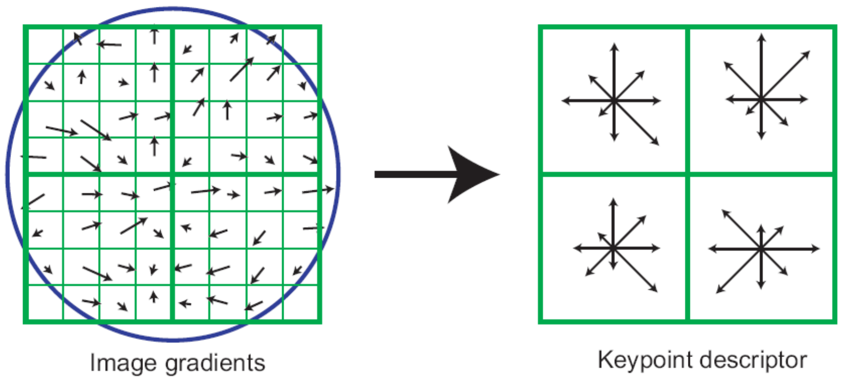


Figure 2.4: The SIFT feature descriptor. For illustration, the figure shows a 2×2 descriptor computed from an 8×8 subarray. The standard descriptor, however, is based on a 4×4 array computed from a 16×16 subarray. Image courtesy of Lowe [2004].

illumination. An illustration of the SIFT feature descriptor is shown in Figure 2.4.

Filter Banks

Other than the SIFT descriptor (and variants of it), another method for constructing a better pixel-level image representation is to run a host of different filters over the image. Per pixel image feature vectors are then constructed by assembling the responses from this *filter bank*. A very popular filter bank for multi-class pixel labeling defines a set of seventeen filters [Shotton et al., 2006]. We will refer to this filter bank as the *texton* filter bank. The filter bank is parameterized by a base scale κ.

The first three filters are Gaussian smoothing filters at scales κ, 2κ, and 4κ that operate on each of the three image color channels. Instead of RGB, the image is first converted to the more visually perceptive CIELab color space. The Gaussian filter at scale σ is defined as

$$G(u, v) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{u^2 + v^2}{2\sigma^2}\right) \tag{2.22}$$

The remaining filters all operate on a grayscale version of the image. The next four filters are x- and y-derivative-of-Gaussian (DoG) filters at scales 2κ and 4κ. The

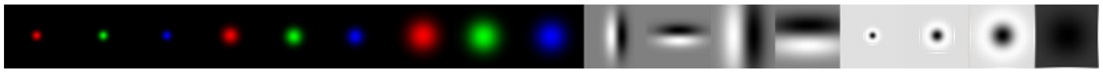


Figure 2.5: Illustration of the 17-dimensional "texton" filters used in multi-class image segmentation [Shotton et al., 2006]. Note that the filters operate in CIELab color space (RGB for the color components shown for illustrative purposes only).

x-derivative-of-Gaussian filter is defined by

$$G_x(u,v) = \frac{\partial}{\partial u} G(u,v) \tag{2.23}$$
$$= -\frac{u}{\sigma^2} r^2 G(u,v) \tag{2.24}$$

where $r^2 = \frac{u^2+v^2}{2\sigma^2}$. A similar equation exists for the y-derivative, $G_y(u,v)$.

The last four filters are Laplacian-of-Gaussian (LoG) filters at scales κ, 2κ, 4κ, and 8κ, defined by

$$\mathrm{LoG}(u,v) = \nabla^2 G(u,v) \tag{2.25}$$
$$= \frac{1}{\pi\sigma^4}(r^2 - 1)\exp(-r^2) \tag{2.26}$$

where $\nabla^2 = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)$ is the Laplacian operator and $r^2 = \frac{u^2+v^2}{2\sigma^2}$.

An illustration of the seventeen texton filters is shown in Figure 2.5. Response images when applied to an example input image are shown in Figure 2.6. We use these filters when building up region appearance features in our scene decomposition model.

2.2.3 Sliding-Window Object Detection

Sliding-window object detection is a simple yet effective technique for simultaneous localization and recognition of objects in images. The approach was popularized by the highly successful work of Viola and Jones [2001, 2004], which aims to find faces in images in real-time. Some forerunners to the Viola-Jones work also suggested the sliding-window idea, e.g., [Papageorgiou et al., 1998], and it is now the dominant

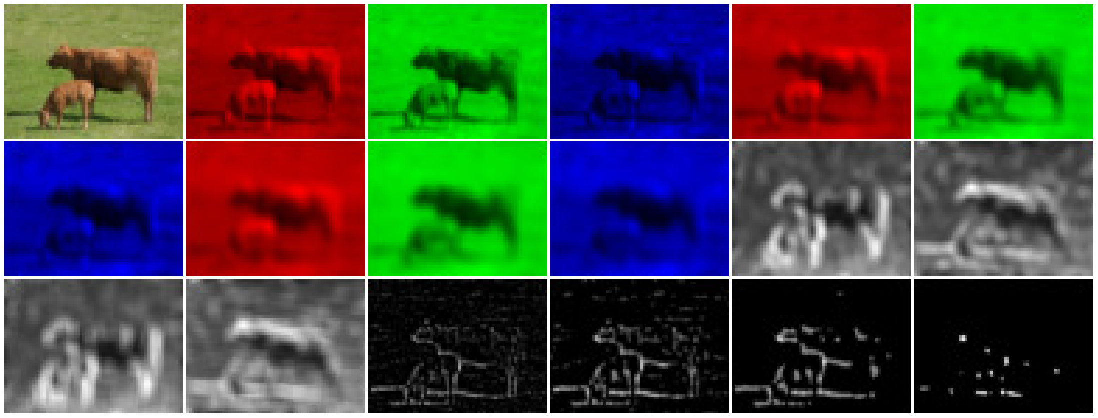


Figure 2.6: Application of 17-dimensional "texton" filters to an image. The original image is shown in the top-left corner.

technique for object detection.

The approach involves scanning the image with a fixed-size rectangular window and applying a classifier to the sub-image defined by the window. The classifier extracts image features from within the window (sub-image) and returns the probability that the window tightly bounds an instance from a particular object category. The process is repeated on successively scaled copies of the image so that objects can be detected at any size. Figure 2.7 depicts this process. The (x, y)-location of the window and scale σ of the image implicitly defines the bounding box, or set of pixels, belonging to the candidate object $\mathcal{B}(x, y, \sigma) = \{(u, v)\} \subseteq \mathcal{I}$. Usually a post-processing set of non-maximal neighborhood suppression is applied to the output set of bounding boxes to remove overlapping detections of the same object.

Object detection is a very active area of research with new features, improved classifier technologies, and better learning techniques being developed regularly. In this thesis we build on two proven sliding-window detectors that vary in the features and classification engine that they use.

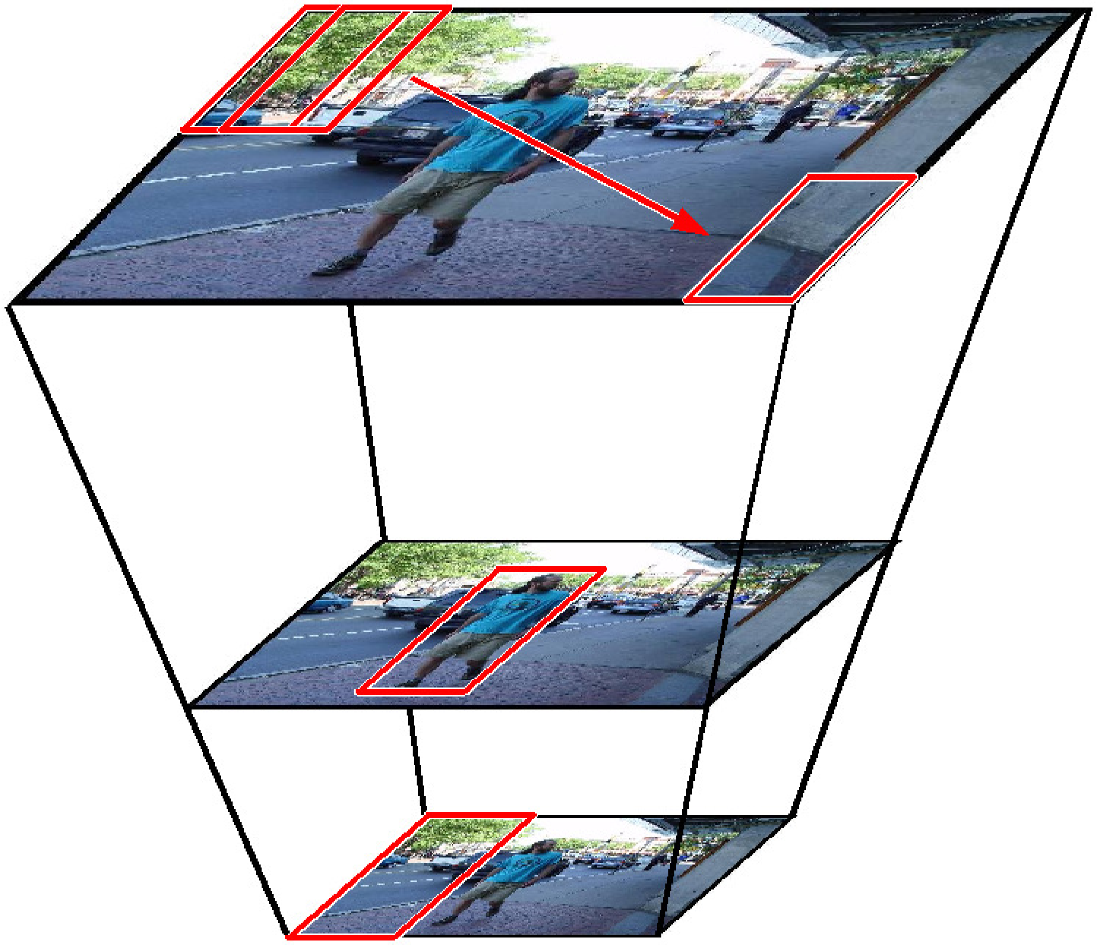


Figure 2.7: Sliding-window object detection. The image is scanned using a fixed-size rectangular window and a classifier applied to the sub-image defined by the window at each location. Scanning is repeated on successively down-sampled copies of the image so that objects can be detected at different scales.

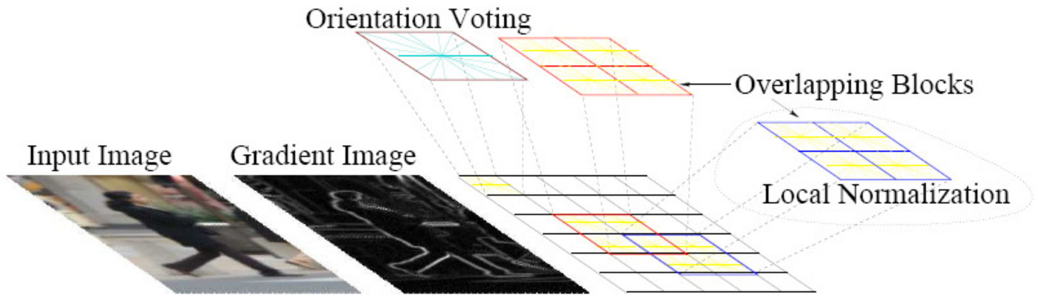


Figure 2.8: Feature construction for HOG-based object detector. Image courtesy of Dalal and Triggs [2005].

Histogram of Oriented Gradients Detector

The histogram of oriented gradients (HOG) object detector of Dalal and Triggs [2005] constructs features motivated by the SIFT descriptor of [Lowe, 2004]. Briefly, x- and y-derivative filters are applied to the image window, and the orientation of the filter responses quantized into discrete bins. A histogram over (quantized) orientations is then constructed in overlapping cells on the detection window by summing the gradient magnitudes within the cell. Finally, the feature vector is normalized, as was done for the SIFT descriptor, to make it invariant to affine changes in illumination. An illustration of the HOG feature construction is shown in Figure 2.8.

The detector learns a separate linear SVM classifier for each object category from a training set of positive and negative examples (for that category). Robust performance is achieved by running the learned detector over a set of training images known to not contain the target object category. This produces a set of false positive examples that are used in a subsequent training iteration to improve the detectors performance. This step can be repeated, but usually produces very good results after just one iteration. We incorporate the same strategy to mine for difficult negative examples when we learn our region-based models.

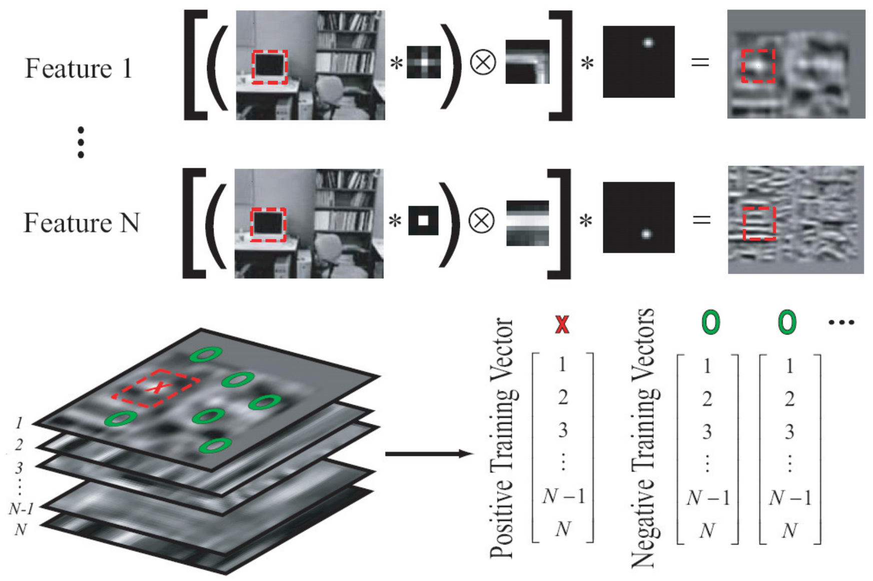


Figure 2.9: Feature construction for patch-based object detector. Image courtesy of Murphy et al. [2006].

Patch-based Object Detector

Another variant of the sliding-window detector is the approach taken by Murphy et al. [2006] (and other works in the series, e.g., [Torralba et al., 2004a]). Here, a dictionary of image patches is constructed at training time by randomly selecting small patches from a set of training images. Associated with each patch is a spatial mask (relative to the detection window) over which the patch response is valid. The mask is derived by taking a small rectangular region around the patch's original location. In addition to patches taken from the grayscale (intensity) image, patches are also extracted from filtered versions of the image.[4] Thus the dictionary is comprised of the triplet of patch, spatial mask, and image filter.

Features are generated by convolving the grayscale image or filtered image with patches from the patch dictionary and taking the feature value to be the maximum response within the spatial mask associated with the patch. This process is illustrated

[4]In our work, we extract patches from the grayscale (intensity) image and the edge-filtered (gradient magnitude) image.

in Figure 2.9. Concretely, the feature value corresponding to the i-th patch is

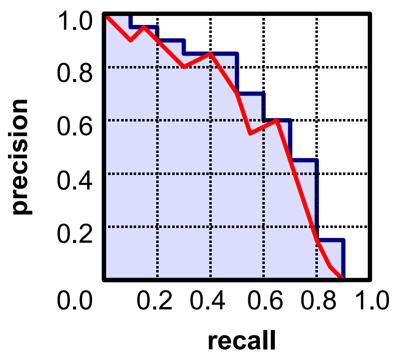
$$f_i = \max_{(u,v)\in w_i} \left|(\mathcal{I}_\sigma * K_i) \otimes g_i\right|(u, v) \tag{2.27}$$

where $\mathcal{I}_\sigma$ is the image at scale σ, K_i is the filter kernel associated with the i-th patch, g_i is the image patch, w_i is the spatial mask, and $\otimes$ is the normalized cross-correlation operator.

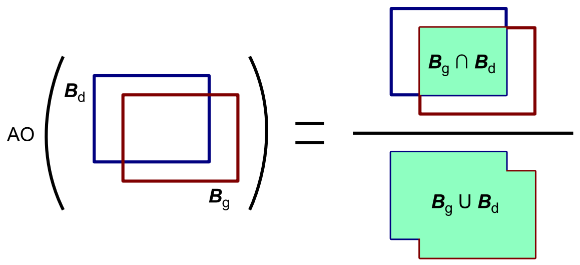
Given the dictionary of patch features and a set of positive and negative training images, the approach learns a gentle-boost classifier [Friedman et al., 1998] over two-split decision stumps for each object category. The patch dictionary is constructed by randomly selecting patches from the positive training examples. In our implementation, the patches vary in size from 4×4 pixels to half the image window, and we fix the spatial mask w_i to 7×7. We then train the object detectors in two stages for improved efficiency: first we select all positive training examples and a subsample of negative training examples at random, and train a boosted classifier for a fixed number of rounds; next, we trim the patch dictionary to remove all patches not used by this initial classifier; and finally, we retrain using all negative training examples to obtain the final object detector. Like the HOG-based detector, the process is repeated on negative examples mined by running the detector over a set of image known not to contain the object of interest.

Evaluating Object Detector Performance

As discussed, the vast majority of object detection algorithms output bounding boxes for candidate objects in the image. Associated with each bounding box is an object class label and score (e.g., probability). Thus, the output of a (bounding box based) object detection algorithm is a set of 3-tuples $\langle \mathcal{B}_i, C_i, s_i \rangle$ where $\mathcal{B}_i$ denotes the bounding box for the object candidate, C_i denotes its class label, and s_i denotes its score. The quality of the object detection algorithm is evaluated by comparing the detected objects with ground-truth object annotations on a hold-out (unseen test) set of images. Of interest are the *recall* of the algorithm—measured as percentage of ground-truth objects detected by the algorithm—and the *precision* of the



(a) Precision-recall curve



(b) Area-of-overlap score

Figure 2.10: Measuring object detection performance. (a) shows a precision-recall curve in red. Performance is summarized by the average precision measured as the 11-point interpolated area under the curve. (b) shows calculation of the area-of-overlap score for determining correspondence between a ground truth objects $\mathcal{B}_g$ and a candidate detection $\mathcal{B}_d$.

algorithm—measured as percentage of detections that are true objects.

The first step in evaluating performance is to decide on a threshold at which to start considering candidate detections. In general, there will be a trade-off between precision and recall: The more candidate detections we allow, the greater the chance that we will detect all the objects in our dataset. However, for a good detector, the higher we set the detection threshold (i.e., fewer detections), the greater the chance that we will not see any false-positive detections. This trade-off is captured by the *precision-recall* (PR) curve which plots precision against recall as we sweep over detection thresholds.

The area under the PR curve provides a convenient measure of object detector performance. It is common to smooth the curve by taking the 11-point *average precision* defined by

$$AP = \frac{1}{11}\sum_{i=0}^{10} \max_t \left\{ p(t) : r(t) \geq \frac{i}{10} \right\} \tag{2.28}$$

where $p(t)$ and $r(t)$ are the precision and recall, respectively, at threshold t. Note that there are a number of other metrics that can be used to measure object detector quality, including maximum F_1-score, recall at a fixed number of false detections per image, etc. However, these will not be considered in this thesis.

We are now left with one final problem: How do we determine if a detection returned by the algorithm corresponds to a ground-truth object? In general, detection windows will not perfectly align with ground-truth annotations, and so a measure of overlap is used to determine correspondence. Specifically, the *area-of-overlap* score (also known as Jaccard coefficient or intersection-over-union score) measures the amount of overlap between a detection candidate and a ground-truth object (with the same class label) by dividing the area of their intersection by the area of their union, i.e.,

$$AO(\mathcal{B}_d, \mathcal{B}_g) = \frac{|\mathcal{B}_d \cap \mathcal{B}_g|}{|\mathcal{B}_d \cup \mathcal{B}_g|} \tag{2.29}$$

where $\mathcal{B}_d$ is the set of pixels belonging to the candidate detection and $\mathcal{B}_g$ is the set of pixels belonging to the ground-truth annotation. A threshold (typically, 0.5 or 0.2) is then used to decide whether the detection corresponds to the ground-truth object or not. Note that this notion of overlap can be extended beyond the bounding box representation to arbitrary pixel sets.

A complication arises when multiple overlapping detections (or ground-truth objects) are present. In this case, we need to decide which of the overlapping objects should correspond since each ground-truth object should be explained by at most one detection. There are two common strategies used to solve this problem. The first strategy is a greedy approach in which we first sort the candidate detections in decreasing score and associate ground-truth objects to the overlapping detection with the highest score.[5] The second approach is to form a bipartite graph between the detection candidates and ground-truth objects where an edge is placed if the area-of-overlap exceeds some predefined threshold. Edges are weighted by detection score and correspondences are then determined by finding the maximum matching.

In summary, for evaluation of object detection performance, detection candidates

[5]If a detection candidate has the choice between two ground-truth objects, it should take the one with highest overlap score.

and ground-truth objects are matched using three basic criteria:

- the detection label C_i must match the ground-truth label;
- the detection must have sufficient overlap with the ground-truth object; and
- at most one detection can be associated with a ground-truth object (that is, we cannot count true-positives multiple times).

With the detection candidates and ground-truth objects matched we can compute precision and recall values at various thresholds to produce a PR curve and calculate average precision.

2.2.4 Image Segmentation

Image segmentation can be generally described as the task of partitioning or segmenting an entire image into distinct regions. In this section, we review three common flavors of image segmentation.

Unsupervised Image Segmentation

Unsupervised image segmentation is a bottom-up procedure that aims to group pixels into regions with similar appearance. The coarseness of the segmentation is determined by how similar we require the regions to be and can usually be controlled by a parameter in the algorithm.

Figure 2.11 shows the output of three popular unsupervised segmentation algorithms. The first algorithm, is a graph-based segmentation algorithm by Felzenszwalb and Huttenlocher [2004]. Briefly, this algorithm creates a graph where the nodes are pixels and edges connect nodes that are adjacent in the image (i.e., neighboring pixels). Each edge is weighted by the Euclidean distance between the pixels' color vectors. We denote the weight for the edge between pixels p and q by w_{pq}, where $w_{pq} = \infty$ if p and q are not adjacent. The algorithm begins by assigning each pixel to its own region. It then greedily merges regions by considering the weight of edges

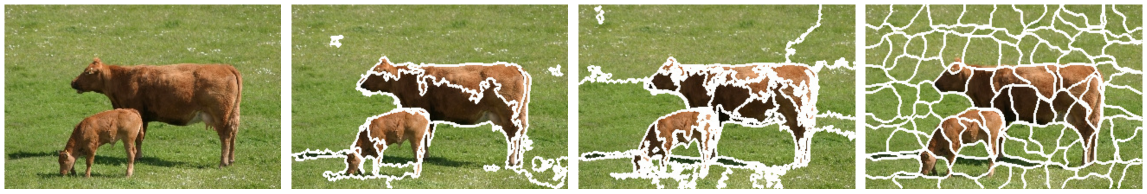

(a) Image (b) Graph-based (c) Mean-shift (d) N-cuts

Figure 2.11: Comparison of different unsupervised image segmentation algorithms. Typical results from (b) the graph-based approach of Felzenszwalb and Huttenlocher [2004], (c) mean-shift segmentation algorithm of Comaniciu and Meer [2002], and (d) normalized-cuts method of Shi and Malik [2000]. In terms of runtime, (b) is the fastest and (d) is the slowest.

that span the regions. Specifically, the merge criterion for two regions $\mathcal{P}_i$ and $\mathcal{P}_j$ is

$$\min_{p \in \mathcal{P}_i, q \in \mathcal{P}_j} w_{pq} \leq \min \left\{ \max_{e \in E(\mathbf{mst}(\mathcal{P}_i))} w_e + \tau(\mathcal{P}_i), \max_{e \in E(\mathbf{mst}(\mathcal{P}_j))} w_e + \tau(\mathcal{P}_j) \right\} \tag{2.30}$$

where $\tau(\mathcal{P}) = \frac{k}{|\mathcal{P}|}$ is a thresholding function that rewards larger clusters (which may have better estimates of their appearance), and $\mathbf{mst}\,(\mathcal{P})$ is the minimum spanning tree over nodes in $\mathcal{P}$. The algorithm is fast, only having to iterate once through all the edges. However, it can produce irregularly shaped segments, sometimes with long long tails (i.e., singly connected trails of pixels).

Another popular unsupervised segmentation algorithm is the mean-shift segmentation algorithm of Comaniciu and Meer [2002]. Mean-shift segmentation treats pixels in an image as 5-dimensional feature vectors (over color and location), and the aim is estimate the modes of the distribution from which these the vectors (in a given image) are drawn. This relies on a technique known as *kernel density estimation*, which defines the distribution as

$$\hat{f}(\boldsymbol{x}) = \frac{1}{Nh^d} \sum_{i=1}^{N} K\left(\frac{\boldsymbol{x} - \boldsymbol{x}^{(i)}}{h}\right) \tag{2.31}$$

where $\boldsymbol{x}^{(i)}$ are the feature vectors, $K(\xi)$ is a kernel function, d is the dimensionality of the feature space (i.e., $d = 5$ for mean-shift image segmentation), and the parameter

Algorithm 2.5: Mean-shift Segmentation [Comaniciu and Meer, 2002].

```
1  Input  : $\{\boldsymbol{x}_p : p \in \mathcal{I}\}$        // 5-dimensional pixel features
2           $(h_s, h_r)$                                      // spatial and color range bandwidths
3  Output: $\{\mathcal{P}_r\}$                                // segmented image (set of regions)
4  foreach pixel $p \in \mathcal{I}$ do
5      set $\boldsymbol{z}_p = \boldsymbol{x}_p$
6      repeat
7          update $\boldsymbol{z}_p \leftarrow \frac{\sum_{q \in \mathcal{I}} K'(\boldsymbol{z}_p - \boldsymbol{x}_q)\boldsymbol{x}_q}{\sum_{q \in \mathcal{I}} K'(\boldsymbol{z}_p - \boldsymbol{x}_q)}$ where $K(\xi)$ is defined by Eq. 2.32
8      until convergence
9  end foreach
10 group all pixels with $\boldsymbol{z}_p$ within neighborhood defined by spatial and range
   bandwidths $(h_s, h_r)$ into a single region $\mathcal{P}$
11 return set of regions $\{\mathcal{P}_r\}$
```

$h \in \mathbb{R}_+$ controls the *bandwidth* of the kernel. For image segmentation the kernel function is defined as

$$K(\xi) = \frac{C}{h_s^2 h_r^3} \exp\left(-\frac{1}{2}\left\|\frac{\xi_s}{h_s}\right\|^2\right) \exp\left(-\frac{1}{2}\left\|\frac{\xi_r}{h_r}\right\|^2\right) \tag{2.32}$$

where h_s and h_r define the spatial and color range bandwidths, respectively.

The mean-shift algorithm is an iterative algorithm that, beginning at each pixel, moves to the centroid of the pixels within its neighborhood (defined by the kernel function). The algorithm iterates until it reaches a stationary point, i.e., a mode. Segmentation is achieved by grouping all pixels that converge to the same stationary point. The algorithm is summarized in Algorithm 2.5.

The final unsupervised image segmentation algorithm shown in Figure 2.11 is the normalized-cuts algorithm. This approach defines a segmentation as a k-way cut on a graph. Like the the graph-based approach of Felzenszwalb and Huttenlocher [2004], normalized-cuts defines a graph over pixels in the image. However, here the edge weights are defined by similarity (i.e., higher is more similar). The cost of any segmentation is a function of the weights for all edges that are cut (i.e., whose adjacent nodes belong to different segments). The key to the algorithm is the method by which cuts are scored. A naive approach to scoring cuts simply sums over the weights of the edges that are cut. However, this leads to pathological cases where a single pixel is segmented from the rest of the image. The crucial insight in normalized-cuts, is that scoring should take into account the size of the segments that are being separated. We will skip the details of the actual algorithm, suffice to say that finding the exact solution defined by the normalized-cuts objective is NP-complete and an approximate solution is obtained by solving a generalized eigenvalue problem. This makes the segmentation algorithm very expensive in both storage requirements and running time.

Most unsupervised segmentation techniques, including all those discussed above, incorporate a post-processing step that removes small regions by merging them with adjacent regions.

In our work, we make use of unsupervised image segmentation algorithms to generate multiple over-segmentations of an image. That is, segmentations in which the regions are, hopefully, subregions of semantically coherent parts of the image. In this case, the segments are often called *superpixels*. The multiple over-segmentations are generated by varying the parameters (e.g., spatial and color range bandwidths) of the segmentation algorithms.

Figure/Ground Segmentation

Sometimes we are just interested in segmenting a foreground object (*figure*) from the rest of the image. This may be the case when a user provides some indication of which areas are foreground and which are background, as in the case of interactive segmentation [Boykov and Jolly, 2001, Rother et al., 2004], or when an instance of a particular object category is known to be within the image (and well-framed) [Kumar et al., 2005, Borenstein and Ullman, 2002]. These works typically build appearance models of the foreground object and surrounding background, and then perform a binary graph-cut to separate foreground from background along high-contrast boundaries.

In this thesis, we are interested in scenes that contain multiple regions and objects that we would like to segment, so we do not consider these methods further.

Multi-Class Image Labeling

Unlike object recognition methods that aim to find a particular object, multi-class image labeling methods are aimed at concurrent classification of *all* pixels in an image. The pixels are classified into one of a predetermined set of labels, e.g., *sky*, *grass*, *road*, etc., and the segmentation defined by the boundaries between regions having different semantic labels.

Most multi-class labeling methods achieve their goal by taking into account local (pixel or superpixel) appearance signals along with a preference for smoothness, i.e., classifying visually-contiguous regions consistently. For multi-class image labeling, this is often achieved by constructing a conditional Markov random field (CRF) over the image that encodes local and pairwise probabilistic preferences (e.g., [He et al., 2004, Shotton et al., 2006]). For example, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ defined over the image, with nodes $\mathcal{V}$ representing pixels or regions and edges $\mathcal{E}$ encoding the neighborhood structure, the CRF is defined as

$$E(\boldsymbol{Y}; \boldsymbol{X}) = \sum_{i \in \mathcal{V}} \psi_i(Y_i; \boldsymbol{X}) + \sum_{(i,j) \in \mathcal{E}} \psi_{ij}(Y_i, Y_j; \boldsymbol{X}) \tag{2.33}$$

where $\boldsymbol{X}$ are features computed from the image (for example, texton filter responses). An example CRF defined over superpixel regions is illustrated in Figure 2.12. Optimizing this energy function is then equivalent to finding the most probable image labeling.

Performance of multi-class image labeling algorithms is often evaluated by measuring per-pixel accuracy over a hold-out (test) data set. Results can either be micro-averaged (favoring the larger classes) or macro-averaged (treating all classes with equal weight). Specifically, for a K-class problem, with predicted labels $\{\hat{y}_i\}_{i=1}^N$ and corresponding ground truth labels $\{y_i\}_{i=1}^N$, these averages are computed as

$$A_{\text{micro}} = \frac{\sum_{i=1}^N \mathbf{1}\{\hat{y}_i = y_i\}}{N} \tag{2.34}$$

$$A_{\text{macro}} = \frac{1}{K} \sum_{k=1}^K \frac{\sum_{i=1}^N \mathbf{1}\{(\hat{y}_i = y_i) \wedge (y_i = k)\}}{\sum_{i=1}^N \mathbf{1}\{y_i = k\}} \tag{2.35}$$

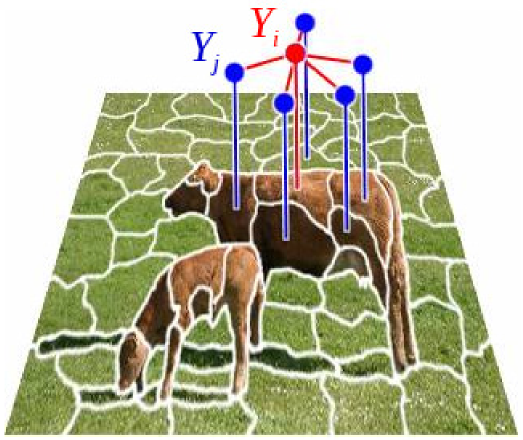


Figure 2.12: Illustration of a conditional Markov random field for multi-class image labeling. Highlighted is one of the random variables Y_i and one of its neighbors Y_j.

where $\mathbf{1}\{P\}$ is one if P is true and zero otherwise. In our work, we use the micro-averaged accuracies when reporting performance.

Like object detection, multi-class image labeling is a very active area of research with improved techniques being published all the time. The majority of successful techniques build on the ideas discussed above.

2.3 Chapter Summary

This chapter reviewed some important background material in machine learning and computer vision. We make use of this material when developing and evaluating our scene understanding algorithms in subsequent chapters of the thesis.

Chapter 3

Image Datasets and Dataset Construction

There have been three significant technological thrusts that have led to the recent advances in high-level computer vision. First, there has been exponential growth in raw computational power (both processor speed and system memory) necessary for running computer vision algorithms. Second, efficient large-scale machine learning techniques have been developed and low-level feature representations improved (both of which were previously discussed in Chapter 2) resulting in more robust performance on novel images. Last, the prevalence of digital cameras and online sharing of media has resulted in the availability of large quantities of labeled data for training and evaluation. In this chapter we describe a few of the standard datasets used by the computer vision community. Our focus will be on datasets applicable to outdoor scene understanding. We also introduce a new dataset for scene understanding and describe its construction leveraging Amazon Mechanical Turk (AMT) for labeling. Unlike previous datasets, this new dataset includes both semantic and geometric annotations.

3.1 Standard Image Datasets

Image datasets vary across many dimensions according to the purpose for which they were originally designed. Annotations can vary from global image descriptions to pixel-level details. For example, a global image description may include the scene category (e.g., urban, rural, highway, coastal, etc.) or location (e.g., bedroom, bathroom, kitchen, etc.). At the pixel-level, annotations may include detailed object outlines or masks.

Datasets intended for object recognition and object detection usually provide cropped images of objects, where each image is tagged with specific object categories. These datasets range from relatively small collections with only a few images per object category, such as the popular Caltech-101 [Fei-Fei et al., 2004] or Caltech-256 [Griffin et al., 2007] datasets, to "Internet-scale" datasets with millions of images divided into thousands of object categories, e.g., ImageNet [Deng et al., 2009]. More sophisticated datasets in this class may also provide bounding box annotations of objects in the context of a larger scene (i.e., uncropped images) or object outlines, which are provided either as an explicit set of points along the boundary or as a segmentation mask. Ponce et al. [2006] give a detailed discussion of the issues involved in constructing datasets for object detection. The authors highlight some of the problems and artifacts present in the early datasets (e.g., Caltech-101 and Caltech-256 [Fei-Fei et al., 2004, Griffin et al., 2007]) such as inadequate intra-class variation (e.g., object pose). These issues are less prevalent in more recent datasets (e.g., LabelMe [Russell et al., 2008] and PASCAL [Everingham et al., 2010]).

Unlike datasets for object detection, all segmentation and multi-class image labeling datasets provide annotations at the pixel level. The Berkeley Segmentation Dataset [Martin et al., 2001] contains 300 images segmented by multiple human subjects and is intended to provide a basis for developing segmentation and boundary detection algorithms. The annotations provide information on region boundaries, but do not provide any semantic information about the regions themselves. Other datasets, intended for multi-class image labeling, annotate each pixel with a semantic (or geometric) class (e.g., the 21-class MSRC dataset [Criminisi, 2004] and the Geometric Context (GC) dataset [Hoiem et al., 2007a]). However, most of these datasets do not distinguish between multiple instances of the same object class. For example, two adjacent cars in an image will be labeled as a single "car" region. More sophisticated datasets (such as the PASCAL VOC dataset [Everingham et al., 2010] discussed later) provide both boundary information to separate instances of the same class and semantic annotations.

Datasets also vary widely in quality. LabelMe [Russell et al., 2008] is a freely available online dataset with user-supplied annotations. The dataset provides tools for adding image sets and updating annotations. Little quality control is enforced, however, resulting in a number of problems including inconsistent names for the same object, poorly annotated object boundaries, and sparsely labeled images (e.g., only one object outlined). Researchers typically select a subset of the images from the LabelMe collection by applying ad hoc filtering rules.

At the other end of the spectrum, the Lotus Hill Research Institute (LHI) has constructed a commercially available dataset [Yao et al., 2007] for image parsing. The dataset contains over 600,000 images, which are labeled with accurate region boundaries and annotated in a part-based hierarchy (e.g., entire cars are segmented from the rest of the scene, and then each car is further segmented into individual parts, such as wheels and doors). A subset of the dataset has recently been made available to academic researchers at no cost.

Between the two extremes of LabelMe [Russell et al., 2008] and LHI [Yao et al., 2007], the PASCAL Network of Excellence has released a well-annotated dataset as part of an annual challenge aimed at advancing the state-of-the-art in object detection techniques. The PASCAL Visual Object Classes (VOC) challenge [Everingham et al., 2010], initiated in 2005, is run on a dataset of over 7,000 images of realistic scenes. The dataset is split into training, validation and test, with the training and validation annotations provided as part of the dataset. Annotations for the test images are withheld by the organizers so that evaluation must be done by submitting algorithm results to an online evaluation server. This prevents researchers from inadvertently over-fitting their models to the test data.

The PASCAL VOC dataset provides bounding box annotations for 20 object categories. A subset of the data is also annotated with pixel-level masks for the same 20 object categories. It is becoming the de facto standard for developing and testing computer vision algorithms. Unfortunately, since the challenge is aimed at object detection, many of the images (e.g., close-up shots or indoor scenes) and object categories (e.g., sofas, chairs, computer monitors) are not applicable to the work presented in this thesis. Furthermore, the dataset does not provide geometric annotations.

To our knowledge, none of the publicly available datasets in the computer vision community provide both semantic and geometric annotations on the same set of images. This is problematic for models such as ours, which attempt to exploit the interaction between semantic classes and scene structure (i.e., geometry). Thus, we constructed our own dataset, the "Stanford Background Dataset", formed by taking images from existing standard datasets and annotating them with suitable semantic and geometry information. Annotation was performed using Amazon Mechanical Turk (AMT)—an online marketplace for simple inexpensive jobs. We describe this dataset and the AMT labeling procedure in Section 3.2. In order to compare to previously published results we also evaluate our methods on other existing datasets where appropriate.

The datasets used in this thesis are:

- **21-Class MSRC Segmentation Dataset [Criminisi, 2004]**. This dataset provides pixel-level annotations for 21 different background and foreground object categories on 591 images. The labeling is approximate with object boundaries tending to be very inaccurate. A further complication is the presence of many unlabeled pixels (denoted as "void"), which are ignored during both training and test. Nevertheless, this dataset is a popular dataset for evaluating the performance of multi-class image labeling algorithms.

- **Geometric Context [Hoiem et al., 2007a]**. Similar to the multi-class image labeling datasets for semantic object categories, this dataset provides pixel-level geometric annotations, e.g., sky, horizontal, vertical, for 300 images of outdoor scenes. The dataset does not annotate the location of the horizon in the images.

- **Stanford Background Dataset [Gould et al., 2009b]**. We constructed this dataset for the purpose of evaluating the performance of our joint semantic and geometric scene decomposition model (see Chapter 4). The dataset contains 715 images of urban and rural scenes assembled from a collection of existing datasets: LabelMe [Russell et al., 2008], 21-class MSRC [Criminisi, 2004], PASCAL VOC [Everingham et al., 2010], and Geometric Context (GC) [Hoiem et al., 2007a]. The construction of this dataset is the subject of the next section. To our knowledge, this is the first dataset to include both semantic and geometric labels.

- **MIT Street Scenes [Bileschi and Wolf, 2005]**. This dataset contains a very large collection of images of Boston street scenes. The images are annotated with polygonal regions corresponding to a small set of background categories as well as three foreground objects (cars, pedestrians and bicycles). Labeling is only approximate and in our experiments we relabeled the background regions using Amazon Mechanical Turk (see Section 3.2.1).

- **CCM (DS1) Dataset [Heitz et al., 2008b]**. This dataset is designed for evaluating holistic scene understanding algorithms and is composed of 422 images of outdoor scenes from various existing datasets (including 21-class MSRC [Criminisi, 2004], PASCAL VOC [Everingham et al., 2010] and LabelMe [Russell et al., 2008]). While the source of the images for this dataset overlaps with the Stanford Background Dataset, very few scenes appear in both datasets. Each image is annotated with object bounding boxes, pixel semantic classes, and high-level scene category (e.g., urban, rural, harbor). The dataset includes six object categories (*boat*, *car*, *cow*, *motorbike*, *person*, and *sheep*) and the same eight pixel-level semantic classes as the Stanford Background Dataset. We use this dataset in Chapter 5 to evaluate our object detection performance.

- **Make3d [Saxena et al., 2005, 2008]**. This dataset is composed of 534 images of outdoor scenes with corresponding depth maps. The depth maps were collected using a laser range finder with maximum range of 81 meters. One of the limitations of this dataset is the lack of foreground objects due to the relatively long time required for the laser range finder to scan the scene during data capture. We use this dataset in Chapter 6 for training and evaluating our semantically-informed depth perception algorithms.

3.2 Stanford Background Dataset

The Stanford Background Dataset is a new dataset introduced as part of this work for evaluating methods for geometric and semantic scene understanding. The dataset contains 715 images chosen from existing public datasets, namely, LabelMe [Russell et al., 2008], 21-class MSRC [Criminisi, 2004], PASCAL VOC [Everingham et al., 2010], and Geometric Context (GC) [Hoiem et al., 2007a]. Our selection criteria were for the images to be of outdoor scenes, have approximately 320×240 pixels, contain at least one foreground object, and have the horizon positioned within the image (it need not be visible).

Ground truth labels for training and testing were obtained using Amazon Mechanical Turk (AMT), as discussed in Section 3.2.1 below. The dataset provides eight semantic region labels: *sky*, *tree*, *road*, *grass*, *water*, *building*, *mountain*, and *foreground object*; and three geometric region labels: *sky*, *horizontal* (support), and *vertical*. In addition, the location of the horizon is also provided. The quality of our annotations (obtained using AMT) is extremely good and in many cases superior to those provided by the original datasets where available (see Figure 3.3 for a comparison with MSRC).

3.2.1 Amazon Mechanical Turk

Amazon Mechanical Turk (AMT) (`http://www.mturk.com/`) is an online marketplace where *requesters* can post simple jobs for non-expert *workers* to complete. Typical jobs—known as Human Intelligence Tasks (HITs)—involve completing online surveys, giving feedback on website designs, writing textual descriptions or recommendations for products, transcribing text from audio recordings, etc. Workers are paid between one cent and a few dollars per task.[1] Researchers in the machine learning community have discovered AMT as a cheap source of supervised training data [Sorokin and Forsyth, 2008, Snow et al., 2008]. For example, construction of the ImageNet [Deng et al., 2009] dataset which contains over nine million images organized into a conceptual-semantic hierarchy (i.e., noun hierarchy) was only possible thanks to the cheap online workforce and infrastructure provided by AMT.

In our work, we used AMT to acquire human-labeled ground truth region boundaries, semantic labels and geometric labels for our Stanford Background Dataset (first introduced in Gould et al. [2009b]), and to relabel some existing datasets such as the MIT Street Scenes [Bileschi and Wolf, 2005] dataset described above. We developed an Adobe Flash application which allowed users to "paint" over an image with a color indicating the category of the underlying pixel (or a unique color for each region in the image). For example, the users were asked to paint grass as green, water as blue, and buildings as red. A screenshot of our application is shown in Figure 3.1.

We configured our application for three different tasks per image:

- **Region labeling:** Here workers were instructed to label each region (i.e., object or distinct background area) in the image with a different color. Specific examples were shown of adjacent objects "painted" in different colors. The workers were asked to pay particular attention to accurate labeling of the boundary between regions.

- **Semantic labeling:** In this task, the workers were asked to label each pixel in the image with the color corresponding to one of eight semantic classes. Pixels not belonging to one of the classes could be left unlabeled (i.e., white). However, since our set of semantic classes covered most background regions found in outdoor scenes, and included a generic *foreground object* label, the vast majority of images were densely labeled.

- **Geometry labeling:** Similar to the semantic labeling task, in this task workers were asked to label each pixel in the image with a color corresponding to one of

[1]A rule-of-thumb is for workers to be paid approximately \$1.00 per hour of work.

three course geometric classes. Slanted surfaces were to be assigned the label (horizontal or vertical) closest to the true orientation.

The tasks were uploaded onto AMT with simple instructions to workers (see Figure 3.1) and examples of both good and bad quality annotations. Workers were paid \$0.10 per task per image. Thus, for under \$250 we were able to label our entire 715-image dataset.[2] Typical labels obtained from AMT are shown in Figure 3.2. As can be seen, the quality of these labels is very good. In particular, when compared to other datasets, our annotations provide a dense labeling of the image (i.e., every pixel is assigned a class label) and the region boundaries are very close to the true object boundaries. Figure 3.3 illustrates this on two example images from the 21-class MSRC dataset [Criminisi, 2004].

We also developed back-end verification tools which, allowed us to quickly verify the quality of the labels and approve or reject the work. Approximately 10% of the jobs were rejected due to poor quality.[3] Figure 3.4 shows examples of jobs that were rejected. In some cases jobs were rejected because of egregious errors (such as the first row in Figure 3.4), while others were due to the AMT worker misunderstanding the labeling instructions. In all cases, the rejected images were re-submitted to AMT and were usually labeled correctly the second time.

The back-end verification process, while streamlined, did require some manual effort on our part (albeit far less effort than had we chosen to label the images ourselves). Some researchers have suggested that AMT can also be used in the verification stage by having a separate set of jobs in which workers rate the quality of each labeling. By providing known good and bad examples, having trusted workers, or getting the same image rated multiple times, one could ensure the integrity of this verification stage, and thus completely automate the labeling process.

Since labeling of the location of the horizon required little effort, for the 715 images in our dataset, we elected to perform this annotation ourselves. However, this task can easily be converted into an AMT job for future labeling of larger scale datasets.

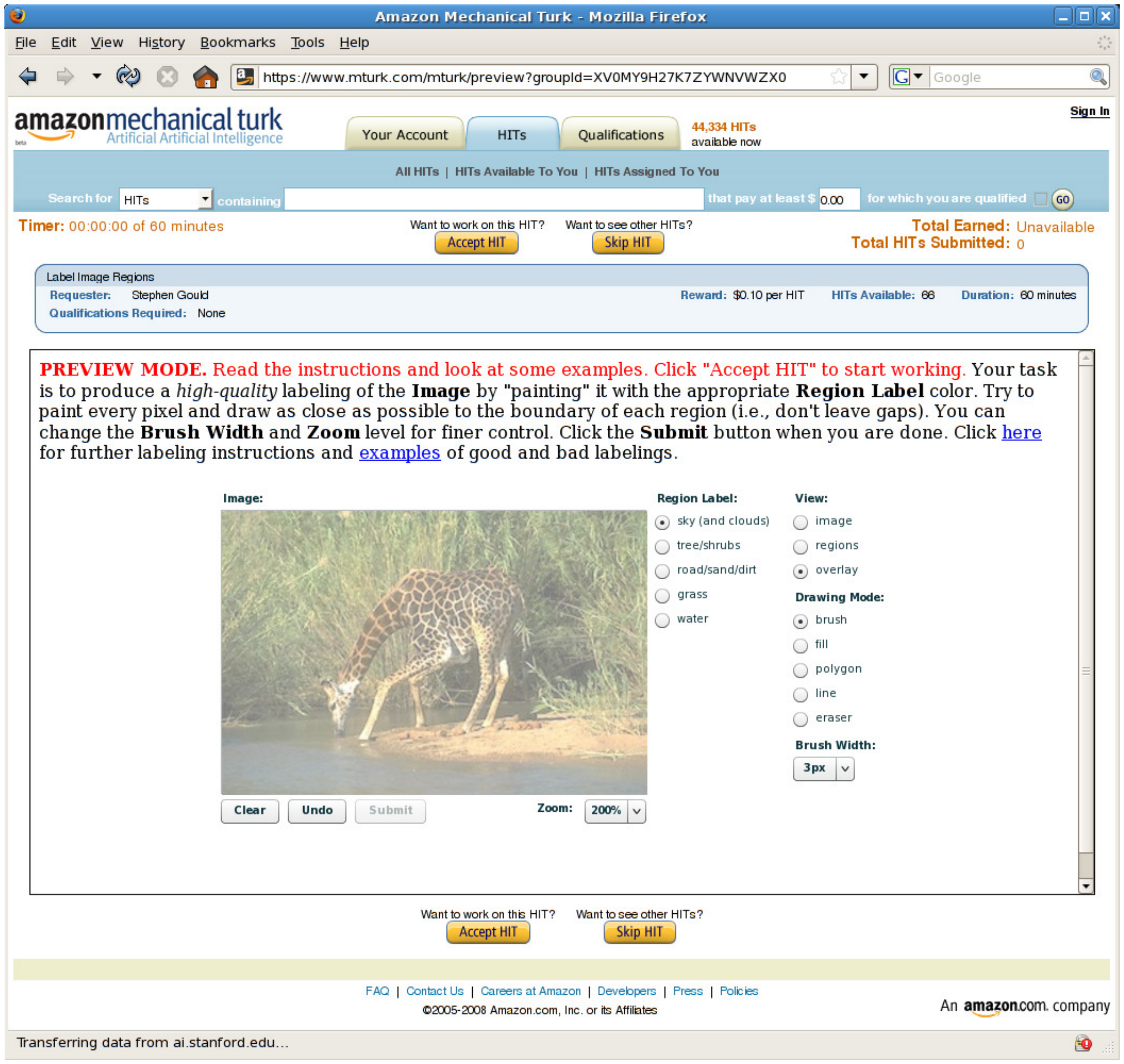
[2]We uploaded our images in batches as we were familiarizing ourselves with the AMT system and debugging our code. Typically, a batch of 100 images would get labeled overnight (i.e., in under 12 hours).

[3]Rejected jobs do not get paid.

3.3 Chapter Summary

Labeled training data is necessary for training the machine learning algorithms used by computer vision researchers, and having standard high-quality datasets is essential for evaluating and comparing the performance of these algorithms.

In this chapter we have provided a brief description of the standard datasets used in the computer vision community and, in particular, for evaluating the methods presented in this thesis. We discussed some of the problems with early datasets that are slowly being corrected in newer (and larger) datasets. To support some of the experiments that we wanted to run, we introduced a new dataset containing both semantic and geometric labels. The labeling of this dataset was performed using Amazon Mechanical Turk at a total cost of under $250.


Amazon Mechanical Turk - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://www.mturk.com/mturk/preview?groupId=XV0MY9H27K7ZYWNVWZX0
amazonmechanical turk Artificial Artificial Intelligence
Your Account
HITs
Qualifications
44,334 HITs available now
Sign In
All HITs | HITs Available To You | HITs Assigned To You
Search for HITs containing that pay at least $ 0.00 for which you are qualified GO
Want to work on this HIT? Accept HIT
Want to see other HITs? Skip HIT
Image:
Clear
Undo
Submit
FAQ | Contact Us | Careers at Amazon | Developers | Press | Policies
An amazon.com. company

Timer: 00:00:00 of 60 minutes

Total Earned: Unavailable
Total HITs Submitted: 0

Label Image Regions
Requester: Stephen Gould
Qualifications Required: None
Reward: $0.10 per HIT
HITs Available: 66
Duration: 60 minutes

PREVIEW MODE. Read the instructions and look at some examples. Click "Accept HIT" to start working. Your task is to produce a *high-quality* labeling of the **Image** by "painting" it with the appropriate **Region Label** color. Try to paint every pixel and draw as close as possible to the boundary of each region (i.e., don't leave gaps). You can change the **Brush Width** and **Zoom** level for finer control. Click the **Submit** button when you are done. Click here for further labeling instructions and examples of good and bad labelings.

Region Label: [✓ sky (and clouds)] tree/shrubs road/sand/dirt grass water

View: image regions [✓ overlay]

Drawing Mode: [✓ brush] fill polygon line eraser

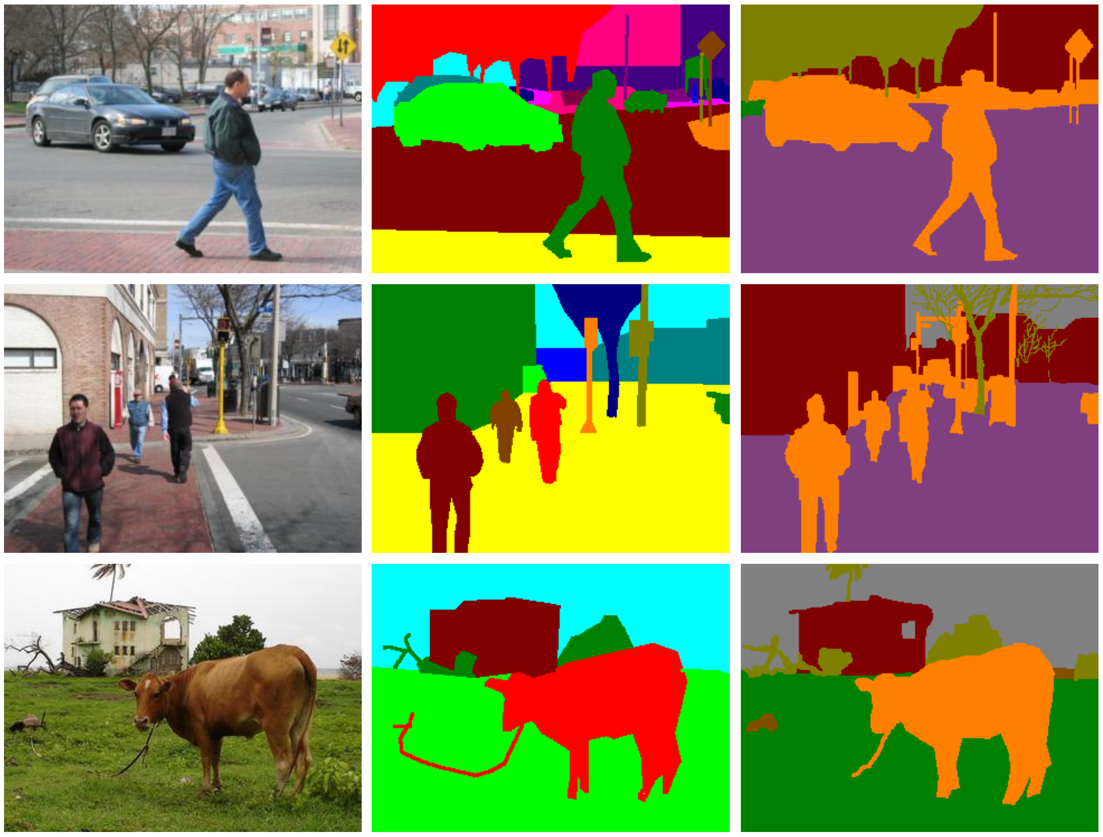
Brush Width: 3px

Zoom: 200%

©2005-2008 Amazon.com, Inc. or its Affiliates

Transferring data from ai.stanford.edu...

Figure 3.1: Screenshot of an Amazon Mechanical Turk HIT for labeling regions in an image. Our tool can be configured for labeling different tasks including regions (e.g., with multiple distinct object instances), semantic classes (by pixel), and geometry.



(a) Image (b) Regions (c) Semantics

⬜ sky 🟨 tree 🟪 road 🟩 grass 🟦 water 🟥 bldg 🟫 mntn 🟧 fg obj.

Figure 3.2: Typical label quality obtained from AMT jobs. Column (a) shows the image being labeled; column (b) shows the annotated regions (each color represents a unique region); and column (c) shows the pixel semantic classes (see semantic legend).

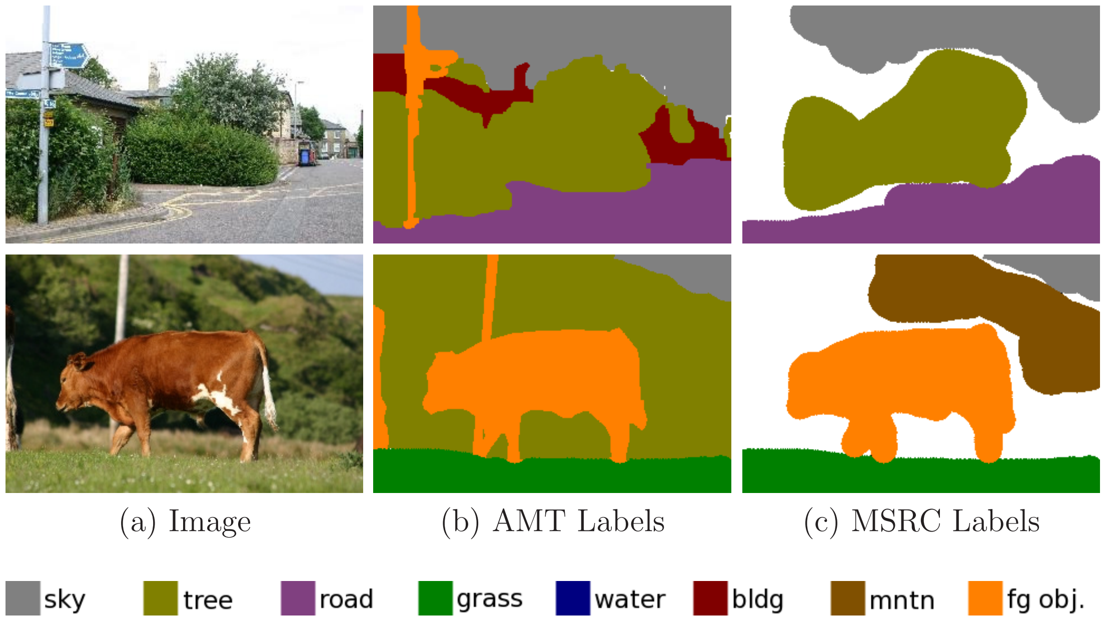


Figure 3.3: Comparison of the label quality obtained from our AMT jobs versus the 21-class MSRC [Criminisi, 2004] dataset. White indicates unlabeled ("void") pixels that are not considered during training or evaluation. The MRSC "cow" label (second row) has been mapped to "foreground" for comparison with our semantic classes.

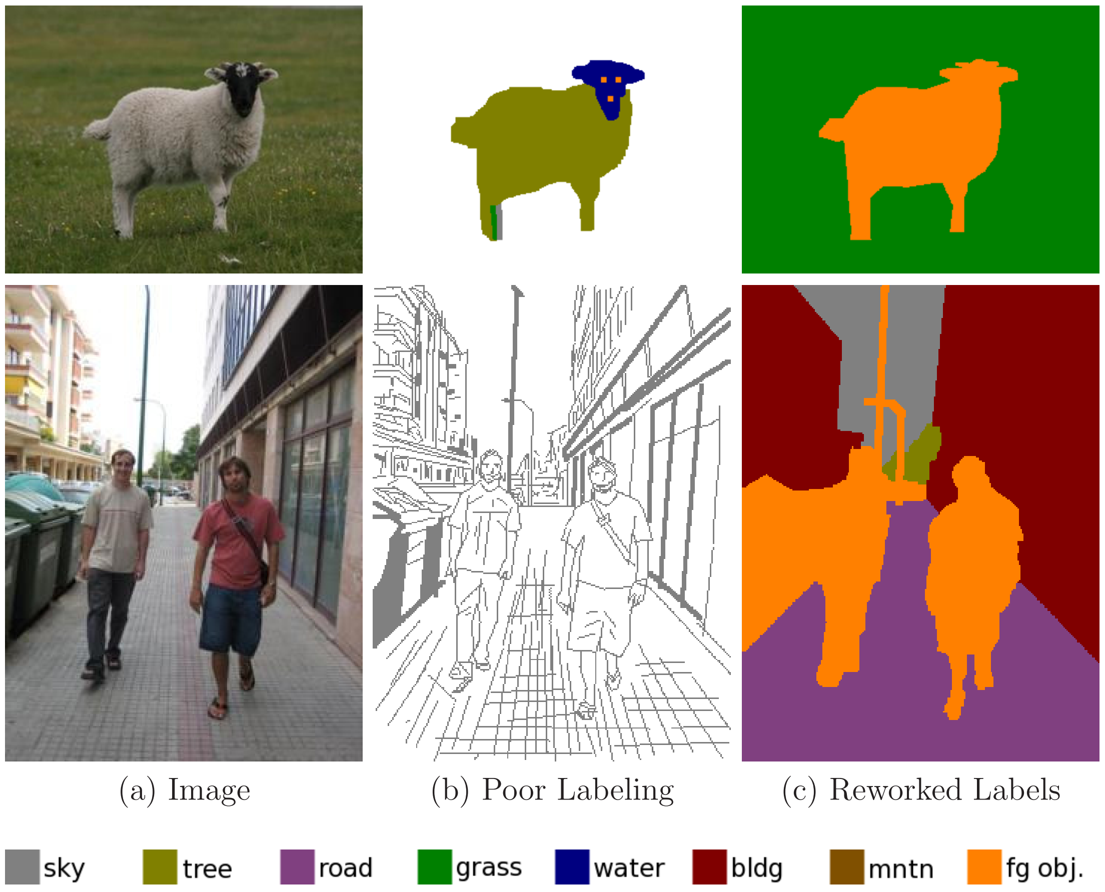


Figure 3.4: Two examples of poorly labeled images obtained from a semantic labeling job on Amazon Mechanical Turk (AMT). The first example (top row) shows correct region boundaries, but incorrect semantic labels. The second example (bottom row) shows a line drawing of the image, indicating a misunderstanding of the labeling instructions by the AMT worker. These images were resubmitted to AMT for subsequent annotation. In total, only about 10% of jobs required rework.

Chapter 4

Scene Decomposition: From Pixels to Regions

In this chapter, we introduce our scene decomposition model. We begin by describing the various entities of the scene that we will reason about and formalize them in terms of random variables in a probabilistic graphical model. We develop the model up to the stage of *scene decomposition*, that is, breaking the image into semantic and geometrically consistent regions, and leave the discussion of specific object identification and outlining until Chapter 5. Our aim, therefore, is to be able to decompose a scene into a number regions that are annotated with a class label from one of many background class types or single *foreground object* class that will later (in Chapter 5) be refined into specific object categories. In addition to annotating the regions with semantic class labels, we will also assign coarse geometric labels to the regions and predict the location of the horizon in the image.

This initial scene decomposition is an important step towards the goal of holistic scene understanding since such an analysis gives a high-level understanding of the overall structure of the scene, allowing us to derive a notion of relative object scale, height above ground, and placement relative to dominant semantic categories such as road, grass, water, buildings or sky. Unlike methods that deal only with multi-class semantic labeling (e.g., [Shotton et al., 2006]) or only with geometric reconstruction (e.g., [Hoiem et al., 2007a]), our approach reasons jointly about both aspects of the

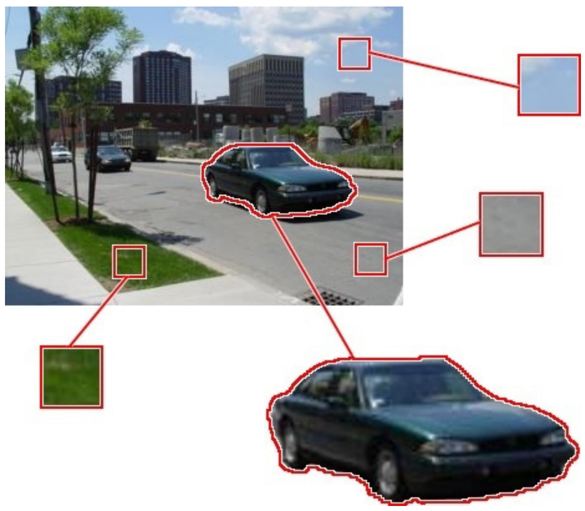


Figure 4.1: Example of a typical urban scene demonstrating that most background regions can be easily recognized from local color and texture cues whereas foreground regions require complex shape and appearance models.

scene, allowing us to avoid inconsistencies (such as vertical roads) and to utilize contextual relationships to reduce false positive objects (such as those not supported by a horizontal region).

Our decision to first concentrate on background semantic classes is not accidental. Consider the various regions present in the typical urban scene shown in Figure 4.1. Notice how the foreground classes ("things")—such as cars and trucks—require complex shape and appearance models, while background classes ("stuff")—such as sky, road and grass—can often be identified from simple texture and color cues. This observation is supported by our own experiments and results from many seminal works in multi-class image labeling. For example, on the 7-class Corel dataset considered by He et al. [2004] 93.7% of the background pixels are classified correctly compared to 82.1% for foreground. On the challenging 21-class MSRC dataset [Criminisi, 2004], the difference is even more stark (e.g., 94.5% versus 63.1% for Shotton et al. [2006] and 90.5% versus 68.3% for Ladicky et al. [2009]).[1] Our hope, then, is that relatively

[1]We computed these performance metrics by grouping the multiple classes considered by each of the works into background and foreground, and then averaging over the results provided in the respective publications.

simple models will allow us to characterize background and separate it from foreground regions, which we can then recognize using more sophisticated methods. As we will see, this is precisely the effect that results from running our decomposition model.

A key aspect of our approach is the use of large, dynamically-defined regions as the basic semantic unit. Most previous methods for doing this type of image decomposition use either individual pixels [Shotton et al., 2006] or predefined superpixels [Yang et al., 2007, Gould et al., 2008]. Here, a superpixel is defined to be a connected set of pixels having similar appearance. Each of these approaches has its trade-offs. The use of individual pixels makes it difficult to utilize more global cues, including both robust statistics about the appearance of larger regions, which can help average out the random variations of individual pixels, and relationships between regions, which are hard to “transmit” by using local interactions at the pixel level. The use of superpixels partially addresses some of these concerns, but as superpixels are constructed in advance using simple bottom-up procedures based on local appearance, their boundaries are often inconsistent with the true region boundaries, making an accurate decomposition of the image impossible. Our approach dynamically associates pixels to regions, allowing region boundaries to adjust so as to accurately capture the true object boundaries. Moreover, our regions are also much larger than superpixels, allowing us to derive global appearance properties for each region, including not only color and texture, but even larger properties such as its general shape, aspect ratio, and characteristics of its boundary.

Reasoning in our model requires that we infer both the pixel-to-region association and the semantic and geometric labels for the regions. We address this challenge using a hybrid inference approach. For the pixel-association task, we propose a novel multiple-segmentation approach, in which different precomputed over-segmentations are used to propose changes to the pixel-to-region associations. These proposed moves take large steps in the energy space and hence help avoid local minima; however, they are evaluated relative to our global energy function, ensuring that each step reduces the energy, and hence improves the decomposition of the scene at hand. The region-labeling task is addressed using global energy-minimization methods over the region space (conditioned on the current pixel-to-region assignments). This step is not too expensive, since the number of regions is significantly lower than the number of pixels. By performing the inference at this level, we also improve labeling accuracy because the adjacency structure between these larger regions allows us to directly exploit correlations between them (such as the fact that ground is below sky).

The parameters of our model are entirely learned from data. Recall that in our model, we are learning to label entire segments, allowing us to exploit global region-level characteristics. We obtain positive examples for region labels from a large training set, which we constructed using Amazon Mechanical Turk (AMT) as discussed in Chapter 3. Negative examples are a bit trickier to acquire, as there are exponentially many "non-regions," most of which are obviously bad choices. Training with all non-regions is clearly intractable, and training with a random subset would be misleadingly easy for the learning algorithm. We therefore propose a novel *closed-loop* training regime, where the algorithm runs inference on the training images given its current model parameters, and mines for mistakes. These mistakes are then used as negative examples for a subsequent parameter learning iteration. This focuses the learning process on improving cases that are likely to occur in practice.

We now turn to describing the various entities in our model. Inference and learning are described in Section 4.2.1 and Section 4.2.2, respectively. In Section 4.3 we apply our method to the challenging Stanford Background Dataset (see Chapter 3) and show that our approach produces multi-class segmentation and surface orientation results that outperform state-of-the-art methods. We conclude the chapter in Section 4.4 with a summary of our model and discussion of the conclusions that can be drawn from this work.

4.1 Scene Decomposition Model

Our goal is to decompose an image $\mathcal{I}$ into an unknown number (K) of geometrically and semantically consistent regions by iteratively optimizing an energy function that measures the quality of the solution at hand. Our model reasons about both pixels and regions. Each pixel in the image $p \in \mathcal{I}$ belongs to exactly one region, which is identified by the pixel's region-association variable $R_p \in \{1, \ldots, K\}$. Let the set of pixels in region r be denoted by $\mathcal{P}_r = \{p : R_p = r\}$. The size of the region (i.e., number of pixels) is simply $N_r = |\mathcal{P}_r| = \sum_p \mathbf{1}\{R_p = r\}$. In our notation we will always use subscripts p and q to denote pixels, and subscripts r and s to denote regions. Double indices are used for pairwise terms between adjacent entities in the scene (e.g., pq or rs).

Each pixel has a local appearance feature vector $\alpha_p \in \mathbb{R}^n$ (described in Section 4.1.2 below). Associated with each region are: a semantic class label S_r, currently grass, mountain, water, sky, road, tree, building and foreground; a geometric class label G_r, currently horizontal, vertical, and sky; and a region appearance A_r that summarizes the appearance of the region as a whole.

The final component in our model is the horizon. We assume that the image was taken by a camera with horizontal axis parallel to the ground. We therefore model the location of the horizon as the row in the image corresponding to the horizon $v^{\text{hz}} \in \{1, \ldots, \text{height}(\mathcal{I})\}$.

Given an image $\mathcal{I}$ and model parameters $\boldsymbol{\theta}$, our unified energy function scores the entire description of the scene: the pixel-to-region associations $\boldsymbol{R}$; the region semantic class labels $\boldsymbol{S}$, geometries $\boldsymbol{G}$, and appearances $\boldsymbol{A}$; and the location of the horizon v^{hz}:

$$\begin{aligned} E(\boldsymbol{R}, \boldsymbol{S}, \boldsymbol{G}, \boldsymbol{A}, v^{\text{hz}} \mid \mathcal{I}, \boldsymbol{\theta}) &= \psi^{\text{horizon}}(v^{\text{hz}}) + \sum_r \psi_r^{\text{region}}(S_r, G_r, v^{\text{hz}}; A_r, \mathcal{P}_r) \\ &+ \sum_{r,s} \psi_{rs}^{\text{pair}}(S_r, G_r, S_s, G_s; A_r, \mathcal{P}_r, A_s, \mathcal{P}_s) + \sum_{p,q} \psi_{pq}^{\text{boundary}}(R_p, R_q; \alpha_p, \alpha_q). \quad (4.1) \end{aligned}$$

Each of the terms in the energy function captures a preference for assignments to a subset of the variables in the model. Together they provide a global score for any given scene decomposition (i.e., complete assignment to all the variables). We now describe each of these terms.

4.1.1 Predicting Horizon Location

The ψ^{horizon} term in our energy function (Equation 4.1) captures the a priori estimate of the location of the horizon in the scene. In our model, it is implemented as a log-gaussian potential function

$$\psi^{\text{horizon}}(v^{\text{hz}}) = -\eta^{\text{horizon}} \log \mathcal{N}(v^{\text{hz}}; \mu, \sigma^2) \tag{4.2}$$

with parameters μ and σ learned from labeled training images. Note that the horizon variable v^{hz} also appears in each of the ψ^{region} terms so that during inference, its optimal value will be determined by taking into consideration the preference from both the ψ^{horizon} and ψ^{region} terms. The constant η^{horizon} in Equation 4.2 weights the contribution of this term relative to the other terms in the energy function.

It should be stressed that this term implements a simple prior on the location of the horizon. A number of techniques have been proposed in the literature [Quan and Mohr, 1989, Kosecka and Zhang, 2002, Hoiem et al., 2006] that attempt to estimate the location of the horizon using more sophisticated approaches such as looking for vanishing lines. We found that a reliable estimate of the horizon could be achieved on our dataset by our simple log-gaussian prior and the inclusion of the horizon variable in the region term. Thus, the additional complexity of these more sophisticated methods was not warranted.

4.1.2 Characterizing Individual Region Appearance

For each pixel p in the image, we construct a local appearance descriptor vector α_p comprised of raw image features and discriminatively learned boosted features. Our raw image features, which are computed in a small neighborhood of the pixel, are identical to the 17-dimensional color and texture features described by Shotton et al. [2006] (see Chapter 2). We augment these raw features with more processed summaries that represent the "match" between the pixel's local neighborhood and each of the region labels. Specifically, for each (individual) semantic and geometric label we learn a one-vs-all boosted classifier to predict the label given the raw image

Pixel Feature	Description
Filter Responses (17)	Color and texture filter responses described in Shotton et al. [2006]
Boosted Classifier Scores (11)	Log-odds ratio from one-vs-all boosted classifiers trained to predict individual pixel semantic and geometric class labels from the pixel's appearance and location, and the appearance of pixels in a small neighborhood around it.

Table 4.1: Local pixel appearance features, α_p.

features in a small neighborhood around the pixel.[2] We then append the score (log-odds ratio) from each boosted classifier to our pixel appearance feature vector α_p. Table 4.1 summarizes these features.

We infer the region appearance A_r by fitting a multi-variate Gaussian over the appearance of pixels within the r-th region. That is, we set $A_r = \left(\mu_r^A, \Sigma_r^A\right)$ where $\mu_r^A \in \mathbb{R}^n$ and $\Sigma_r^A \in \mathbb{R}^{n \times n}$ are the maximum-likelihood Gaussian mean and covariance parameters, respectively:

$$\mu_r^A = \frac{1}{N_r} \sum_{p \in \mathcal{P}_r} \alpha_p \tag{4.3}$$

$$\Sigma_r^A = \frac{1}{N_r} \sum_{p \in \mathcal{P}_r} \left(\alpha_p - \mu_r^A\right) \left(\alpha_p - \mu_r^A\right)^T \tag{4.4}$$

These summary statistics give us a more robust estimate of the appearance of the region than would be obtained by considering only small neighborhoods of the individual pixels.

[2]In our experiments, we append to the pixel's 17 "texton" features, the average and variance for each feature over a 5×5-pixel window in 9 grid locations around the pixel and the image row to give a total of 324 features. We use the GentleBoost algorithm [Friedman et al., 1998] with 2-split decision stumps and train for 500 rounds. Our results appeared robust to the choice of parameters.

4.1.3 Individual Region Potentials

To define the potentials that help infer the label of individual regions, we extract features $\phi_r(A_r, \mathcal{P}_r) \in \mathbb{R}^n$ describing the region appearance and basic shape. These features are summarized in Table 4.2. Our appearance features include the mean and variance over the appearance of the pixels in the region as described above, and the log-determinant of the covariance matrix over pixel appearances (i.e., $\log |\Sigma_r^A|$) to capture total variation in pixel appearance. We also include the average contrast at the region boundary and region interior which gives a sense of the region's internal texture versus its surrounding. In addition to relating to semantic class—grass is green; sky is blue—these appearance features provide a measure for the quality of a region since well-formed regions will tend to have strong boundary contrast and (depending on the class) little variation of interior appearance.

We also want to capture more global characteristics of our larger regions. For example, we would like to capture the fact that buildings tend to be vertical with many straight lines, trees tend to be green and textured, and grass tends to be green and horizontal. Thus, we incorporate shape features that include normalized region area, perimeter length, first- and second-order shape moments, and residual to a robust line fit along the top and bottom boundary of the region.[3] The latter features capture the fact that buildings tend to have straight boundaries while trees tend to be rough.

We also include the horizon variable in the region-specific potential, allowing us to include features that measure the ratio of pixels in the region above and below the horizon. These features give us a sense of the scale of the object and its global position in the scene. For example, buildings are tall and tend to have more mass above the horizon than below it; foreground objects are often close and will have most

[3]Specifically, let $\mathcal{P}_r$ be the set of pixels in the region and let $\mathcal{B} = \{(u_i, v_i)\} \subseteq \mathcal{P}_r$ be the set of pixels along its perimeter (boundary). Then the normalized region area is $\frac{|\mathcal{P}_r|}{|\mathcal{I}|}$; the length of the perimeter is simply $|\mathcal{B}|$; the first- and second-order shape moments are $\mu_x = \frac{1}{|\mathcal{P}_r|} \sum_{(u,v) \in \mathcal{P}_r} u$ and $\sigma_x^2 = \frac{1}{|\mathcal{P}_r|} \sum_{(u,v) \in \mathcal{P}_r} (u - \mu_x)^2$ and similarly for y, and $\sigma_{xy}^2 = \frac{1}{|\mathcal{P}_r|} \sum_{(u,v) \in \mathcal{P}_r} (u - \mu_x)(v - \mu_y)$; and, finally, the residual along the top and bottom boundary of the region is computed by linear least-squares regression to the subset of pixels in $\mathcal{B}$ that are the topmost and bottommost in their columns, respectively.

Region Feature	Description
Appearance (57)	Mean and standard deviation of the 17-dimensional filter responses and boosted classifier score for each pixel in the region, and log-determinant of the appearance covariance matrix Σ_r^A.
Boundary Contrast (4)	Measure of the contrast along the boundary of the region relative to its interior. They include the mean pixelwise contrast along the boundary; the average boundary contrast divided by the square-root of the region's area; the average internal contrast; and the ratio of boundary contrast to total (boundary plus internal) contrast.
Shape (19)	Basic region shape such as size and extent (relative to image dimensions), perimeter length, residual to lines fitted along the top and bottom boundary (as a measure of "straightness"), and first- and second-order shape moments. They also include binary features indicating whether the region has been truncated by the edge, top, or bottom of the image.
Geometry (9)	These features capture the relative location of the region with respect to the horizon v^{hz} and include the percentage of pixels in the region above and below the horizon; the spread of pixels about the horizon (i.e., mean distance above and below the horizon).

Table 4.2: Summary of the features $\phi_r(A_r, \mathcal{P}_r)$ used to describe regions in our model.

of their mass below the horizon. Conversely, these potentials also allow us to capture the strong positional correlation between the horizon and semantic classes such as sky or ground, allowing us to use the same potential to place the horizon within the image.

To put all of these features together, we learn a multi-class logistic classifier over the joint assignment to semantic and geometric class labels $S_r \times G_r$ with a quadratic kernel over ϕ_r (see Section 4.2.2). The score for any assignment to the region variables is then

$$\psi_r^{\text{region}}(S_r, G_r, v^{\text{hz}}; A_r, \mathcal{P}_r) = -\eta^{\text{region}} N_r \log \sigma\left(S_r \times G_r \mid \phi_r(A_r, \mathcal{P}_r), v^{\text{hz}}\right), \tag{4.5}$$

where η^{region} weights the importance of the region energy term relative to the other terms in the energy function and $\sigma(\cdot)$ is the multi-class logistic function with learned parameters. We scale the potential by the region size N_r so that our score gives more weight to larger regions and is independent of the number of regions in the image.

4.1.4 Inter-Region Potentials

Our model contains two types of inter-region potentials. The first of these terms is $\psi_{pq}^{\text{boundary}}$, which is a standard contrast-dependent pairwise boundary potential [Shotton et al., 2006]. Concretely, for two adjacent pixels p and q, we define

$$\psi_{pq}^{\text{boundary}}(R_p, R_q; \alpha_p, \alpha_q) = \begin{cases} \eta^{\text{bndry}} \exp\{-\frac{1}{2}\beta^{-1}\|\alpha_p - \alpha_q\|^2\}, & \text{if } R_p \neq R_q \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

where β is an image-specific normalization constant set to the average contrast between all adjacent pixels in the image. Specifically, we have

$$\beta = \frac{1}{|\mathcal{E}|} \sum_{(p,q)\in\mathcal{E}} \|\alpha_p - \alpha_q\|^2 \tag{4.7}$$

where the summation is over all pairs of adjacent pixels in the image denoted by the set $\mathcal{E}$. In practice, we restrict α_p and α_q to the first three color components (i.e., CIELab color coordinates) of the full pixel appearance feature vector. This term penalizes adjacent regions that do not have an edge between them; it has the effect of trying to merge adjacent regions that are not clearly demarcated. We note that, since the penalty is accumulated over pairs of adjacent pixels, the region-level penalty is proportional to the pixel-length of the boundary between the regions.

Our second inter-region potential, ψ^{pair}, models the affinity of two semantic or geometric classes to appear adjacent to each other (as separate regions). Similar to the within-region potentials, we extract features $\phi_{rs}(A_r, \mathcal{P}_r, A_s, \mathcal{P}_s) \in \mathbb{R}^m$ for every pair of adjacent regions r and s. We then learn independent multi-class logistic classifiers for $S_r \times S_s$ and $G_r \times G_s$ given these features. Note that these pairwise potentials are asymmetric (e.g., exchanging S_r and S_s gives a different preference since, as we shall see, the features include orientation information). The features ϕ_{rs} are intended to capture contextual properties between regions, for example, the boundary between building and sky tends to be straight and building is more likely to appear above a foreground object than below it. To capture these properties, our features include the difference between centroids of the two regions, the proportion of pixels along the boundary in which region r is above/below region s, the length of the boundary, the residual after fitting a straight line to the boundary pixels, and the orientation of this line.[4] In addition to these layout-based features, we include appearance difference between the regions normalized by the total appearance variance within each region. This captures signals such as foreground objects tend to contrast highly with other regions, whereas background regions are more similar in appearance, such as adjacent buildings in a city. The pairwise region features ϕ_{rs} are summarized in Table 4.3.

We normalize each pairwise potential by the sum of the number of pixels in each region divided by the number of neighbors for the region: $\left(\frac{N_r}{|\mathbf{nbr}(r)|} + \frac{N_s}{|\mathbf{nbr}(s)|}\right)$. This makes the total influence on a region independent of its number of neighbors while still giving larger regions more weight. The final form of our second inter-region

[4]As for the region boundary features, the pairwise boundary terms are computed using a linear least-squares fit to the boundary pixels. Specifically, let $\mathcal{B} = \{(u_i, v_i)\}$ be the set of pixels along the boundary between regions r and s. Then the length of the boundary is simply the number of pixels $|\mathcal{B}|$; the residual is computed as $\min_{\boldsymbol{\theta}} \sum_{(u_i,v_i)\in\mathcal{B}} \|v_i - \theta_1 u_i + \theta_0\|$ (i.e., performing a linear least-squares fit); and the orientation of this line is taken as θ_1.

potential is then

$$\psi^{\text{pair}}_{rs}(S_r, G_r, S_s, G_s; A_r, \mathcal{P}_r, A_s, \mathcal{P}_s) = -\eta^{\text{pair}} \left(\tfrac{N_r}{|\mathbf{nbr}(r)|} + \tfrac{N_s}{|\mathbf{nbr}(s)|} \right) \cdot \\ \Big(\log \sigma \left(S_r \times S_s \mid \phi_{rs} \right) + \log \sigma \left(G_r \times G_s \mid \phi_{rs} \right) \Big), \quad (4.8)$$

where, as above, η^{pair} weights the pairwise region potential relative to the other terms in the energy function, and $\sigma(\cdot)$ is the multi-class logistic function (with quadratic kernel over the features ϕ_{rs} as was done for the individual region potentials in the previous section).

Pairwise Feature	Description
Centroid (2)	Difference between the centroids of regions r and s (in normalized image coordinates).
Relative Position (2)	Percentage of pixels along the boundary between regions r and s for which (i) region r is above region s, and (ii) region r is below region s.
Boundary (3)	Number of pixels along the boundary between regions r and s, the residual after fitting a straight line to the boundary pixels, and the orientation of this line.
Appearance (28)	The difference between the average appearance features of regions r and s normalized by the standard deviation of the feature, i.e., $\mathbf{diag}\left(\sigma^A_r + \sigma^A_s\right)^{-1} (\mu^A_r - \mu^A_s)$ where σ^A_r is the square-root of the diagonal of Σ^A_r.

Table 4.3: Summary of the features $\phi_{rs}(A_r, \mathcal{P}_r, A_s, \mathcal{P}_s)$ used to describe the pairwise relationship between two adjacent regions in our model.

4.2 Inference and Learning

4.2.1 Inference Algorithm

Exact inference in our decomposition model is clearly intractable. We therefore adopt a two-stage hill climbing approach to approximately minimize the energy defined by Equation 4.1. In the first stage, we modify the pixel-to-region association variables by allowing a set of pixels to change the region to which they are assigned. Given the new pixel assignments, we then jointly optimize the region and horizon variables in the second stage. The global energy of the resulting configuration is then evaluated, and the move is accepted only if this energy decreases, ensuring that our inference is continuously improving a coherent global objective.

The proposal moves for region associations are drawn from a precomputed, image-specific dictionary of image segments Ω, an example of which is shown in Figure 4.2. We build the segments by computing a number of different over-segmentations of the image. Here, we use the mean-shift algorithm [Comaniciu and Meer, 2002], but could equally have used other approaches for generating over-segmentations (e.g., graph-based clustering [Felzenszwalb and Huttenlocher, 2004] or normalized-cuts [Shi and Malik, 2000]).[5] We generate different over-segmentations by varying the spatial and range bandwidth parameters and add all generated segments to the dictionary. This produces a rich set of fine and coarse granularity proposal moves. We sort the moves in the dictionary by the entropy of pixel appearance within each segment so that more uniform segments are proposed first. In addition to moves proposed by the dictionary, we also allow moves in which two adjacent regions are merged together. The set of allowed pixel-to-region association proposal moves is thus: (i) pick a segment $\omega \in \Omega$ and assign all R_p for $p \in \omega$ to a new region; (ii) pick a segment $\omega \in \Omega$ and assign all R_p for $p \in \omega$ to one of the regions in its neighborhood; or (iii) pick two neighboring regions r and s and merge them, that is, $\forall R_p = s$ set $R_p = r$.

[5]Publicly available code exists for many of the over-segmentation algorithms. We used an implementation of the mean-shift algorithm that can be downloaded from:
`http://www.caip.rutgers.edu/riul/research/~code/EDISON/index.html`

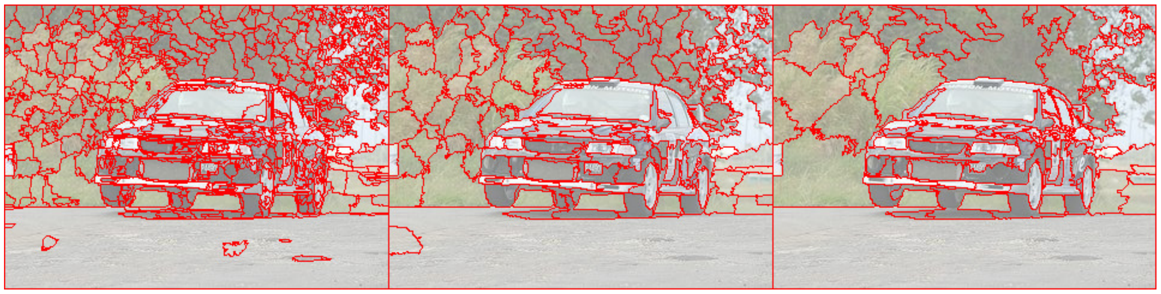


Figure 4.2: Example over-segmentation dictionary, Ω, for a given image generated by running mean-shift [Comaniciu and Meer, 2002] with three different parameter settings, $(h_r, h_s) \in \{(6, 7), (6, 11), (6, 19)\}$. The figure shows the boundary of each segment overlayed on top of the original image. In our inference algorithm, each segment $\omega \in \Omega$ defines a set of pixels-to-region association variables $\{R_p : p \in \omega\}$ that may be re-assigned to another region.

We note that, in our current implementation, we do not allow regions to be disconnected, although such situations might occur in practice (for example, a bus whose middle is occluded by a foreground tree). This is ensured during the proposal move by reassigning region identifiers if the move results in a disconnected component. Since checking for connected components has to be done after every proposal move, we have developed fast techniques for performing this step (see Appendix A.2).

Our overall inference procedure is summarized in Algorithm 4.1. Briefly, we initialize our pixel-to-region association variables $\boldsymbol{R}$ using one of the over-segmentations used to produce our dictionary, and greedily infer the remaining variables in the model ($\boldsymbol{S}$, $\boldsymbol{G}$ and v^{hz}). We then cycle over proposal moves: Given our current associations $\boldsymbol{R}$, we select a proposal move and reassign pixels to form new regions. Next, we update the appearance model A_r and features of any region that was affected by the move. We maintain sufficient statistics over pixel appearance, making this step very fast (see Appendix A.1). Keeping the pixel-to-region association variables and horizon fixed, we run max-product belief propagation on the region class and geometry variables. We then update the horizon v^{hz} using a greedy minimization step conditioned on the remaining variables.[6] The new configuration is evaluated relative to

[6]We experimented with including v^{hz} in the belief propagation inference but found that it changed very little from one iteration to the next and was therefore more efficient to infer conditionally once the other variables were assigned.

Algorithm 4.1: Scene Decomposition Inference.

```
1  Input  : $\mathcal{I}$                          // image
2           $\boldsymbol{\theta}$                  // model parameters
3  Output: $\boldsymbol{R}, \boldsymbol{S}, \boldsymbol{G}$, and $v^{\text{hz}}$        // scene decomposition
4  generate over-segmentation dictionary $\Omega$
5  initialize $R_p$ using one of the over-segmentations
6  infer initial $S_r$, $G_r$ and $v^{\text{hz}}$ variables
7  compute energy of initial assignment and store in $E^{\text{min}}$
8  repeat
9      foreach proposal move $\omega$ do
10         move pixels $\{R_p : p \in \omega\} \leftarrow r$
11         update region appearance $\boldsymbol{A}$ and features
12         run inference over region variables $(\boldsymbol{S}, \boldsymbol{G})$ and horizon $v^{\text{hz}}$
13         compute total energy $E$ of new assignments
14         if $(E < E^{min})$ then
15             accept move and set $E^{\text{min}} = E$
16         else
17             reject move
18         end
19     end foreach
20 until convergence
21 return $\boldsymbol{R}, \boldsymbol{S}, \boldsymbol{G}$, and $v^{\text{hz}}$
```

our global energy function, and kept if it provides an improvement. The algorithm iterates until convergence, i.e., none of the proposal moves result in a lower energy assignment. In our experiments (Section 4.3) inference took between 30 seconds and 10 minutes to converge depending on image complexity (i.e., number of segments in the dictionary Ω).

4.2.2 Learning Algorithm

We train our model using a labeled dataset where each image is segmented into regions that are semantically and geometrically coherent. Thus, our ground truth specifies both the region association for each pixel, R_p, and the semantic and geometric labels for each region, S_r and G_r.

We learn each term ψ^{horizon}, ψ^{region} and ψ^{pair} in our energy function separately, using our labeled training data. Learning is done using a maximum likelihood objective with small ℓ_2-regularization on the parameters (see Chapter 2 for details). We then cross-validate the weights between the terms to maximize our pixelwise labeling accuracy on a subset of our training data. Since only the relative weighting between terms matter, we fixed η^{region} to one.

For the horizon singleton term, we learn a Gaussian over the location of the horizon in training images and set ψ^{horizon} to be the log-likelihood of v^{hz} given this model. We normalize v^{hz} by the image height to make this model resolution invariant. Our learned Gaussian has a mean of approximately 0.5 and standard deviation of 0.15 (varying slightly across experiment folds). This suggests that the horizon in our Stanford Background Dataset is quite well spread around the center of the image.

The within-region term, ψ^{region}, and the between-region term, ψ^{pair}, are learned using multi-class logistic regression. However, the training of the within-region term involves an important subtlety. One of the main roles of this term is to help recognize when a given collection of pixels is actually a coherent region—one corresponding to a single semantic class and a single geometry. Although all of the regions in our training set are coherent, many of the moves proposed during the course of inference are not. For example, our algorithm may propose a move that merges together pixels containing (horizontal) grass and pixels containing (vertical) trees. We want to train our classifier to recognize invalid moves and penalize them. To penalize such moves, we train our multi-class logistic regression classifier with an additional "invalid" label. This label cannot be assigned to a candidate region during inference, and so if the proposed region r appears incoherent, the "invalid" label will get high probability, reducing the probability for all (valid) labels in $S_r \times G_r$. This induces a high energy for the new proposed assignment, making the move likely to be rejected.

To train a discriminative classifier that distinguishes between coherent and incoherent regions, we need to provide it with negative (incoherent) training instances. Here, we cannot simply collect arbitrary subsets of adjacent pixels that do not correspond to coherent regions: Most arbitrary subsets of pixels are easily seen to be incoherent, so that a discriminative model trained with such subsets as negative examples is unlikely to learn a meaningful decision boundary. Therefore, we use a novel "closed-loop" learning procedure, where the algorithm trains on its own mistakes. We begin by training our classifier where the negative examples are defined by merging pairs of adjacent ground truth regions (which are not consistent with each other). We then perform inference (on our training set) using this model. During each proposal move we compare the outcome of inference against the ground truth annotations. We append to our training set moves that were incorrectly accepted or rejected, or moves that were accepted (resulted in lower energy) but produced an incorrect labeling of the region variables. Here, to determine whether the algorithm has made a mistake, we measure how consistent the regions are with respect to the ground annotations. Specifically, we examine all regions affected by the move, and deem that a move should be rejected if more than 25% of the pixel in a region do not overlap with the same ground truth region (as the majority of the pixels). We deem that a move should be accepted if it merges two regions that overlap with the same ground truth region. For determining incorrect labeling of region variables, we take the true label for a region to be the maximum occurring ground truth label over all pixels within the region. We deem a region to be incorrectly labeled if the inferred label does not match this label (calculated from ground truth). In this way, we can target the training of our decision boundary on the more troublesome examples.

4.3 Experiments

We conduct experiments on the 715-image Stanford Background Dataset described in Chapter 3. We perform 5-fold cross-validation with the dataset randomly split into 572 training images and 143 test images for each fold. For the semantic class we used labels: *sky*, *tree*, *road*, *grass*, *water*, *building*, *mountain*, and *foreground object*; and for the geometry we used: *sky*, *horizontal* (support), and *vertical*.

We also compare a restricted version of our method on other datasets and compare to published state-of-the-art results.

	CLASS		GEOMETRY	
Pixel CRF (baseline)	74.3	(0.80)	89.1	(0.73)
Region-based energy	**76.4**	(1.22)	**91.0**	(0.56)

Table 4.4: Multi-class image segmentation and surface orientation (geometry) accuracy on the Stanford Background Dataset. Standard deviation (over five folds) shown in parentheses.

4.3.1 Baselines

To validate our method and provide strong baselines for comparison, we performed experiments on independent multi-class image segmentation and geometry prediction using standard pixelwise conditional Markov random field (CRF) models. Here, the pixel class S_p (or surface geometry G_p) is predicted separately for each pixel $p \in \mathcal{I}$ given the pixel's appearance α_p (see Section 4.1.2). A contrast-dependent pairwise smoothness term (similar to the ψ^{boundary} potential in our model) is added to encourage adjacent pixels to take the same value. The baseline models have the form

$$E(\boldsymbol{S} \mid \mathcal{I}) = \sum_p \psi_p(S_p; \alpha_p) + \eta^{\text{contrast}} \sum_{p,q} \psi_{pq}(S_p, S_q; \alpha_p, \alpha_q) \tag{4.9}$$

and similarly for $E(\boldsymbol{G} \mid \mathcal{I})$. In this model, each pixel can be thought of as belonging to its own region. The parameters are learned as described above with ψ_p a multi-class logistic over boosted appearance features and ψ_{pq} the boundary penalty. The baseline results on the Stanford Background Dataset are shown in Table 4.4.

4.3.2 Region-based Approach

Multi-class image segmentation and surface orientation results from our region-based approach are shown below the baseline results in Table 4.4. Our improvement of 2.1% over baseline for multi-class segmentation and 1.9% for geometry is significant. In particular, we observed an improvement over the baseline in each of our five folds. Our horizon prediction was within an average of 6.9% (relative to image height) of the true horizon.

In order to evaluate the quality of regions found by our scene decomposition model, we computed the overlap score between our boundary predictions and the hand-annotated region boundaries. To make this metric robust, we first dilate both the predicted and ground truth boundaries by five pixels. We then compute the overlap score by dividing the total number of overlapping pixels by half the total number of (dilated) boundary pixels (ground truth and predicted). A score of one indicates perfect overlap. We averaged 0.499 on this metric across the five folds, indicating that on average we get about half of the semantic boundaries correct. For comparison, using the between-class boundaries defined by the baseline semantic class predictions gives a boundary overlap score of 0.454.

The boundary score result reflects our algorithm's tendency to break regions into multiple segments. For example, it tends to leave windows separated from buildings and people's torsos separated from their legs (as can be seen in Figure 4.3). This is not surprising given the strong appearance difference between these different parts. We will see in the next chapter that for foreground objects these types of errors can be corrected the introduction of the object entities, comprised of multiple regions, into our model.

Figure 4.3 shows four example scene decompositions illustrating typical results from our method. Further good examples and some bad examples are shown in Figures 4.4 and 4.5, respectively. Notice the high quality of the class and geometry predictions particularly at the boundary between classes, and how our algorithm deals well with both near and far objects. There are still many mistakes that we would like to address in future work. For example, our algorithm is often confused by strong shadows and reflections in water as can be seen in some of the examples in Figure 4.5. We hope that with stronger geometric reasoning we can avoid this problem. Also, without knowledge of foreground subclasses (which we will address in the next chapter), our algorithm sometimes merges a person with a building or confuses boat masts with buildings.

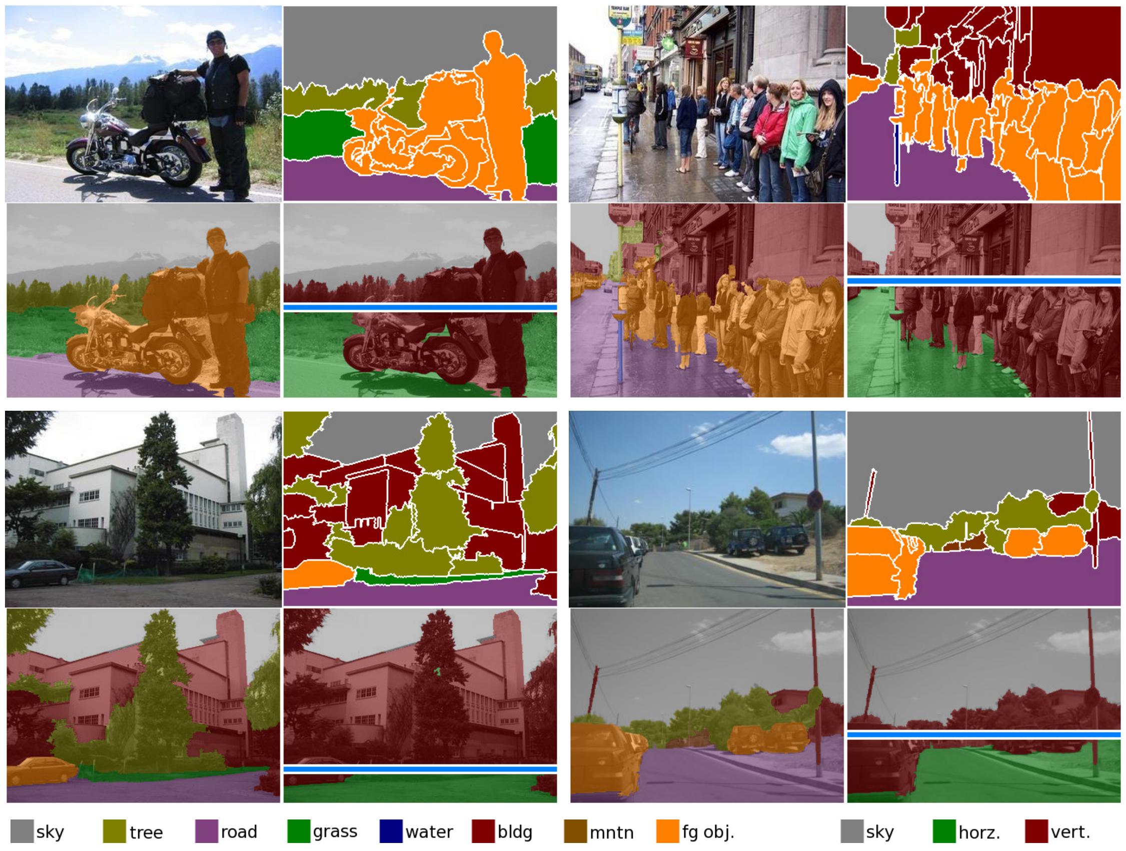


Figure 4.3: Examples of typical scene decompositions produced by our method. Shown for each image are regions colored by semantic class (top right), semantic class overlay (bottom left), and surface geometry with horizon (bottom right).

4.3.3 Comparison with Other Methods

We also compared our region-based approach with state-of-the-art techniques on the 21-class MSRC [Criminisi, 2004] and 3-class Geometric Context (GC) [Hoiem et al., 2007a] datasets. To make our results directly comparable with published works, we removed components from our model not available in the ground truth labels for the respective datasets. That is, for MSRC we only use semantic class labels and for GC we only use the (main) geometry labels. Neither model used horizon information. Despite these restrictions, our region-based energy approach is still competitive with state-of-the-art. Results are shown in Tables 4.5 and 4.6.

METHOD	ACCURACY
TextonBoost [Shotton et al., 2006]	72.2
Yang et al. [2007]	75.1
Gould et al. [2008]	**76.5**
Pixel CRF	75.3
Region model	76.4

Table 4.5: Comparison with state-of-the-art 21-class MSRC results against our restricted model. Table shows mean pixel accuracy over all semantic classes.

METHOD	ACCURACY
Hoiem et al. [2005a]	86.0
Hoiem et al. [2007a] (pixel)	82.1
Hoiem et al. [2007a] (superpixel)	**88.1**
Pixel CRF	86.5
Region model	86.9

Table 4.6: Comparison with state-of-the-art Geometric Context results against our restricted model. Table shows mean pixel accuracy over major geometric classes.

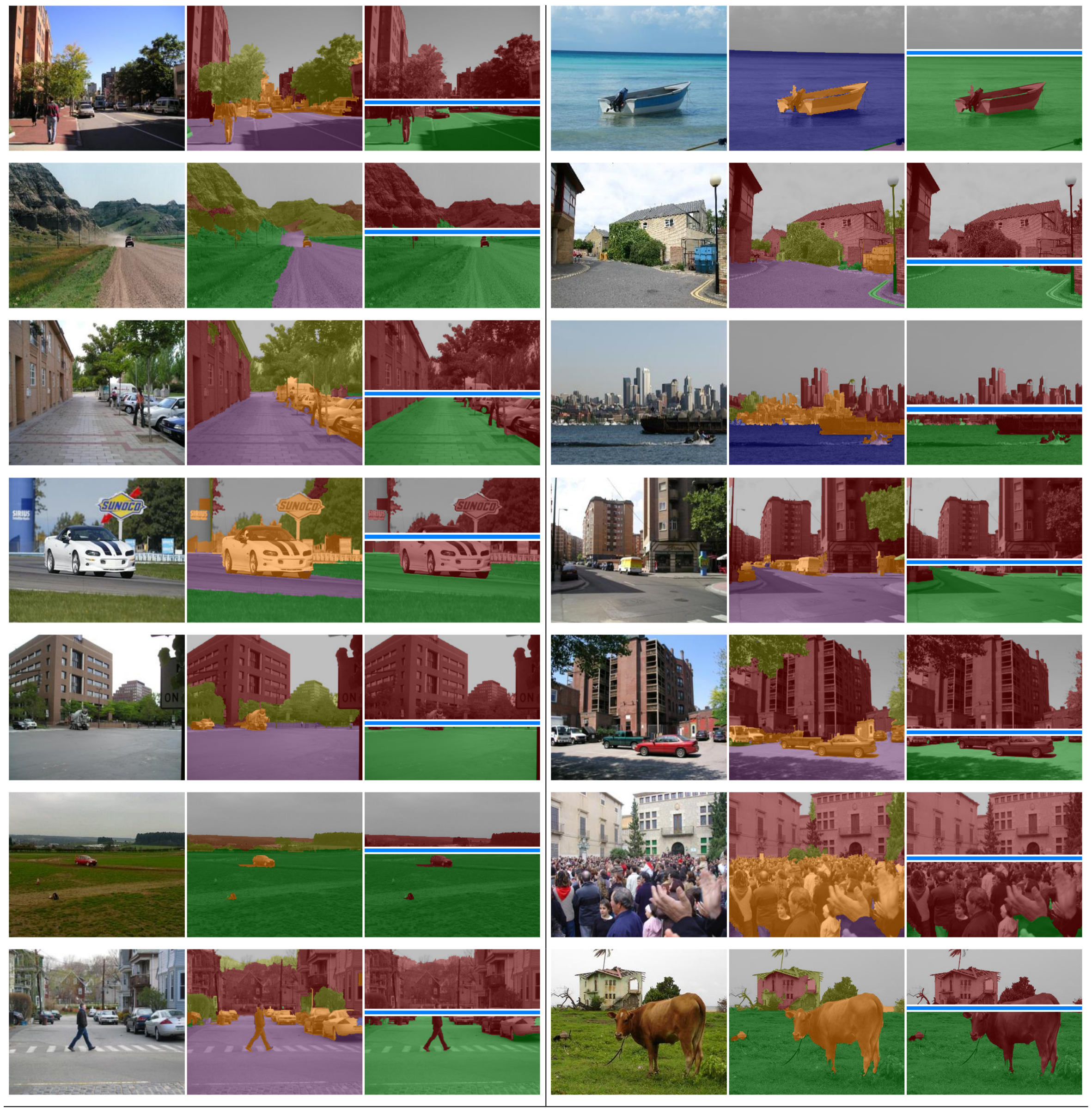
In work appearing simultaneously with ours, Ladicky et al. [2009] achieve a pixelwise score of 86.0% accuracy on the 21-class MSRC dataset. This impressive result was achieved, in part, by the introduction of very strong pixel-level features—the same used by many object detection algorithms—which better discriminate between the many foreground object classes in the MSRC dataset. We would also expect to see similar gains (for pixelwise accuracy on the MSRC dataset) by using these features. However, recall that the goal of this chapter was to decompose the scene into regions, with generic foreground class, and leave refinement of foreground objects to the next chapter.

4.4 Chapter Summary

The contribution of this chapter has been the development of a model for decomposing a scene into semantic and geometrically coherent regions. Our method combines reasoning over both pixels and regions through a unified energy function. In contrast to other multi-class image labeling approaches, our method incorporates regions as variables to be inferred along with the class labels. However, this makes inference in our model significantly more challenging. We proposed a data-driven energy minimization technique that effectively deals with this challenge. Our technique iteratively improves the current scene decomposition by proposing changes to the regions and re-inferring the class labels and horizon location. Currently, convergence is slow, taking up to 10 minutes per image. However, there may be a number of ways in which this can be improved, such as more intelligent scheduling of the proposal moves or making use of parallel hardware to produce multiple "good" partial solutions and then merging them. A similar approach, known as *fusion moves*, has successfully applied to energy minimization in low-level vision problems (e.g., stereo reconstruction) [Lempitsky et al., 2009b].

We showed how the parameters of our unified energy function could be learned from data using a piecewise learning approach. The learning algorithm employed a novel closed-loop procedure to mine for difficult negative examples when learning the parameters of the region term ψ^{region}. This procedure is reminiscent of the techniques used in the computer vision community to train object detectors [Torralba et al., 2004a, Dalal and Triggs, 2005, Felzenszwalb et al., 2010] and constraint generation approaches used in the machine learning community when learning max-margin classifiers for structured prediction [Tsochantaridis et al., 2004, Taskar et al., 2005, Joachims et al., 2009]. The latter techniques are applicable to learning the parameters in energy functions such as ours and may provide better models through global parameter estimation (as opposed to our piecewise method). Unfortunately, the methods rely on running inference a large number of times on each image (once for each parameter update) which is currently expensive. However, with the speed improvements to the energy minimization procedure such as the ones discussed above and the incorporation of warm-starts, this would make an interesting direction to pursue.

Our decomposition framework provides a basis on which many valuable extensions can be layered. We will consider two very natural extensions in the remainder of this thesis. In the next chapter, we incorporate object-based reasoning into our model, thereby allowing us to refine our generic foreground object class into subclasses representing specific object categories. Then, in Chapter 6, we use the output of our decomposition model to reason about the 3D structure of a scene and inform depth estimation. The combination of these two extensions can be used, for example, to understand the relative positions of objects in 3D-space and help infer the activities of the scene or find inconspicuous, or otherwise difficult-to-detect, objects (e.g., a soccer ball in a crowded football stadium).



⬜ sky 🟨 tree 🟪 road 🟩 grass 🟦 water 🟥 bldg 🟫 mntn 🟧 fg obj. ⬜ sky 🟩 horz. 🟥 vert.

Figure 4.4: Representative results when our scene decomposition model does well. Each cell shows original image (left), overlay of semantic class label (center), and surface geometry (right) for each image. Predicted location of horizon is shown on the geometry image. Best viewed in color.

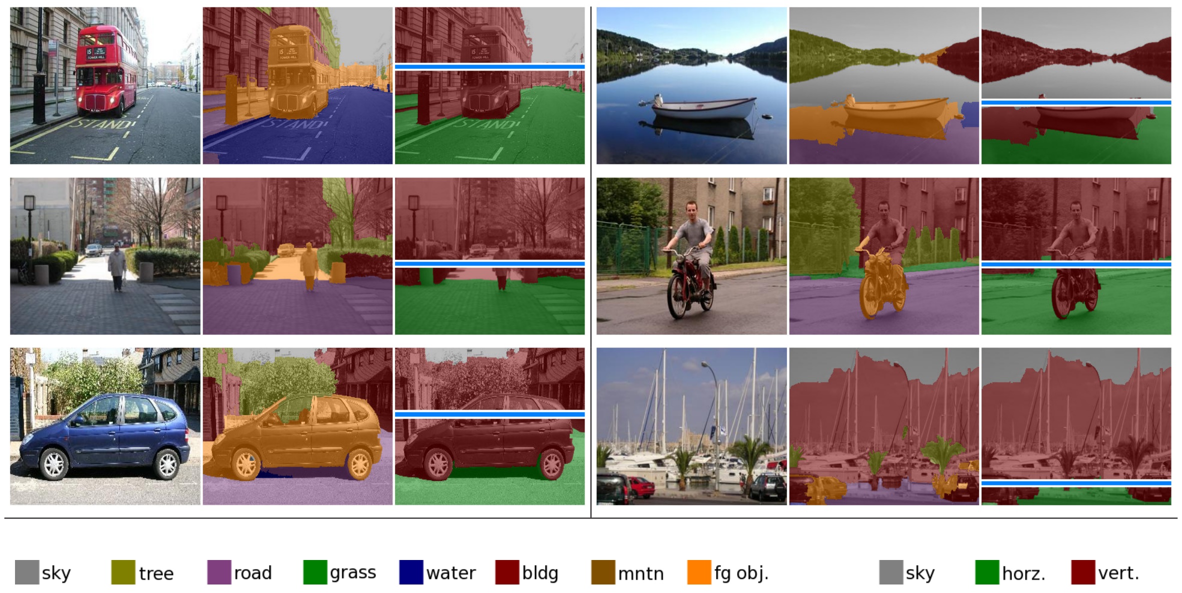


Figure 4.5: Examples of where our scene decomposition algorithm makes mistakes, such as mislabeling of road as water (top left), or confusing boat masts as buildings (bottom right). We also have difficulty with shadows and reflections. Best viewed in color.

Chapter 5

Object Detection and Outlining: From Regions to Objects

Object detection—the task of identifying and localizing objects in a novel image—is one of the great challenges of computer vision, and has received continuous attention since the birth of the field. As discussed in Chapter 2, the most common modern approaches to object detection scan the image for candidate objects and provide a score for each candidate. This is typified by the sliding-window object detection approach [Viola and Jones, 2004, Torralba et al., 2004b, Dalal and Triggs, 2005, Felzenszwalb et al., 2010], but is also true of most other detection schemes (such as centroid-based methods [Leibe et al., 2004] or boundary edge methods [Ferrari et al., 2008]). Many successful approaches for scoring object candidates combine cues from inside the object boundary (local features) with cues from outside of the object (contextual cues), e.g., [Heitz and Koller, 2008, Torralba et al., 2004b, Fink and Perona, 2003]. Most often the contextual cues are a function of appearance (e.g., "the area below the candidate window is *gray and textured*") rather than a function of scene semantics or geometry (e.g., "the area *underneath* the candidate object is *road*").

In almost all cases, the goal is to place a tight bounding box around the object of interest. Despite its popularity, this bounding box representation has many problems. For example, a bounding box includes many pixels within each candidate detection window that are not part of the object itself. Furthermore, multiple overlapping candidate detections contain many pixels in common. How these pixels should be treated is ambiguous in such bounding box approaches, and usually left completely unspecified. Ideally, we would like a method that uniquely identifies which pixels in the image belong to which objects rather than lackadaisically returning a set of bounding boxes with the claim that "the objects are somewhere in there." Uniquely identifying pixels is not only more elegant, but is also more likely to produce reliable results since it encodes a bias of the true world (i.e., a visible pixel belongs to only one object).[1]

Some notable exceptions to the standard bounding box approach attempt to produce object outlines or segmentation masks. These include the landmark-based model of Heitz et al. [2008a], the implicit shape model of Leibe et al. [2004], the figure-ground segmentation approach of Kumar et al. [2005], and the boundary fragment model of Opelt et al. [2006b]. However, these approaches still do not attempt to understand the scene holistically and as a result either leave the majority of the pixels in the image unexplained, or fail to take into consideration important contextual cues.

In this chapter, we build on our scene decomposition model from Chapter 4 to include the concept of objects. Our model has the flavor of some of the earliest works in scene understanding [Barrow and Tenenbaum, 1981, Marr, 1982] as well as that of more recent works that adopt a holistic approach to computer vision by combining the output of multiple vision tasks [Heitz et al., 2008b, Hoiem et al., 2008a]. Specifically, in this chapter, we develop an integrated region-based model that combines multi-class image labeling with object detection. We propose a hierarchical model, allowing us to reason simultaneously about pixels, regions and objects in the image, rather than scanning arbitrary overlapping windows. As in the previous chapter, we associate pixels with regions and label each region with a semantic class label from a fixed set of background classes (currently *sky*, *tree*, *road*, *grass*, *water*, *building*, *mountain*)

[1]Uniquely associating a pixel with a single (visible) object is not strictly true for scenes containing transparent objects. Some recent works (e.g., [Fritz et al., 2009]) have begun to consider the problem of detecting objects such as drinking glasses. However, dealing with transparent objects is still in its infancy and thus not considered in our work. When such techniques mature, the notion of pixel ownership can easily be extended to multiple objects.

or a single foreground class. The foreground class is then further classified, at the object level, into one of our known object classes (currently *boat*, *car*, *cow*, *motorbike*, *person*, *sheep*) or *unknown*.

While we have described the labeling process as a feed-forward pipeline, the actual inference procedure is a joint optimization over all variables in the model. Importantly, it allows cues from the object level to influence random variables at the region level. An example of where this would be useful was shown previously in Figure 1.1, where a sliding-window object detector correctly identified a person in the scene, but the pixel labeling task confuses his shirt as belonging to the tree class. In our model, the strong evidence for an object (which may come from global shape features) can override local appearance features, resulting in the model preferring the corresponding regions to be labeled as *foreground* rather than one of the background classes (in the case of Figure 1.1 the *tree* label). However, as we shall see, inferring the joint MAP assignment to all variables in our model requires more sophisticated proposal moves during inference than were discussed in the previous chapter. Specifically, the introduction of objects requires top-down reasoning.

One of the difficulties in recognizing complex objects such as people or cars, is that they are composed of several dissimilar regions (e.g., a person's head, torso, and legs) that cannot be combined by the simple bottom-up approach proposed in Chapter 4. Indeed, we have already seen that objects ("things") are characterized by global features such as shape and internally localized parts (e.g., the wheels of a car), rather than average color and texture. The hierarchical approach developed in this chapter facilitates both bottom-up and top-down reasoning about the scene. For example, during inference we can propose an entire object (from top-down information) comprised of multiple regions and evaluate this joint move against our global objective. Thus, our hierarchical model enjoys the best of two worlds: Like multi-class image segmentation, our model uniquely explains every pixel in the image and groups these into semantically coherent regions. Like object detection, our model uses sophisticated shape and appearance features computed over candidate object locations with precise boundaries.

A further benefit of our holistic approach to object detection is that our joint

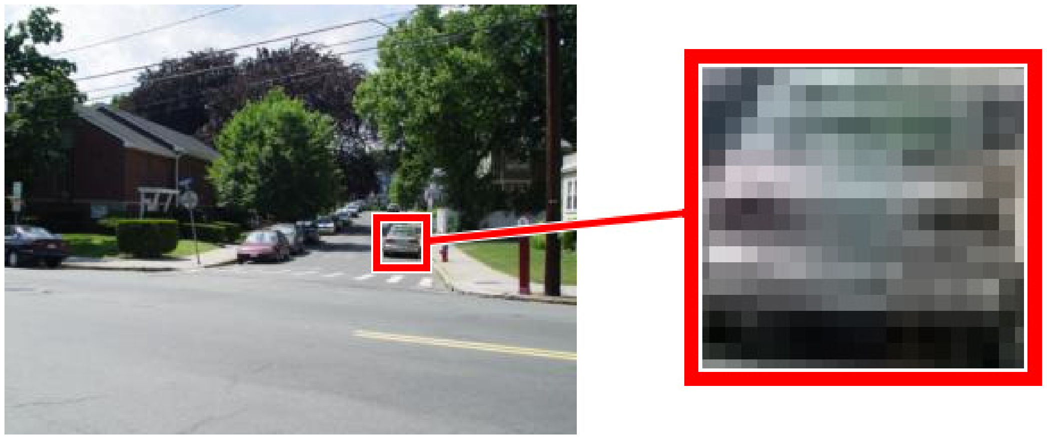


Figure 5.1: Contextual information is critical for recognizing small objects. The car in this image is less than 32×32 pixels in size and would be difficult to recognize without the surrounding context.

model over regions and objects allows contextual priors to be encoded through direct semantic relationships (e.g., cars are usually found on roads, and boats are usually found on water). These types of relationships are particularly important when objects are small. Consider, for example, the image in Figure 5.1. Here, a standard sliding-window object detection algorithm that only considers local features would have difficulty detecting the cars due to their small size—the largest car in the image has fewer than 32×32 pixels. However, by making use of contextual information such as proximity to the road and, perhaps, other car-like objects, as well as an understanding of global scale, we as humans have no trouble identifying the cars.

5.1 Region-based Model for Object Detection

In this section, we present our joint object detection and scene segmentation model. This model combines scene structure with region and object level semantics in a coherent energy function. Unlike the previous chapter, which included both geometric and semantic labels for each region, we restrict our attention here to just the semantic labels. However, we still maintain the horizon in our model as it constrains on the location of semantic classes and allows us to encode relative object size.

5.1.1 Energy Function

Our model builds on the model from the previous chapter, which aims to decompose a scene into a number (K) of semantically consistent regions. Recall that each pixel p in the image $\mathcal{I}$ is assigned to exactly one region, identified by its region-association variable $R_p \in \{1, \ldots, K\}$. The r-th region is simply the set of pixels $\mathcal{P}_r$ whose region-association variable equals r, i.e., $\mathcal{P}_r = \{p : R_p = r\}$. As in the previous chapter, we use p and q to denote pixels, and r and s to denote regions. In this chapter, we introduce the index o to denote objects. Double subscripts indicate pairwise terms between adjacent entities (e.g., or references the pairwise term between an object o and a region r adjacent to it).

Regions, while visually coherent, may not encompass entire objects. Indeed, in Chapter 4 we saw that foreground objects tend to be over-segmented into multiple regions. We address this complication by allowing an object to be composed of many regions rather than trying to force dissimilar regions to merge. The object to which a region belongs is denoted by its object-association variable $O_r \in \{\varnothing, 1, \ldots, N\}$. Some regions, such as background, do not belong to any object, which we denote by $O_r = \varnothing$. Like regions, the set of pixels that comprise the o-th object is denoted by $\mathcal{P}_o = \bigcup_{r:O_r=o} \mathcal{P}_r$. Currently, we do not allow a single region or object to be composed of multiple disconnected components.

Random variables are associated with the various entities (pixels, regions, objects, and horizon) in our model. Each pixel has a local appearance feature vector $\alpha_p \in \mathbb{R}^n$. Each region has an appearance variable A_r that summarizes the appearance of the whole region, a semantic class label S_r (such as "road" or "foreground object"), and an object-association variable O_r. Each object, in turn, has an object class label C_o (such as "car" or "pedestrian"). As discussed in Chapter 4, our model also includes the location of the horizon, which captures global geometry information. Recall, we assume that the image was taken by a camera with horizontal axis parallel to the ground and model the horizon $v^{\text{hz}} \in [0, 1]$ as the normalized row in the image corresponding to its location. We quantize v^{hz} into the same number of rows as the image.

We combine the variables in our model into a single coherent energy function

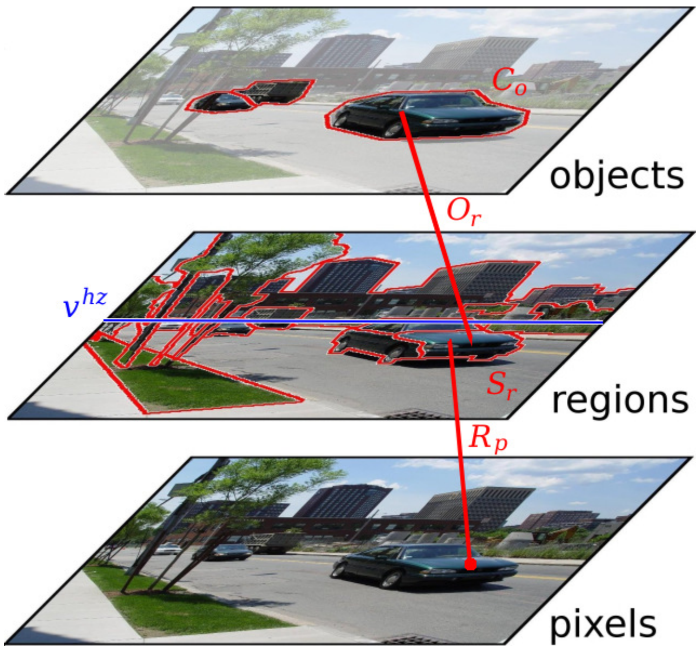


Figure 5.2: An illustration of random variables defined by our joint object detection and scene decomposition model.

that captures the structure and semantics of the scene. The energy function includes terms for modeling the location of the horizon, region label preferences, region boundary quality, object labels, and contextual relationships between objects and regions. These terms are described in detail below. The combined energy function has the form:

$$E(\boldsymbol{R}, \boldsymbol{S}, \boldsymbol{O}, \boldsymbol{C}, v^{\text{hz}} \mid \mathcal{I}, \boldsymbol{\theta}) = \psi^{\text{horizon}}(v^{\text{hz}}) + \sum_r \psi_r^{\text{reg}}(S_r, v^{\text{hz}}) \\ + \sum_{r,s} \psi_{rs}^{\text{bdry}} + \sum_o \psi_o^{\text{obj}}(C_o, v^{\text{hz}}) + \sum_{o,r} \psi_{or}^{\text{ctxt}}(C_o, S_r) \quad (5.1)$$

where for notational clarity the subscripts on the individual energy terms indicate that they are functions of the pixels (appearance and shape) belonging to the corresponding regions or objects, i.e., ψ_r^{reg} is also a function of $\mathcal{P}_r$, etc. It is assumed that all terms are conditioned on the observed image $\mathcal{I}$ and learned model parameters $\boldsymbol{\theta}$.

The summation over context terms includes all ordered pairs of adjacent objects and regions, while the summation over boundary terms is over unordered pairs of regions. An illustration of the variables in the energy function is shown in Figure 5.2.

The first three energy terms are adapted from the scene decomposition model of Chapter 4. We briefly review them here:

Horizon term. The ψ^{horizon} term captures the a priori location of the horizon in the scene and, in our model, is implemented as a log-gaussian

$$\psi^{\text{horizon}}(v^{\text{hz}}) = -\eta^{\text{horizon}} \log \mathcal{N}(v^{\text{hz}}; \mu, \sigma^2) \tag{5.2}$$

with parameters μ and σ learned from labeled training images.

Knowing the location of the horizon allows us to compute the world height of an object in the scene. Using the derivation from Hoiem et al. [2008b], it can be shown that the height y_k of an object (or region) in the scene can be approximated as $y_k \approx h\frac{v_t - v_b}{v^{\text{hz}} - v_b}$ where h is the height of the camera origin above the ground, and v_t and v_b are the row of the top-most and bottom-most pixels in the object/region, respectively. In our current work, we assume that all images were taken from the same height above the ground, allowing us to use $\frac{v_t - v_b}{v^{\text{hz}} - v_b}$ as a feature in our region and object terms.

Region term. The region term ψ^{reg} in our energy function captures the preference for a region to be assigned different semantic labels (currently *sky*, *tree*, *road*, *grass*, *water*, *building*, *mountain*, *foreground*). For convenience we include the v^{hz} variable in this term to provide information about scene structure. If a region is associated with an object, then we constrain the assignment of its class label to *foreground* (e.g., a “sky” region cannot be part of a “car” object).

More formally, let N_r be the number of pixels in region r, i.e., $N_r = \sum_p \mathbf{1}\{R_p = r\}$, and let $\phi_r\left(A_r, \mathcal{P}_r\right) \in \mathbb{R}^n$ denote the features for the r-th region (see Section 4.1.3).

The region term is then

$$\psi_r^{\text{reg}}(S_r, v^{\text{hz}}) = \begin{cases} \infty & \text{if } O_r \neq \varnothing \text{ and } S_r \neq \textit{foreground} \\ -\eta^{\text{reg}} N_r \log \sigma\left(S_r \mid \phi_r, v^{\text{hz}}; \theta^{\text{reg}}\right) & \text{otherwise} \end{cases} \tag{5.3}$$

where $\sigma(\cdot)$ is the multi-class logistic function and η^{reg} is the relative weight of the region term versus the other terms in the model.

Boundary term. The term ψ^{bdry} penalizes two adjacent regions that have similar appearance or lack of boundary contrast. This helps to merge coherent pixels into a single region. We combine two metrics for boundary quality: The first captures region similarity as a whole, the second captures contrast along the common boundary between the regions. Specifically, let $d\left(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{S}\right) = \sqrt{(\boldsymbol{x}-\boldsymbol{y})^T \boldsymbol{S}^{-1}(\boldsymbol{x}-\boldsymbol{y})}$ denote the Mahalanobis distance between vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, and let $\mathcal{E}_{rs}$ be the set of pixels along the boundary between regions r and s. Then, the boundary term is

$$\psi_{rs}^{\text{bdry}} = \eta_A^{\text{bdry}} \cdot |\mathcal{E}_{rs}| \cdot e^{-\frac{1}{2}d(A_r, A_s; \Sigma_A)^2} + \eta_\alpha^{\text{bdry}} \sum_{(p,q)\in\mathcal{E}_{rs}} e^{-\frac{1}{2}d(\alpha_p, \alpha_q; \Sigma_\alpha)^2} \tag{5.4}$$

where the Σ_A and Σ_α are the image-specific pixel appearance covariance matrix computed over all pixels and neighboring pixels, respectively. In our experiments, we restrict Σ_A to be diagonal and set $\Sigma_\alpha = \beta\mathbf{I}$ with $\beta = \frac{1}{|\mathcal{E}|}\sum_{(p,q)\in\mathcal{E}} \|\alpha_p - \alpha_q\|^2$ as in Chapter 4. Furthermore, we only consider the first three color components of the pixel appearance feature vectors α_p (and α_q). The parameters η_A^{bdry} and $\eta_\alpha^{\text{bdry}}$ encode the trade-off between the region similarity and boundary contrast terms and weight them against the other terms in the energy function (Equation 5.1).

Note that, unlike the pairwise term ψ^{pair} in Section 4.1.4, the boundary term here does not include semantic class or object information, since pairwise semantic context is captured by our new object-to-region context term ψ^{ctxt} discussed below.

Object term. Going beyond the scene decomposition model of Chapter 4, we include object terms ψ^{obj} in our energy function that score the likelihood of a group of regions being assigned a given object label. In addition to our known object classes (currently, *boat*, *car*, *cow*, *motorbike*, *person*, and *sheep*), an object (or foreground region) can take the class label *unknown*. This *unknown* class includes objects not currently explicitly modeled, such as trash cans, street signs, telegraph poles, traffic cones, bicycles, etc. Similar to the region term, the object term is defined by a logistic function that maps object-specific features $\phi_o\left(\mathcal{P}_o, v^{\text{hz}}, \mathcal{I}\right) \in \mathbb{R}^n$ to a probability for each object class. We discuss the construction of these features in Section 5.1.2 below.

Instead of taking the log-likelihood from the logistic function and incorporating it directly into our energy function, we first perform an important refinement that acts to normalize this term against information already present in the region layer. Specifically, since our region layer already identifies foreground regions, we would like our energy to improve only when we recognize known object classes. We therefore bias the object term to give zero contribution to the energy for the class *unknown*.[2] Formally, we have

$$\psi_o^{\text{obj}}(C_o, v^{\text{hz}}) = -\eta^{\text{obj}} N_o \left(\log \sigma\left(C_o \mid \phi_o; \theta^{\text{obj}}\right) - \log \sigma\left(\mathit{unknown} \mid \phi_o; \theta^{\text{obj}}\right)\right) \qquad (5.5)$$

where $N_o = \sum_{r:O_r=o} N_r$ is the number of pixels belonging to the object.

Context term. Intuitively, contextual information which relates objects to their local background can improve object detection. For example, Heitz and Koller [2008] showed that detection rates improve by relating "things" (objects) to "stuff" (background). Our model has a very natural way of encoding such relationships through pairwise energy terms between objects C_o and regions S_r. We do not encode contextual relationships between region classes (i.e., S_r and S_s) although this could also be done. However, most interesting contextual relationships exist between foreground objects and their neighboring regions in the scene. Contextual relationships between two adjacent foreground objects (i.e., C_o and C_m) may be beneficial (e.g., people found on bicycles), but are not considered in this work. Formally, the context term

[2]Technically, this allows the O_r variable to take the value $\varnothing$ for *unknown* foreground regions without affecting the energy.

Context Feature	Description
Relative Position (8)	Orientation of the centroid of region r with respect to the centroid of object o quantized into eight compass directions.

Table 5.1: Summary of the features $\phi_{or}(\mathcal{P}_o, \mathcal{P}_r)$ used to describe the contextual pairwise relationship between an object o and adjacent region r.

in the energy function is

$$\psi_{or}^{\text{ctxt}}(C_o, S_r) = -\eta^{\text{ctxt}} \log \sigma\left(C_o \times S_r \mid \phi_{or}; \theta^{\text{ctxt}}\right) \tag{5.6}$$

where $\phi_{or}(\mathcal{P}_o, \mathcal{P}_r, \mathcal{I}) \in \mathbb{R}^n$ is a pairwise feature vector for object o and region r, $\sigma(\cdot)$ is the multi-class logistic function, and η^{ctxt} weights the strength of the context term relative to other terms in the energy function. Our context features, summarized in Table 5.1, simply capture the orientation of the region with respect to the object allowing us to encode simple preferences such as "on top of", "next to", etc.

Since the pairwise context term is between objects and (background) regions it grows linearly with the number of object classes. This has a distinct advantage over including a pairwise term between all classes, which would result in quadratic growth.

5.1.2 Object Features

Performing well at object detection requires more than simple aggregated region appearance features. Indeed, the power of state-of-the-art object detectors comes from their ability to model localized appearance and general shape characteristics of an object class. Thus, in addition to raw appearance features (similar to our region-level appearance features), we append to our object feature vector ϕ_o features derived from such object detection models. We discuss two methods for adapting state-of-the-art object detector machinery for this purpose.

Black-box Object Detectors

In the first approach, we treat the object detector as a black-box that returns a score per (rectangular) candidate window. However, recall that an object in our model is defined by a contiguous set of pixels $\mathcal{P}_o$ that can have an arbitrary shape, not necessarily a rectangular window. In the black-box approach, we naively place a bounding box (at the correct aspect ratio) around these pixels and classify the entire contents of the box. To make classification more robust we search candidate windows in a small neighborhood (defined over scale and position) around this bounding box, and take as our feature the output of highest scoring window. Any state-of-the-art object detection algorithm can be used in the black-box model allowing this approach to easily take advantage of advances in object detection technology. In our experiments, we test this approach using the HOG detector of Dalal and Triggs [2005], which learns a linear support vector machine (SVM) classifier over feature vectors constructed by computing histograms of gradient orientations in fixed-size overlapping cells within the candidate window (see Chapter 2).

An important practical consideration is how to interface the black-box object detector (for which we may only have an external executable) to our model. It appears that we have a "chicken and egg" situation: Before running inference we do not know which pixels belong to objects, however, we cannot run inference without computing features over the objects. Since our inference procedure (discussed in Section 5.2.1 below) is proposal-move based, one solution is to make a call to the object detector each time we propose a new object. However, this solution incurs significant overhead resulting in a dramatic increase in running time.

A more pragmatic approach is to first run the (external) object detector over the entire image to collect object detection candidate windows. This is the usual operating mode of an object detector. We can then limit the location of objects in the scene to the subset of windows returned by the external object detector. This, in effect, forces ψ_o^{obj} to take the value ∞ whenever $\mathcal{P}_o$ does not sufficiently overlap with a candidate detection. Note that we can tune the threshold of the object detector to increase the number of detection candidates making it more likely that we cover all true objects in the scene.

Integrated Object Detectors

Despite its simplicity, the black-box approach is somewhat unsatisfying in that many of the pixels within the bounding box being evaluated are not actually part of the object. For example, consider an L-shaped region. To a black-box approach, this is indistinguishable from a full bounding box positioned tightly around the "L".

A more elegant approach is to mask out all pixels not belonging to the object. In our implementation, we use a soft mask that attenuates the intensity of pixels outside of the object boundary as a function of their distance to the boundary (see Figure 5.3). This has the dual advantage of preventing hard edge artifacts and being less sensitive to small segmentation errors. Consider, for example, the true-positive window for detecting cars shown in Figure 5.3(c). Here, with the correct region mask, the effect of masking (either soft or hard) is not significant. However, in the case of a false-positive window (e.g., Figure 5.3(h)), applying a hard mask introduces strong edge artifacts that may cause the window to be incorrectly classified, whereas the soft mask produces a region that is unlikely to be confused with a car.

The soft mask can be thought of as a matrix the same size as the window being classified. It is generate by setting entries corresponding to the pixels in the object to one, and all other entries to zero. (This gives the hard mask.) We then dilate the mask by 7 pixels and apply a Gaussian smoothing filter with a radius of 15 pixels. This ensures a gradual decay in pixel intensity beyond the object boundary. Finally, we multiply the image color values by the soft mask to get the window for classification. The soft mask is used at both training and test time to ensure consistency between training and test examples.[3]

Many different object detectors can be adapted for this more integrated approach, and in our experiments, we evaluate the approach using the patch-based approach of Torralba et al. [2004a,b]. Here, features are extracted by matching small rectangular patches at fixed locations within the masked window and combining these weak

[3]At training time we have ground truth labels for pixels belonging to regions and can therefore generate the masks. It is also possible to use standard black-box training methods which do not have access to pixel masks and still apply the trained classifier to masked regions during test. However, we did not evaluate performance under this regime.

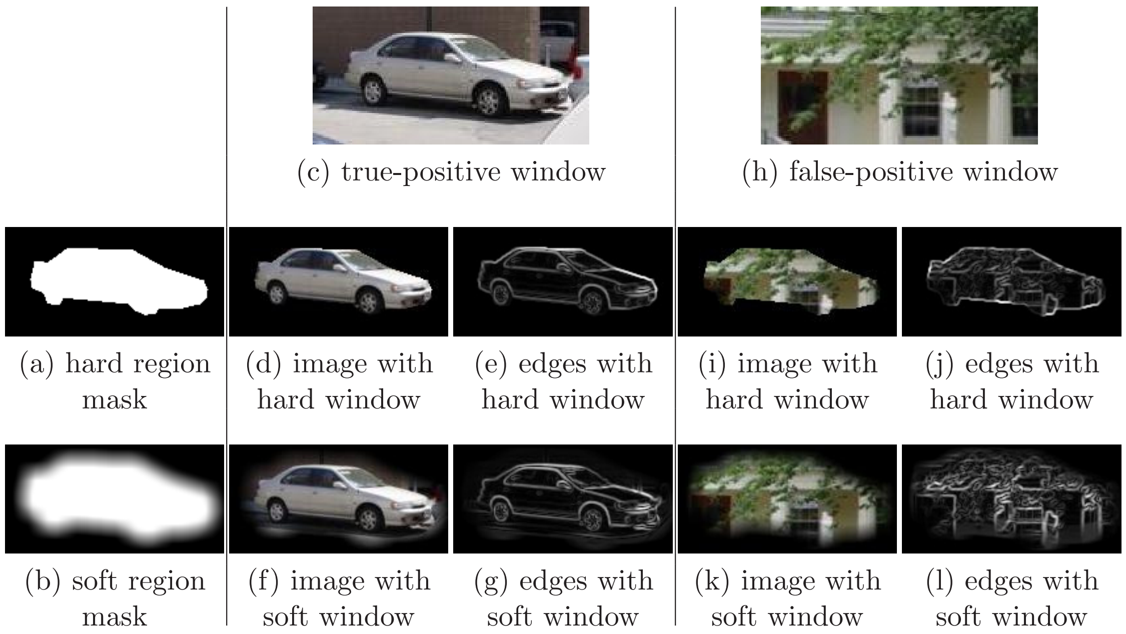

(c) true-positive window
(h) false-positive window
(a) hard region mask
(d) image with hard window
(e) edges with hard window
(i) image with hard window
(j) edges with hard window
(b) soft region mask
(f) image with soft window
(g) edges with soft window
(k) image with soft window
(l) edges with soft window

Figure 5.3: Illustration of soft masks for proposed object regions. Using a hard region mask is problematic since it introduces edge artifacts that can result in misclassification of the window. These artifacts are reduced when a soft mask is used.

responses using boosted decision trees. Object appearance and shape are captured by matching the patches on both the original (intensity) image and the edge-filtered image. A more complete description of this patch-based object detector is provided in Section 2.2.3.

For both black-box and integrated approaches, we append the score (for each object category) from the object detection classifiers—linear SVM for HOG-bases or boosted decision trees for patch-based—to the object feature vector ϕ_o. Our object-level features are summarized in Table 5.2.

Image Resolution

An important parameter for object detection is the base scale at which features are extracted. Scale-invariance is usually achieved by successively down-sampling the image. Below the base-scale, feature matching becomes inaccurate, so most detectors will only find objects above some minimum size. Clearly there exists a trade-off

Object Feature	Description
Appearance (51)	Aggregated region appearance: Mean and standard deviation of the 17-dimensional filter responses and boosted classifier score for each pixel in the object, and log-determinant of the appearance covariance matrix Σ_o^A computed over all pixels in the object.
Boundary Contrast (4)	Measure of the contrast along the boundary of the object relative to its interior.
Shape (19)	Basic region shape such as size and extent (relative to image dimensions), perimeter length and “straightness”, and first- and second-order shape moments. These features also include binary indicators capturing whether the region has been truncated by the edge, top, or bottom of the image.
Geometry (10)	Like its region counterpart, these features capture the relative location of the object with respect to the horizon v^{hz} and include the percentage of pixels in the object above and below the horizon; the spread of pixels about the horizon (i.e., mean distance above and below the horizon). We also add the term $\frac{v_t - v_b}{v^{\text{hz}} - v_b}$ where v_t and v_b are the topmost and bottom most row in the object, respectively. This captures the relative height of objects in the scene.
Detector Score (1)	Score from running trained object detector on the pixels belonging to the object. The score can either be derived by running a black-box detector on a bounding box covering the pixels in the object, or from an integrated object detector applied to a masked-window as shown in Figure 5.3.

Table 5.2: Summary of the features $\phi_o(\mathcal{P}_o, v^{\text{hz}}, \mathcal{I})$ used to describe an object in our model. The first five sets of features are extended variants of the region features described in the previous chapter. The last feature is a powerful top-down feature that captures global cues (such as shape and localized appearance).

between the desire to detect small objects, feature quality, and computational cost. To reduce the computational burden of running our full scene decomposition model on high-resolution images while still being able to identify small objects, we employ a multi-scale approach. In this approach, we run our scene decomposition algorithm on a low-resolution (320×240) version of the scene, but extract features from the original high-resolution version (if available in the dataset).[4] That is, when we extract object-detector features we map the object pixels $\mathcal{P}_o$ onto the original image and extract our features at the higher resolution.

5.2 Inference and Learning

In Section 4.2 we developed the inference and learning procedures for the basic scene decomposition model. In this section we briefly review those methods and describe the extensions necessitated by the inclusion of object variables into our model.

5.2.1 Inference

We use a modified version of the hill-climbing inference algorithm described in Chapter 4, which uses multiple over-segmentations to propose large moves in the energy space. An overview of this procedure is shown in Algorithm 5.1. As in Chapter 4, we initialize the variables in our model by segmenting the image using an off-the-shelf unsupervised segmentation algorithm (e.g., mean-shift [Comaniciu and Meer, 2002]). We then run inference using a two-phased approach.

In the first phase, we want to build up a good set of initial regions before trying to classify them as objects. Thus, we remove the object variables $\boldsymbol{O}$ and $\boldsymbol{C}$, and corresponding energy terms ψ^{obj} and ψ^{ctxt}, from the model, and perform inference over the reduced state space. In this phase, the algorithm behaves exactly as in the previous chapter by iteratively proposing re-assignments of pixels to regions (variables $\boldsymbol{R}$) and recomputing the optimal (conditional) assignment to the remaining variables (in this case, $\boldsymbol{S}$ and v^{hz}). If the overall energy for the new configuration is lower, then

[4]In our experiments, high-resolution images were only available for the Street Scenes dataset.

the move is accepted, otherwise the previous configuration is restored and the algorithm proposes a different move. The algorithm proceeds until no further reduction in energy can be found after exhausting all proposal moves from a predefined set (see Section 5.2.2).

In the second phase, we introduce object variables over all foreground regions. Initially, the object class variable is set to *unknown* so that this operation does not affect the overall energy. We then iteratively propose merges and splits of both regions and objects (variables $\boldsymbol{R}$ and $\boldsymbol{O}$) as well as top-down proposal moves (see Section 5.2.2 below) to generate new regions from sliding-window object candidates (again, affecting both $\boldsymbol{R}$ and $\boldsymbol{O}$). After a move is proposed, we recompute the optimal assignment to the remaining variables ($\boldsymbol{S}$, $\boldsymbol{C}$ and v^{hz}). As in the first phase, this process repeats until the energy cannot be reduced by any of the proposal moves.

Since only part of the scene is changing during any iteration we only need to recompute the features and energy terms for the entities affected by a move. However, inference is still slow given the sophisticated features that need to be computed and the large number of moves considered. To improve running time, we leave the context terms ψ^{ctxt} out of the model until the last iteration through the proposal moves. This allows us to maximize each region term independently during each proposal step. We then greedily update v^{hz} after the region labels have been inferred. After introducing the context term, we use max-product belief propagation to infer the optimal joint assignment to $\boldsymbol{S}$ and $\boldsymbol{C}$.

5.2.2 Proposal Moves

We now describe the set of pixel and region proposal moves considered by our algorithm. These moves are relative to the current best scene decomposition and are designed to take large steps in the energy space to avoid local minima. As discussed above, each move is accepted if it results in a lower overall energy after inferring the optimal assignment to the remaining variables.

The main set of pixel moves are described in Chapter 4 but briefly repeated here

Algorithm 5.1: Scene Decomposition with Objects Inference.

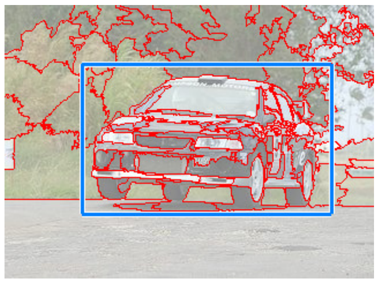
```
Input  : $\mathcal{I}$                                  // image
         $\boldsymbol{\theta}$                                  // model parameters
Output: $\boldsymbol{R}, \boldsymbol{S}, \boldsymbol{O}, \boldsymbol{C},$ and $v^{\mathrm{hz}}$     // scene decomposition
// phase one:
generate over-segmentation dictionary $\Omega$ (see § 4.2.1)
initialize variables $\boldsymbol{R}$, $\boldsymbol{S}$ and $v^{\mathrm{hz}}$
repeat
    foreach pixel proposal move do
        re-assign region-association variables $\boldsymbol{R}$ (see § 4.2.1)
        update region and boundary features $\boldsymbol{\phi}$
        run inference over region variables $\boldsymbol{S}$ and horizon $v^{\mathrm{hz}}$
        compute total energy $E$ of new assignments
        if $(E < E^{min})$ then
            accept move and set $E^{\mathrm{min}} = E$
        else
            reject move
        end
    end foreach
until $E^{min}$ does not change
// phase two:
generate top-down object dictionary (see § 5.2.2)
initialize variables $\boldsymbol{O}$ and $\boldsymbol{C}$
repeat
    foreach pixel or region proposal move do
        re-assign region-association $\boldsymbol{R}$ or object-association variables $\boldsymbol{O}$
        update region, boundary and object features $\boldsymbol{\phi}$
        run inference over region and object variables $(\boldsymbol{S}, \boldsymbol{C})$ and horizon $v^{\mathrm{hz}}$
        compute total energy $E$ of new assignments
        if $(E < E^{min})$ then
            accept move and set $E^{\mathrm{min}} = E$
        else
            reject move
        end
    end foreach
until $E^{min}$ does not change
return $\boldsymbol{R}, \boldsymbol{S}, \boldsymbol{O}, \boldsymbol{C},$ and $v^{\mathrm{hz}}$
```

for completeness. The most basic move is to merge two adjacent regions. More sophisticated moves involve local re-assignment of pixels to neighboring regions. These moves are proposed from a precomputed dictionary of image segments Ω. The dictionary is generated by varying the parameters of an unsupervised over-segmentation algorithm (in our case mean-shift [Comaniciu and Meer, 2002]) and adding each segment ω to the dictionary. During inference, these segments are used to propose a re-assignment of all pixels in the segment to a neighboring region or creation of new region. As shown in Chapter 4, these bottom-up proposal moves work well for background classes, but tend to result in over-segmentation of foreground objects that have heterogeneous appearance. For example, one would not expect the wheels and body of a car to be grouped together by a bottom-up approach.

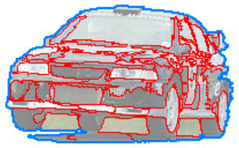
An analogous set of moves can be used for merging two adjacent objects or assigning regions to objects. For example, merging two adjacent objects o and s involves re-assigning all regions $\{r : O_r = s\}$ to o, or creating a new object by assigning O_r to a new object for some region r. However, if an object is decomposed into multiple regions, this bottom-up approach is problematic as multiple such moves may be required to produce a complete object. When performed independently, these moves are unlikely to improve the energy. We get around this difficulty by introducing a new set of powerful top-down proposal moves based on object detection candidates. Here, we use precomputed candidates from a sliding-window detector to propose new foreground regions with corresponding object variable.[5] In the case of the black-box approach, described in Section 5.1.2, these are the only candidate objects that we allow.

Directly proposing moves involving all pixels inside the bounding box obtained from the sliding-window detector for the candidate object, is likely to result in an increase in energy due to the introduction of artificial boundaries into the scene decomposition. We implement two solutions to this problem. First, instead of proposing a move over all pixels in the bounding box, we propose move that re-assign only the pixels inside the set of segments (from our segment dictionary Ω) fully contained within the bounding box. An illustration of this procedure is shown in Figure 5.4.

[5]In our experiments, we take as candidates the top twenty detections for each object category.



(a) Over-segmentation and top-down bounding box



(b) Proposal move

Figure 5.4: Illustration of a preprocessed top-down object proposal move. First, a bounding box from a sliding-window object detector is combined with one of the over-segmentations of the image in (a). Then, all pixels inside segments fully contained within the bounding box are proposed in the move. The set of pixels in the proposal move does not rely on the energy function and can be precomputed.

This solution has the advantage that the moves can be precomputed from the bounding box output and the set of over-segmentations. If the set of segments fully contained within the bounding box are disconnected, then we only propose the largest connected component to prevent creating objects comprised of disconnected regions.

An alternative approach is to allow the energy to temporarily increase when the move is initially proposed. We then iterate through the segment dictionary Ω proposing moves that overlap with the current proposal (i.e., object bounding box), and accept those moves that result in a lower energy. If, after iterating through the segment dictionary, the resulting scene decomposition has a higher energy than before proposing the initial bounding box move, then the variables are reverted to their previous state (i.e., before the bounding box move). Otherwise the sequence of moves has resulted in a better scene decomposition and we continue with the variable assignments at hand. This solution can be thought of as using our inference algorithm to erode the bounding box proposals. The approach is illustrated in Figure 5.5. This more costly approach has the advantage that the moves can adapt to the current decomposition rather than be precomputed.

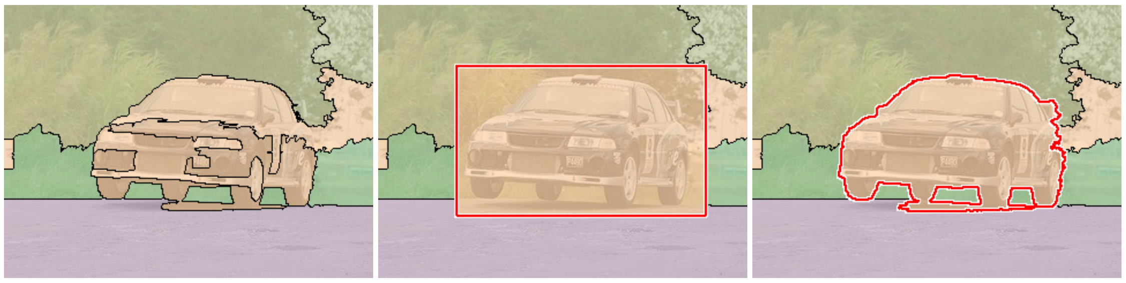

(a) Starting decomposition (b) Proposed bounding box (c) Final eroded move

Figure 5.5: Illustration of eroding a top-down bounding box move. The proposed bounding box move in (b) is likely to cause the energy to increase above the energy of starting decomposition (a) due to the region boundary artifacts introduced by the move. After running inference the bounding box is eroded resulting in lower energy scene decomposition in (c). Region boundaries are shown in black. Since the proposal move makes use of the current scene decomposition and energy function, the set of pixels in the final eroded move cannot be predetermined.

5.2.3 Learning

Object detectors are learned separately using black-box training procedures and their output provided as input features to the combined model (as discussed in Section 5.1.2). Then, as in Chapter 4, we learn the parameters of our model from labeled training data in a piecewise fashion. First, the individual terms are learned using an ℓ_2-regularized maximum-likelihood objective for the subset of variables within each term. The relative weights (η^{reg}, η^{obj}, etc.) between the terms are learned through cross-validation on a subset of the training data to optimize object detection performance (as measured by area under the precision-recall curve).

For both the base object detectors and the parameters of the region and object terms, we use a closed-loop learning technique where we first learn an initial set of parameters from training data. We then run inference on our training set and record mistakes made by the algorithm (false-positives for object detection and incorrect region proposal moves for the full algorithm as described in Section 4.2.2). We augment the training data with these mistakes and re-train each term. This process gives a significant improvement to the final results.

5.3 Experiments

We conducted experiments on two challenging datsets. The first dataset, Street Scenes [Bileschi and Wolf, 2005], contains a very large collection of images, but is limited in both the variety of the scenes (i.e., all images were taken in an urban environment) and in the number of object classes—predominantly cars and pedestrians. The second dataset is the CCM (DS1) dataset from Heitz et al. [2008b], which was designed for evaluating holistic scene understanding methods. It is smaller than the Street Scenes dataset, but contains a wider variety of scenes (urban, rural and harbor) and six different object classes. In both cases, we evaluate our model using HOG-based black-box detectors and patch-based integrated detectors, and compare to baselines that use the same detectors. In the case of the CCM dataset, we also compare to the state-of-the-art results of Heitz et al. [2008b]. To be consistent with these results, when evaluating object detector performance, we use an area-of-overlap score of 0.2 for determining correspondence with a ground-truth object.

5.3.1 Street Scenes Dataset

Experiments on the Street Scenes dataset [Bileschi and Wolf, 2005] were conducted using a single fold containing 2837 images for training and 710 images for testing. The images are high-resolution so we rescaled them to 320×240 before running our algorithm, but as discussed in Section 5.1.2, we used the original resolution for extracting object features. The dataset comes with hand-annotated region labels and object boundaries. However, the annotations use rough overlapping polygons, so, as discussed in Chapter 3, we used Amazon Mechanical Turk (AMT) to improve the labeling of the background classes. We kept the original polygons for the foreground object classes during evaluation to be consistent with other results on this dataset.

The multi-class image segmentation component of our model achieves an overall pixel-level accuracy of 84.2% across the eight semantic classes compared to 83.0% for the pixel-based baseline method described in Chapter 4. More interesting was our object detection performance. The test set contained 1183 cars and 293 pedestrians with average size of 86×48 and 22×49 pixels, respectively. Many objects are occluded

	Patch-based			HOG-based		
	Baseln.	Unary	Ctxt.	Baseln.	Unary	Ctxt.
Car	0.40	0.55	0.56	0.35	**0.58**	0.57
Pedestrian	0.15	0.22	0.21	**0.37**	0.35	0.35
Mean	0.28	0.39	0.39	0.36	**0.47**	0.46

Table 5.3: Object detections results for the Street Scenes dataset [Bileschi and Wolf, 2005]. Performance is measured by area under the precision-recall curve (average precision).

making this a very difficult dataset.

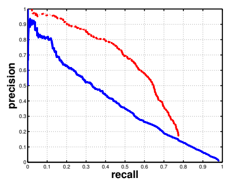
Since our algorithm produces MAP estimation for the scene we cannot simply generate a precision-recall curve by varying the object classifier threshold as is usual for reporting object detection results. Instead, we take the max-marginals for each C_o variable at convergence of our algorithm and sweep over thresholds for each object class separately to generate a curve. An attractive aspect of this approach is that our method does not have overlapping candidates and hence does not require arbitrary post-processing such as non-maximal suppression of sliding-window detections.

Our results are summarized in Table 5.3. We include a comparison against two baseline sliding-window approaches. For the patch-based detector, our method improves over the baseline for both object categories. Our method also shows significant improvement over the HOG baseline for car detection. Interestingly, for pedestrian detection, our method shows only comparable performance to the HOG baseline. However, this should not be surprising since the HOG detector was specifically engineered for the task of pedestrian detection. Furthermore, as we discuss below, context is not as helpful here due to the limited variety of scenes.

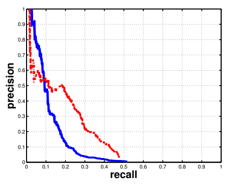
Complete precision-recall curves are shown in Figure 5.6 for both the patch-based and HOG-based detectors. Notice that our method does not achieve 100% recall (even at low precision) due to the curves being generated from the MAP assignment in which pixels have already been grouped into regions. Unlike the baselines, this forces only one candidate object per region. However, by trading off the strength (and hence operating point) of the energy terms in our model it is possible to increase the maximum recall for a given object class at the expense of precision.

Removing the pairwise context term does not have a significant effect on our

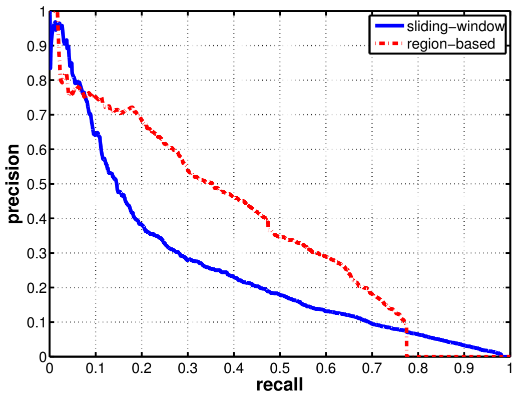
Patch-based



(a) Cars

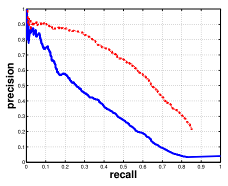


(b) Pedestrians

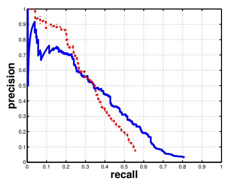


(c) Combined cars and pedestrians

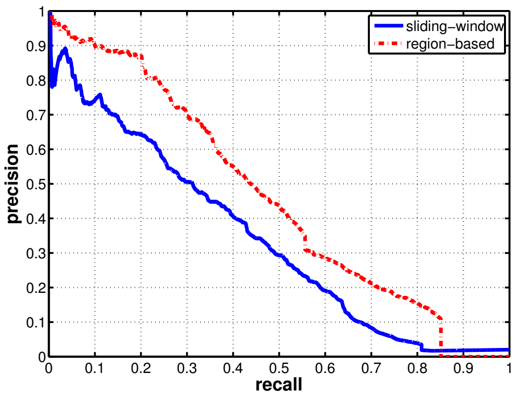
HOG-based



(a) Cars



(b) Pedestrians



(c) Combined cars and pedestrians

Figure 5.6: Object detection precision-recall curves for patch-based (left) and HOG-based (right) detectors on the Street Scenes dataset [Bileschi and Wolf, 2005]. Plots (a) and (b) show individual PR curves for each object; plot (c) shows combined curve.

results. This is due to the encoding of some context through the region term (e.g., the relative position of the horizon) and the fact that all images were of urban scenes. Hence most detections—both true and false—were supported by road regions and surrounded by buildings. As we shall see, context does play an important role when we consider the more varied scenes in the CCM dataset.

Example output for a typical image from our dataset is shown in Figure 5.7. The example shows the original image, semantic overlay with object outlines, scene decomposition (in terms of regions), and estimated horizon location. Notice how the two foreground regions get grouped together into a single "car" object.

We also show a selection of qualitative object detection results in Figure 5.8. Here, we only show the original image and semantic overlay for each example. Notice that our algorithm is able to detect multiple (adjacent) objects per image. In some cases, for example, the scene containing two cars in the first column and second row, the

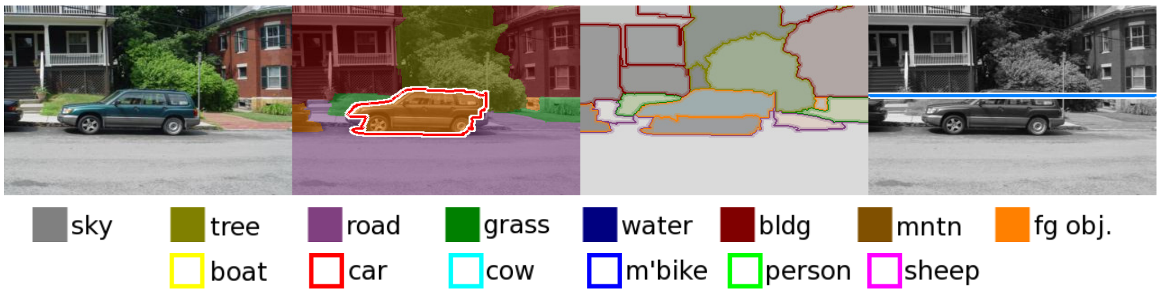


Figure 5.7: Example output from our joint scene decomposition and object detection algorithm. Shown are: (a) the original image, (b) semantic overlay (including object outlines), (c) decomposition by region, and (d) estimated location of horizon.

algorithm incorrectly groups parts of one object with the adjacent object. However, as can be seen in the other examples, we generally get very good boundaries for cars. Pedestrians are much more difficult in this dataset, and their small size makes it hard to obtain precise outlines even when correctly detected (e.g., bottom-right example in Figure 5.8).

5.3.2 Cascaded Classification Models Dataset

Our second set of experiments were conducted on the more diverse, but smaller CCM dataset [Heitz et al., 2008b]. Here we conducted five-fold cross-validation on the 422 images and report object detection results averaged across the five folds.

Again, we compare our approach to the HOG-based and patch-based baselines. However, for the HOG-based baseline we use post-processed detection candidates from the *independent* method of Heitz et al. [2008b] rather than directly from the Dalal and Triggs [2005] detector. This produces a significantly stronger baseline. Aggregated results from running our inference procedure on the five evaluation folds are shown in Table 5.4. The table includes the number of instances, precision, and recall for each object category individually and combined across all object categories (i.e., weighted average). As can be seen, our model favors high precision at the expense of recall for some of the more difficult object categories (e.g., pedestrians and boats). That is, objects that get labeled by our inference procedure tend to be correct. However,

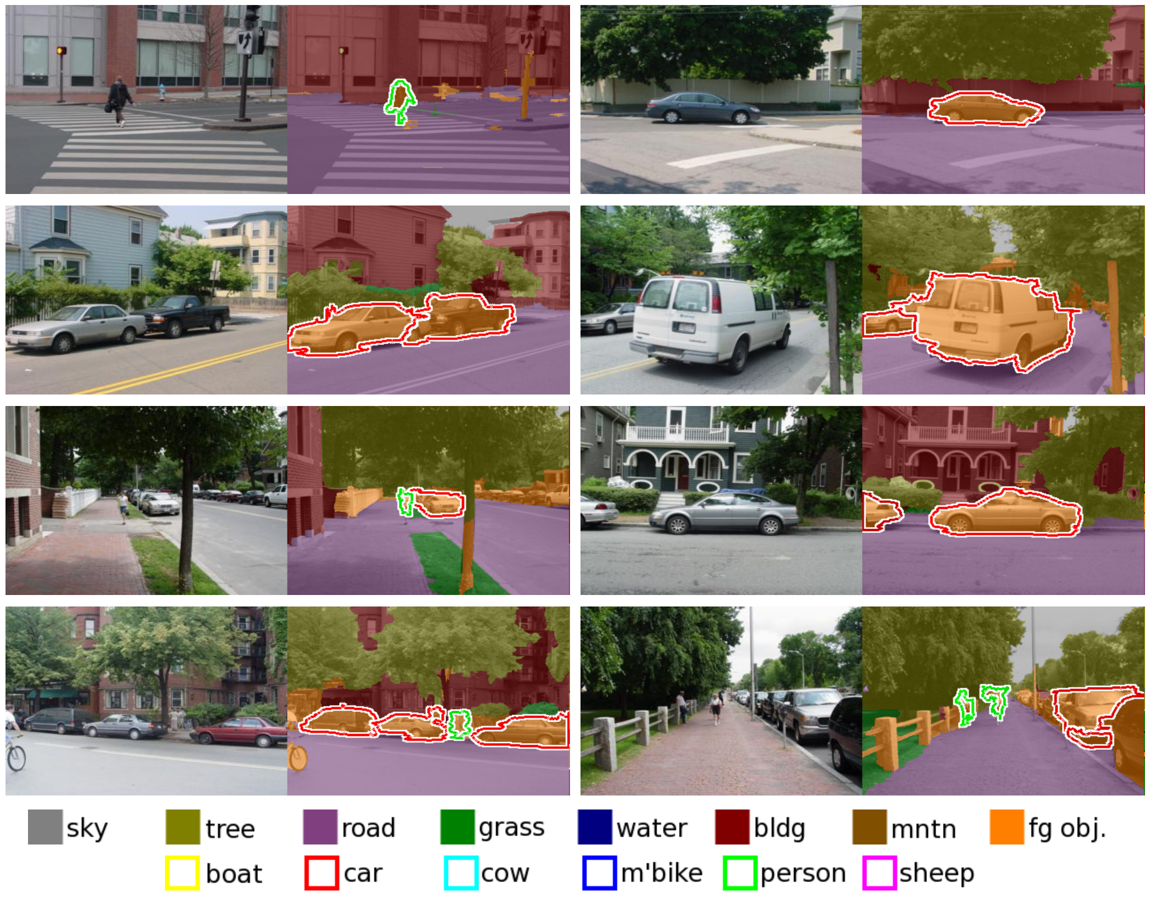


Figure 5.8: Qualitative object detection results for the Street Scenes dataset [Bileschi and Wolf, 2005]. Each example show original image (left) and overlay colored by semantic class and detected objects (right).

many of the objects in the dataset are missed by our algorithm at this high-precision operating point.

As discussed previously, there is a trade-off between precision and recall that can be tuned by modifying the weight of the object terms in the energy function. For example, setting the weight for one of the object terms very high will result in more proposal moves being accepted for that object category, which, in turn, will result in higher recall. Note, however, that unlike the standard sliding-window approach to object detection, we do not allow overlapping detections and therefore may still not be able to achieve 100% recall (for example, if a large high-scoring false-positive overlaps

	Obj. Count	Recall	Precision
Boat	94	0.15	0.82
Car	148	0.23	0.79
Cow	127	0.37	0.92
Motorbike	54	0.31	0.94
Person	240	0.10	0.83
Sheep	141	0.40	0.95
Combined	804	0.24	0.87

Table 5.4: Precision and recall for each object category calculated on the MAP assignment, i.e., on objects detected after convergence of inference. Results were generated using HOG-based detectors on the CCM dataset [Heitz et al., 2008b].

with a smaller true-positive detection). In these experiments we evaluate the trade-off between precision and recall for each object category by generating precision-recall curves based on scores obtained from the top-down proposal moves. These moves are derived from the same set of bounding boxes that are used to generate the baseline results. Specifically, after our inference algorithm converged, we iterated over each top-down object candidate and performed a single max-product message passing step between the object and its neighbors.[6] The score for each move is determined from the beliefs obtained from this single message passing step, and the precision-recall curve plotted by thresholding on this score.

Our results are summarized, in terms of *average precision*, in Table 5.5 with full precision-recall curves shown in Figure 5.9. The curves include the MAP operating point of our algorithm (from Table 5.4) for the HOG-based object detection method. For the integrated patch-based detectors, our method significantly improves over the baseline for all six object classes. However, for the strong HOG-based approach (with post-processed scores from Heitz et al. [2008b]), our approach improves over the baseline for four of the six object classes, and is only marginally better than the CCM results on average. We attribute this to the very strong baseline used, and as we saw in the Street Scenes results, the fact that the HOG detector was specifically engineered to work well on pedestrians, which is our weakest category.

[6]We do this rather than full inference for efficiency. Note, also, that since inference had converged at this stage, none of the proposal moves can result in a decrease in energy.

	Patch-based			HOG-based			
	Baseln.	Unary	Ctxt.	Indep.	Unary	Ctxt.	CCM
Boat	0.19	0.40	0.45	0.31	0.52	**0.53**	0.48
Car	0.23	0.40	0.45	0.55	0.47	0.52	**0.58**
Cow	0.30	0.50	0.55	0.49	0.52	**0.59**	0.53
Motorbike	0.20	0.49	0.58	0.57	**0.70**	**0.70**	0.65
Person	0.20	0.28	0.29	0.53	0.36	0.39	**0.55**
Sheep	0.19	0.53	0.55	0.39	0.53	**0.57**	0.45
Mean	0.22	0.43	0.48	0.47	0.52	**0.55**	0.54

Table 5.5: Object detections results for the CCM dataset [Heitz et al., 2008b]. Performance is measured by area under the precision-recall curve (average precision). The independent and CCM results are taken from Heitz et al. [2008b].

There is a clear advantage in using contextual information in this dataset. This is due to the strong correlation between objects and their environments: Boats are found on water, cars on roads, and sheep on grass. The CCM framework (for the HOG-based detector) also makes use of context by constructing auxiliary object features from other computer vision tasks (multi-class image labeling and scene categorization). However, our unified model allows tighter coupling of context and results in slightly better performance. It also provides object outlines, which the other methods do not.

Some qualitative results are shown in Figure 5.10. The quality of the outlines is outstanding and our method is even able to correctly detect difficult objects such as the small car in the second row and the boat in the last row. Some outlines are not perfect, for example the person walking her dog, and we miss one of the people in the top-left image (a consequence of our model operating at high precision). Other—more dramatic—failure modes are shown in Figure 5.11. These vary from missing objects completely to producing incorrect object labels. Some failures come from misclassification of unknown objects, portions of which may resemble known object categories, such as the bus in Figure 5.11(g). Other errors are caused by incorrect semantic labels at the region level. For example, our model does not handle reflections and shadows well and is confused by them in the image of the car in Figure 5.11(h). Here, the object term has difficulty overcoming the strong beliefs of the region terms. In most cases, the failure modes could be corrected with stronger object models.

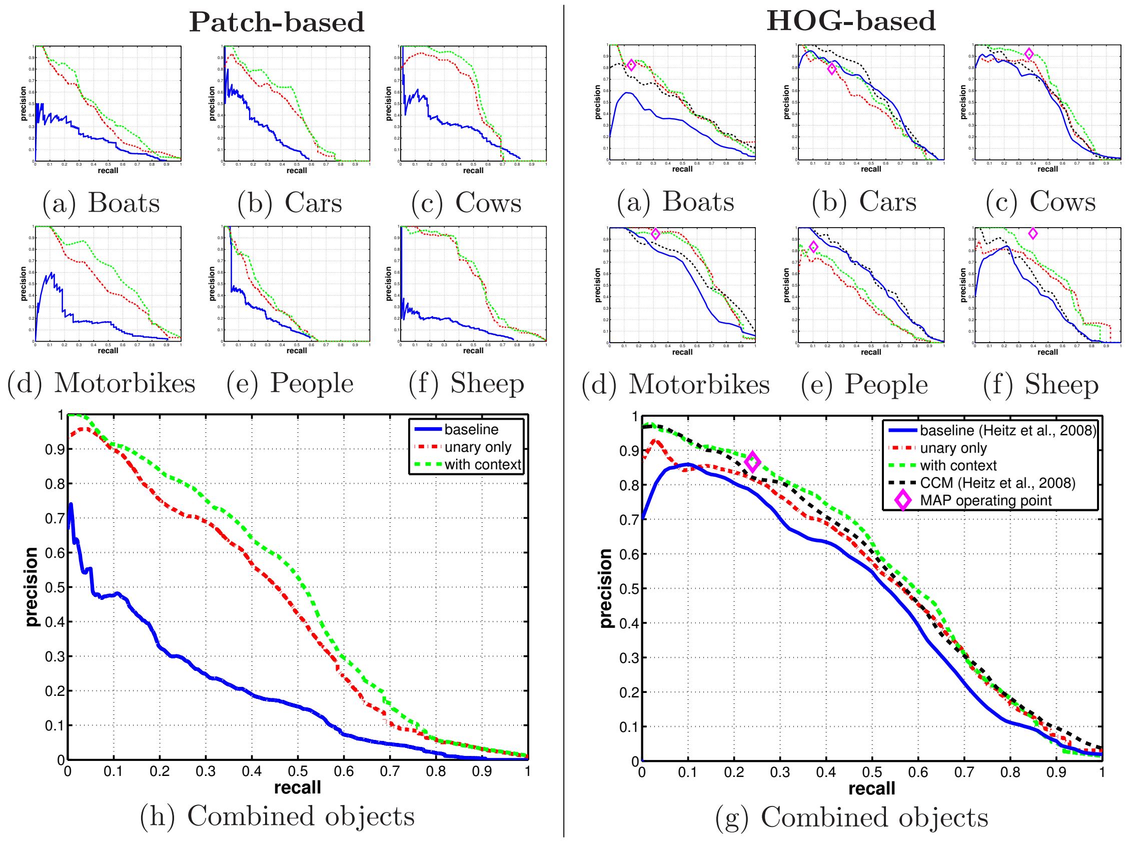


Figure 5.9: Object detection precision-recall curves for patch-based (left) and HOG-based (right) detectors on the CCM dataset [Heitz et al., 2008b]. Plots (a)-(f) show individual PR curve for each object category; plot (g) shows combined curve. The HOG-based plots include the operating point for the MAP assignment indicated by a diamond (⋄).

5.4 Chapter Summary

In this chapter, we have presented a hierarchical model for joint object detection and image segmentation. Our model builds on the scene decomposition model of the previous chapter and overcomes many of the problems associated with trying to combine the related computer vision tasks of object detection and multi-class image labeling. The main contribution of our model is that it explains every pixel and enforces consistency between random variables from the different tasks. We achieve this by representing the scene hierarchically, where pixels are associated with regions, and regions are associated with objects. A major benefit of this representation is that it avoids ad hoc decisions inherent in sliding-window object detectors such as post-processing to remove overlapping detection candidates. Furthermore, our model is encapsulated in a modular energy function that can be easily analyzed and improved as new computer vision technologies become available.

Our method is able to incorporate object features from existing state-of-the-art object detector frameworks through two mechanisms: Either as external black-box algorithms or tightly coupled within our inference procedure. The black-box approach makes it very easy to adapt to the latest algorithms, but limits the object candidates that we can consider. The integrated approach is not limited to predetermined object candidates, but requires additional work in integrating the feature computation and classification methods from the object detectors into our code.

The output of our model is a MAP estimate of the scene structure that includes annotated regions and objects. We believe that this is the desired output of a scene understanding algorithm (unless part of some larger pipeline) since it commits us to a consistent interpretation of the scene. The implication, however, is that our algorithm operates at a single point on the precision-recall curve. This makes comparison to existing approaches for object detection difficult, since the standard performance metric computes area under the precision-recall curve. To get around this difficulty, we compared precision-recall curves based on scoring proposal moves (and also report the single operating point from the MAP output).

Finally, we showed how the greedy inference procedure from Chapter 4 could be extended with top-down proposal moves, which are better suited for finding objects. The top-down moves were generated from preprocessing step that collected candidate bounding boxes produced when running a sliding-window detector over the image. There are many ways in which these bounding boxes can be used to propose object moves, and we discussed two of them. In both approaches, the bounding box is pruned in some way to obtain a better object outline. Other interesting approaches along these lines, which we did not discuss, are to use image editing techniques from the computer graphics community for cropping objects out of images, e.g., [Rother et al., 2004, Lempitsky et al., 2009a]. These techniques could produce very good proposal moves (at a small additional computational cost) when the candidate detections correctly encompass an object.

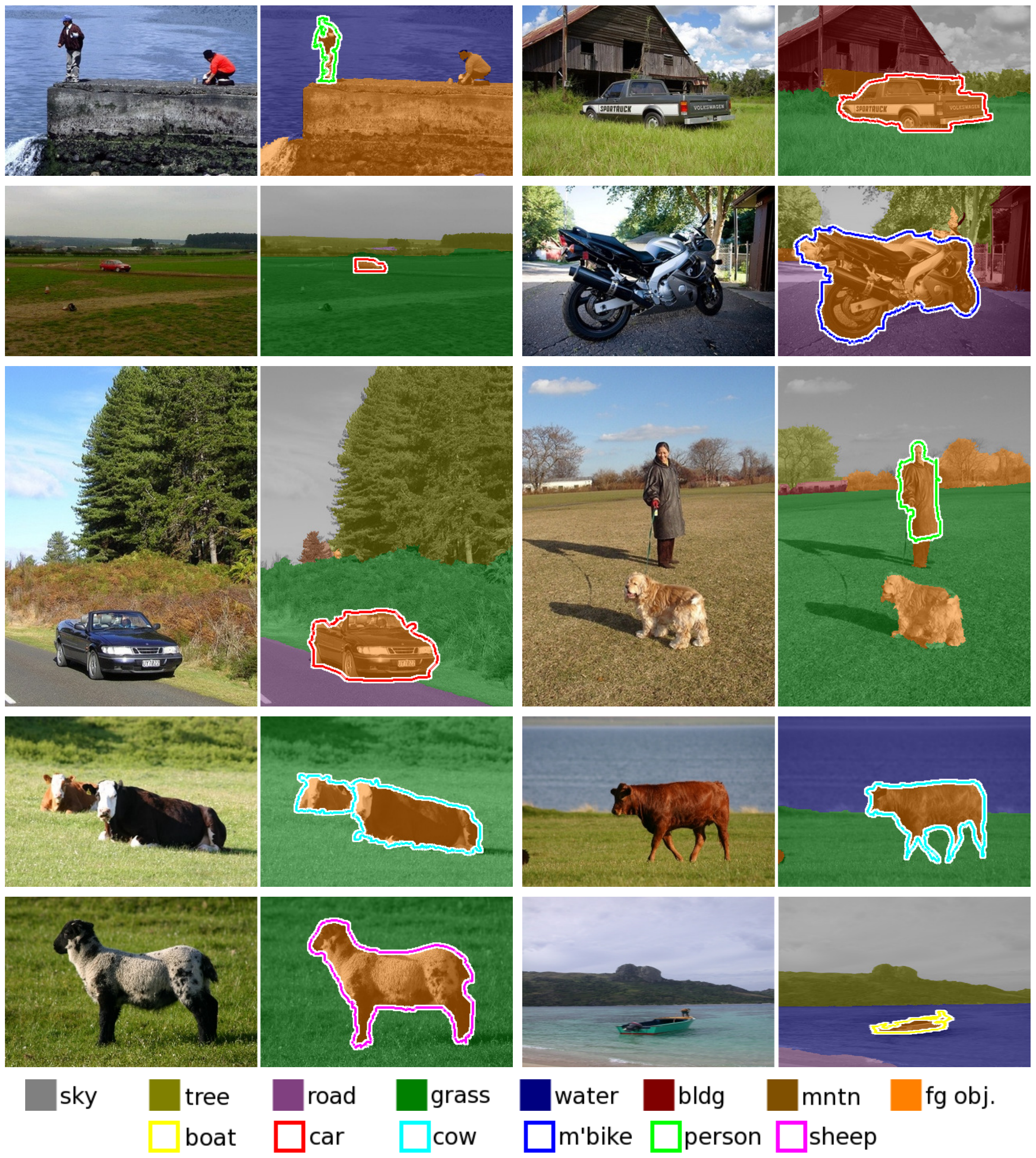


Figure 5.10: Qualitative object detection results for the CCM dataset [Heitz et al., 2008b]. Each example shows the original image (left) and overlay colored by semantic class and detected objects (right).

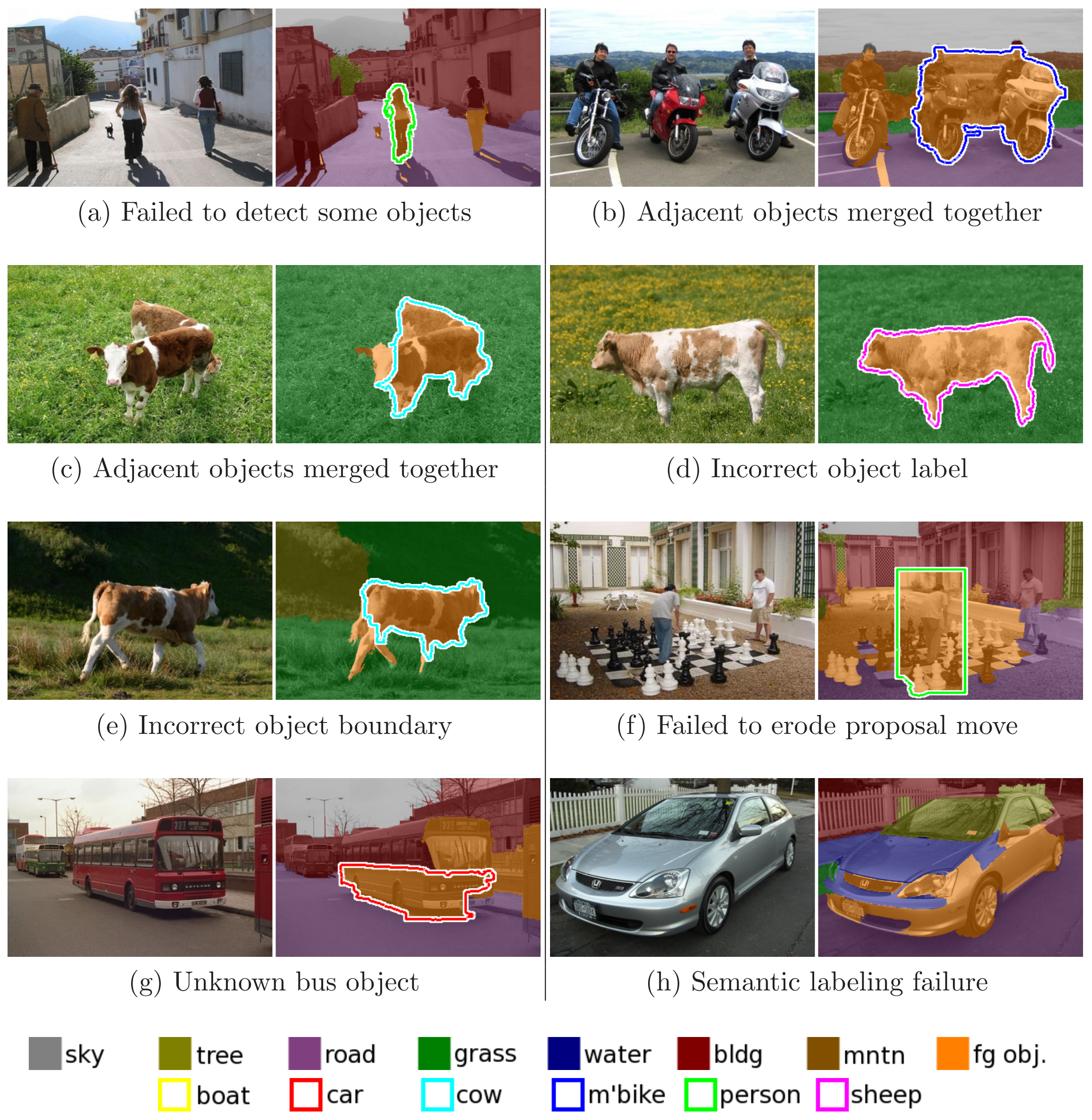

(a) Failed to detect some objects
(b) Adjacent objects merged together
(c) Adjacent objects merged together
(d) Incorrect object label
(e) Incorrect object boundary
(f) Failed to erode proposal move
(g) Unknown bus object
(h) Semantic labeling failure
sky
tree
road
grass
water
bldg
mntn
fg obj.
boat
car
cow
m'bike
person
sheep

Figure 5.11: Failure modes of our scene understanding algorithm. Each example shows the original image (left) and overlay colored by semantic class and detected objects (right). For example, (a) shows two missed person detections; (d) shows a cow incorrectly classified as a sheep; and (h) shows the hood of a car incorrectly labeled as water.

Chapter 6

Scene Reconstruction: From Semantics to Geometry

Recovering the 3D structure of a scene from a single image is a fundamental problem in computer vision that has application in robotics, surveillance and general scene understanding—if we can estimate scene structure then we can better understand the scene by knowing the 3D relationships between the objects within it. However, estimating structure from raw image features is extremely difficult since local appearance is insufficient to resolve depth ambiguities (e.g., sky and water regions in an image can have similar appearance but dramatically different geometric placement within the scene). In this chapter, we demonstrate how semantic understanding of a scene, such as that obtained from our region-based scene understanding framework, can be used to better interpret the 3D structure of a scene, resulting in a complete understanding of regions, objects and geometry.

Producing spatially plausible 3D reconstructions of a scene from monocular images annotated with geometric cues, such as horizon location, vanishing points, and region/surface boundaries, is a well understood problem [Criminisi et al., 2000]. In the first part of this chapter, we show how our region-based decomposition of the scene provides such geometric cues automatically and can be used to generate novel 3D views of the scene. Our approach is very simple and obtains its power from our region-based decomposition rather than sophisticated features tuned for the task.

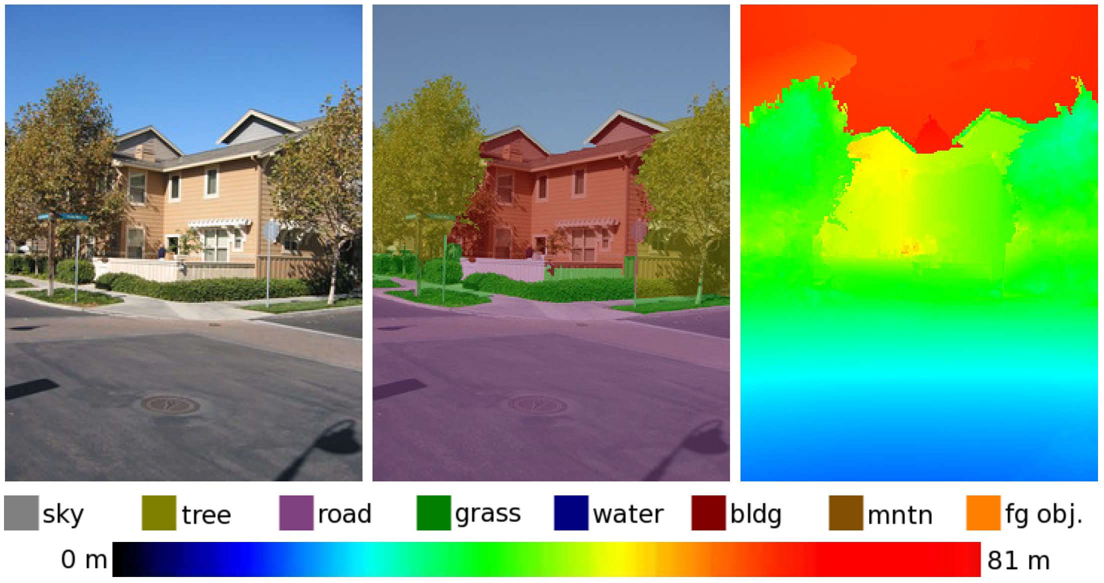


Figure 6.1: Example output from our semantically-aware depth estimation model showing how semantic class prediction (center) strongly informs depth perception (right). Semantic classes are shown overlayed on image. Depth indicated by color map (red is more distant).

Nevertheless, the results from our approach are surprisingly good. Since our model does not provide true depth estimates our goal here is to produce planar geometric reconstructions of each region with accurate relative distances rather than absolute distance. Given an estimate of the distance between any two points in the scene, our 3D reconstruction can then be scaled to the appropriate size.

Determining absolute (rather than relative) depths from monocular images is a notoriously difficult task—even for humans—and has been the subject of much recent work [Saxena et al., 2008, Hoiem et al., 2007a, Delage et al., 2006, Hoiem et al., 2005a, Saxena et al., 2005]. In order to produce metric (i.e., depth-accurate) reconstructions one needs information such as camera parameters (pose and focal length) and estimates of real-world distances in addition to pure geometric cues such as region boundaries. The main thrust of these recent works has been on how to extract this additional information (either explicitly or implicitly) from raw image features such as texture gradients or height of known objects. In the second part of this chapter, we build on our intuition from the first and show how our reasoning about the semantic content of a scene can be used as context for depth estimation. Figure 6.1 shows example output from our scene decomposition and depth estimation models. It is clear that each of the semantic regions in the scene should be treated differently when it comes to estimating depth.

The incorporation of semantic class knowledge allows us to do two things: First, we can take advantage of class-related depth and geometry priors. For example, sky is always at the farthest depth possible; grass and road form supporting ground planes for other objects. Second, by conditioning on semantic class, we can better model depth as a function of local pixel appearance. For example, uniformity of texture may be a good indicator for the depth of a tree, but not useful when estimating the depth of a building. Our model is therefore able to use much simpler image features than previous approaches while still achieving state-of-the-art depth perception results.

We adopt a two-phase approach to depth estimation. In the first phase, we perform a semantic decomposition of the scene using our region-based model from Chapter 4. This provides us with a semantic label for each pixel as well as region boundary information. As we have seen, the seven background semantic class labels (*sky*, *tree*, *road*, *grass*, *water*, *building*, and *mountain*) cover a large portion of regions in outdoor scenes, while the *foreground* class captures the eclectic set of foreground objects such as cars, street signs, people, animals, etc., which "pop-up" from the ground plane. In this chapter, we do not refine the *foreground* class into specific object categories.

In the second phase of our approach, we use the predicted semantic class labels and region boundaries to inform our depth reconstruction model. Here, we first learn a separate depth estimator for each semantic class. We incorporate these predictions into a Markov random field (MRF) model that includes semantic-aware reconstruction priors such as smoothness and orientation of different semantic classes, and builds on the geometric intuition developed in the first part of the chapter. However, in the new model, all the constraints are soft and are learned from data. In addition, we are predicting true depths.

We compare our model to other published results on the Make3d dataset [Saxena et al., 2005, 2008] for which ground truth depth data is publicly available. Our experiments show that our model outperforms state-of-the-art approaches and yields qualitatively excellent reconstructions.

6.1 Geometric Reconstruction

We start with a simple geometric model that allows us to generate novel 3D views of a scene from the output of the scene decomposition model presented in Chapter 4. Since our model does not provide true depth estimates, our goal here is to produce planar geometric reconstructions of each region with accurate relative distances rather than absolute distance. Later, in Section 6.2, we will develop a model that makes use of semantic information for true depth estimation.

Recall that the output of our scene decomposition model is a set of regions, each annotated with a semantic (and rough geometric) label. In addition, our model also predicts the location of the horizon v^{hz}. Our rules for reconstruction are simple: First, we assume an ideal camera model with horizontal (x) axis parallel to the ground. We fix the camera origin at 1.8m above the ground (i.e., $y = 1.8$). We then estimate the yz-rotation of the camera from the location of the horizon (assumed to be at depth ∞) as

$$\theta = \tan^{-1}\left(\frac{v^{\text{hz}} - v_0}{f}\right) \tag{6.1}$$

where v_0 is half the image height and f is the focal length of the camera. This is illustrated for the simple scene in Figure 6.2. The above formula can be verified trivially from the geometry. Note that if the horizon is located at the image center, i.e., $v^{\text{hz}} = v_0$, then the camera's imaging plane will be perpendicular to the ground, and when the horizon is above the image center, the image plane points downwards.

As discussed in Chapter 2, a pixel p with coordinates (u_p, v_p) (in the camera plane) is the image of a point in 3D space that lies on the ray extending from the camera origin through (u_p, v_p) in the camera plane. The ray $\mathbf{r}_p$ in the world coordinate system

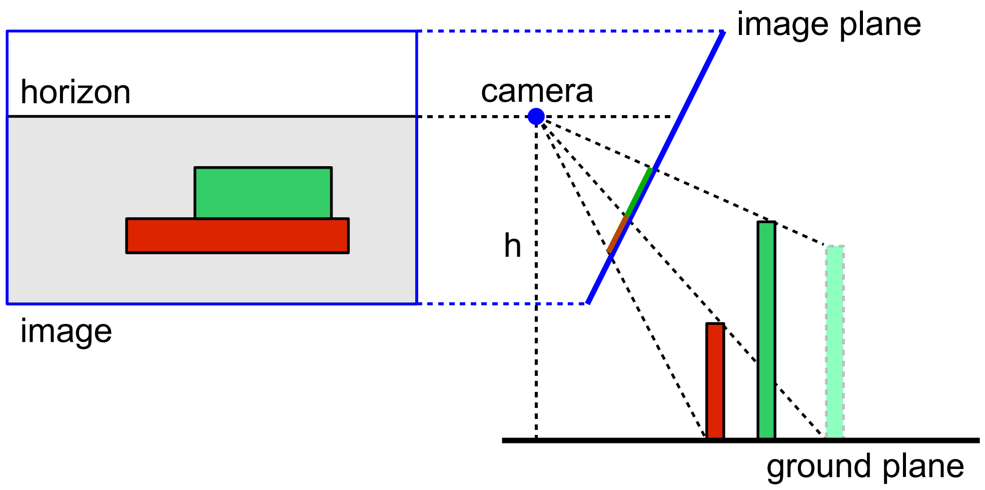


Figure 6.2: Schematic illustrating geometric reconstruction of a simple scene.

is given by

$$\mathbf{r}_p \propto \boldsymbol{R}(\theta)^{-1}\boldsymbol{K}^{-1}\begin{bmatrix} u_p \\ v_p \\ 1 \end{bmatrix} \tag{6.2}$$

where $\boldsymbol{R}(\theta) \in SO(3)$ defines the transformation (rotation) from camera coordinate system to world coordinate system, and $\boldsymbol{K} \in \mathbb{R}^{3\times3}$ is the camera matrix [Ma et al., 2005].[1] The actual 3D point in the world coordinate system is obtained by scaling the ray by the distance to the point, i.e., $(x_p, y_p, z_p) = d_p\mathbf{r}_p$.

Consider again the simple scene shown in Figure 6.2. The scene contains two objects (a red box and a green box) resting on a ground plane. Since we can see the point of contact between the red box and the ground, we can compute the distance to the object by extending the ray $\mathbf{r}_p$ from the camera origin through a pixel p at the base of the object in the image plane until it intersects with the ground plane.

[1]In our model, we assume that there is no translation between the world coordinate system and the camera coordinate system, implying that the images were taken from approximately the same height above the ground. Furthermore, as in Saxena et al. [2008] we assume a reasonable value for the focal length (in our experiments we set $f_u = f_v = 348$ for a 240×320 image).

Specifically, for a pixel p at the base of the red box we have, for the distance to the pixel,

$$d_p = -\left(\frac{\mathbf{r}_p^T e_3}{\mathbf{r}_p^T e_2}\right) h \tag{6.3}$$

where h is the height of the camera above the ground, and e_i is the i-th canonical vector (i.e., vector with i-th element one and the rest zero). Assuming that the red box is vertical, this gives the distance to the object. That is, all pixels in the object above this base pixel will be at the same depth.

The green box is more difficult since we cannot see the point at which the object makes contact with the ground. It appears to be occluded by the red box. However, it could also be either sitting on top of the red box, or at a point where we can see the base over the top of the red box, but not the ground immediately below it. This gives us a range of distances for the green box, and a reasonable guess (not having any other information) would be to place the object half way between its minimum and maximum possible depths.

Using these simple insights, we can make use of our scene decomposition to produce realistic geometric reconstructions: We process each region in the image depending on its semantic class. For ground plane regions (road, grass and water) we scale the ray at each point to make the height zero (see Equation 6.3). We model each vertical region (tree, building, mountain and foreground) as a planar surface whose orientation and depth with respect to the camera are estimated by fitting a robust line over the pixels along its boundary with the ground plane. This produced qualitatively good results despite the fact that not all of these pixels are actually adjacent to the ground in 3D (such as the belly of the cow in Figure 6.3). We can then pop-up these regions as we did for the red box in Figure 6.2.

When a region does not touch the ground (that is, it is occluded by another object), we estimate its orientation using pixels on its bottom-most boundary. We then place the region half way between the depth of the occluding object and maximum possible depth (either the horizon or the point at which the ground plane would become visible beyond the occluding object). The 3D location of each pixel p in the region

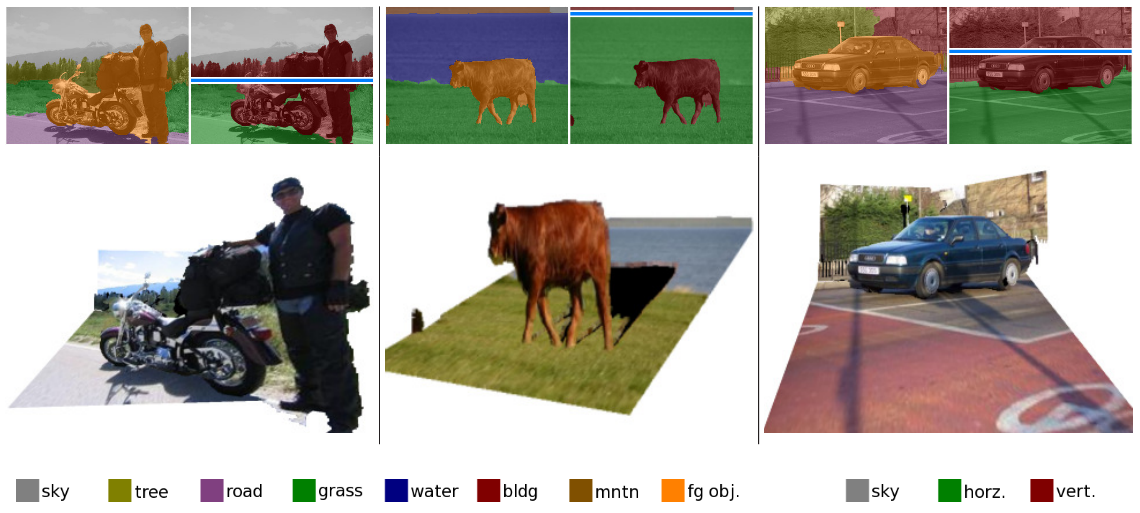


Figure 6.3: Novel views of a scene with foreground objects generated by geometric reconstruction. Shown above each novel view are the semantic and geometric class overlays from the model described in Chapter 4.

is determined by the intersection of the ray $\mathbf{r}_p$ with this plane. The placement of the green box in Figure 6.2 is an example of this type of reasoning. Finally, sky regions are placed behind the last vertical region.[2]

Some example 3D views for scenes in our dataset are shown in Figure 6.3. These examples show that semantic information together with an understanding of image formation can lead to very good geometric reconstructions. In the following section we go one step further and present a model that uses predicted semantic classes to guide estimation of real-world distances.

6.2 Depth Estimation

The previous section showed how we could make use of the semantic labels and region boundaries produced by our scene decomposition algorithm to generate plausible geometric reconstructions of a scene. We will now turn our attention to the more difficult task of depth estimation. Here, the goal is to predict the depth of each

[2]Technically, sky regions should be placed at the horizon, but since the horizon has infinite depth, we choose to render sky regions closer, so as to make them visible in our viewer.

pixel in the image rather than construct a geometric "pop up". To be consistent with other published works (e.g., [Saxena et al., 2008]), we use the term *depth* to mean the distance from the camera origin to a point in the world coordinate system, i.e., $\sqrt{x^2 + y^2 + z^2}$. When referring to the distance into the scene (i.e., along the z-coordinate) we will use the term *z-value*.

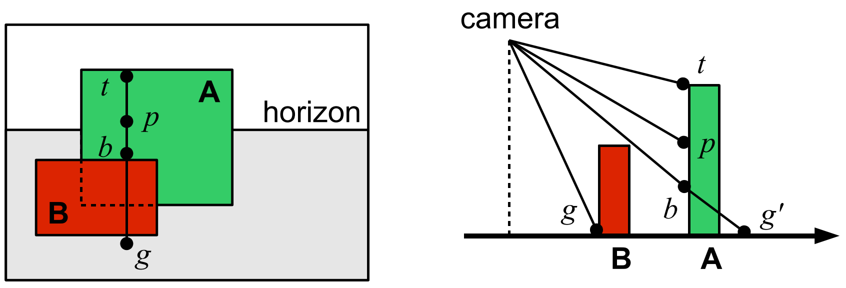
Our region-based scene decomposition model from Chapter 4 provides us with the location of the horizon v^{hz} in the scene, and a set of regions $\{\mathcal{P}_r\}$. Each region has a semantic class label S_r, and the region to which a pixel belongs is denoted by its region-association variable R_p. In this section we will make use of both the semantic labels for each pixel and geometric cues based on the region in which the pixel belongs.

To demonstrate the importance of region-based semantics rather than just pixelwise semantics we also compare our results against semantics derived from the pixelwise baseline developed in Chapter 4. Note, however, that any multi-class image labeling method (e.g., [Shotton et al., 2006, He et al., 2004, Gould et al., 2009b]) could be used for this purpose. Here, we will define regions as connected components with the same semantic class label (i.e., a region boundary exists between two pixels of different semantic class).

We also require an estimate for the location of the horizon, which is not produced by standard multi-class image labeling methods. When using the pixel baseline, we adapt the ideas from Chapter 4 to generate an estimate from the (pixelwise) semantic labeling itself. As in Chapter 4, we assume that the horizon v^{hz} can be modeled as a row within the image and define a prediction model as

$$E\left(v^{\text{hz}} \mid \mathcal{I}, \boldsymbol{S}\right) = -\log \mathcal{N}(v^{\text{hz}}; \mu, \sigma) + \sum_p \psi_p(v^{\text{hz}}; S_p) \tag{6.4}$$

where $\mathcal{N}(v^{\text{hz}}; \mu, \sigma)$ is a normal distribution reflecting our prior estimate of the horizon location and $\psi_p(v^{\text{hz}}; S_p)$ penalizes inconsistent relative location between the horizon and pixels with given semantic class (e.g., ground plane pixels should be below the horizon and sky pixels above). The location of the horizon is determined as the minimizing assignment to v^{hz}.



(a) Image view (b) Side view

Figure 6.4: Illustration of semantically derived geometric constraints.

6.2.1 Encoding Geometric Constraints

The geometric reconstruction model in Section 6.1 showed how geometric reasoning could be used to position vertical planes in 3D space by knowing where they contact the ground. We now relax the planarity and verticality assumption on regions, but use the same reasoning developed in Section 6.1 to impose soft constraints on pixel depths. As described, the point in 3D space corresponding to a pixel p, with coordinates (u_p, v_p) in the image plane, can be written as $(x_p, y_p, z_p) = d_p\mathbf{r}_p$, where $\mathbf{r}_p$ is the unit ray extending through pixel p into the world coordinate system and d_p is the distance to the point.

With pixels grouped into regions by our scene decomposition model, we are now ready to encode constraints about the geometry of a scene. Consider, for example, the simple scene in Figure 6.4, and assume that we would like to estimate the depth of a pixel p located somewhere on a vertical object (or region) **A** attached to the ground. We define three key points that are strong indicators of the depth of pixel p. First, let g be the topmost ground pixel in the same column below p. As we saw in the previous section, the z-value of g is a lower bound on the z-value of p. Second, let b be the bottommost visible pixel b on the object **A** (and in the same column as p). By extending the camera ray through b to the ground, we can calculate an upper bound on the z-value of p. Third, the topmost point t on the object (and, again, in the same column as p) may also be useful since a non-sky pixel high in the image (e.g., an overhanging tree) tends to be close to the camera.

Simple geometric reasoning allows us to encode the first two constraints as

$$d_g \left(\frac{\mathbf{r}_g^T e_3}{\mathbf{r}_p^T e_3} \right) \;\leq\; d_p \;\leq\; d_g \left(\frac{\mathbf{r}_g^T e_2}{\mathbf{r}_b^T e_2} \right) \left(\frac{\mathbf{r}_b^T e_3}{\mathbf{r}_p^T e_3} \right) \tag{6.5}$$

where d_p and d_g are the distances to the points p and g, respectively, and e_i is the i-th canonical vector (i.e., vector with i-th element one and the rest zero). The point $d_g \left(\frac{\mathbf{r}_g^T e_2}{\mathbf{r}_b^T e_2} \right) \mathbf{r}_b$ is the point at which the ray through b touches the ground plane, and is denoted by g' in Figure 6.4. The third constraint can be similarly encoded as $d_t \mathbf{r}_t^T e_3 \approx d_p \mathbf{r}_p^T e_3$.

In the following sections we show how these constraints are incorporated as features and potential terms in our depth perception MRF model.

6.2.2 Features and Pointwise Depth Estimation

Our goal is to predict the depth to every pixel in the image. We begin by constructing a descriptor $f_p \in \mathbb{R}^n$ for each pixel, which includes local appearance features and global geometry features derived from our semantic understanding of the scene. Standard depth perception models utilize the fact that the local appearance of a pixel changes with depth and attempt to learn a function that describes this relationship. However, this relationship depends strongly on the semantic class (e.g., a distant tree will appear less textured than a nearby one, but this is not necessarily so for more uniform classes such as building or road). Furthermore, the distance of some classes is tightly constrained by the class itself (e.g., sky). Accordingly, we learn a different local depth predictor for each semantic class.

Motivated by the desire to more accurately model the depth of nearby objects and the fact that relative depth is more appropriate for scene understanding, we learn a model to predict log-depth rather than depth itself. We thus estimate pointwise log-depth as a linear function of the pixel features (given the pixel's semantic class),

$$\log \hat{d}_p = \theta_{S_p}^T f_p \tag{6.6}$$

where $\hat{d}_p$ is the pointwise estimated depth for pixel p, S_p is its semantic class label

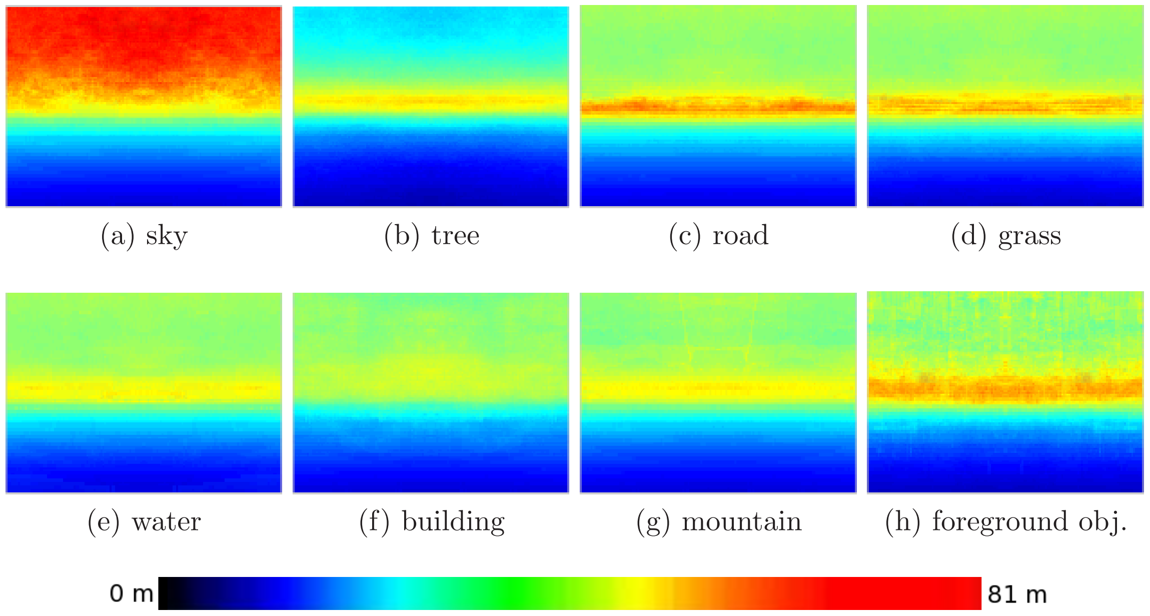


Figure 6.5: Smoothed per-pixel log-depth prior for each semantic class calculated on the Make3d dataset [Saxena et al., 2005]. Images are shifted so that the horizon appears at the center. Colors indicate distance (red is further away and blue is closer). The classes "water" and "mountain" had very few samples and so are close to the global log-depth prior (not shown).

predicted from our scene decomposition model, $f_p \in \mathbb{R}^n$ is a local feature vector, and $\{\theta_s\}_{s \in \mathcal{S}}$ are the learned parameters of the model—one parameter vector for each semantic class label.

Our basic pixel appearance features are the 17 raw filter responses used in the semantic model and also the log of these features. We also include the (u, v) coordinates of the pixel and an a priori estimated log-depth for a pixel with coordinates (u, v) and semantic label S_p. These are all adjusted to a consistent coordinate system (i.e., shifted so that the horizon lies at the center row of the image). The prior log-depth is learned, for each semantic class, by averaging the log-depth at a particular (u, v)-pixel location over the set of training images. Since not all semantic class labels appear at all pixel locations, we smooth the priors with a global log-depth prior (the average log-depth over all the classes at the particular location). We also enforce symmetry about the vertical axis by averaging with a mirror-image of the prior.

Figure 6.5 illustrates these priors for the eight semantic classes in our model. It is interesting to note the differences between the semantic classes. A tree pixel towards the top of the image, for example, is likely to be closer than a tree pixel near the horizon (center of the image). More important, any pixel within a tree region that extends towards the top of the image, is likely to be closer than a pixel in a tree region that does not extend much beyond the horizon. This supports the observation made in Section 6.2.1 that the location of the topmost pixel in a region may be a more useful (a priori) indicator for region depth than an arbitrary pixel within the region.

We encode additional geometric constraints as features by examining the three key pixels discussed in Section 6.2.1. For each of these pixels (bottommost and topmost pixel within the same region R_p and topmost ground pixel, all in the same column as p), we use the pixel's prior log-depth to calculate a depth estimate for p (assuming that most objects are roughly vertical) and include this estimate as a feature. We also include the (horizon-adjusted) vertical coordinate of these pixels as features. Note that, unlike Section 6.1, our verticality assumption is not a hard constraint, but rather a soft one that can be overridden by the strength of other features.

As discussed previously, these key pixels are motivated by the observation that most foreground objects are vertical and therefore pixels on the object within the same column are likely to be at the same depth. Furthermore, the topmost ground plane pixel constrains the minimum depth of the object, and a non-sky pixel high in the image is likely to be closer than one lower in the image. By including these as features, we allow our model to learn the strength of these a priori constraints.

Finally, we add the square of each feature, allowing us to model quadratic correlations between feature values and depth.

We learn the parameters of the model $\{\theta_s\}_{s \in \mathcal{S}}$ by linear regression on the log-depths from a set of training images (see Section 2.1). For numerical stability, we first normalize each feature to zero mean and unit variance. We also include a small regularization term.

6.2.3 MRF Model for Depth Reconstruction

The pointwise depth estimation provided by Equation 6.6 is somewhat noisy and can be improved by including priors that constrain the structure of the scene. We develop a continuous pixelwise Markov random field (MRF) model to encode such priors. The priors constrain the depth relationships between two or more pixels, weighted by the contrast between the pixels to allow for depth discontinuities at strong boundaries.

Our MRF model incorporates terms for encoding three preferences. First, we would like our predicted depths to be close to the pointwise depth estimates derived above. Second, we would like to constrain regions to be roughly planar. This amounts to adding a potential term over three consecutive pixels (in the same row or column) that prefers co-linearity. Third, we encode semantically-derived depth constraints which penalize vertical surfaces from deviating from geometrically plausible depths (as described in Section 6.2.1). Formally, we define the energy function over pixel depths $\boldsymbol{D}$ as

$$\begin{aligned} E\left(\boldsymbol{D} \mid \mathcal{I}, \boldsymbol{S}\right) = &\underbrace{\sum_{p} \psi_p(d_p)}_{\text{data term}} + \underbrace{\sum_{pqr} \psi_{pqr}(d_p, d_q, d_r)}_{\text{smoothness}} \\ &+ \underbrace{\sum_{p} \psi_{pg}(d_p, d_g) + \sum_{p} \psi_{pb}(d_p, d_g) + \sum_{p} \psi_{pt}(d_p, d_t)}_{\text{geometry (see §6.2.1)}} \end{aligned} \tag{6.7}$$

where the data term, ψ_p, attempts to match the depth for each pixel d_p to the pointwise estimate $\hat{d}_p$, and ψ_{pqr} represents the co-linearity prior. The terms ψ_{pg}, ψ_{pb} and ψ_{pt} represent the geometry constraints described in Section 6.2.1. Recall that the pixel indices g, b and t are determined from p and the region to which it belongs, i.e., they are functions of p.

The data term in our model is given by

$$\psi_p(d_p) = h(d_p - \hat{d}_p; \beta) \tag{6.8}$$

where $h(x; \beta)$ is the Huber penalty, which takes the value x^2 for $-\beta \leq x \leq \beta$ and

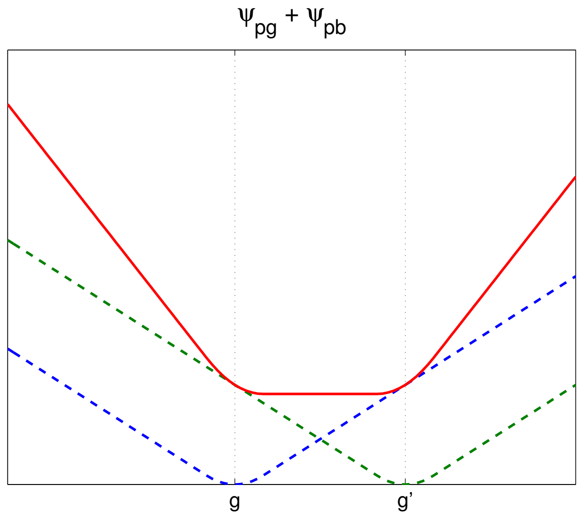


Figure 6.6: Graph showing the effect of using the Huber penalty for constraining the depth range for a pixel p to be between two reference depths g and g'.

$\beta(2|x| - \beta)$ otherwise. We choose the Huber penalty because it is more robust to outliers than the more commonly used ℓ_2-penalty and, unlike the robust ℓ_1-penalty, is continuously differentiable (which simplifies inference). In our model, we set $\beta = 10^{-3}$.

Our smoothness prior encodes a preference for co-linearity of adjacent pixels within uniform regions. Assuming pixels p, q, and r are three consecutive pixels (in any row or column), we have

$$\psi_{pqr} = \lambda^{\text{smooth}} \cdot \sqrt{\gamma_{pq}\gamma_{qr}} \cdot h(2d_q - d_p - d_r; \beta) \tag{6.9}$$

where the smoothness penalty is weighted by a contrast-dependent term and the prior strength λ^{smooth}. Here, $\gamma_{pq} = \exp\left(-c^{-1}\|\alpha_p - \alpha_q\|^2\right)$ measures the contrast between two adjacent pixels, where α_p and α_q are the CIELab color vectors for pixels p and q, respectively, and c is the mean square-difference over all adjacent pixels in the image. Note that this is the same pixel-wise contrast term used by the scene decomposition model of Chapter 4. We choose the prior strength by cross-validation on a set of training images.

The soft geometry constraints ψ_{pg}, ψ_{pt} and ψ_{pb} model our prior belief that certain semantic classes are vertically oriented (e.g., buildings, trees and foreground objects). Here, we impose the soft constraint that a pixel within such a region should be the same depth as other pixels above and below it in the region (i.e., by linking to the topmost and bottommost pixels in the region), and be between the nearest and farthest ground plane points g and g' defined in Section 6.2.1. The constraints are encoded using Huber penalties. For example, the latter constraint can be encoded as

$$h\left(d_p - d_g\left(\frac{\mathbf{r}_g^T e_3}{\mathbf{r}_p^T e_3}\right); \beta\right) + h\left(d_p - d_g\left(\frac{\mathbf{r}_g^T e_2}{\mathbf{r}_b^T e_2}\right)\left(\frac{\mathbf{r}_b^T e_3}{\mathbf{r}_p^T e_3}\right); \beta\right). \tag{6.10}$$

Figure 6.6 provides a visualization of this term, clearly showing that the sum of the two Huber penalties has the desired effect by putting a preference on p lying between g and g'. In our energy function, we multiply each term by a semantic-specific prior weight $\{\lambda_s^{\mathrm{g}}, \lambda_s^{\mathrm{t}}, \lambda_s^{\mathrm{b}}\}_{s\in\mathcal{S}}$. The final form of each term is

$$\psi_{pg}(d_p, d_g) = \lambda_{S_p}^{\mathrm{g}} h\left(d_p - d_g\left(\frac{\mathbf{r}_g^T e_3}{\mathbf{r}_p^T e_3}\right); \beta\right) \tag{6.11}$$

$$\psi_{pt}(d_p, d_t) = \lambda_{S_p}^{\mathrm{t}} h\left(d_p - d_t\left(\frac{\mathbf{r}_t^T e_3}{\mathbf{r}_p^T e_3}\right); \beta\right) \tag{6.12}$$

$$\psi_{pb}(d_p, d_g) = \lambda_{S_p}^{\mathrm{b}} h\left(d_p - d_g\left(\frac{\mathbf{r}_g^T e_2}{\mathbf{r}_b^T e_2}\right)\left(\frac{\mathbf{r}_b^T e_3}{\mathbf{r}_p^T e_3}\right); \beta\right). \tag{6.13}$$

These weights $\{\lambda_s^{\mathrm{g}}, \lambda_s^{\mathrm{t}}, \lambda_s^{\mathrm{b}}\}_{s\in\mathcal{S}}$ are learned by cross-validation on the training images.

Our Markov random field formulation in Equation 6.7 defines a convex objective, which we solve using the L-BFGS algorithm [Nocedal, 1980, Liu and Nocedal, 1989] to obtain a depth prediction for every pixel in the image. In our experiments, inference takes about 2 minutes per image (of size 240×320).

6.3 Experiments

We tested our depth estimation mode by running experiments on the publicly available Make3d dataset from Saxena et al. [2005]. The dataset consists of 534 images with corresponding depth maps and is divided into 400 training and 134 testing images. We use Amazon Mechanical Turk (see Chapter 3) to hand-annotate the images with semantic class labels. The 400 training images were used for learning the parameters of the semantic and depth models. All images were resized to 240×320 before running our algorithm.

We report results on the 134 test images. Since the maximum range of the sensor used to collect ground truth measurements was 81m, we truncate our depth predictions to the range $[0, 81]$. Table 6.1 shows our results compared against previous published results. As is standard on this dataset, we compare both the average log-error and average relative error, defined as

$$|\log_{10} g_p - \log_{10} d_p| \quad \text{and} \quad \frac{|g_p - d_p|}{g_p}, \tag{6.14}$$

respectively, where g_p is the ground truth depth for pixel p. We also compare our results to our own baseline implementation which does not use any semantic information.

Method	$\log_{10}$	Rel.
SCN [Saxena et al., 2005]†	0.198	0.530
HEH [Hoiem et al., 2005a]†	0.320	1.423
Pointwise MRF [Saxena et al., 2008]†	0.149	0.458
PP-MRF [Saxena et al., 2008]†	0.187	**0.370**
Baseline (no semantics)	0.208	0.469
Pixelwise semantics	0.149	0.375
Region-based semantics	**0.145**	0.372

† Results reported in Saxena et al. [2008].

Table 6.1: Quantitative results comparing variants of our "semantic-aware" approach with strong baselines and other state-of-the-art methods. Baseline models do not use semantic class information.

We compare three variants of our model. The **Baseline** variant does not include any semantic information so provides a measure on how well our raw features predict depth. The second variant, **Pixelwise semantics**, includes semantic information, but not region information. As discussed above, this model derives region boundaries (for finding topmost and bottommost pixels t and b) from semantic class boundaries. The last variant, **Region-based semantics**, includes region information from our scene decomposition model.

Both our semantic-aware models achieve state-of-the-art performance for the $\log_{10}$ metric and comparable performance to state-of-the-art for the relative error metric. Importantly, they achieve good results on both metrics unlike the previous methods which perform well at either one or the other. This can be clearly seen in Figure 6.7 where we have plotted the performance metrics on the same graph.

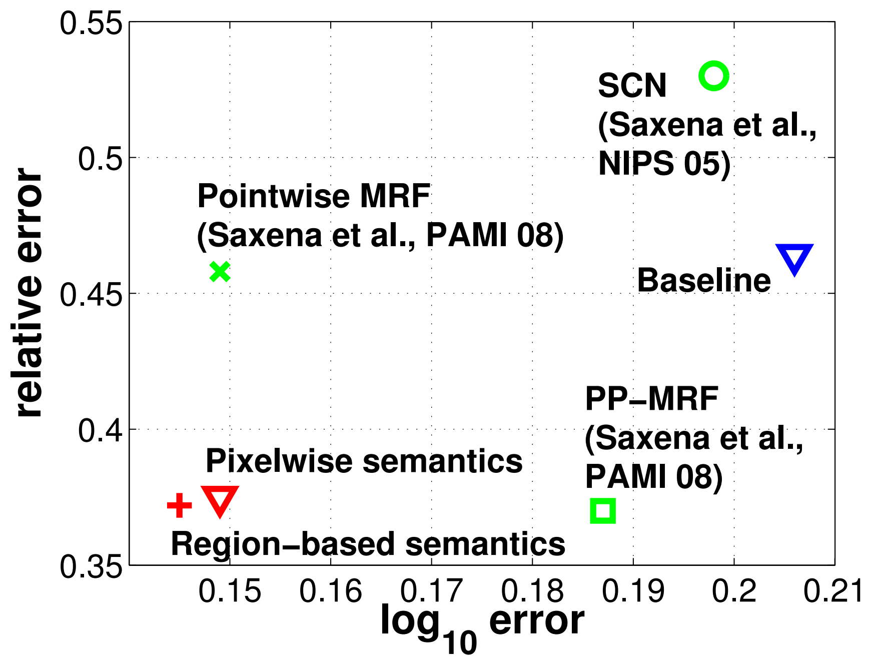


Figure 6.7: Plot of log_{10} error metric versus relative error metric comparing algorithms from Table 6.1 (HEH [Hoiem et al., 2005a] not shown). Bottom-left indicates better performance.

Having semantic class labels allows us to break down our results by (predicted) class as shown in Tables 6.2 and 6.3. Our best performing results are the ground plane classes (especially road), which are easily identified by our semantic model and tightly constrained geometrically. We achieve poor performance on the foreground class, which we attribute to the lack of foreground objects in the training set (less

	$\log_{10}$ Error			
	All Pixels	Correct	Incorrect	Baseline
Sky	0.155	0.118	0.304	0.434
Tree	0.207	0.208	0.197	0.232
Road	0.058	0.052	0.141	0.079
Grass	0.098	0.084	0.140	0.139
Water	0.089	-	0.089	0.167
Building	0.189	0.178	0.241	0.288
Mountain	0.353	-	0.353	0.270
Foreground	0.223	0.318	0.209	0.245
All Classes	**0.145**	**0.133**	**0.207**	**0.208**

Table 6.2: Quantitative results by predicted semantic class ($\log_{10}$ error).

	Relative Error			
	All Pixels	Correct	Incorrect	Baseline
Sky	0.612	0.492	1.095	0.678
Tree	0.504	0.517	0.423	0.587
Road	0.132	0.111	0.402	0.202
Grass	0.233	0.198	0.337	0.273
Water	0.222	-	0.222	0.473
Building	0.442	0.360	0.826	0.665
Mountain	0.860	-	0.860	0.387
Foreground	0.584	1.657	0.424	0.609
All Classes	**0.372**	**0.334**	**0.567**	**0.469**

Table 6.3: Quantitative results by predicted semantic class (relative error).

than 1% of the pixels). Likewise, the small number of water and mountain pixels limit the conclusions we can draw from these classes.

Unexpectedly, we also perform poorly on sky pixels which are easy to predict and should always be positioned at the maximum depth. This is due, in part, to errors in the ground truth measurements (caused by sensor misalignment) and the occasional misclassification of the reflective surfaces of buildings as sky by our semantic model. Note that the nature of the relative error metric is to magnify these mistakes since the ground truth measurement in these cases is always closer than the maximum depth.

Given our heavy reliance on inferred semantic evidence at the depth estimation stage, it is also important to consider the robustness of our approach to errors in the semantic labeling. Quantitatively, about 16% of pixels were classified with the wrong semantic class. Tables 6.2 and 6.3 include results calculated on the subset of pixels that were either correctly or incorrectly classified. As expected, the depth estimation accuracy is better on correctly classified pixels than those misclassified. Surprisingly, the depth estimation accuracy over the incorrect pixels was comparable to the baseline model (0.207 versus 0.208 for $\log_{10}$ error and 0.567 versus 0.469 for relative error) showing that the depth-estimation stage of our approach is able to partially overcome mistakes made in the semantic classification stage.

Finally, we show some qualitative results in Figure 6.8 and example 3D reconstructions in Figure 6.9. The results show that we correctly model co-planarity of the ground plane and building surfaces. Notice our accurate prediction of the sky (which is sometimes penalized by misalignment in the ground truth, e.g., the example in the bottom row). Our algorithm also makes mistakes, such as positioning the building too close in the second row and missing the ledge in the foreground (a mistake that many humans would also make). We also miss some low contrast objects such as the post in the example at the top of the figure. However, in general, we produce very accurate results.

6.4 Chapter Summary

In this chapter we presented two models for 3D reconstruction of the scene. Both models make heavy use of predicted semantic information. The first model used basic geometric reasoning over the region output of our scene decomposition model to produce 3D "pop ups" of the scene. The second model predicts real-world depths guided by the semantic labels for each pixel. The inclusion of semantic information allowed us to model appearance and geometry constraints that were not possible in previous works (e.g., [Saxena et al., 2008]).

Our depth estimation models did not make use of refined foreground object information which, as other researchers have also noted [Saxena et al., 2008, Hoiem et al., 2007a], could provide strong constraints on 3D distances. For example, knowing the distribution over object sizes (in a particular category, say, people), puts a constraint on distances within the scene. Unfortunately, the lack of ground truth training data containing foreground objects complicates their inclusion into the model. We note that Saxena et al. [2008] demonstrates the use of detected objects for encoding (vertical) orientation priors, much like our use of foreground semantics. However, they do not make use of object size.

A possible solution is to use manufacturer's specifications (or anatomical statistics, in the case of animals) to estimate distributions over object sizes. These could then be encoded as soft constraints in the model like many of our other terms. It remains to be able to learn the relative strength of these terms which will depend on their interactions with other energy terms and on the quality of the object detectors. Weakly supervised data containing foreground objects, and with a few pairwise distances hand-annotated (per image) may be sufficient for this purpose. Alternatively, synthetic data (e.g., extracting ground truth depth from ray-traced scenes) may be employed, or a combination of synthetic and weakly labeled images used. These approaches present interesting directions for future work.

Objects can be used in other ways, too. Our depth estimation model (and those in previous works) make a strong assumption about the focal length of the camera, which is necessary for computing the orientation of rays $\mathbf{r}_p$ extending through the camera plane into the scene. A natural extension, of this work, is to remove this assumption and infer the focal length from features in the image. One possibility is to make use of foreground objects for this purpose. However, purely scaling for the height of an object is not sufficient, since there remains a scaling ambiguity that object height alone does not address: Is the object twice as large because it is half the distance to the camera, or because the camera has half the focal length? To answer this question, an algorithm would need to extract z-value distances, for example, between objects (or landmarks) at different depths, in addition to object width and height parallel to the imaging plane.

Finally, this chapter has addressed the problem of using predicted semantic information to guide 3D reconstruction as a two-stage process. An interesting question, partially addressed in Chapter 4 and also in some other works on holistic scene understanding (e.g., [Heitz et al., 2008b]), is whether we can go in the other direction.

That is, how can we use 3D constraints and, in particular, depth estimates derived from image features to guide semantic understanding? One of the most compelling directions for future research, as we shall discuss in Chapter 7, is the joint inference over these aspects of the scene in a single unified model.

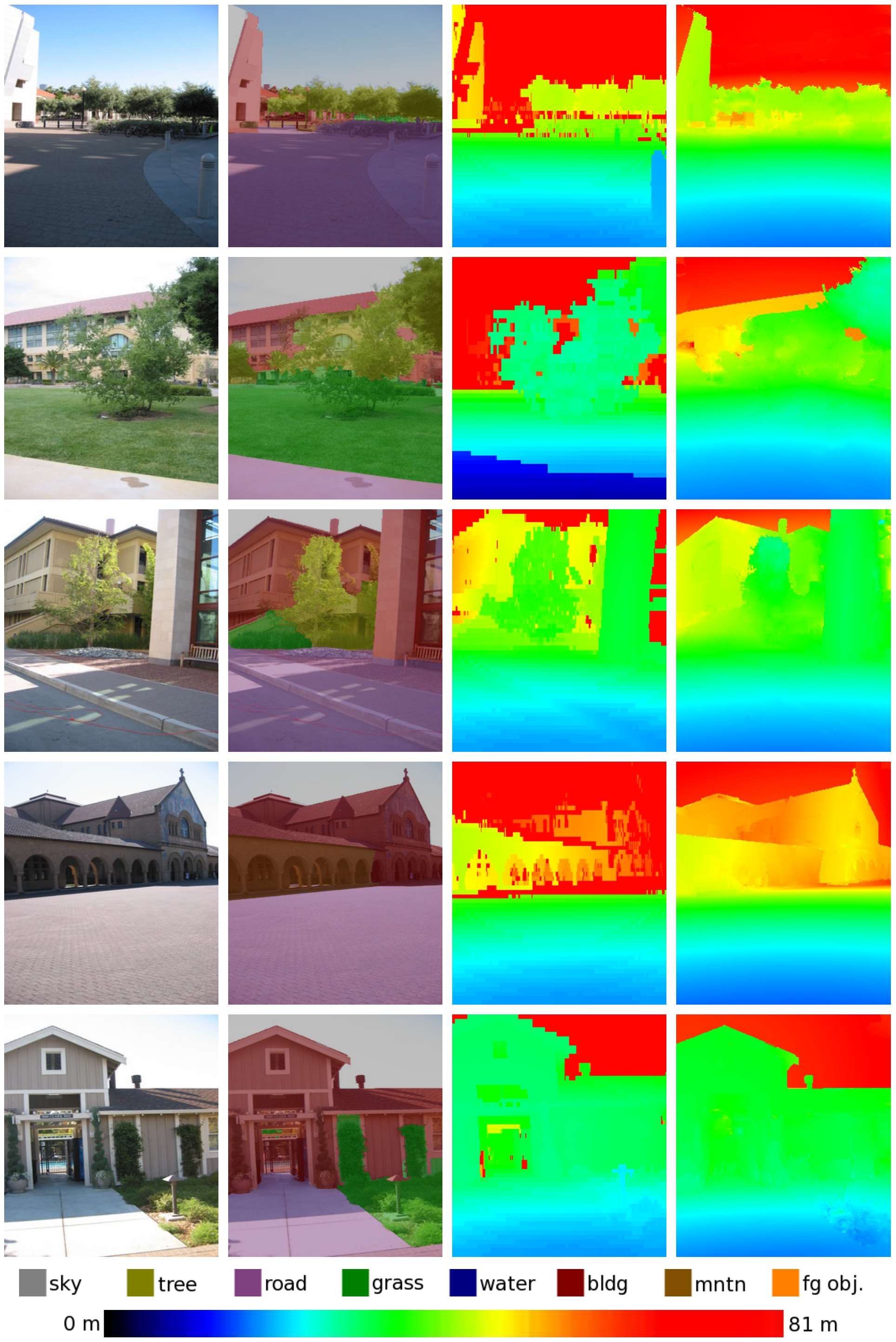


Figure 6.8: Qualitative depth estimation results on the Make3d dataset [Saxena et al., 2005]. Shown (from left to right) are the image, semantic overlay, ground truth depth measurements, and our predicted depth estimates. The legends shows semantic color labels and depth scale.

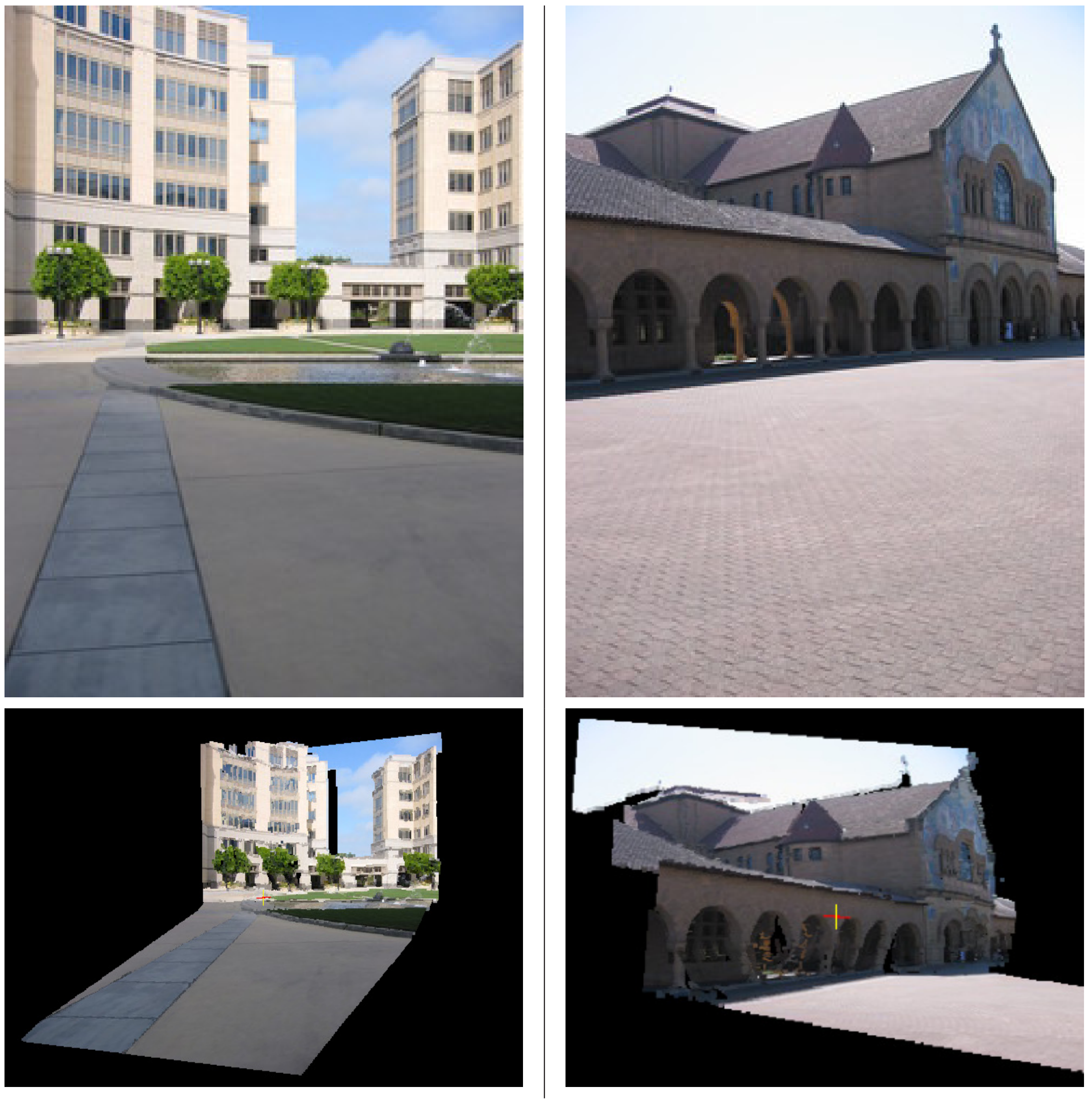


Figure 6.9: Novel 3D views of different scenes generated from our semantically-aware depth reconstruction model.

Chapter 7

Conclusions and Future Directions

Our work has focused on understanding three main aspects of a scene—regions, objects, and geometry. We have done so using a holistic model that enforces consistency and allows information to propagate between these different aspects of a scene. In this final chapter, we summarize our model and discuss its contributions. We then present some open problems and challenges for future research.

7.1 Summary

In this dissertation, we tackled the problem of holistic scene understanding. We presented a unified model for understanding scenes in terms of their dominant regions, objects, and geometry. These three aspects of the scene are key to higher-level interpretation. The most important feature of our model is that it is holistic. That is, we jointly reason over the three scene aspects rather than treat them as independent tasks. This enforces consistency in our representation and hence interpretation of the scene. In addition, the introduction of regions as variables in our model means that we are not committing to early bottom-up segmentation decisions that may result in degraded performance.

In Chapter 4 we introduced the first components of our model for decomposing a scene into semantic and geometrically consistent regions. Entities in the scene (pixels, regions and geometry) were encoded as random variables in a conditional

Markov random field (CRF) model, which allows for preferences of assignments to subsets of variables to be specified. We showed how these preferences could be learned from labeled training data. Exact inference in our model is intractable so we proposed a move-making approach that performs very well in practice.

When decomposing the scene into regions, we focused on a number of background classes that are typically found in outdoor scenes. These were easily characterized by local color and texture information. We also included a (single) foreground object class label. Foreground objects are harder to characterize in terms of local features and we found that many objects, while separated from the background, were themselves segmented into multiple regions. For example, people tended to be divided into head, torso, and legs, and cars into wheels and body.

We extended our model in Chapter 5 to incorporate the notion of objects. This involved the refinement of (one or more) foreground regions into specific object categories. Random variables representing objects and their corresponding energy terms were introduced into the model. The energy terms scored how likely a proposed object was to be one of the known object categories. Other energy terms encoded contextual preferences between objects and their neighboring regions (e.g., cars are found on roads; boats are found on water). An important benefit of our holistic model was that we were able to provide a precise (pixel-level) boundary for objects rather than the standard bounding box representation. However, we were still able to take advantage of strengths of bounding box approaches in proposing top-down moves that suggested candidate locations for objects. Each move was then evaluated against our unified energy function ensuring that we were optimizing a consistent objective.

In the final technical chapter, Chapter 6, we showed how simple geometric reasoning applied to our region-based decomposition of a scene could lead to plausible 3D planar reconstructions. Building on this geometric reasoning, we then developed a continuous CRF model that incorporated semantic labels and region boundary information to guide depth prediction in outdoor scenes. This model outperformed current state-of-the-art approaches on the task of depth estimation from monocular images.

7.2 Open Problems and Future Directions

Our work has made some progress towards the long-term goal of scene understanding. However, there are still many questions left unanswered and plenty more work to be done. This section discusses some of the limitations of our current approach and suggests a number of possible directions for building on our work.

7.2.1 Learning from Multiple Sources

As discussed in Chapter 3, the availability of high-quality datasets in computer vision is becoming more prevalent. However, there are still a significant number of object categories and scene types that are not represented in existing datasets. Furthermore, for some computer vision tasks, data is still difficult to collect. For example, one of the limitations of the depth estimation model was that it required image-aligned ground truth depth measurements for training. This is difficult to acquire, especially for distant and moving objects.

In addition, many existing datasets are labeled for a specific task in mind. In order to leverage these valuable resources (as opposed to relabeling all the images every time we wish to consider a new task), it will be beneficial to develop approaches that can learn from multiple disjoint sources. Our existing framework does this to a limited extent. For example, the object detectors are learned from a separate set of cropped images and later integrated into the larger scene model. However, a more integrated approach that allows disjoint training sources at all levels of the algorithm is still needed.

An exciting direction for future work that addresses these limitations is to extend our framework to make use of large corpora of partially labeled data, synthetic training data, and video sequences. This has the potential to fill the gap and provide depth information (e.g., from ray-traced synthetic scenes) and segmentation cues (e.g., by extracting motion cues from video). As discussed in previous chapters, the use of max-margin learning algorithms for structured prediction are a promising direction for training our models, and extensions that allow for latent variables (e.g., [Yu and Joachims, 2009]) make training with partially labeled data feasible.

7.2.2 Structure and Occlusion

Our current work has focused on understanding images of outdoor scenes, in particular, urban, rural and harbor. However, there are many other scenes types with markedly different geometric structure that we would like to understand. For example, indoor scenes present a very different challenge to outdoor scenes.

Even more interesting are scenes with considerably different scales. For example, understanding the difference between a close-up of a person's face or a zoomed-in view of some workspace versus an entire urban landscape. Our current model makes strong assumptions about the scene structure, and in these scenarios those assumptions are clearly violated. Instead of relying on global geometry, understanding close-up images requires local geometric reasoning similar to the early work of inferring geometry from line drawings [Waltz, 1975] or structure from shading [Horn, 1989].

Other than scale, one of the most important cues that we can derive from scene structure is knowing the relative location of objects and thus be able to reason about occlusion. Presently, occlusion reasoning is absent from our model. An interesting research direction is in developing models that make better use of our geometric understanding of a scene to determine what parts of an object (candidate) are occluded and taking that information into account. For example, if a person is standing behind a car so that their legs are occluded, then the object term in our energy function should not look for visual evidence of legs. Likewise, if we have no evidence that an object is occluded, then the object term should penalize missing features.

Occlusion modeling can be extended to take into account transparent objects (which would require allowing a pixel to be associated with two different objects). For example, a person looking out of a glass window. This becomes important when we start trying to understand the activities in a scene, as we discuss below. For example, the question "is there a person inside the car?" can only be answered if we can see a person through the car's windows or windshield.

Finally, as discussed in Chapter 6, our current model performs depth estimation after decomposing the scene into regions. While we were able to produce state-of-the-art results for depth estimation, it did not allow us to make use of this depth information when decomposing the scene or identifying objects (other than implicit cues from the location of the horizon and region configuration). Developing a model that jointly optimizes over depth prediction and scene decomposition poses significant challenges, but would allow stronger coupling between these aspects of the scene.

7.2.3 Expanding the Object Classes

Our approach currently only recognizes objects from a small set of categories. The region layer handles this by allowing regions to be annotated with a generic foreground object class label. However, at the object layer we would like to be able to identify many thousands of object types. Scaling to this magnitude is a significant challenge in many respects. First, we would need to collect labeled training instances for each object class. Second, the object energy term would now have many competing hypotheses (due to the increased label space). Intuitively, this increases the chance of false positives. Last, as the number of object classes increases, the similarity between object categories complicates training and evaluation. For example, is mistaking a car for a truck as bad an error as mistaking a car for a horse?

Regarding the first challenge, an important limitation of our current approach is its reliance on a large amount of hand-labeled training data. As discussed in Section 7.2.1 above, extending our framework to make use of diverse data sets is a viable solution to this problem.

An interesting direction for addressing the second challenge is to develop model-free methods for object labeling within a higher-level model of scene structure. For example, object candidates can be matched to a large dictionary of labeled samples. The matched instances then need to be coherently placed within the scene using the region and geometry aspects of the model. An advantage of this approach is that it is an *open universe* allowing new object instances and categories to be incrementally added without changing the underlying model. It also presents exciting opportunities for active learning paradigms and object discovery.

The last challenge gets at the heart of the scene understanding problem. As discussed in the introduction, the real goal is to enable autonomous agents to interpret scenes so as to provide a description, answer questions, or perform some action. Thus, it is the final application that determines the quality of the scene understanding framework. For example, a safety system in an autonomous vehicle need not worry about confusing a car with a horse, so long as it avoids an accident. Alternatively, a content retrieval system may wish to have high precision so that when a user searches for images of horses, the system does not return cars. The next two sections deal with two different subtleties raised by this challenge.

7.2.4 Hierarchical Decomposition

Currently, our model decomposes a scene over a very shallow hierarchy. We associate pixels to regions, and regions to objects. However, many objects that we are interested in are made up of different parts (such as the wheels of a car), which are not captured in our model. Decomposing the scene over a deeper hierarchy, also known as *image parsing* will present three key advantages.

First, in describing or answering questions about a scene, we may need to know about object parts and poses (e.g., answering the question "does the car have a flat tire?" requires that know how to decompose a car into parts, and what a flat tire looks like independent of the general appearance of cars). As we discuss in the following section, this can be achieved if we already have a hierarchical decomposition rather than as a post-processing step where the relevant information may not be readily available.

The second benefit of a hierarchical decomposition is that it addresses some of the issues raised above in expanding the set of recognizable object categories. For example, learning models of common parts and attributes, and sharing these learned models between objects may allow us to learn better object models (especially when training data is scarce).

Last, the hierarchical approach may present some opportunities for accelerating inference when the number of object categories grows large. For example, by eliminating unlikely object hypotheses while building up our understanding of a scene (e.g., all objects that do not have wheels), the value space of object variables can be significantly pruned.

7.2.5 Activity Recognition

In claiming to have understood a scene, we have developed a model that can recognize objects, annotate background regions, and reconstruct coarse geometry. However, this is not enough to really be able to describe what the scene is about. Complete scene understanding also requires the ability to reason about the activities that are happening within the scene. This opens up a whole new thrust of research into how to make use of our region-based understanding to infer the actions behind the scene.

We expect that by building on our decomposition of the scene into regions and object, many activities can be readily identified by making use of the 3D relationships between interacting objects and their parts. As discussed above, this requires a hierarchical decomposition so the we can identify relevant parts. For example, a person kneeling down next to the wheel of a car is possibly changing a flat tire, and a cow with its head down next to a trough is probably drinking water.

7.3 Conclusion

This thesis has explored the problem of holistic scene understanding by integrating the tasks of segmentation, multi-class image labeling, object detection, and depth reconstruction into a single unified model. However, as discussed in the previous section, there is still much work to be done on improving these methods and exciting directions in which they can be extended. We hope that our work can provide the foundations on which further research can stand.

Appendix A

Implementation Details

Inference in our region-based models, as discussed in Chapter 4 and Chapter 5, involves repeated execution of two expensive operations—computing region features and checking for connectedness of the regions. Specifically, every time a proposal move is made it changes the pixel-to-region associations and these two operations must be performed. They become the computational bottleneck during inference, so to achieve good runtime performance they need to be optimized. In this section we present some implementation details that describe the data structures and algorithms that enable fast execution of region feature calculation and connected component finding.

We will assume that the inference algorithm is running on a W-by-H image $\mathcal{I}$. A pixel $p \in \mathcal{I}$ is indexed by its (u, v) coordinates in the image plane, with $u \in \{1, \ldots, W\}$ and $v \in \{1, \ldots, H\}$, and we store our pixel-to-region association variables in an integer matrix $\boldsymbol{R} \in \mathbb{Z}^{H \times W}$ so that R_{vu} indicates the region to which pixel (u, v) belongs.[1] The set of pixels belonging to region r is then $\mathcal{P}_r = \{(u, v) \mid R_{vu} = r\}$. Likewise, we will assume that any feature computed at the pixel level, specifically, the α_p's, are arranged in feature matrices $F^{(1)}, \ldots, F^{(n)} \in \mathbb{R}^{H \times W}$ where the element $F^{(i)}_{vu}$ stores the i-th component of feature vector α_p for pixel (u, v).[2] This is illustrated in Figure A.1.

[1]Note that matrices are indexed in row-major order, i.e., the v coordinate first and the u coordinate second. To avoid confusion in dealing with image indexing and matrix indexing, we will use parentheses for column-major indexing and subscripts for row-major indexing, so that $R_{vu} = R(u, v)$.

[2]Arranging the storage of these matrices so that $F^{(1)}_{vu}, \ldots, F^{(n)}_{vu}$ are contiguous in memory will result in better cache locality and thus faster execution.

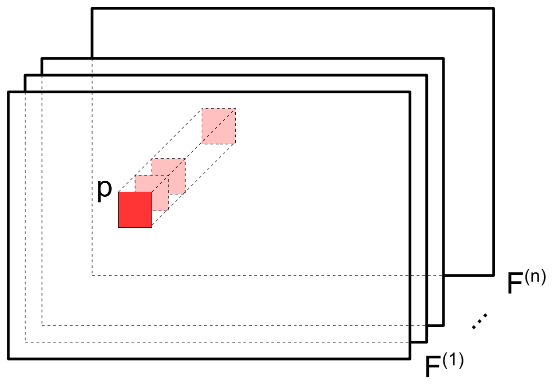


Figure A.1: Feature matrices for storing pixel feature vectors. Each feature is arranged in an $H \times W$ matrix. The feature vector for pixel p is constructed by taking the element corresponding to p from each feature matrix.

A.1 Region Appearance Features

Recall from Chapter 4 that we model the appearance of each region r in the scene as $A_r = \left(\mu_r^A, \Sigma_r^A\right)$ where

$$\mu_r^A = \frac{1}{N_r} \sum_{p \in \mathcal{P}_r} \alpha_p \tag{A.1}$$

$$\Sigma_r^A = \frac{1}{N_r} \sum_{p \in \mathcal{P}_r} \left(\alpha_p - \mu_r^A\right)\left(\alpha_p - \mu_r^A\right)^T \tag{A.2}$$

$$= \frac{1}{N_r} \sum_{p \in \mathcal{P}_r} \alpha_p \alpha_p^T - \mu_r^A (\mu_r^A)^T \tag{A.3}$$

and $\alpha_p \in \mathbb{R}^n$ are the appearance feature vectors for each pixel, $\mathcal{P}_r \subseteq \mathcal{I}$ are the set of pixels belonging to the region, and $N_r = |\mathcal{P}_r|$ is the size of the region.

Now, if we were given a single segmentation (decomposition) of the image $\boldsymbol{R}$, we could simply iterate over the pixels p accumulating α_p and $\alpha_p \alpha_p^T$ for each region and compute the above appearance features in $O(Nn^2)$ where $N = H \cdot W$ is the total number of pixels in the image. Unfortunately, during inference, we end up considering numerous different segmentations of the image and a naive re-computation of each region's appearance would be prohibitively expensive. Accelerating this computation warrants some upfront work to organize the feature values in a more convenient form. As we shall see, one such convenient form is to precompute integrals along each row of the feature matrix. However, before describing these *row integrals* we begin by discussing the more familiar *integral images*.

Integral images were first introduced by Crow [1984] and now used extensively in computer vision. The integral image for a feature matrix $F^{(i)}$, which we denote by $F^{(i)}_{\text{int}}$ is an $(H+1) \times (W+1)$ matrix whose (u,v)-th entry contains the the sum of all entries in $F^{(i)}$ above and to the left of pixel (u,v), i.e.,

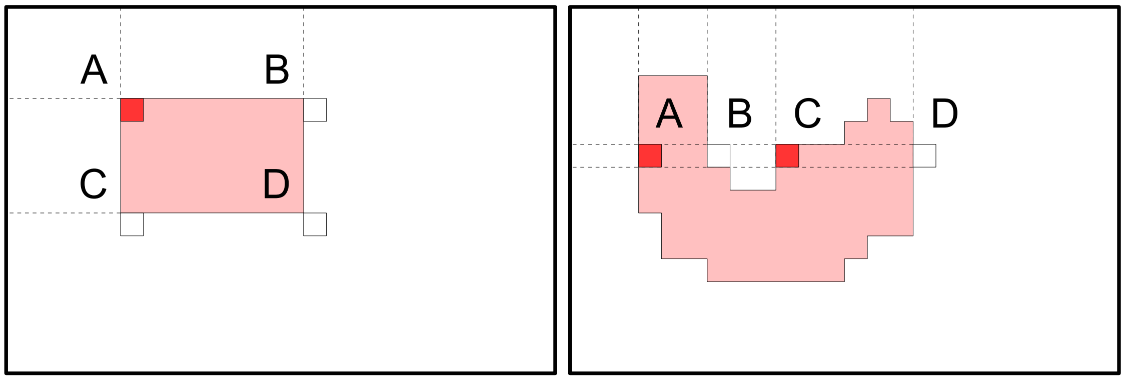
$$F^{(i)}_{\text{int}}(u,v) = \sum_{u'<u, v'<v} F^{(i)}(u',v'). \tag{A.4}$$

The integral image can be computed in a single pass through the feature matrix, and once computed can be used to find the sum of feature values within any rectangular region in the image in constant time. Consider, for example, the rectangle $ABCD$ in Figure A.2(a). It is straightforward to see that the sum of features within this rectangle can be calculated as

$$\sum_{\substack{u_A \le u < u_D \\ v_A \le v < v_D}} F^{(i)}(u,v) = F^{(i)}_{\text{int}}(u_A,v_A) + F^{(i)}_{\text{int}}(u_D,v_D) - F^{(i)}_{\text{int}}(u_B,v_B) - F^{(i)}_{\text{int}}(u_C,v_C). \tag{A.5}$$

The idea of integral images can be readily extended to squared integral images for computing the sum over features squared, and even adapted to compute integrals over rectangular regions rotated by 45°.

Unfortunately, for non-rectangular regions, integral images fail to be beneficial. For arbitrary shaped regions some work has been done in extending Green's Theorem (which relates the double integral over a closed surface to the line integral around its contour) to the discrete domain [Tang, 1982, Yang and Albregtsen, 1996]. However, these approaches require the boundary of each region to be traversed (and therefore computed and stored explicitly). We elect to use a simpler approach that shares ideas developed in Kumar et al. [2004] for accelerating similar computations in their work. Our approach has the added advantage of being applicable to the case of computing



(a) Integral image (b) Row integral

Figure A.2: Integral images and row integrals. Integral images are useful for efficiently computing the sum of feature values within a rectangular region, but fail to be of use on arbitrary shaped regions.

features over all regions in the image in a single pass.

We define the row integral $F^{(i)}_{\text{r-int}}$ to be an $H \times (W + 1)$ matrix whose (u, v)-th entry contains the the sum of all entries in $F^{(i)}$ to the left of pixel (u, v), i.e.,

$$F^{(i)}_{\text{r-int}}(u, v) = \sum_{u' < u} F^{(i)}(u', v). \tag{A.6}$$

Like the integral image, the row integrals can be computed in a single pass through the feature matrix. Similarly, we can define the squared row integral as

$$F^{(i,j)}_{\text{sq-r-int}}(u, v) = \sum_{u' < u} F^{(i)}(u', v) \cdot F^{(j)}(u', v). \tag{A.7}$$

With these row integrals precomputed, we can now compute the region appearance features in $O(N + Bn^2)$ time using Algorithm A.1. Here B is the number of boundary pixels (transitions) between regions.[3] Note that usually we have $B \ll N$. In addition, the algorithm can be modified to include an explicit check for the special-case of single pixel transitions and resort to using the feature matrices directly in this case (saving

[3]The boundary pixels include pixels at the left and right image boundaries. In cases where regions are likely to wrap around the edge of the image (e.g., a large contiguous sky region), a *raster scan integral* as opposed to *row integrals* may provide additional speedup.

an overhead of $n^2 + n$ operations).

The computational gains can be understood by considering the example region in Figure A.2(b). For the line indicated, a naive approach would require summing over all shaded pixels. However, using our integral rows we can compute the sum over the two connected shaded area using four operations (per feature value), e.g.,

$$F^{(i)}_{\text{r-int}}(u_B, v) - F^{(i)}_{\text{r-int}}(u_A, v) + F^{(i)}_{\text{r-int}}(u_D, v) - F^{(i)}_{\text{r-int}}(u_C, v) \tag{A.8}$$

for the i-th feature. Note that in Algorithm A.1 we use the notation $\left(F^{(i)}_{\text{r-int}}(u, v)\right)_{i=1}^{n}$ and $\left(F^{(i)}_{\text{sq-r-int}}(u, v)\right)_{i=1,j}^{n}$ to represent the n-vector and $n \times n$ matrix formed by taking the corresponding entries of $F^{(i)}_{\text{r-int}}(u, v)$ and $F^{(i,j)}_{\text{sq-r-int}}(u, v)$, respectively.

While the algorithm has been described for calculating the aggregate features for an entire image, we can further improve performance by maintaining sufficient statistics over each region. Then whenever a move is proposed, we only need to adjust the sufficient statistics by iterating over the modified pixels, resulting in dramatically faster computation.

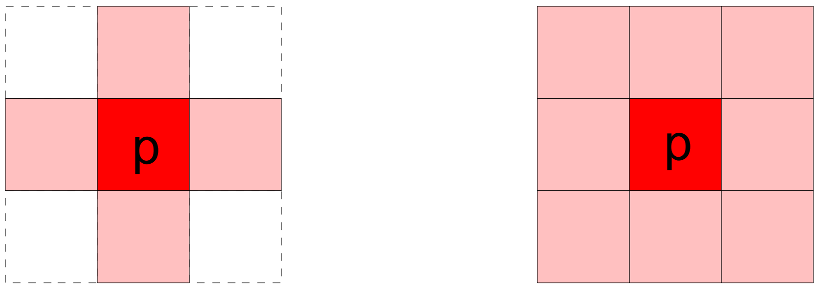
A.2 Checking for Connected Components

In Chapter 4 we noted that currently, our method for scene decomposition requires that the regions be *connected components*. We now make precise our definition of connected component and describe an algorithm which efficiently checks for connectedness. We begin by defining the neighborhood of a pixel:

Definition A.2.1: The *neighborhood* of pixel p is a set of pixels $\mathcal{N}(p)$ is such that $p \notin \mathcal{N}(p)$ and if pixel $q \in \mathcal{N}(p)$ then $p \in \mathcal{N}(q)$. We will use $\mathcal{N}(\mathcal{P})$ to denote the neighborhood of a set of pixels $\mathcal{P}$, i.e., $\mathcal{N}(\mathcal{P}) = \bigcup_{p \in \mathcal{P}} \mathcal{N}(p)$.

Of interest to us will be 4-connected and 8-connected neighborhoods. These are illustrated in Figure A.3 and formalized by the following definitions:

Definition A.2.2: The *4-connected neighborhood* of pixel $p = (u, v)$ is the set of pixels $\mathcal{N}_4(p) = \{(u-1, v), (u, v-1), (u, v+1), (u+1, v)\}$. Similarly, the *8-connected*



(a) 4-connected neighborhood (b) 8-connected neighborhood

Figure A.3: 4-connected and 8-connected neighborhoods.

neighborhood of pixel $p = (u, v)$ is the set of pixels $\mathcal{N}_8(p) = \{(u-1, v-1), (u-1, v), (u-1, v+1), (u, v-1), (u, v+1), (u+1, v-1), (u+1, v), (u+1, v+1)\}$.

We are now ready to define a connected component. Recall that a region r contains the set of pixels $\mathcal{P}_r$. The following definition encapsulates what it means for region r to be connected.

Definition A.2.3: Two pixels p and q, both in region r, are *connected* with respect to some neighborhood $\mathcal{N}$ if there exists a set of pixels $\{i_1, \ldots, i_n\} \subseteq \mathcal{P}_r$ such that $i_1 = p$, $i_n = q$, and for all $k < n$, $i_k \in \mathcal{N}(i_{k+1})$. Region r is a *connected component* if $\forall (p, q) \in \mathcal{P}_r \times \mathcal{P}_r$, p and q are connected.

The most naive approach to determining whether a region is connected or not is to perform a breadth-first or depth-first search over neighboring pixels. Specifically, starting from any pixel $p \in \mathcal{P}_r$ we place all neighbors $q \in \mathcal{N}(p) \cap \mathcal{P}_r$ into a queue. Then, while the queue is not empty, we pop an element p' from the queue and, if p' has not been visited previously, add all of its neighbors $q' \in \mathcal{N}(p') \cap \mathcal{P}_r$ to the queue. The set of visited pixels forms a connected component, and if all pixels in $\mathcal{P}_r$ were visited, then the region is a connected. If the region is not connected, then we can break it into connected components, by iteratively finding a pixel in the region that has not yet been visited and applying the above algorithm.

While the above approach to finding connected components is linear in the size of the image, it suffers from needing to maintain a queue data structure and having to examine the entire neighborhood of each pixel. Instead of this approach, we propose a method that groups connected components along rows in the image and makes use of a *disjoint sets* (or *union-find*) data structure [Cormen et al., 2001] to merge connected groups. Other researchers (e.g., [Strandh and Lapeyre, 2004]) have found performance gains using similar approaches.

Our method is presented in Algorithm A.2. Briefly, we iterate through the image from top-left to bottom-right maintaining disjoint sets of pixels. At each pixel p, we examine the pixel's region-association variable R_p to determine whether it belongs to the same region as the previous pixel *in the same row*. If so, we add the pixel to the same set as the previous pixel. Otherwise, we start a new set $\mathcal{P} = \{p\}$. In addition to maintaining the set of (sets of) pixels, we also maintain a list of edges between neighboring sets. Again, at each pixel p, we examine the pixels in the row above (that are also within the pixel's neighborhood). For each of these pixels q, if q belongs to the same region as p we add edge between the set containing p and the set containing q. Finally (in Line 25 of Algorithm A.2), we merge all pixel sets with an edge between them (using *union* and *find* operations on the *disjoint sets* data structure [Cormen et al., 2001]).

The above algorithm, in addition to not having to search the full neighborhood of each pixel (since we only ever examine the pixel to the left and the row above), allows us to re-use calculations when proposing moves. Like the algorithm for computing region appearance, by caching the connected row segments $\mathcal{V}$ and the neighborhood set $\mathcal{E}$ (and some housekeeping data), Algorithm A.2 can be modified to operate on the set of pixels modified by a move resulting significant performance gains.

We have presented the algorithms for updating region appearance and finding connected components as two separate stages. However, the astute reader will have noticed that both algorithms iterate through the image performing operations on connected row segments. By maintaining a data structure over connected row segments, we can iterate over modified pixels once and perform both computations quickly.

Algorithm A.1: Compute Region Features.

```
1  Input  : $\boldsymbol{R}$                                  // region association variables
2           $\{F^{(i)}_{\text{r-int}}\}_{i=1}^{n}$            // row integrals
3           $\{F^{(i,j)}_{\text{sq-r-int}}\}_{i,j=1}^{n}$     // squared row integrals
4  Output:  $\boldsymbol{A}$                                  // region appearance variables
5  initialize $N_r = 0$, $\mu_r^A = \mathbf{0}_{n\times 1}$ and $\Sigma_r^A = \mathbf{0}_{n\times n}$ for each region $r$
6  for $v = 1, \ldots, H$ do
7      set $s = \boldsymbol{R}(u, v)$
8      for $u = 1, \ldots, W$ do
9          set $r = \boldsymbol{R}(u, v)$
10         $N_r \leftarrow N_r + 1$
11         if $r \neq s$ then
12             $\mu_r^A \leftarrow \mu_r^A - \left(F^{(i)}_{\text{r-int}}(u, v)\right)_{i=1}^{n}$ and $\Sigma_r^A \leftarrow \Sigma_r^A - \left(F^{(i,j)}_{\text{sq-r-int}}(u, v)\right)_{i,j=1}^{n}$
13             $\mu_s^A \leftarrow \mu_s^A + \left(F^{(i)}_{\text{r-int}}(u, v)\right)_{i=1}^{n}$ and $\Sigma_s^A \leftarrow \Sigma_s^A + \left(F^{(i,j)}_{\text{sq-r-int}}(u, v)\right)_{i,j=1}^{n}$
14             $s \leftarrow r$
15         end
16     end
17     $\mu_s^A \leftarrow \mu_s^A + \left(F^{(i)}(W + 1, v)\right)_{i=1}^{n}$ and $\Sigma_s^A \leftarrow \Sigma_s^A + \left(F^{(i,j)}_{\text{sq-r-int}}(W + 1, v)\right)_{i,j=1}^{n}$
18 end
19 foreach region $r$ do
20     set $\mu_r^A \leftarrow \frac{1}{N_r}\mu_r^A$
21     set $\Sigma_r^A \leftarrow \frac{1}{N_r}\Sigma_r^A - \mu_r^A(\mu_r^A)^T$
22 end foreach
23 return $\{A_r = (\mu_r^A, \Sigma_r^A)\}$
```

Algorithm A.2: Find Connected Components.

```
Input  : $\boldsymbol{R}$                // region association variables
         $\mathcal{N}$                // neighborhood operator (e.g., $\mathcal{N}_4$ or $\mathcal{N}_8$)
Output: $\mathcal{C}$                // set of connected components
initialize $\mathcal{V} = \emptyset$ and $\mathcal{E} = \emptyset$
for $v = 1, \ldots, H$ do
    initialize active pixel set $\mathcal{P}$ to $\{(1, v)\}$
    for $u = 2, \ldots, W$ do
        if $\boldsymbol{R}(u, v) = \boldsymbol{R}(u-1, v)$ then
            $\mathcal{P} \leftarrow \mathcal{P} \cup \{(u, v)\}$
        end
        if $\boldsymbol{R}(u, v) \neq \boldsymbol{R}(u-1, v)$ or $u = W$ then
            add to $\mathcal{P}$ to $\mathcal{V}$
            foreach $(u', v') \in \mathcal{N}(\mathcal{P})$ such that $v' = v - 1$ do
                if $\boldsymbol{R}(u', v') = \boldsymbol{R}(u-1, v)$ then
                    let $P'$ be the pixel set containing $(u', v')$
                    add edge $(\mathcal{P}, \mathcal{P}')$ to $\mathcal{E}$
                end
            end foreach
            if $u \neq W$ then
                re-initialize active pixel set $\mathcal{P} = \{(u, v)\}$
            end
        end
    end
end
$\mathcal{C} = \mathrm{union}(\mathcal{V}, \mathcal{E})$
return $\mathcal{C}$
```

Bibliography

H. Barrow and J. Tenenbaum. Computational vision. *Proc. of the IEEE*, pages 572–295, 1981.

J. Besag. Statistical analysis of non-lattice data. *The Statistician*, pages 179–195, 1975.

J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society Series B*, 48:259–302, 1986.

S. Bileschi and L. Wolf. A unified system for object detection, texture recognition, and context analysis based on the standard model feature set. In *Proc. of the British Machine Vision Conference (BMVC)*, 2005.

C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1996.

C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2007.

E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2002.

S. P. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge, 2004.

Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *Proc. of the International Conference on Computer Vision (ICCV)*, 1999.

Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2001.

R. Brooks. Model-based 3-D interpretations of 2-D images. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, pages 140–150, 1983.

S. C. Brubaker, J. Wu, J. Sun, M. D. Mullin, and J. M. Rehg. On the design of cascades of boosted ensembles for face detection. *International Journal of Computer Vision (IJCV)*, 2007.

J. Canny. A computational approach to edge detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, pages 679–714, 1986.

D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 24: 603–619, 2002.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2001.

A. Criminisi. Microsoft research cambridge (MSRC) object recognition image database (version 2.0), 2004. Available for download from `http://research.microsoft.com/vision/cambridge/recognition`.

A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. *International Journal of Computer Vision (IJCV)*, 2000.

F. Crow. Summed-area tables for texture mapping. In *Proc. of the International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 207–212, 1984.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. of the IEEE Conference of Computer Vision and Pattern Recognition (CVPR)*, 2005.

E. Delage, H. Lee, and A. Y. Ng. A dynamic Bayesian network model for autonomous 3D reconstruction from a single indoor image. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

S. K. Divvala, D. Hoiem, J. H. Hays, A. A. Efros, and M. Hebert. An empirical study of context in object detection. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley-Interscience, 2000.

G. Elidan, I. McGraw, and D. Koller. Residual belief propagation: Informed scheduling for asynchronous message passing. In *Proc. of the Uncertainty in Artificial Intelligence (UAI)*, 2006.

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results, 2010.

L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. In *Proc. of the IEEE Conference of Computer Vision and Pattern Recognition (CVPR)*, 2004.

L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. In *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2006.

P. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2010.

P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision (IJCV)*, 2004.

R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proc. of the IEEE Conference of Computer Vision and Pattern Recognition (CVPR)*, 2003.

V. Ferrari, L. Fevrier, F. Jurie, and C. Schmid. Groups of adjacent contour segments for object detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2008.

M. Fink and P. Perona. Mutual boosting for contextual inference. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proc. of the European Conference on Computational Learning Theory (ECCLT)*, pages 23–37, 1995.

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression. Technical report, Stanford, 1998.

M. Fritz, M. Black, G. Bradski, and T. Darrell. An additive latent feature model for transparent object recognition. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

C. Galleguillos and S. Belongie. Context based object categorization: A critical survey. Technical Report CS2008-0928, UCSD, 2008.

V. Ganapathi, D. Vickrey, J. Duchi, and D. Koller. Constrained approximate maximum entropy learning. In *Proc. of the Uncertainty in Artificial Intelligence (UAI)*, 2008.

A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller. Multi-class segmentation with relative location prior. *International Journal of Computer Vision (IJCV)*, 2008.

S. Gould, F. Amat, and D. Koller. Alphabet SOUP: A framework for approximate energy minimization. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009a.

S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2009b.

S. Gould, T. Gao, and D. Koller. Region-based segmentation and object detection. In *Advances in Neural Information Processing Systems (NIPS)*, 2009c.

D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society*, 51(2):271–279, 1989.

G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical report, California Institute of Technology, 2007.

C. Gu, J. J. Lim, P. Arbelaez, and J. Malik. Recognition using regions. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

F. Han and S.-C. Zhu. Bottom-up/top-down image parsing with attribute grammar. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2009.

A. R. Hanson and E. M. Riseman. VISIONS: A computer system for interpreting scenes. In *Computer Vision Systems*, pages 303–333, 1978.

R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

X. He, R. Zemel, and M. Carreira-Perpinan. Multiscale conditional random fields for image labeling. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2009.

G. Heitz and D. Koller. Learning spatial context: Using stuff to find things. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2008.

G. Heitz, G. Elidan, B. Packer, and D. Koller. Shape-based object localization for descriptive classification. In *Advances in Neural Information Processing Systems (NIPS)*, 2008a.

G. Heitz, S. Gould, A. Saxena, and D. Koller. Cascaded classification models: Combining models for holistic scene understanding. In *Advances in Neural Information Processing Systems (NIPS)*, 2008b.

T. Heskes. Convexity arguments for efficient minimization of the Bethe and Kikuchi free energies. *Journal of Artificial Intelligence Research (JAIR)*, 26:153–190, 2006.

D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2005a.

D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. In *Proc. of the International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2005b.

D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. In *Proc. of the IEEE Conference of Computer Vision and Pattern Recognition (CVPR)*, 2006.

D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. *International Journal of Computer Vision (IJCV)*, 2007a.

D. Hoiem, A. N. Stein, A. A. Efros, and M. Hebert. Recovering occlusion boundaries from a single image. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2007b.

D. Hoiem, A. A. Efros, and M. Hebert. Closing the loop on scene interpretation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008a.

D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. *International Journal of Computer Vision (IJCV)*, 2008b.

B. K. P. Horn. *Obtaining shape from shading information*, pages 123–171. 1989.

T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

E. Klingbeil, B. Carpenter, O. Russakovsky, and A. Y. Ng. Autonomous operation of novel elevators for robot navigation. In *Proc. of the International Conference on Robotics and Automation (ICRA)*, 2010.

P. Kohli, L. Ladicky, and P. Torr. Robust higher order potentials for enforcing label consistency. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques.* MIT Press, 2009.

V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2006.

V. Kolmogorov and C. Rother. Minimizing nonsubmodular functions with graph cuts—A review. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2007.

V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2004.

N. Komodakis, N. Paragios, and G. Tziritas. MRF optimization via dual decomposition: Message-passing revisited. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2007.

J. Kosecka and W. Zhang. Video compass. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2002.

M. P. Kumar, P. H. S. Torr, and A. Zisserman. Extending pictorial structures for object recognition. In *Proc. of the British Machine Vision Conference (BMVC)*, pages 789–798, 2004.

M. P. Kumar, P. H. S. Torr, and A. Zisserman. OBJ CUT. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

S. Kumar and M. Hebert. A hierarchical field framework for unified context-based classification. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2005.

L. Ladicky, C. Russell, P. Kohli, and P. Torr. Associative hierarchical CRFs for object class image segmentation. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2009.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of the International Conference on Machine Learning (ICML)*, 2001.

B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2004.

V. Lempitsky, P. Kohli, C. Rother, and T. Sharp. Image segmentation with a bounding box prior. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2009a.

V. Lempitsky, C. Rother, S. Roth, and A. Blake. Fusion moves for markov random field optimization. Technical report, Microsoft Research, 2009b.

A. Levin and Y. Weiss. Learning to combine bottom-up and top-down segmentation. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2006.

M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Trans. on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 2(1):1–19, 2006.

L.-J. Li, G. Wang, and L. Fei-Fei. OPTIMOL: automatic object picture collection via incremental model learning. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

L.-J. Li, R. Socher, and L. Fei-Fei. Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *Proc. of the IEEE Conference of Computer Vision and Pattern Recognition (CVPR)*, 2009.

J. Lim, P. Arbelaez, C. Gu, and J. Malik. Context by region ancestry. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2009.

B. Liu, S. Gould, and D. Koller. Single image depth estimation from predicted semantic labels. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing: Label transfer via dense scene alignment. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

D. Liu and J. Nocedal. On the limited memory method for large scale optimization. In *Math. Prog. B*, volume 45, pages 503–528, 1989.

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 2004.

Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An Invitation to 3-D Vision.* Springer, 2005.

D. Marr. *Vision.* W.H. Freeman and Company, 1982.

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 *Proc. of the International Conference on Computer Vision (ICCV)*, 2001.

D. Mirota, R. H. Taylor, M. Ishii, and G. D. Hager. Direct endoscopic video registration for sinus surgery. In *Medical Imaging 2009: Visualization, Image-guided Procedures and Modeling. Proceedings of the SPIE*, 2009.

E. N. Mortensen and W. A. Barrett. Interactive segmentation with intelligent scissors. In *GMIP*, 1998.

K. P. Murphy, A. Torralba, and W. T. Freeman. Using the forest to see the tree: a graphical model relating features, objects and the scenes. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

K. P. Murphy, A. Torralba, D. Eaton, and W. T. Freeman. Object detection and localization using local and global features. In J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, editors, *Toward Category-Level Object Recognition*, Cambridge, MA, 2006. MIT Press.

P. Newman, D. Cole, and K. Ho. Outdoor SLAM using visual appearance and laser ranging. In *Proc. of the International Conference on Robotics and Automation (ICRA)*, 2006.

J. Nocedal. Updating quasi-newton matrices with limited storage. In *Mathematics of Computation*, volume 35, pages 773–782, 1980.

Y. Ohta, T. Kanade, and T. Sakai. An analysis system for scenes containing objects with substructures. In *Proc. of the International Joint Conference on Pattern Recognition (IJCPR)*, pages 752–754, 1978.

A. Opelt, A. Pinz, and A. Zisserman. Incremental learning of object detectors using a visual shape alphabet. In *Proc. of the IEEE Conference of Computer Vision and Pattern Recognition (CVPR)*, 2006a.

A. Opelt, A. Pinz, and A. Zisserman. A boundary-fragment-model for object detection. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2006b.

C. P. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *Proc. of the International Conference on Computer Vision (ICCV)*, 1998.

J. Ponce, T. Berg, M. Everingham, D. Forsyth, M. Hebert, S. Lazebnik, M. Marszalek, C. Schmid, B. Russell, A. Torralba, C. Williams, J. Zhang, and A. Zisserman. Dataset issues in object recognition. *Toward Category-Level Object Recognition*, pages 29–48, 2006.

L. Quan and R. Mohr. Determining perspective structures using hierarchical hough transform. *Pattern Recognition Letters*, 9:279–286, May 1989.

M. Quigley, S. Batra, S. Gould, E. Klingbeil, Q. V. Le, A. Wellman, and A. Y. Ng. High-accuracy 3D sensing for mobile manipulation: Improving object detection and door opening. In *Proc. of the International Conference on Robotics and Automation (ICRA)*, 2009.

A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2007.

C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *Proc. of the International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2004.

C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. Optimizing binary MRFs via extended roof duality. In *Proc. of the IEEE Conference of Computer Vision and Pattern Recognition (CVPR)*, 2007.

B. C. Russell and A. Torralba. Building a database of 3D scenes from user annotations. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

B. C. Russell, A. A. Efros, J. Sivic, W. T. Freeman, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *Proc. of the IEEE Conference of Computer Vision and Pattern Recognition (CVPR)*, 2006.

B. C. Russell, A. B. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: A database and web-based tool for image annotation. *International Journal of Computer Vision (IJCV)*, 77(1-3):157–173, 2008.

S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2002.

A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

A. Saxena, L. Wong, M. Quigley, and A. Y. Ng. A vision-based system for grasping novel objects in cluttered environments. In *International Symposium on Robotics Research (ISRR)*, 2007.

A. Saxena, M. Sun, and A. Y. Ng. Learning 3-D scene structure from a single still image. In *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2008.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2000.

J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2006.

R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and fast—but is it good? evaluating non-expert annotations for natural language tasks. In *Proc. of Empirical Methods in Natural Language Processing (EMNLP)*, 2008.

D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening LP relaxations for MAP using message passing. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2008.

A. Sorokin and D. Forsyth. Utility data annotation with Amazon Mechanical Turk. In *the First IEEE Workshop on Internet Vision CVPR*, 2008.

R. Strandh and J.-O. Lapeyre. An efficient union-find algorithm for extracting the connected components of a large-sized image. Technical report, Laboratoire Bordelais de Recherche en Informatique, 2004.

C. Sutton and A. McCallum. Piecewise training of undirected models. In *Proc. of the Uncertainty in Artificial Intelligence (UAI)*, 2005.

G. Y. Tang. A discrete version of green's theorem. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 1982.

B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *Proc. of the International Conference on Machine Learning (ICML)*, 2005.

D. Thirde, M. Borg, J. Ferryman, F. Fusier, V. Valentin, F. Bremond, and M. Thonnat. A real-time scene understanding system for airport apron monitoring. In *Proc. of the International Conference on Computer Vision Systems (ICCVS)*, 2006.

A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *Proc. of the IEEE Conference of Computer Vision and Pattern Recognition (CVPR)*, 2004a.

A. Torralba, K. P. Murphy, and W. T. Freeman. Contextual models for object detection using boosted random fields. In *Advances in Neural Information Processing Systems (NIPS)*, 2004b.

A. Torralba, A. Oliva, M. S. Castelhano, and J. M. Henderson. Contextual guidance of eye movements and attention in real-world scenes: the role of global features in object search. *Psychol Rev*, 113(4):766–786, October 2006.

E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.

R. Y. Tsai. An efficient and accurate camera calibration technique for 3D machine vision. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 364–374, 1986.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector learning for interdependent and structured output spaces. In *Proc. of the International Conference on Machine Learning (ICML)*, 2004.

Z. Tu. Auto-context and its application to high-level vision tasks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

Z. Tu, X. Chen, A. L. Yuille, and S.-C. Zhu. Image parsing: Unifying segmentation, detection, and recognition. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2003.

Z. Tu, X. Chen, A. L. Yuille, and S.-C. Zhu. Image parsing: unifying segmentation, detection and recognition. *International Journal of Computer Vision (IJCV)*, 2005.

P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision (IJCV)*, 2001.

P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision (IJCV)*, 57(2):137–154, 2004.

M. Wainwright, T. Jaakkola, and A. Willsky. MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Trans. on Info. Theory*, 2005.

D. Waltz. Understanding line drawings of scenes with shadows. In *The Psychology of Computer Vision*. McGraw-Hill, 1975.

J. Winn and N. Jojic. LOCUS: Learning object classes with unsupervised segmentation. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2005.

J. Winn and J. Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2005.

C. Wojek and B. Schiele. A dynamic conditional random field model for joint labeling of object and scene classes. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2008.

L. Yang and F. Albregtsen. Fast and exact computation of cartesian geometric moments using discrete green's theorem. *Pattern Recognition*, 29:1061–1073, July 1996.

L. Yang, P. Meer, and D. J. Foran. Multiple class segmentation using a unified framework over mean-shift patches. In *Proc. of the IEEE Conference of Computer Vision and Pattern Recognition (CVPR)*, 2007.

Y. Yang, S. Hallman, D. Ramanan, and C. Fowlkes. Layered object detection for multi-class segmentation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

B. Yao, X. Yang, and S.-C. Zhu. Introduction to a large scale general purpose groundtruth dataset: methodology, annotation tool, and benchmarks. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, 2007.

C.-N. Yu and T. Joachims. Learning structural SVMs with latent variables. In *Proc. of the International Conference on Machine Learning (ICML)*, 2009.