GRAPHICAL MODELS FOR HIGH-LEVEL COMPUTER VISION

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

Geremy Heitz
March 2009

For my family and Roxana.

Acknowledgments

I would like to begin by thanking my advisor Daphne Koller. She has provided me with several years of guidance, teaching, and support. She has taught me how to approach a problem, what questions to ask, how to measure progress in a research project. I have learned a lot from Daphne's extensive experience about presenting or selling a piece of work in papers, posters, and oral presentations. Her creativity, knowledge and drive has been an inspiration to me, and the example she has provided will stick with me long past my time at Stanford as I venture out into the world. In the end, her help in shaping my projects and this thesis were a large part in me getting through this daunting undertaking of getting a Ph.D.

My first graduate advisor, Calvin Maurer, was instrumental in keeping me on the academic track toward my degree. When my early research projects were slow to develop and I wasn't sure what I was doing, he was very positive and helped me gain the confidence that I would need when I moved to a new group in a new field under Daphne.

To the members of my thesis committee, thank you for being flexible in accommodating me. Sebastian Thrun has had many other things on his plate while I've been bugging him for advice and scheduling surrounding reading my thesis and attending my defense. Despite this, he has always been very helpful and supportive. Much thanks to Andrew Ng for his advice about when and where to submit my papers and his laid-back approach to the thesis process. I would also like to thank Trevor Hastie for chairing my defense committee without even knowing me!

Other than Daphne, the biggest influence during my Ph.D. research was Gal Elidan. When I was lost in the group without a project, reading papers day in and day out, but not trying anything of my own, Gal convinced me to jump in and try *something*. We struggled and fought for three years on the LOOPS work, and finally, through our blood, sweat, and tears, managed to get it accepted. He gave me great perspective on research: sometimes things work out right away, but sometimes it's a long drawn-out struggle. His perpetual (outward) optimism kept me fighting even when everything seemed to be going wrong.

The other two members of the coffee crew, Ben Packer and Steve Gould, have also been brave comrades in this battle. To Ben, thanks for being ready to talk each and everyone one of the 3,449 times that I turned around in my chair and called your name over the past three years. Saving your life that one fateful morning before the NIPS deadline actually turned out to be a good decision after all. And Steve, the wise sage from Australia, has been a great technical mentor for me. His work efficiency, coding competency, extensive knowledge, and analytical brain have been a great resource, and he taught me (or helped me learn) many of the little things that make working on projects a little bit easier. In addition to helping me get through the past four years, Gal, Steve, Ben and I have solved many of the problems of the world while spending on the order of $5,000 at the Clark Center Peet's.

In addition, I've also collaborated in research with a number of people both within DAGS and without. At the beginning of my work in DAGS, Drago Anguelov was an important factor in my research getting going. Even starting in a brand new group, working with Drago was very easy and he was always laid-back and understanding of early slow progress. Ben Taskar also provided some important guidance in those early days. Some early work with Gal Chechik, Pieter Abbeel, Jim Rodgers, Gary Huang, Brian Noguchi, Joni Laserson also paved the way for my later work that appears in this thesis. Thanks to Ashu Saxena for his work on the CCM paper. Thanks also to the other people that provided me with interesting discussions, helpful lessons in the areas of machine learning that I didn't know as much about, and open ears to my presentations and constant questions. For this, I would like to thank: Suchi Saria, Su-in Lee, Haidong Wang, David Vickrey, Varun Ganapathi, Alexis Battle, Vasco Chatalbashev, Evan Rosen, Rick Fulton, James Diebel, Rahul Biswas, Farshid Moussavi.

I would like to thank my parents, George and Betsy, for encouraging me to focus on education while I was growing up and supporting my decision to pursue my graduate degree. I hope I have made you proud! Last but not least, thanks to Roxana, my wife-to-be. While the work that has gone into this thesis has sometimes been painful or tedious, my life with you has always been happy and full of love. Your smiles and support have really helped me get through this journey.

Abstract

The goal of image understanding has long been a shared goal in the field of computer vision. Extracting the required scene-level information from image data is a formidable task, however. While the raw data comes in the form of matrices of numbers, the inferences that we must perform occur at a much higher level of abstraction. Much progress has been made in recent years in extracting the primitives of an image in isolation, for example detecting the objects, labeling the regions, or extracting the surfaces. Modeling the interactions and fine-grained distinctions between these primitives is an important next step along the path to scene understanding. Because this involves reasoning about relationships between heterogeneous entities at a high level of abstraction, this problem lends itself well to the tools of probabilistic graphical models.

In this thesis we consider two important challenges in this space. In the first, we model interactions between primitives of various types in order to capture the contextual relationships between them. For example, we can expect to find a sheep (a detected object) more often on a field of grass (a labeled region) than on a patch of water. By modeling the interactions between these components, we can expect to improve the quality of classification by leveraging these contextual cues. We first introduce Cascaded Classification Models (**CCM**), a flexible framework for combining various state-of-the-art vision models in a way that allows for improved performance of each model. We next consider the tasks of object detection and region labeling and develop a more sophisticated probabilistic model aimed at capturing the contextual relationships between these types of primitives in a more targeted and meaningful way. Our Things and Stuff (TAS) context model learns to leverage contextual cues directly from data.

Following this exploration of interactions *between* objects, we consider the interactions of “parts” *within* an object, and in particular tackle the problem of descriptive querying of objects. This involves making distinctions about the object at a refined level, beyond mere categorization. For example, we may want to know whether a cheetah in an image is running or standing still. We introduce a probabilistic deformable shape model (LOOPS) and a method for matching this model to an image that allows for precise localization of several object classes. Using this localization, we show how descriptive distinctions can be drawn with a small amount of training data.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Computer vision is the study of designing machines that see, or interact with the world through visual input. Some of the applications that computer vision addresses include medical imaging, image search, computational photography, data mining, robotics, surveillance, etc. These tasks range greatly in complexity, from a simple surveillance task of determining whether an intruder is blocking an LED emission from being received at a detector, to a difficult robotics task such as navigating a robot through a busy environment. While these applications arise in a variety of areas, they share many commonalities. In many of them, an important subgoal is to extract the abstract information of what is happening in the scene. This challenge of image understanding has been a shared motivation in the field of computer vision since computers were first able to process images.

Extracting this level of information from image data is a formidable task, however. While the raw data comes in the form of matrices of numbers, the inferences that we seek occur at a much higher level of abstraction. Based on this intuition, many researchers in the field have posited a hierarchical architecture for solving this problem. One such representation includes the concepts of early vision, mid-level vision and high-level vision. Early vision involves processing of the raw pixels of an image to produce useful features, mid-level vision involves making inferences over groups of these features to determine the composition of the image in terms of objects, surfaces, and geometry, and high-level vision involve producing a symbolic representation of

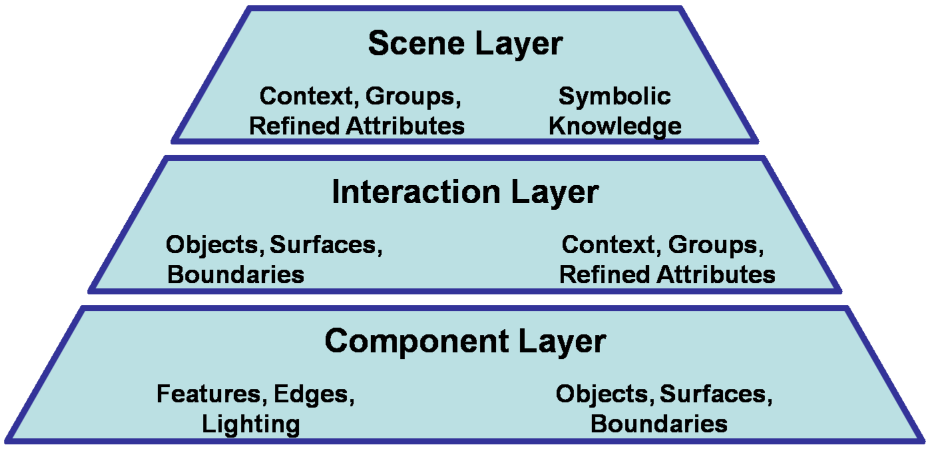

Scene Layer
Context, Groups, Refined Attributes
Symbolic Knowledge
Interaction Layer
Objects, Surfaces, Boundaries
Context, Groups, Refined Attributes
Component Layer
Features, Edges, Lighting
Objects, Surfaces, Boundaries

Figure 1.1: Image understanding hierarchy.

the image.

Another formulation of this hierarchy, which will motivate the development in this thesis, is depicted in Figure 1.1. At the lowest level of the hierarchy is the **Component Layer**, which involves processing the image at the feature level. This involves modeling the edges, local image features, or illumination of the image. The goal of analysis at this level is to produce decisions about what objects are contained in the image, which pixels correspond to these objects, where the boundaries are, what the major structures are, etc. These are the primitives on which scenes are built, and much work has gone into solving these tasks in isolation, with separate models and methods for producing each type of primitive.

The second level of the hierarchy is the **Interaction Layer**. This layer incorporates both the features of the image and the primitives extracted at the **Component Layer**. It is at this level that isolated structures are combined to form coherent reconstructions of large parts of the scene. Contextual relationships between objects will also be incorporated at this level, allowing modeling of the cooccurence and spatial relationships between objects. Finally, it is at this level that more refined characterizations of the various objects and structures can occur, based on the relationship between parts of the objects or the relationship between the object and it's environment.

The ultimate goal is represented by the top level of this hierarchy, the **Scene**

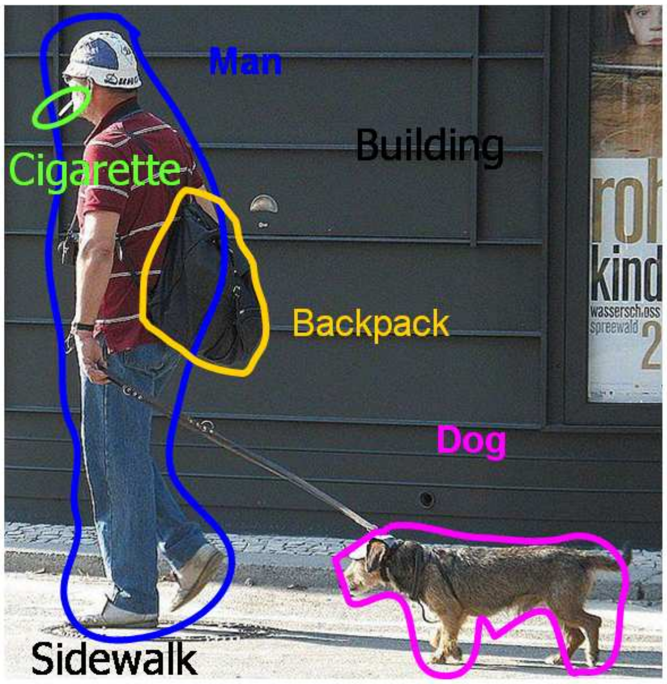


Figure 1.2: What is scene understanding?

Layer. This is the holy-grail of image understanding, where the objects, structures, relationships, reconstructions, and refined characterizations of objects are combined to produce a symbolic representation of the scene. A system capable of solving this layer can extract abstract knowledge from these images.

Figure 1.2 shows an example input image into system that implements this paradigm. The Component Layer converts the raw pixel data into detections of the man and the dog in the image. The Interaction Layer might model the fact that the man and dog are walking on a sidewalk, both in the same direction, and the man is leading the dog by a leash. Finally, the Scene Layer is responsible for mapping these objects and relationships into knowledge about the image, perhaps in the form of a sentence like, “The man is walking the dog.”

The bulk of this thesis focuses on models that fall in the Interaction Layer of the scene understanding hierarchy. The tools that are most appropriate for producing innovations at the Component Layer are based on feature engineering, intelligent image processing, and “flat” machine learning methods focused on efficiently learning

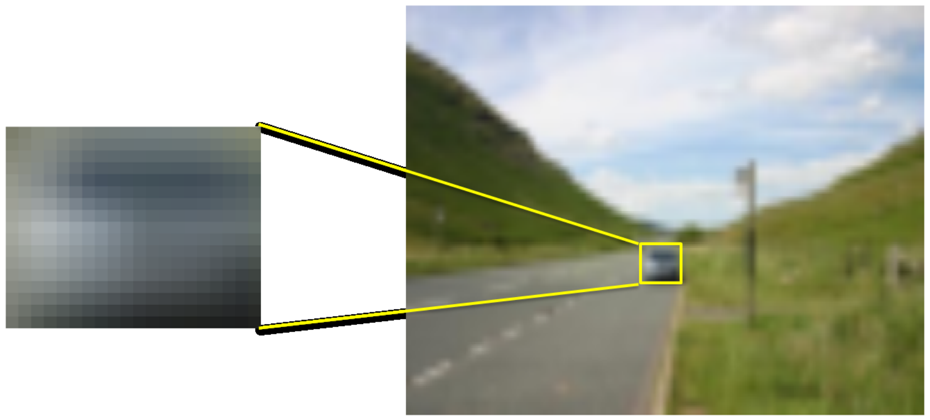


Figure 1.3: What is the object on the left? On it's own, this looks like it could be many different things, including a face, mug, or car. By taking into account the context of the whole scene, including the road passing through the image, we increase our belief that the object is a car.

the statistics of image features toward independent classification decisions. The Interaction Layer, on the other hand, often involves reasoning about entities at a higher level of abstraction. These might include the identities of objects that we have found, or the spatial layout of the parts of these entities. The relationships between objects in an image depend in probabilistic ways on the structure of the scene, the types of objects present, and the activities being performed. This complex web of interactions lends itself to more sophisticated probabilistic modeling. As a result, one of the more useful tools in this domain is the probabilistic graphical model. In this thesis we will use the formalism of probabilistic graphical models to solve some of the problems at this middle level of the scene understanding hierarchy.

We build on many state-of-the-art methods developed over the past several years to introduce models that produce refined characterizations of objects and models that combine the base sub-components to allow for improved effectiveness in solving these sub-tasks. While at the end of the day these models and capabilities will be judged by their role in producing useful inputs to the Scene Layer, we do not pursue this very difficult final level.

As described above, one of the important goals of models at the Interaction Layer is the combination of the base components into contextual groups. Intuitively, we believe that these components interact in ways that inform each other (e.g., in Figure 1.3, the presence of a road informs our belief about what the object is), and that produce higher level constructs such as a group of cars together on the same road. We argue that the interaction between these various components are perhaps most naturally characterized in terms of "context relationships," where the outputs of one module (model for a particular task) can be thought of as inputs of context cues for the other modules. For instance, our region labeling module might indicate that the bottom-left quadrant of the image in Figure 1.3 contains mostly road, which allows our object detector to be more confident in predicting that the object in the middle of the image is a car, and together these two increase the chances that our scene categorizer will call this a street scene.

We begin our exploration with the development of a flexible, straightforward model for combining arbitrary component modules, where each produces one or a small number of primitives. This framework uses black-box classifiers for the various modules and sets of training data that are labeled with groundtruth values for each module. The idea here is to develop a single global model that combines these components in a way that they receive mutual benefit from each other. In Chapter 4, we describe our method for **Integrating Vision Models** using "Cascaded" Classification Models (CCMs), which share information through a simple input/output interface that allows each component to operate as a black box.

Following this, we reconsider the goal of modeling context with the aim to more faithfully capture the probabilistic interactions occurring in real scenes. We explore this direction with two component modules, object detection and region labeling. In particular, if we want to improve the quality of our object detection, for example, we might try to learn to use the surrounding image regions to improve the accuracy of our detector outputs. In Chapter 5 we develop a probabilistic model for **Learning Spatial Context** which ties these two pieces together by discovering which contextual relationships to leverage. Our Things and Stuff (TAS) model allows a supervised object detection model to leverage a clustering of image regions toward a context aware object detection system.

Even if we were to successfully identify all of the primitives in a scene, there are many times that we require a more refined characterization of the objects in a scene in order to make distinctions that will be important for scene understanding. This implies that another important goal of the Interaction Layer is to produce refined characterizations of the components in the scene. In Chapter 6 we explore methods that perform these characterizations.

Our goal is to capture actions, or the shape or appearance properties, of a single object, based on the spatial relationship of the parts of the object. To differentiate this approach from classifying the category of an object, we call this procedure **Descriptive Querying** of objects. A descriptive query is defined to be one that considers characteristics of an object that vary between instances within the object category. Towards this goal, we describe a corresponded object model representation and method (LOOPS: Localizing Object Outlines using Probabilistic Shape) for matching an object model to an image and answering descriptive queries about the object.

This thesis begins with introductions to the tools that are used, including the formalism of probabilistic graphical models and the basics of high-level computer vision. Following this, a series of methods are developed for addressing problems at the Interaction Layer of the image understanding hierarchy. We begin with the development of the **CCM** framework which flexibly integrates many vision models in a way that mutually benefits each of them. The Thing and Stuff (TAS) context model is then developed, in order to explore the proper modeling of spatial contextual interactions between the labeling of image regions and object detection. We conclude our model development with a description of our corresponded object model for descriptive querying, which addresses questions that are at the heart of refined characterization of a scene. Finally, we discuss future directions related to this work.

1.1 Contributions

There are three primary contributions in this thesis:

1. **Integrating models for vision sub-tasks**. We propose a flexible framework (CCMs) for integrating arbitrary vision sub-task models into a joint system that

allows each to benefit. While previous works have explored the combination of particular tasks, our framework allows various models to be included in the framework under some very mild assumptions about their input/output functionality. This allows designers to leverage context using state-of-the-art models for the various sub-tasks, without needing the expertise of understanding the inner workings of the included methods.

2. **Learning Spatial Context**. We develop a probabilistic model (TAS) that links object detection with region labeling. Unlike existing methods, the TAS model allows learning of context when the region labels are unknown (unsupervised). As a result, our model "discovers" what categories of regions it should use to leverage contextual cues. The detections and regions are coupled in a Bayesian network through contextual relationship variables, which allow probabilistic influence to flow between the two sub-tasks. Furthermore, we introduce a structure learning approach based on the standard structural EM algorithm for selecting which spatial relationships should be considered.

3. **Descriptive Querying of Images**. We address the problem of localizing objects in instances to a degree that allows for answering questions beyond the category level. Our registration algorithm corresponds a LOOPS model, which is learned from supervised outlines of instances from the target class, to a new test image. Once the object has been localized and corresponded, we show that many classification tasks that rely on the shape or pose of the object can be more easily answered.

1.2 Thesis Outline

Here is a summary of the chapters:

Chapter 2 Probabilistic Graphical Models. This chapter outlines the basics of graphical model formulations of probability distributions. It begins with a description of the Bayesian Network representation as well as the Markov Random Field and Conditional Random Field representations. Methods for performing inference in such models are described, as well as algorithms for learning the structure and parameters for these models.

Chapter 3. Fundamentals of High-Level Computer Vision. This chapter introduces the current state-of-the-art in high level computer vision. It illustrates the types of models that are used, as well as the applications and data to which they are applied.

Chapter 4 Integrating Vision Models: Cascades of Classification Models Toward the goal of combining the sub-tasks necessary for holistic scene understanding, we develop a general framework for combining "black-box" models for the various tasks into a whole that allows the modules to benefit from the inclusion of each other. This framework seamlessly handles arbitrary annotation of the data and the inclusion of varied datasets.

Chapter 5 Learning Spatial Context: Using Stuff to Find Things. Here we consider the case where we want to use contextual cues, but we have a scarcity of labeled data, including only objects annotated with bounding boxes. We show how to learn a model over image regions in an unsupervised way that still allows for the inclusion of contextual cues for object detection. Results indicate that leveraging this type of context can lead to large improvements in detector performance.

Chapter 6 Descriptive Querying of Images. This chapter considers more fine-grained classification than mere object detection. We show how to learn a corresponded outline model for various object classes that operates on the bounding box detections from the previous methods. For each object, the model is registered to the image of the object and our system considers questions about the object's pose, articulation, etc.

Chapter 7 Future Directions The thesis concludes with a summary, and a consideration of future extensions of our work, including open questions that remain.

1.3 Related Work

The field of computer vision gained steam in the 1970's, when computers first became powerful enough to acquire and process image data. Through the years, as large databases of digital images have become more common, the field has progressed and expanded in many directions. In this section, we discuss the most relevant papers to the broad topics that will be covered in this thesis.

1.3.1 Early Attempts at Scene Understanding

The idea of "understanding the scene" was one of the early concepts in computer vision [11]. Groups of researchers in the field developed large systems, such as the VISIONS system of Hanson and Riseman [61], with the goal of solving the entire problem in one bottom-up system. Many authors believed that this bottom-up approach, based on parsing the image into primitive descriptions of the gray level changes, followed by higher level processing, was the correct approach for understanding of the scene. Marr [89] calls this primitive, intermediate description of the image the *primal sketch*. As a result of this belief, a great deal of effort was put towards finding the correct representation for visual primitives [90]. These works typically operated on line drawings of scenes (rather than natural images of scenes) in order to "reconstruct" the shape of the scene by parsing the various edges and junctions that were found [87, 133]. The work of Waltz [133] exemplifies this line of research. An example image that this method operates on is shown in Figure 1.4. The so-called Waltz's algorithm defines a large library of junctions between sets of lines, and uses a constraint propagation technique to find a description of the scene that both makes sense in 3D, and respects the set of junctions observed in the image.

While the ambition of these techniques is impressive, they were never ever to scale to complex scenes, and certainly not to natural images, where even extracting the edge structure requires a great deal of work and the result will always be noisy. In an initial attempt to overcome some of these problems, Rosenfeld et al. [108] extended the work of Waltz [133], solving the same problem with an algorithm that used parallel relaxation updates on pairs of entities (such as junctions). Their method also allowed for fuzzy interpretations of scenes which incorporate both noise, non-deterministic signals from the data, and user preferences. More recently, Mohan and Nevatia [92] used a recursive grouping scheme to "perceptually organize" the observed edges in an

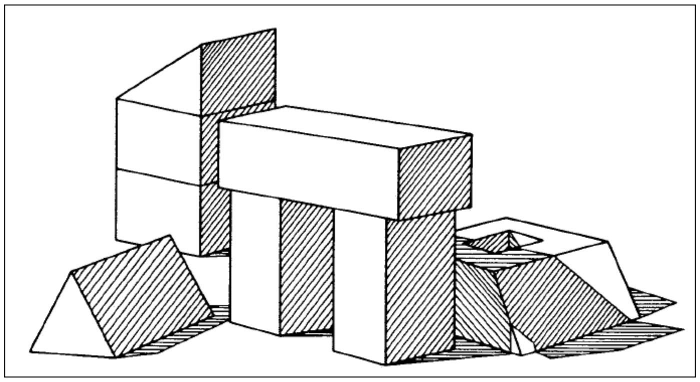


Figure 1.4: An example line drawing with shadows. Such an image would be the input into the scene understanding algorithms of Waltz [133] or Mackworth [87].

image, in the hopes of extracting the various objects in the scene. They used simple natural scenes, and the noisy edgemaps produced from them. While this approach has not seen a lot of work lately, recent advances in features and probabilistic modeling techniques may allow more progress to be made in this direction.

In a parallel line of work, "generalized cylinders" were developed by Nevatia and Binford [96]. This approach represents solid objects as groups of geometric primitives, in an attempt to reconstruct the "shape" of objects, in a similar vein to how the above approaches tried to reconstruct the shape of a scene. While this work focused on relatively complex objects that occur in the real world (such as people, hammers, gloves, etc.), the underlying principle is the same as the scene reconstruction methods. Both target the production of a true-to-life 3D reconstruction of the surfaces and objects in the image. Rather than a sparse labeling of the image, a correct answer decomposes the image into the set of structural elements and their poses. This idea is described in more detail by Biederman [10], who argues that this makes sense as a model for how the human visual system parses a scene.

An alternative decomposition, based on "intrinsic images" was proposed by Barrow and Tenenbaum [6]. In this model, maps of various image properties (depth, reflectance, color) are computed from the image. The assumption is that the final observed image is produced through the interactions of the various intrinsic images, each of which is the same size as the original image, but rather than each element being an intensity, it is instead a quantification of some other property of the image, such as the distance from the camera, the amount of reflectance of the surface at that point, or the pigment (color) of the surface. The original work was on very simple images (as above), but has been extended more recently by Tappen et al. [123] to work on natural images.

The work in this thesis, and indeed most recent work in scene understanding, is more similar to this intrinsic image approach, where the various components of an image are understood as interacting groups of entities or features, rather than requiring a full 3D explanation for every pixel.

1.3.2 Recent Work on Image Understanding

Over the past few decades, great progress has been made in the processing and classification of large sets of natural images. One of the primary contributors to this progress has been the widespread use of machine learning techniques to make sense of the large quantity of noisy data. Human understanding of images comes in very abstract terms, while the raw data we get from imaging sensors are concrete, quantitative measurements of light. We discuss this disconnect more at the beginning of Chapter 2. From this insight, strong probabilistic methods have been developed to solve many pieces of the scene understanding problem.

In the past five years, researchers have returned to the bigger picture problem, armed with the knowledge and successful methods for the sub-parts of the problem. In 2003, Murphy et al. [93] introduced a graphical model approach to jointly reason over the type of scene in an image, as well as the presence and location of the various object types. Other researchers followed this with more sophisticated models over subsets of the variables that we might be interested in for scene understanding. Kumar and Hebert [76], for example, develop a large MRF-based probabilistic model linking multi-class segmentation and object detection. Both pixel labels and object locations are sites in the MRF, and potentials between them enforce believable layouts of regions and objects. Tu et al. [130] also consider these two primitives (segment labeling and object detection), introducing a generative model for the image. Parsing an image (i.e., performing inference) is accomplished with an MCMC-based algorithm.

Following up on this line of work, Hoiem et al. [70] showed that considering properties of the camera and the typical orientation of natural scenes allowed for a model that determined the likely sizes and locations of instances from various object classes in an image. This led to a large improvement in detector performance. Leveraging this discovery, and returning to the paradigm of intrinsic images described above, the authors, in later work [71], developed an approach that integrated object detection, surface estimation, camera viewpoint reasoning and occlusion boundary classification toward the goal of full scene interpretation.

Much of this work was developed in parallel with the work in this thesis, and validates our idea that solving these problems simultaneously is a promising direction. The work in this thesis explores similar ideas to these works, with a heavy focus on the contextual relationships between the primitives. The momentum in the field shows that this area needs, and will receive increasing attention in coming years.

1.3.3 Refined Characteristics of Objects

A vast majority of the work on classifying "objects" in images considers either identification of a particular object or determining the category to which particular object belongs. Much less common are works that consider other characteristics of objects. We briefly review some of this work below, including research that targets human pose estimation, shape analysis for medical research, and the modeling of visual attributes for deeper analysis of objects.

Perhaps one of the most common lines of work is the problem of identifying the pose of people in images. Typically, these methods use a probabilistic parts-based approach. Lee and Cohen [79] uses such a model over the pose of a body, and achieves relatively fast inference through a smart proposal distribution in an MCMC inference scheme. Ramanan and Sminchisescu [106] shows that training such a model to optimize the conditional likelihood is more effective than using maximum likelihood, allowing the model to discriminate better between very similar and very distinct poses across a large dataset. Wang and Mori [134] shows that inferring over multiple tree structures over the human body parts produces superior results to a single tree, as it is able to capture constraints between parts that are not directly connected in the standard human body tree structure. Balan et al. [4] takes a different approach, matching a full 3D graphics model of a human to test images using a stochastic search procedure. The SCAPE model [2] that he uses models both pose deformation and body shape deformation to cover the full space of human shapes in images. In Balan and Black [3], he extends this to the case of loose-fitting clothing, localizing the underlying naked body using constraints from a mere four views of the person. In addition to this work, there are many papers that localize people in videos, both for tracking [105] and for classifying the action that the person is performing [135].

In the medical imaging community, the class of the object that is being imaged is usually known beforehand. The goal, generally, is to classify other aspects of the imaged organ, including various pathological conditions. Often, this can be accomplished by some analysis of the shape of the object. Visual measurement of the shape of the vocal cords in video sequences, for instance, allow for the accurate localization of atrophied vocal fold tissue [95], which may allow for automated diagnosis or aid in planning of surgery. Many works measure various aspects of the brain in order to understand disease processes. Measurement of the hippocampus, for example, have shown that victims of childhood sexual abuse have a statistically significantly smaller left hippocampus (in volume) than controls [17]. Similarly, f. Mangin et al. [41] used shape analysis of the brain cortical sulci, developing sophisticated descriptors based on 3D shape invariants, to discover statistically significant differences based on gender and handedness.

In the computer vision community, there has been recent interest in attributes of objects in images that go beyond the mere category of the object. This idea is expressed explicitly in Ferrari and Zisserman [48], where the goal is to model concepts such as ‘red’ and ’striped,’ toward the eventual refined characterization of objects. In addition, some work has addressed the idea that leveraging adjectives or properties of objects can give improved recognition or detection performance [60]. While this line of work is still in its infancy, there is reason to believe that continued exploration in this direction will yield important results.

The work in Chapter 6 of this thesis uses shape-based analysis of objects in images to classify either the pose or shape characteristics of objects. It is a natural extension of these three types of refined characterizations, that should help in a more holistic understanding of the object that looks beyond the mere category membership.

1.4 Previously Published Work

A majority of the work presented in this thesis has been previously published in the form of conference proceedings or journal articles. The LOOPS model and descriptive querying of objects using the LOOPS model (material in Chapter 6) was published in Heitz et al. [64] and is currently in submission at the International Journal of Computer Vision. An early version of the model and the pipeline for learning the shape model from cartoon images was published in Elidan et al. [38]. The Cascaded Classification Model framework that is presented in Chapter 4 was published in Heitz et al. [65]. Finally, the Things and Stuff Context model described in Chapter 5 was published in Heitz and Koller [66].

Chapter 2

Probabilistic Graphical Models

One of the major advancements in computer vision over the past 20 years was the realization that statistical or probabilistic methods were necessary to attack many of the problems in the field. In particular, there was a realization that a large gap exists between the "data" in the form of an image and the abstract representation of an image that is present in human brains (imagine having to figure out whether Figure 2.1 is a picture of a woman from a huge list of 320×240 numbers). Because we are aware of a very high-level signal present in the data (it's a picture of a woman), but we have no intuitive concept of how to extract that signal, statistical methods are a natural fit.

Some of the statistical analysis focuses on the image pixels themselves. Figure 2.2, for example, shows histograms of the pixel intensities for two types of images, faces and beach scenes. Even this simple analysis demonstrates that statistical signals exist within the matrix of numbers. In the next chapter, we describe several representations of image features that occur at the level of single pixels or small groups of pixels. Typically, the probabilistic methods used to represent and understand these image features are heavily engineered to be maximally useful for further levels of processing. Some of the typical image features and the statistical methods used to model them are presented in Chapter 3.

Much of the information present in images, however, lies at a higher level of abstraction. Indeed, when an observers looks at an image, he sees a number of

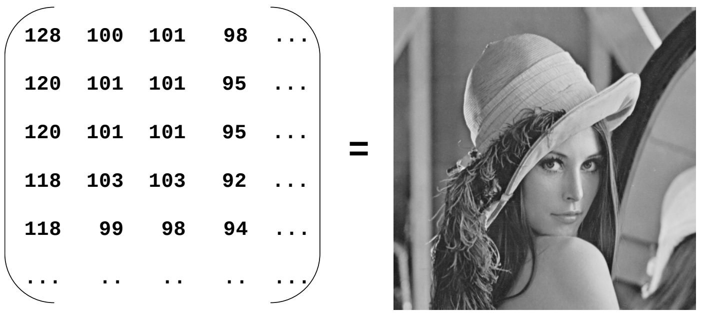


Figure 2.1: The famous "Lena" image is represented by the computer as a large matrix of numbers. The task of computer vision is to make sense of these numbers.

objects made up of a bunch of surfaces composed of materials with a set of salient visual textures. This situation provides a rich playground for deep statistical methods over a number of unknown entities with a complex web of dependencies. Representing information and distributions over this complex web of random quantities is well matched by the formalism of *Probabilistic Graphical Models* (PGMs). These models provide principled representations, notation, and visualization for domains with many random variables that interact in probabilistic ways.

In this chapter we review some of the basics of graphical models, beginning with definitions of Bayesian networks and Markov networks that assume a basic probability background. These definitions are followed by a description of techniques for probabilistic inference, which determines probabilities for outcomes given observed evidence, and of model learning, where the underlying probability distribution must be estimated from data. The chapter concludes with a presentation of the basics of the Gaussian distribution, which plays a central role in the development of this thesis.

2.1 Representing Probability Distributions

The goal of a PGM is to encode a probability distribution over a set of random variables $\mathcal{X} = \{X_1, \ldots, X_N\}$. In general these variables may be discrete (take one of a predefined set of values) or continuous (take one of an infinite number of real values). For the purposes of this development, assume that the variables X_i are discrete, but most of the definitions extend easily to the continuous case. Define the

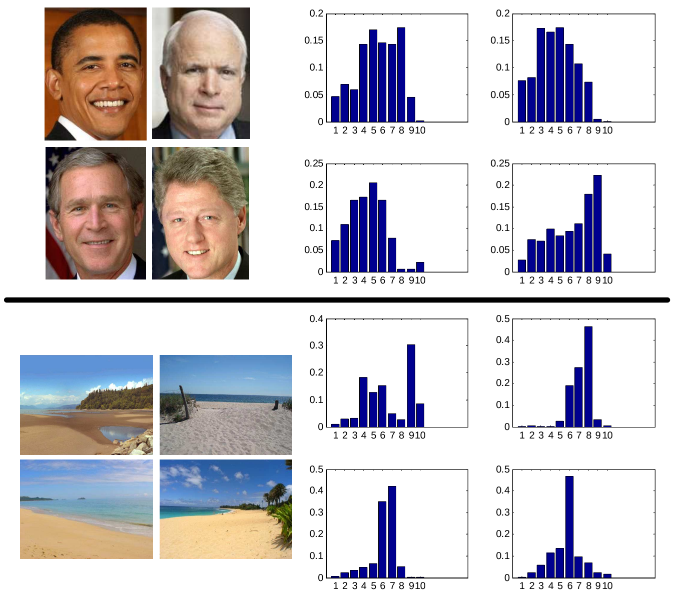


Figure 2.2: Example histograms of the intensities of the pixels in images. Even from just this simple statistic, we can already see a difference between the two classes.

domain $dom(X_i)$ of X_i to be the set of all values x_i that X_i can take.

Now suppose that we are interested in representing the probability over all of the outcomes in the entire space. Denote this probability by $P(X_1, X_2, \ldots, X_N)$. In general, this probability distribution can be represented as a large table with a real number for every possible outcome, such that these numbers sum to 1. With N variables, and $K = |dom(X_i)|$ possible values for each, this requires specifying K^N numbers. Because they must sum to 1, we have $K^N - 1$ "free" parameters (i.e., the last value is fully determined by the first $K^N - 1$ values). This representation is called a *probability table*, and obviously, for even modest values of N and K, is too large to be practical for representing the entire joint distribution. This raises the question of whether there are more compact representations of joint distributions, which require fewer parameters.

In addition to requiring less memory to store, there are other reasons to desire a "smaller" parametrization of our distribution. From a computational perspective, a model with fewer parameters is likely to lead to less costly (in time and memory)

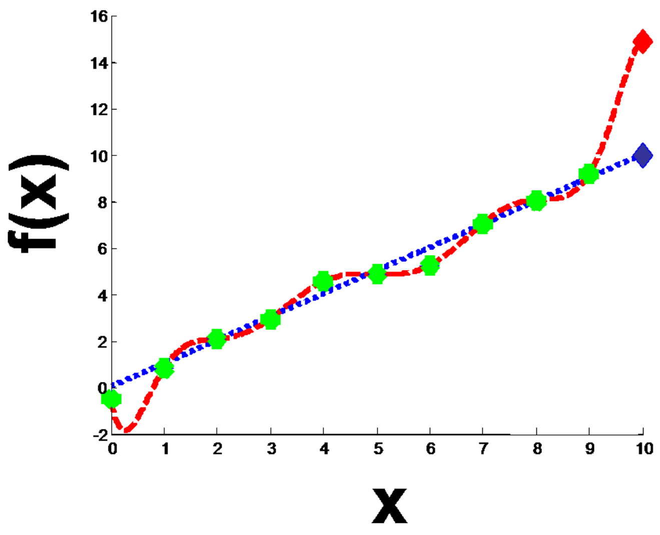


Figure 2.3: The problem of overfitting.

inference and learning algorithms. An even bigger concern is the problem of overfitting. Because one of the main goals of learning a probability distribution is to use it to generalize to unseen data, we want to have a plausible model. Intuitively, if a model has many parameters, then it can "match" any training data by becoming arbitrarily complex. By having fewer parameters, we force our model to be simpler, and thus it is more likely to correctly generalize to new instances. This is a form of *regularization*, in which the goal is to "smooth out" the learned distribution in an attempt to improve it's generalization performance. Figure 2.3 shows an example of this phenomenon. The goal is to fit a curve to the green points, which is assumed to be generated by a polynomial function plus white noise. The red curve is a degree 10 polynomial (11 parameters), that exactly hits the green points, while the blue curve is a line (2 parameters) that only "comes close" to the target points. However, because the blue line is simpler (has fewer parameters), we are more likely to believe its prediction of the next point (the blue diamond), rather than the higher order prediction given by the red diamond.

With these concerns in mind, the goal is to reduce the number of parameters required to encode this distribution to a practical level. Recall from basic probability that any joint distribution can be decomposed into a product of conditional distributions according to the chain rule. In this case, we can write:

$$P(X_1, \ldots, X_N) = P(X_1)P(X_2 \mid X_1) \ldots P(X_N \mid X_{N-1} \ldots X_1) \tag{2.1}$$

where $P(\mathcal{Y} \mid \mathcal{Z})$ is a *conditional probability distribution* (CPD). A conditional probability distribution $P(\mathcal{Y} \mid \mathcal{Z})$ is a probability distribution over the variables in the set $\mathcal{Y}$ conditioned on a particular value for all of the variables in the set $\mathcal{Z}$. This can be represented as a *conditional probability table* (CPT), which is a probability table for the variables in $\mathcal{Y}$ for each value of $\mathcal{Z}$.

With this parametrization of the distribution, there are $K-1$ free parameters for $P(X_1)$, and $K*(K-1)$ free parameters for $P(X_2 \mid X_1)$, etc. Because the chain rule is mathematically equivalent to our original representation, the number of parameters must be the same: $K^N - 1$. However, by rewriting our probability in this form, we can take advantage of the fact that X_i may only truly depend on a subset of the variables to the right of the conditioning bar.

Let's look at an example to illustrate this point. Suppose that two people take two photographs $\mathbf{I}_1$ and $\mathbf{I}_2$ of the same scene $\mathbf{X}$ from different angles. Clearly the two images are related, because they are both of the same object (say the Notre Dame Cathedral). But suppose we are interested in $P(\mathbf{I}_1 \mid \mathbf{I}_2, \mathbf{X})$, the probability distribution of the first image given the second and all the details of the scene. The mechanics of this setup tell us that all of the information needed to determine the image $\mathbf{I}_1$ is provided by the scene itself. In this case, knowing the contents of the second image $\mathbf{I}_2$ *provides no new information* about the first, once we know what the scene is. As a result, we might say:

$$P(\mathbf{I}_1 \mid \mathbf{I}_2, \mathbf{X}) = P(\mathbf{I}_1 \mid \mathbf{X})$$

This is an example of the notion of *conditional independence.* More formally:

Definition 2.1.1: Given three sets of variables $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$, we say that $\mathcal{X}$ and $\mathcal{Y}$

are **conditionally independent** given $\mathcal{Z}$ if and only if $P(\mathcal{X} \mid \mathcal{Y}, \mathcal{Z}) = P(\mathcal{X} \mid \mathcal{Z})$ whenever $P(\mathcal{Z}) > 0$. We denote this property by:

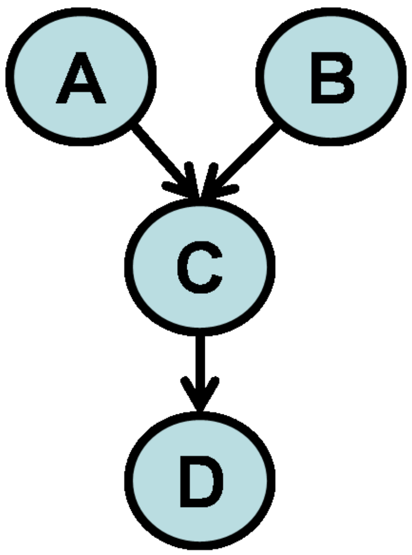
$$(\mathcal{X} \perp \mathcal{Y} \mid \mathcal{Z})$$

∎

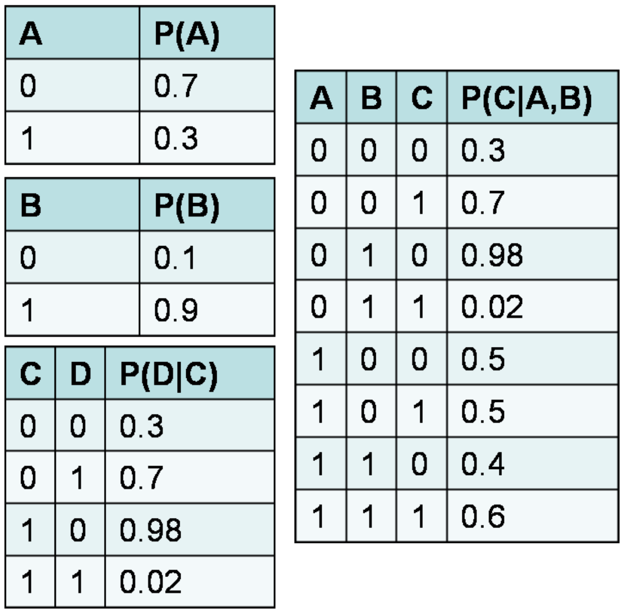
When the set of conditioning variables $\mathcal{Z}$ is empty, we say that $\mathcal{X}$ and $\mathcal{Y}$ are **marginally independent**. It is interesting to note that two variables might be marginally independent, but not conditionally independent given certain conditioning sets $\mathcal{Z}$. For example, suppose that, for a random family, we can assume that the eye color of the mom E_m and of the dad E_d are marginally independent (since they were not genetically related). Thus $(E_m \perp E_d)$ holds. Now you are told that their son has blue eyes ($E_s = blue$). Knowing this fact probabilistically couples E_m and E_s. For example, if you found out that $E_m = brown$, you would be more likely to think that $E_d = blue$ (the son must have gotten his blue eyes from somewhere!). Therefore, we have $(E_m \not\perp E_d \mid E_s)$.

Armed with the notion of conditional independence, we can now find a way to reduce the number of parameters required to describe our joint distribution. If we can identify some conditional independencies present in the probabilities in our chain rule decomposition of Eq. (2.1), we end up with fewer variables to the right of the conditioning bar in these terms. As a result, we will require fewer entries in the CPT for that term. In the end, this leads to fewer parameters needed to describe the distribution.

Probabilistic graphical models are a representation for encoding the conditional independencies and the resulting CPT structures for a joint probability distribution. In the next sections, we consider two common forms of graphical models that will be used throughout this thesis.



(a)



A	P(A)
0	0.7
1	0.3

B	P(B)
0	0.1
1	0.9

C	D	P(D|C)
0	0	0.3
0	1	0.7
1	0	0.98
1	1	0.02

A	B	C	P(C|A,B)
0	0	0	0.3
0	0	1	0.7
0	1	0	0.98
0	1	1	0.02
1	0	0	0.5
1	0	1	0.5
1	1	0	0.4
1	1	1	0.6

(b)

Figure 2.4: An example Bayesian network

2.2 Bayesian Networks

Perhaps the most well-known type of PGM is the Bayesian network (BN). A BN is represented by a pair $(\mathcal{G}, \Theta)$, where $\mathcal{G}$ is a directed acyclic graph (DAG) with the nodes corresponding to the variables that we wish to model. An example graph is depicted in Figure 2.4(a), modeling the four binary variables A, B, C, and D.

The DAG $\mathcal{G}$ visually illustrates the conditional independencies in the distribution represented by the BN. In order to describe these independencies, we must define two concepts. The *parents* of a node X_i in a BN are the nodes X_j that have an arrow beginning at X_j and ending at X_i. The *descendants* of a node X_i are all nodes X_j such that there is a directed path from X_i to X_j in $\mathcal{G}$. The graph structure $\mathcal{G}$ for a Bayesian network encodes a set of *Markov independence assumptions* amongst the variables:

Definition 2.2.1: For a Bayesian network over the variables $\{X_1, \ldots, X_N\}$, the BN graph $\mathcal{G}$ encodes the set of **Markov independence assumptions**: each variable X_i

is conditionally independent of its non-descendants given its parents. More formally:

$$\forall X_i \, (X_i \perp NonDescendants(X_i) \mid Parents(X_i))$$

▮

The other component of a BN is the set of parameters Θ. A BN is parameterized by a CPD for each node (variable) in the network, where the distributions are conditioned on the parents of the variable. Figure 2.4(b) shows some of the CPDs for our example network. The probability for a full assignment to all of the variables in the network is computed using the chain rule:

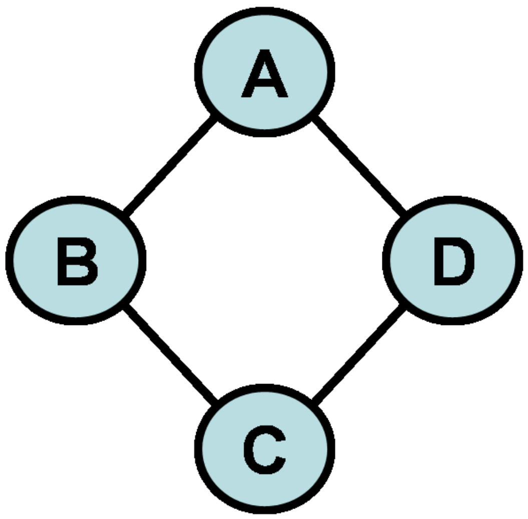
$$P(\mathcal{X}) = \prod_i P(X_i \mid Parents(X_i)) \tag{2.2}$$

In the example of Figure 2.4, this decomposition gives us that $P(A, B, C, D) = P(A)P(B)P(C \mid A, B)P(D \mid C)$. In this case, representing the full joint distribution as a single table would take $2^4 - 1 = 15$ independent parameters. In our BN representation, however, we need 1 free parameter each for $P(A)$ and $P(B)$, 4 free parameters for $P(C \mid A, B)$, and 2 free parameters for $P(D \mid C)$, for a total of 8 free parameters. This more compact representation should require less data to learn accurately, and will allow many inference questions to be answered more easily.

2.3 Markov Networks

The directed nature of the BN representation has many important benefits. Because the idea of causality is intuitive to humans, it is often relatively easy to produce the graph corresponding to a set of variables by identifying the causes that directly effect each variable. The variables that *directly causally effect* a variable X become its parents.

Sometimes, however, we don't want to enforce a directionality to the probabilistic relationships between variables. To illustrate this, we take a study group example. In this example, we have four students, Alice, Bob, Cathy, and David together in the



(a)

ψ(A,B)	B=0	B=1	B=2
A = 0	4	3	1
A = 1	3	6	3
A = 2	1	3	4

ψ(B,C)	C=0	C=1	C=2
B = 0	8	5	1
B = 1	5	2	1
B = 2	1	1	1

(b)

Figure 2.5: An example Markov network. The variables here represent students (e.g., B = Bob), and the values represent grades (e.g., B=0 means that Bob get an ‘A’ in the class). The potentials ψ represent the affinity between two students (e.g., $\psi(B = 0, C = 1)$ is the affinity between Bob getting an ‘A’ and Cathy getting a ‘B’).

same class. They study together in pairs, but the boys don't like each other very much, and neither do the girls. We can represent this study group pattern with the undirected graph of Figure 2.5(a). The nodes represent the students, and the links represent a study pair, a pair of students that study together.

Now suppose we want to put a probability distribution over the grade that each student receives in the class. We have nothing to differentiate the students from each other *a priori*, but we do believe that students studying together will learn the same ideas. As a result, we might expect the grade between two students in a study pair to be correlated. However, there is no clear directionality between a given pair. This situation is naturally encoded as Markov network (MN) or Markov random field (MRF).

In the MRF representation, we encode *affinities* between variables using a potential function:

Definition 2.3.1: A **potential function** or **factor** $\psi(\mathcal{X})$ over a set of random variables $\mathcal{X}$ is a mapping from the joint domain $dom(\mathcal{X})$ to a non-negative real number:

$$\psi(\mathcal{X}) : dom(\mathcal{X}) \rightarrow \Re^{+}$$

▮

A factor is reminiscent of a probability table, in that it has a real value for each assignment to the variables. However, these numbers have no normalization constraint (they don't have to sum to 1), and instead of a probability, the values indicate preferences for certain assignments to the variable in the scope. Some example potentials for the study group example are shown in Figure 2.5(b).

An MRF for the variables $\{X_1, \ldots, X_N\}$ is defined by a pair $(\mathcal{H}, \Psi)$ where $\mathcal{H}$ is an undirected graph over nodes that correspond to the variables, and Ψ is a set of potential functions where the set of variables in the scope of each potential function ψ_c corresponds to a fully connected subgraph of $\mathcal{H}$, called a *clique*. Note that we do not require that *every* clique has a potential function, only that every potential function is over a clique.

Similarly to the Bayesian network representation, we can "read off" some conditional independencies from the graph for an MRF. We call node X_i a *neighbor* of node X_j if there is an edge directly from one to the other. For an MRF, we can define the set of Markov independence assumptions:

Definition 2.3.2: A graph $\mathcal{H}$ for a Markov network over the variables $\mathcal{X} = \{X_1, \ldots, X_N\}$ encodes the following **Markov independence assumptions**: X_i is independent from all other variables given (i.e., conditioned on) the values of its neighbors in $\mathcal{H}$. Formally:

$$\forall X_i (X_i \perp X_{-i} \mid Neighbors_{\mathcal{H}}(X_i)),$$

where X_{-i} indicates the variables in $\mathcal{X}$ other than X_i. ▮

The joint probability distribution for all of the variables in an MRF is given by:

$$P(\mathcal{X}) = \frac{1}{Z} \prod_{c} \psi_c(X_c), \tag{2.3}$$

where $\mathcal{C}$ is the set of cliques, and each $c \in \mathcal{C}$ has scope $\mathcal{X}_c$ and potential function ψ_c. The normalization factor Z is known as the **partition function**, and is required to make the probability sum to 1. The partition function can be computed as:

$$Z = \sum_{\mathbf{X}} \prod_{c} \psi_c(X_c). \tag{2.4}$$

In the study group example of Figure 2.5, we have factors for every pair of students with a link between them. Our joint probability over the variables is given by:

$$P(A, B, C, D) = \frac{1}{Z}\psi(A, B)\psi(B, C)\psi(C, D)\psi(A, D) \tag{2.5}$$

For some problems, it is natural to split our variables into two sets. The first set contains quantities that are observed for both the training and test data. In an image segmentation example, these variables might correspond to the features extracted for each pixel of the image. The other set of variables are the target or classification variables (i.e., the ones we wish to infer from the data at test time). In our image example, these variables might be binary foreground/background labels for each pixel. If we let $\mathcal{X}$ be our feature variables, and $\mathcal{Y}$ be our target variables, what we are really interested in in this case is the distribution $P(\mathcal{Y} \mid \mathcal{X})$.

One option for modeling this situation is to learn a Bayesian network where the variables in $\mathcal{X}$ appear as the roots of the network (have no parents), and the $\mathcal{Y}$ variables are conditioned on the $\mathcal{X}$ variables. However, we often want to have undirected potentials between the Y's, indicating, for instance, that neighboring pixels are likely to share the same label. This situation is well modeled by a *conditional Markov random field* (CRF). A CRF is a special case of an MRF where the potential functions are *conditioned* on the value of the feature variables. This means that each data instance corresponds to a MRF with different potentials. In our image labeling example, we might learn that more colorful pixels tend to be in the foreground. As a result, we might have singleton potentials over each Y_i that favor the foreground label in cases where the corresponding pixel X_i is very colorful.

2.3.1 Log-linear CRF Potentials and Logistic Regression

One of the most common methods for representing the potentials of a CRF is using so-called "log-linear" potential functions. Suppose we have a node Y_i in our CRF with an associated feature vector $\mathbf{x}_i$ (e.g., measurements from the image in the neighborhood of this pixel). We represent the singleton potential function for this variable using the following form:

$$\psi_i(Y_i = y \mid \mathbf{x}_i) = \exp\left(\theta_y^T \mathbf{x}_i\right). \tag{2.6}$$

This formulation is called log-linear because the logarithm of the potential is a linear function of the feature values. If we consider a CRF with a single variable and potential function, we get a discriminative classification model known as *logistic regression*. The logistic regression model is within many of the larger models developed in this thesis. In this specific case, we get a probability for our variable of the form:

$$P(Y = y \mid \mathbf{x}) = \frac{\exp\left(\theta_y^T \mathbf{x}\right)}{\sum_{y'} \exp\left(\theta_{y'}^T \mathbf{x}\right)}. \tag{2.7}$$

2.4 Probabilistic Inference

Once we have a probabilistic model in graphical form, such as a Bayesian network or a Markov network, what can we use it for? The most common use of a PGM is to perform inference. The problem of inference involves answering probabilistic queries about sets of variables, possibly while conditioning on observed values of other sets of variables. The two most common inference queries are *conditional probability* queries, and *maximum a-posteriori* (MAP) queries.

In a conditional probability query, we are asking for a distribution over the target set of variables $\mathcal{Y}$, conditioned on the evidence set $\mathcal{E}$. In particular, the goal is to produce the probability table $P(\mathcal{Y} \mid \mathcal{E})$ that is consistent with the joint distribution encoded in the PGM. In our study group example of Figure 2.5, for example, we might want a distribution over Alice's grade, given that Bob received an **A** ($B = 0$). In probabilistic terms, we want to compute $P(A \mid B = 0)$.

For a MAP query, we are interested in the most likely assignment to our target variables $\mathcal{Y}$ given $\mathcal{E}$. We want to determine $\mathcal{Y}^* = \text{argmax}_{\mathcal{Y}} P(\mathcal{Y} \mid \mathcal{E})$. In our study group example, we could ask for the most likely grade for Cathy given that Alice and Bob both got **B**'s. One way of solving this problem is to perform a conditional probability query over the same set of variables, and simply "look-up" the assignment to the target set that gives the highest conditional probability. In some cases, however, we can develop algorithms that directly target the MAP problem.

In general, the problem of probabilistic inference is NP-hard [22], which means that solving it exactly is intractable for many cases. Indeed, in many real world applications, obtaining exact posteriors or MAP assignments is often too expensive in memory or in time (or both). As a result of this, many computationally feasible algorithms have been developed that give an *approximate* answer to inference queries. In the next two sections we briefly describe two such methods that will used in the development of this thesis.

2.4.1 Belief Propagation

The Belief Propagation (BP) algorithm was introduced by Pearl [101], and is based on the idea of *message passing.* The algorithm proceeds by passing messages between nodes, where the message indicates how much each node should update its belief based on the neighboring node's current information. In this manner, beliefs propagate (hence the name) across the network until all nodes agree on the messages they are sending.

In tree structured networks, the algorithm performs exact inference, obtaining the true marginals for each variable and pair of variables with an edge between them. This set of networks includes Bayesian networks where each node has at most one parent, and singly-connected Markov networks (at most one path exists between any two variables).

In addition, for networks that do not fit into this category, BP has been empirically shown to give good performance [39], and recent theoretical work has justified the method as well (see Yedidia et al. [141]). We will briefly describe a variant of BP that operates on pairwise Markov networks. In pairwise MNs, all potentials are over at most two variables. It can be easily shown that any BN or any non-pairwise MN can be converted into a pairwise MN, so this is not a great restriction. Often, in practice, different variants of the algorithm are used for different cases, but the main ideas are the same, and can be understood from this variant.

Assume that we have a pairwise Markov network, for which the probability is given by:

$$P(\mathbf{X}) = \frac{1}{Z}\prod_i \psi_i(X_i)\prod_{ij}\psi_{ij}(X_i, X_j).$$

Our goal is to compute the marginal probabilities $P(X_i)$ for each variable, possibly conditioned on any evidence that we have received. At each point in the algorithm, each variable keeps an estimate of the marginal, called a *belief*, denoted by $b_i(X_i)$. Node i updates its beliefs by receiving messages from each of its neighbor nodes j. We denote the message passed from node j to node i by $m_{j\to i}$.

For the case we consider here, of discrete inference with tabular potentials, both the beliefs b and the messages m are in the form of tables, and admit the same set of operations (multiplication and summing out of variables) as the original potentials.

The algorithm begins by initializing all of the messages to uniform (i.e., $m_{j\to i} = 1$). It then proceeds to update each of the messages according to the *message update* formula:

$$m_{j\to i}(x_i) = \sum_{x_j}\psi_j(x_j)\psi_{ij}(x_i, x_j)\prod_{k\in Neighbors(j)\setminus i} m_{k\to j}(x_j). \tag{2.8}$$

From the messages, we can also compute the beliefs for each node at each iteration, according to:

$$b_i(x_i) = \alpha\psi_i(x_i)\prod_{j\in Neighbors(i)} m_{j\to i}(x_i), \tag{2.9}$$

where the proportionality coefficient α is calculated to enforce the requirement that the beliefs sum to 1. The messages and beliefs are iteratively updated according to this rule until convergence, when all of the messages stay constant from one iteration to the next.

In the exact case, for singly connected networks, this algorithm is guaranteed to converge to the correct answer. In graphs with loops, however, there is no such guarantee. There maybe be oscillations in the messages, and even if it does converge, it may be far off from the true answer. Despite these problems, the BP algorithm is often effective in practice, and is used in a few of the methods described in this thesis.

In the above discussion, we assumed that each of the messages was updated in parallel, based on the previous value for all of the messages. This form of BP, which is called *synchronous* BP is usually effective, but it often spends a great deal of effort in updating messages. *Asynchronous* BP was introduced to combat this problem, and has been empirically found to converge faster and more often than the synchronous variety [39]. This version of the algorithm selects a single message to update at each iteration. Asynchronous variants have been instrumental in the decoding literature [142], for example.

Once we have decided to use the asynchronous version of BP, we must select a message schedule (i.e., an order in which to update the messages). Typical implementations of BP choose a round-robin message scheduling, where the messages get updated in the same order during each round [101]. However, recent work has shown that clever scheduling can make convergence more likely, and can greatly improve the convergence rate and accuracy in cases where it does converge. For example, Wainwright et al. [132] developed the tree-based re-parametrization algorithm with nice theoretical guarantees, as well as showing it to be empirically successful. The TRP algorithm can be viewed as either a new approximation for inference, or a new scheduling algorithm for BP messages.

One particular example, the Residual Belief Propagation (RBP) of Elidan et al. [39] chooses to update the message with the highest residual, which is defined as the distance between the current value of the message and the value of the message after it is updated. The residual is calculated as:

$$r^t_{i\to j} = \|m^t_{i\to j} - m^{t-1}_{i\to j}\| \tag{2.10}$$

In Chapter 6 we use RBP for our discrete inference. Pseudocode for the Residual Belief Propagation algorithm is provided in Figure 2.6.

```
Algorithm Residual Belief Propagation
Input:    Pairwise Markov network with potentials $\psi_i$ and $\psi_{ij}$
Output:   Marginal probabilities $P(X_i)$ for each variable

    Initialize $m_{i\to j} = 1$ for all neighboring nodes $i$,$j$
    Compute all residuals $r_{i\to j}$ according to Eq. (2.10)
    For $t = 1, 2, \ldots, T$ or until convergence
        Select the message $m_{i\to j}$ with largest residual $r^t_{i\to j}$
        Update: $m_{j\to i}(x_i) \leftarrow \sum_{x_j} \psi_j(x_j)\psi_{ij}(x_i, x_j) \prod_{k\in Neighbors(j)\setminus i} m_{k\to j}(x_j)$
        Update residuals $r_{m\to n}$ for all messages involving node $j$
        if all $r^t_{i\to j} = 0$, report convergence
    Compute the final beliefs $b_i(x_i) = \alpha\psi_i(x_i) \prod_{j\in Neighbors(i)} m_{j\to i}(x_i)$
    Return $P(x_i) = b_i(x_i)$, $\forall i$
```

Figure 2.6: Residual Belief Propagation Inference

2.4.2 Gibbs Sampling

Another set of inference algorithms are based on sampling methods [52]. In these algorithms, the goal is to draw samples from the posterior distribution that you seek to approximate, and use the resulting samples to estimate the posterior. Suppose we had a magic box that could draw samples from $P(Y_6 \mid \mathcal{E})$, and we wanted to know the quantity: $P(Y_6 = 1 \mid \mathcal{E})$. We can estimate this quantity as the proportion of samples that have this assignment amongst all the samples that were drawn:

$$P(Y_6 = 1 \mid \mathcal{E}) \approx \frac{\sum_{m=1}^{M} 1\{Y_6[m] = 1\}}{M}, \tag{2.11}$$

where M is the total number of samples drawn, and $Y_i[m]$ is the assignment to variable Y_i in the m^{th} sample, and $1\{\}$ is the "indicator" function which returns 1 when the argument is true, and 0 otherwise.

Now we get to the difficult problem of drawing samples from this posterior. Many algorithms have been developed that solve this problem. Most of these are based on the idea of a Markov chain, where an initial sample is iteratively updated according to a particular rule, and in the limit of many updates, the sample approaches the desired posterior. While many algorithms have been developed with particular goals

```
Algorithm Gibbs Sampling
Input:   Bayesian network $\mathcal{B}$ over variables $\mathcal{X} = \{X_1, \dots, X_N\}$
         Assignments $\mathbf{E}$ to evidence variables $\mathcal{E} \subseteq \mathcal{X}$
Output:  Sample $\mathbf{X}[m]$ drawn approximately from the posterior $P(\mathcal{X} \mid \mathcal{E})$

    $\mathbf{X}^{(0)}[m] \leftarrow$ random assignment
    $X_i^{(0)}[m] \leftarrow E_i \quad \forall X_i \in \mathcal{E}$
    For $t = 1$ to $T$
        For each $i$ s.t. $X_i \notin \mathcal{E}$
            Draw $X_i^{(t)}[m] \sim P(X_i \mid X_{-i}^{(t-1)}[m])$
    Return $\mathbf{X}^{(T)}[m]$
```

Figure 2.7: Gibbs Sampling Inference

in mind (e.g., speed of convergence, efficiency of update), one of the simplest method is **Gibbs sampling**, which was introduced by Geman and Geman [55].

The Gibbs sampling algorithm induces a Markov chain using a very simple update rule. At each update step we update each variable not in the evidence set by sampling from the conditional distribution of that variable given the previous assignment of all of the other variables. Because of the Markov independence assumptions inherent in BNs and MNs, computing this conditional probably is usually very efficient. For example, in the BN example of Figure 2.4, if we want to resample C given A, B, and D, we must compute:

$$P(C \mid A, B, D) \propto P(C \mid A, B)P(D \mid A, B, C) = P(C \mid A, B)P(D \mid C),$$

where the two terms in the product can be directly looked up from the CPDs for C and D, respectively. We can renormalize these values into a probability simply by dividing each entry by the sum over all possible values for C.

Pseudocode for the Gibbs sampling algorithm is given in Figure 2.7. An important choice for this algorithm is the value of T, the number of iterations to perform [104]. It is easy to prove that as $T \to \infty$, the chain will converge to the true posterior (i.e., the sample will be a true sample from the posterior). However, for any practical T, we cannot, in general, know whether we have converged, or how close we are to convergence. Researchers must trade off computationally efficiency with accuracy to choose a T that works for their application.

In Chapter 5, we use a small modification to this algorithm, inspired by the idea of distributional samples (described below). Recall that in the pure sampling case, we use the indicator function Eq. (2.11) to "count" the number of times that each target outcome occurs in our samples. This means that each sample only contributes to one entry in the final probability table $P(\mathcal{Y} \mid \mathbf{E})$. But we saw in the Gibbs algorithm of Figure 2.7 that we are required to compute conditional probabilities for each of the variables in order to perform an update step. Instead of sampling from this distribution, in the final step of Gibbs, we record the conditional distribution for the variable that is our inference target.

Suppose that (following the case of Eq. (2.11)), we remember the final distribution $P(Y_6 \mid \mathbf{Y}_{-6}[m], \mathbf{E})$. The pair $(\mathbf{Y}_{-6}[m], P(Y_6 \mid \mathbf{Y}_{-6}[m], \mathbf{E}))$ is called a "distributional" sample, because it contains a full assignment (sample) to most of the variables, and a distribution on the remaining variable(s), conditioned on the sampled variables. Using a set of distributional samples, we can approximate the posterior of Eq. (2.11) using:

$$P(Y_6 = 1 \mid \mathcal{E}) \approx \frac{\sum_{m=1}^{M} P(Y_6 = 1 \mid Y_{-6}[m], \mathbf{E})}{M}. \tag{2.12}$$

By doing this, we hope that the approximation of Eq. (2.12) provides a good estimate of the probability with a smaller number of samples, since each sample now contributes to the resulting table for each value of the target variable(s).

2.5 Learning Bayesian Networks

In some cases we can directly design a BN or MRF to fit the problem that we wish to model. However, in most cases we do not know the exact model that we want to use. In this case, we often often take the approach of learning the model. In general, this may involve learning both the graph structure and the parameters of the model. Furthermore, we must consider both cases where we have *fully supervised* data, with a value for each variable for every instance in the training set, and *partially supervised* data, where some of the variables are missing (unobserved) across the training set. Finally, we sometimes wish to introduce *hidden* variables, which are inherently always unobserved, but are useful to model certain relationships.

Beginning with the most general case, where we must learn the structure, including any hidden variables, as well as the parameters, with an arbitrary number of missing values in our data, is a formidable task. In order to make progress, we will begin our development with the case of known structure and no missing data, including no hidden variables. In this section, we will describe learning in the context of Bayesian networks, but many of the principles are the same for undirected networks. In this setting, we often want to learn the parameters that maximize the likelihood of the observed training data. We will now describe the basics of maximum likelihood estimate of parameters.

2.5.1 Maximum Likelihood Parameter Estimation

For maximum likelihood learning, we assume that we have a training set $\mathcal{D} = \{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(M)}\}$ of data instances, where each instance $\mathbf{X}^{(m)}$ includes full assignments to all of the variables. Suppose that we are given a Bayesian network graph structure $\mathcal{G}$, and our goal is to learn the CPD parameters $\Theta = \{\theta_1, \ldots, \theta_N\}$ for each node (variable) X_i in the graph.

The maximum likelihood learning objective is to find the parameters that maximize the log-likelihood (log probability) of the training set:

$$\Theta_{\text{ML}} = \text{argmax}_{\Theta} \ell(\Theta; \mathcal{D}), \tag{2.13}$$

where the log-likelihood ℓ is given by:

$$\ell(\Theta; \mathcal{D}) = \sum_{m=1}^{M} \log P(\mathbf{X}^{(m)}; \Theta). \tag{2.14}$$

One of the beneficial properties of the BN framework is that the probability of a data instance decomposes according to the graph. Recall from Eq. (2.2) that we can write this probability as:

$$P(\mathcal{X}; \Theta) = \prod_i P(X_i \mid \mathbf{Y}_i = Parents(X_i); \Theta).$$

where $\mathbf{Y}_i$ are the parent variables of X_i in the BN. In addition, because the parameters for a BN represent CPDs for each such term, we can rewrite our likelihood function as:

$$\begin{aligned}\ell(\Theta; \mathcal{D}) &= \sum_{m=1}^{M} \sum_i \log P(X_i^{(m)} \mid \mathbf{Y}_i^{(m)}; \theta_i) \\ &= \sum_i \sum_{m=1}^{M} \log P(X_i^{(m)} \mid \mathbf{Y}_i^{(m)}; \theta_i), \qquad (2.15)\end{aligned}$$

From Eq. (2.15) we can see that the learning problem decomposes across CPDs.

This means that we can optimize the parameters θ_i for each CPD independently, given only the variables in the "family" of that CPD (i.e., the variable corresponding to that node of the graph, as well as all parents of that variable). In particular, we want to set:

$$\theta^{\text{ML}}_{x_i|y_i} = \text{argmax}_{\theta_i} \sum_{m=1}^{M} \log P(X_i^{(m)} \mid \mathbf{Y}_i^{(m)}; \theta_i)$$

The sum over m indicates that the $\log P(X \mid Y)$ term gets added in the number of times that it occurs in the training set. This can be seen by rewriting the above equation as:

$$\theta^{\text{ML}}_{x_i|\mathbf{y}_i} = \text{argmax}_{\theta_i} \sum_{m=1}^{M} \sum_{x_i, \mathbf{y}_i} 1\{X_i^{(m)} = x_i, \mathbf{Y}_i^{(m)} = \mathbf{y}_i\} \log P(x_i^{(m)} \mid \mathbf{y}_i^{(m)}; \theta_i)$$

We can rearrange the summations to get:

$$\theta^{\text{ML}}_{x_i|\mathbf{y}_i} = \text{argmax}_{\theta_i} \sum_{x_i, \mathbf{y}_i} \left[\sum_{m=1}^{M} 1\{X_i^{(m)} = x_i, \mathbf{Y}_i^{(m)} = \mathbf{y}_i\} \right] \log P(x_i^{(m)} \mid \mathbf{y}_i^{(m)}; \theta_i)$$

And we use $M[x_i, \mathbf{y}_i]$ to denote the term in the brackets. These are called the "counts" of the data, because they indicate the number of times that this configuration of values occurs in the data. Also, in a BN, the probability is encoded directly in the parameter, as $P(x_i^{(m)} \mid \mathbf{y}_i^{(m)}; \theta_i) = \theta_{x_i|\mathbf{y}_i}$. With this simplification, we can write our local likelihood function directly in terms of counts and the parameters:

$$\ell_i = \sum_{x_i, \mathbf{y}_i} M\left[x_i, \mathbf{y}_i\right] \log \theta_{x_i|\mathbf{y}_i} \tag{2.16}$$

Now our goal is to find parameters that maximize this local likelihood. In order to do this, we must solve the problem:

$$\begin{aligned} \max_{\theta_{x_i|\mathbf{y}_i}} \quad & \sum_{x_i, \mathbf{y}_i} M\left[x_i, \mathbf{y}_i\right] \log \theta_{x_i|\mathbf{y}_i} \\ \text{s.t.} \quad & \sum_{x} \theta_{x_i|\mathbf{y}_i} = 1 \quad \forall \mathbf{y}_i \end{aligned} \tag{2.17}$$

We can solve this using the Lagrangian function:

$$f = \sum_{\mathbf{y}_i} \left[\sum_{x_i} M\left[x_i, \mathbf{y}_i\right] \log \theta_{x_i|\mathbf{y}_i} + \lambda_{\mathbf{y}_i} \sum_{x_i} \theta_{x_i|\mathbf{y}_i} \right].$$

Taking the derivative with respect to one of the parameters $\theta_{x_i|\mathbf{y}_i}$ and setting it equal to zero gives us that:

$$\theta_{x_i|\mathbf{y}_i} \propto M\left[x_i, \mathbf{y}_i\right]$$

Enforcing the constraint of Eq. (2.17) give us:

$$\theta^{\text{ML}}_{x_i|\mathbf{y}_i} = \frac{M\left[x_i, \mathbf{y}_i\right]}{\sum_{\mathbf{y}_i} M\left[x_i, \mathbf{y}_i\right]}$$

This gives us a very intuitive solution to the maximum likelihood learning problem: the parameters for a particular set of variable values are proportional to the number of times the corresponding values occur in the training set. Because the ML parameters are completely determined by these counts, we say that they are *sufficient statistics*, meaning that we can fully summarize the dataset by it's counts, rather than having to remember all the values for each data instance.

2.5.2 Missing Data: EM Learning

In many cases life is not so simple. One of the most common complications during parameter learning is the problem of missing data. In these cases our training data instances have assignments to a subset of the variables. The particular observed subset might be different for each instance, and we may even have variables that are never observed (so-called "hidden" variables).

In theory, we still want to maximize the likelihood of the observed data. If we suppose that, for each instance m, our variables are broken up into two sets, the observed variables $\mathbf{O}^{(m)}$ and the hidden variables $\mathbf{H}^{(m)}$, we can write the likelihood of the observed data as:

$$\ell(\Theta; \mathcal{D}) = \sum_m \log P(\mathbf{O}^{(m)}; \Theta). \tag{2.18}$$

Unfortunately, this function is multimodal and admits no straightforward solution. However, an appealing solution to the problem was developed by Dempster et al. [35], and is called the *Expectation-Maximization* (EM) learning algorithm.

In this algorithm, we introduce a new learning objective, different but related to the likelihood of Eq. (2.18), called the *expected log-likelihood*. In order to formulate this objective, we introduce an auxiliary distribution over the hidden variables in the entire training set, denoted by $Q(\mathcal{H})$. Later we will describe how to choose this distribution. Having defined the Q distribution, we define the expected log-likelihood as:

$$E_Q\left[\ell(\Theta; \mathcal{D})\right] = \sum_{m=1}^{M} E_Q^{(m)}\left[\log P(\mathbf{O}^{(m)}, \mathbf{H}^{(m)}; \Theta)\right] \tag{2.19}$$

The EM algorithm begins from an initial "guess" for the parameters Θ^0, and then attempts to optimize this function by iterating two steps:

1. **Expectation Step:** Choose the auxiliary function to be the posterior distribution of the hidden variables, conditioned on the observed variables, using the

current setting for the model parameters. If Θ^t is the current set of parameters, we set:

$$Q(\mathbf{H}^{(m)}) = P(\mathbf{H}^{(m)} \mid \mathbf{O}^{(m)}; \Theta^t)$$

2. **Maximization Step:** With this choice of auxiliary distribution held fixed, maximize the expected log-likelihood over the parameters, to obtain a new estimate for the parameters:

$$\Theta^{t+1} = \text{argmax}_\Theta E_Q \left[\ell(\Theta; \mathcal{D})\right]$$

We will now explore how to maximize the expected log-likelihood (solve the M-step). We first extend our distribution Q to be over the hidden $\mathbf{H}$ and observed $\mathbf{O}$ variables for each instance, and put these variables together into a vector $\mathbf{X}$ (i.e., $Q_m(\mathbf{X}) = Q_m(\mathbf{O}, \mathbf{H}) = Q(\mathbf{H} \mid \mathbf{O})1\{\mathbf{O} = \mathbf{O}^{(m)}\}$). We begin by converting the function into a form similar to Eq. (2.15):

$$\begin{aligned}
E_Q\left[\ell(\Theta; \mathcal{D})\right] = & \sum_{m=1}^{M} E_Q^{(m)}\left[\log P(\mathbf{X}^{(m)}; \Theta)\right] && (2.20)\\
= & \sum_{m=1}^{M} E_Q^{(m)}\left[\sum_i \log P(X_i^{(m)} \mid \mathbf{Y}_i^{(m)}; \Theta)\right] && (2.21)\\
= & \sum_{m=1}^{M} \sum_i E_Q^{(m)}\left[\log P(X_i^{(m)} \mid \mathbf{Y}_i^{(m)}; \Theta)\right] && (2.22)\\
= & \sum_{m=1}^{M} \sum_i \sum_{x_i, \mathbf{y}_i} Q^{(m)}(x_i, \mathbf{y}_i) \log \theta_{x_i|\mathbf{y}_i} && (2.23)\\
= & \sum_i \sum_{x_i, \mathbf{y}_i} \left[\sum_{m=1}^{M} Q^{(m)}(x_i, \mathbf{y}_i)\right] \log \theta_{x_i|\mathbf{y}_i}, && (2.24)
\end{aligned}$$

and we notice that instead of a sum over indicators for the values of variables, we have a sum over marginals of Q. We can now introduce the notation $\hat{M}\left[x_i, \mathbf{y}_i\right] =$

$\sum_{m=1}^{M} Q^{(m)}(x_i, \mathbf{y}_i)$, and our expected likelihood becomes:

$$E_Q\left[\ell(\Theta; \mathcal{D})\right] = \sum_i \sum_{x_i, \mathbf{y}_i} \hat{M}\left[x_i, \mathbf{y}_i\right] \log \theta_{x_i|\mathbf{y}_i}. \tag{2.25}$$

We see that the $\hat{M}$ performs the same role as the counts M in the fully observed case. As a result, we call these "expected" counts, and these are easily computed using probabilistic inference in our BN with the previous set of parameters.

The EM algorithm has the nice property that, in addition to optimizing the expected log-likelihood, it is guaranteed to improve the observed data log-likelihood function at each iteration as well [35]. Because the log-likelihood function is upper bounded, this means that the EM algorithm is guaranteed to converge, although not necessarily to a global optimum of the true likelihood function.

2.5.3 Structure Learning

In the development above, we've assumed that the graph structure of the Bayesian network is known. In this section we consider the challenge of learning the graph structure in addition to the parameters.

We begin by supposing that we have a candidate graph $\mathcal{G}$, and that we would like a way of scoring the graph to decide how good a fit it is to our training data set $\mathcal{D}$. The most obvious, and seemingly natural score for the structure is the *likelihood score*. The likelihood score of a graph is the likelihood function evaluated for that graph structure with the maximum likelihood parameters:

$$score_\ell(\mathcal{G}; \mathcal{D}) = \ell_{\mathcal{G}}(\Theta_{\mathcal{G},\mathrm{ML}}; \mathcal{D}),$$

where $\Theta_{\mathcal{G},\mathrm{ML}}$ are the maximum likelihood parameters for this graph and dataset. This score is very appealing, because it encourages us to use structures that are able to model the training data better, once we learn the parameters.

Unfortunately, this scoring mechanism suffers from one critical drawback. Intuitively, if we are trying to produce the highest likelihood on the training data, we would like to have as many free parameters as possible. This corresponds to introducing additional edges into our Bayesian network, and in the limit, we can even use the complete graph (denoted by $\mathcal{G}_C$). If we are trying to maximize the likelihood of the training data, why would we ever use anything other than the complete graph? It turns out that a more quantitative analysis matches this intuition. In particular, we can easily show that for any pair of graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, where the second contains *at least* all of the edges in the first, we have:

$$score_\ell(\mathcal{G}_2; \mathcal{D}) \geq score_\ell(\mathcal{G}_1; \mathcal{D}),$$

which implies:

$$score_\ell(\mathcal{G}_C; \mathcal{D}) \geq score_\ell(\mathcal{G}; \mathcal{D}),$$

for any graph $\mathcal{G}$. Therefore we are guaranteed to maximize the likelihood score by choosing the complete graph. However, our discussion of regularization above implies that we usually want to have *fewer* parameters, rather than more.

In order to overcome this, we need a method for regularizing the graph structure. There has been a lot of work on this problem, and many solutions have been proposed, including the BIC score [114], the AIC score [1], and the Bayesian score [33] (and closely related Bayes Factors approach [56]). The Bayesian score evaluates the structure by marginalizing out the parameters, assuming a weak prior over the value of the parameters. Intuitively, this approach can be thought of as evaluating each instance as if it were in the test set for a model learned using the previously observed instances. Often, however, this Bayesian score can be difficult or expensive to compute. We can therefore use the BIC score as an alternative. In the limit of infinite data, the BIC score approaches the Bayesian score, and is hence considered to be a good approximation in many cases. The BIC score can be thought of as a penalty for "complexity" of the graph, measured as the number of parameters required for the particular graph structure.

One of the simplest ways of doing this is to introduce a structure prior, or a probability distribution over graphs, that penalizes more complex networks (i.e., graphs with more edges). We can use this distribution, denoted by $P(\mathcal{G})$ to choose the graph with the highest log "posterior" probability, given by:

$$\begin{aligned} score_{\text{posterior}} &= \log P(\mathcal{G} \mid \mathcal{D}) && (2.26) \\ &= \log P(\mathcal{D} \mid \mathcal{G}) + \log P(\mathcal{G}) - \log P(\mathcal{D}) && (2.27) \\ &= score_{\ell}(\mathcal{G}; \mathcal{D}) + \log P(\mathcal{G}) + C, && (2.28) \end{aligned}$$

where C is a constant that does not depend on the graph. In Chapter 5 we will use this scoring function to choose between BN graph structures.

Once we have selected a scoring function to evaluate our candidate graphs, we are now faced with the problem of searching for the highest scoring candidate. We can solve this problem approximately, using a greedy search procedure, where we begin with an empty graph, and iteratively apply the best structure move from a set of candidate operations. These candidate operators might include adding an edge, deleting an edge, or swapping the direction of an edge.

At each iteration we evaluate all of the possible next graphs, and pick the one with the highest score. Once we can no longer improve the score, we return the best graph found so far as our final structure. Because this process was greedy, this is not guaranteed to be the best structure according to our score, but we know that no single search move will produce a better graph.

2.5.4 Structural EM

In the above approach for structure learning, we assumed that our data was fully observed, allowing us to compute the likelihood function, and therefore the structure score, for each candidate structure. In most cases, however, we have some missing data in our training set, and may even want to have hidden variables. How do we learn the structure in this case?

The most straightforward thing to do is to combine the EM parameter learning method described above with our structure search. That is, for each candidate structure, we perform EM parameter learning to produce the parameters, and then evaluate our score on the observed data.

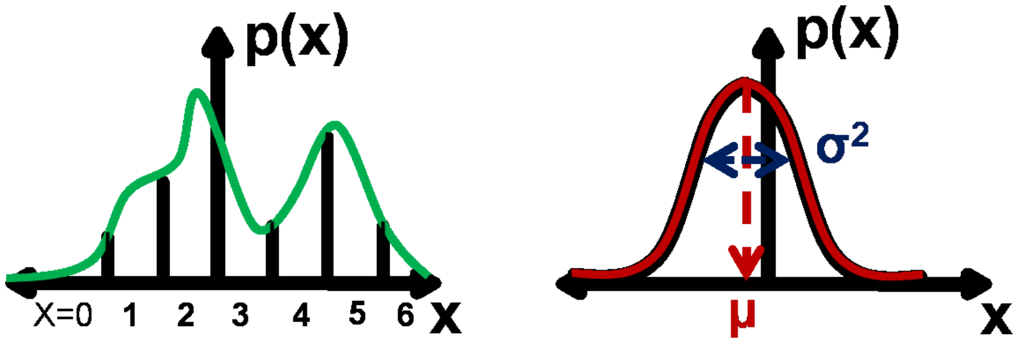
```
Algorithm **Structural EM**
Input:    Training Set of partially labeled data $\mathcal{D} = \{\mathbf{O}^{(1)}, \ldots \mathbf{O}^{(M)}\}$
Output:   Bayesian Network Structure $\mathcal{G}$

    Initialize $\mathcal{G}^0 \leftarrow$ Empty Graph
    Loop for $n = 0, 1, \ldots$ until convergence
        $\Theta_G^n \leftarrow$ LearnEM$(\mathcal{G}^n, \mathcal{D})$
        Store final expected counts $\hat{M}^n$ w.r.t $\mathcal{G}^n$, $\mathcal{D}$, and $\Theta_G^n$
        Find a structure $\mathcal{G}^{n+1}$ that maximizes $score(\mathcal{G}; \mathcal{D}, \hat{M}^n)$
    Return final $\mathcal{G}$
```

Figure 2.8: Structural EM

Clearly this is an extremely expensive process, however. At each stage in the search procedure, we must evaluate a large number of candidate graph structures, and the EM learning for each requires many iterations of expensive inference. Despite this, we can make a few small modification to this procedure, allowing it to become usable in many real-world cases.

The procedure that we can use in this situation is called *structural EM*, and was developed by Friedman [54]. It's based on the following intuition: over a small number of structure moves, the posterior distribution over the hidden variables given the observed variables does not change very much. Following this intuition, we also believe that the resulting expected counts that come out of the E-step of EM will probably not change very much between nearby structures. Structural EM holds these expected counts fixed, and evaluates many structures (perhaps even performing multiple greedy structure update steps) with a fixed set of expected counts. Pseudocode for the algorithm is given in Figure 2.8.

In Chapter 5, we will use this algorithm to learn a Bayesian network structure over a small set of candidate structures.



(a) (b)

Figure 2.9: Continuous random variables.

2.6 Gaussian Random Variables

While discrete random variables cover many of the situations that we would like to model in the world, there are also times when we require a continuous distribution that allows the variables to take on a real value. While the grade a student receives in a class is naturally quantized into discrete bins according to the letter, representing the IQ of the student is not so easy. One option, when we are faced with an inherently continuous random variable, is to try to quantize the value into "bins" and use a discrete approach. While this approach, illustrated in Figure 2.9(a) is feasible, it somehow feels dissatisfying. In addition, in the particular case of a person's IQ, we have a strong prior that this distribution is well modeled by the bell-curve or *Gaussian* distribution.

The Gaussian distribution for a single variable X, which is visualized in Figure 2.9(b), is distributed according to the probability density function:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right), \tag{2.29}$$

where the parameters of the distribution are given by μ, the mean value of the variable, and σ^2, the variance of the distribution. We denote that a variable X is distributed according to a Gaussian with these parameters by $X \sim \mathcal{N}(\mu, \sigma^2)$.

Usually we have more than a single variable that we wish to model. In this case, we might have a set of variables $\mathcal{X} = \{X_1, \ldots, X_N\}$ that are jointly gaussian distributed. In this case, we can stack the values of the variables into a vector $\mathbf{X}$, and we call this a *Gaussian random vector* (GRV). A Gaussian random vector is parameterized by its mean μ and a covariance matrix $\mathbf{\Sigma}$. The pdf for the vector is given by:

$$p(\mathbf{x}) = \frac{|\mathbf{\Sigma}^{-1}|^{1/2}}{(2\pi)^{N/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)\right). \tag{2.30}$$

Jointly Gaussian random variables, represented as a Gaussian random vector, have several nice properties. One of the most important properties, which we will use in the development of Chapter 6, is that the conditional distribution of any subset of the variables, given any other subset, is also a Gaussian. In particular, if we have subvectors $\mathbf{X}$ and $\mathbf{Y}$ which are distributed according to:

$$\begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_y \\ \mu_Y \end{bmatrix}, \begin{bmatrix} \mathbf{\Sigma}_X & \mathbf{\Sigma}_{XY} \\ \mathbf{\Sigma}_{XY}^T & \mathbf{\Sigma}_Y \end{bmatrix}\right),$$

then we have an efficiently computed closed-form distribution for $\mathbf{X}$ conditioned on the value of $\mathbf{Y}$ given by:

$$\mathbf{X} \mid \mathbf{Y} \sim \mathcal{N}\left(\ \mu_x + \mathbf{\Sigma}_{XY}\mathbf{\Sigma}_Y^{-1}(\mathbf{y}-\mu_y),\ \ \mathbf{\Sigma}_X - \mathbf{\Sigma}_{XY}\mathbf{\Sigma}_Y^{-1}\mathbf{\Sigma}_{XY}^T\ \right). \tag{2.31}$$

Because of the ease of use of the Gaussian distribution, it is often used to model continuous quantities, even if the underlying distribution is not truly Gaussian. In this thesis, we will use the Gaussian distribution to model both distributions over the shape of various objects and the distribution over appearance features for regions or patches.

2.6.1 Learning Parameters

Suppose we have a training set $\{\mathbf{X}^{(1)} \ldots \mathbf{X}^{(M)}\}$ of continuous data, and we want to learn parameters for a Gaussian random vector model of the data. As in the discrete case, the simplest thing to do is to learn the maximum likelihood parameters

$(\mu_{\mathrm{ML}}, \mathbf{\Sigma}_{ML})$. In the case of fully observed data, we can write the log-likelihood of a dataset as:

$$\begin{aligned}
\ell(\mathcal{D}) = & \sum_m -\log(2\pi)^{N/2} + \log\left[|\mathbf{\Sigma}^{-1}|^{1/2}\right] - \frac{1}{2}(\mathbf{X}^{(m)} - \mu)^T \mathbf{\Sigma}^{-1}(\mathbf{X}^{(m)} - \mu) && (2.32)\\
= & \; C + \frac{M}{2}\log\det\mathbf{\Sigma}^{-1} - \sum_m \frac{1}{2}(\mathbf{X}^{(m)} - \mu)^T\mathbf{\Sigma}^{-1}(\mathbf{X}^{(m)} - \mu), && (2.33)
\end{aligned}$$

where C is a constant that does not depend on the parameters. Now, if we take the derivative of this with respect to the mean μ, and set it equal to zero, we get:

$$\begin{aligned}
\frac{\delta}{\delta\mu}\ell(\mathcal{D}) = & -\sum_m \mathbf{\Sigma}^{-1}(\mathbf{X}^{(m)} - \mu) = 0 && (2.34)\\
\rightarrow & \; \mathbf{\Sigma}^{-1}\sum_m(\mathbf{X}^{(m)} - \mu_{\mathrm{ML}}) = 0 && (2.35)\\
\rightarrow & \; \sum_m(\mathbf{X}^{(m)} - \mu_{\mathrm{ML}}) = 0 && (2.36)\\
\rightarrow & \; M\mu_{\mathrm{ML}} = \sum_m \mathbf{X}^{(m)} && (2.37)\\
\rightarrow & \; \mu_{\mathrm{ML}} = \frac{1}{M}\sum_m \mathbf{X}^{(m)}. && (2.38)
\end{aligned}$$

And if we differentiate with respect to the inverse covariance $\mathbf{\Sigma}^{-1}$, setting the result equal to zero, we get:

$$\begin{aligned}
\frac{\delta}{\delta\mathbf{\Sigma}}\ell(\mathcal{D}) = & \; \frac{M}{2}\mathbf{\Sigma} - \sum_m \frac{1}{2}(\mathbf{X}^{(m)} - \mu)(\mathbf{X}^{(m)} - \mu)^T = 0 && (2.39)\\
\rightarrow & \; M\mathbf{\Sigma}_{\mathrm{ML}} = \sum_m(\mathbf{X}^{(m)} - \mu)(\mathbf{X}^{(m)} - \mu)^T && (2.40)\\
\rightarrow & \; \mathbf{\Sigma}_{\mathrm{ML}} = \frac{1}{M}\sum_m(\mathbf{X}^{(m)} - \mu)(\mathbf{X}^{(m)} - \mu)^T && (2.41)
\end{aligned}$$

Restated succinctly, for the case of fully observed data instances, these parameters are given by:

$$\mu_{\text{ML}} = \frac{1}{M}\sum_m \mathbf{X}^{(m)} \tag{2.42}$$

$$\mathbf{\Sigma}_{\text{ML}} = \frac{1}{M}\sum_m (\mathbf{X}^{(m)} - \mu_{\text{ML}})(\mathbf{X}^{(m)} - \mu_{\text{ML}})^T \tag{2.43}$$

$$= \left(\frac{1}{M}\sum_m \mathbf{X}^{(m)}\mathbf{X}^{(m),T}\right) - \mu_{\text{ML}}\mu_{\text{ML}}^T \tag{2.44}$$

In other words, the maximum likelihood mean is the empirical mean of the training data also known as the "first moment", and the maximum likelihood covariance matrix is the empirical "second moment" minus the outer product of the mean with itself.

For a Gaussian random vector of N dimensions, we have $K = N + \frac{N(N-1)}{2}$ parameters. This number rises rapidly with N, and as a result, overfitting is likely. Furthermore, for even a relatively small N, we can get into a situation where there are more parameters than data instances ($K > M$). In this case, the maximum likelihood covariance matrix will be singular (non-invertible). Unfortunately, the covariance matrix for a GRV must be positive definite, and therefore must be invertible. Given these two issues, we would like to find a way to regularize our parameter estimation to ensure a positive definite covariance as well as to avoid overfitting.

One of the most common regularization methods for estimating Gaussian parameters is the so-called "ridge" regularization [69]. Ridge regularization involves adding a small number (the ridge parameter r) to the diagonal of our maximum likelihood covariance matrix. Our ridge-regularized estimate for the covariance matrix is:

$$\mathbf{\Sigma}_{\text{ridge}} = \mathbf{\Sigma}_{\text{ML}} + r\mathbf{I},$$

where $\mathbf{I}$ is the identity matrix of the same size as the covariance matrix. While this form of regularization will always produce a positive definite covariance matrix, and will often help with overfitting to a large degree, it still has a very large number of parameters. Another solution to the regularization problem for Gaussians it to learn a “sparse” representation of the Gaussian, which actually reduces the number of parameters required. We will explore this method in the next section.

2.6.2 Sparse Gaussians

If we need to have a covariance matrix, and our covariance matrix inherently contains $\frac{N(N-1)}{2}$ values, how can we reduce the number of parameters? If we look more closely at the form of Eq. (2.30), we can see that the numbers that contribute to the probability calculation are the entries of the inverse of the covariance matrix. We will call this the precision matrix, and denote it by $\mathbf{\Omega} = \mathbf{\Sigma}^{-1}$. We begin by rewriting Eq. (2.30) in terms of $\mathbf{\Omega}$:

$$p(x) = \frac{|\mathbf{\Omega}|^{1/2}}{(2\pi)^{N/2}} \exp \frac{1}{2} \left((\mathbf{x} - \mu)^T \mathbf{\Omega} (\mathbf{x} - \mu) \right). \tag{2.45}$$

and then we take the log of the probability, and write it as a sum, rather than in matrix notation:

$$\begin{aligned} \ell(x) = \log p(x) = & \log \left(\frac{|\mathbf{\Omega}|^{1/2}}{(2\pi)^{N/2}} \right) + (\mathbf{x} - \mu)^T \mathbf{\Omega} (\mathbf{x} - \mu) \\ & \frac{1}{2} \log |\mathbf{\Omega}| - \frac{N}{2} \log(2\pi) + \frac{1}{2} \sum_{ij} \mathbf{\Omega}_{ij} (x_i - \mu_i)(x_j - \mu_j). \end{aligned} \tag{2.46}$$

The third term in this equation makes explicit what the entries of the precision matrix are doing. The entries $\mathbf{\Omega}_{ij}$ are coefficients for the quadratic terms that multiply x_i and x_j.

Based on this analysis, one way of reducing the parameters of the Gaussian is to set some (or most) of the $\mathbf{\Omega}_{ij}$ to zero. This prevents some of the quadratic products from influencing the probability. Put another way, we are looking for a sparse $\mathbf{\Omega}_{\text{S}}$ (a matrix with many zeros) that gives a good approximation to the maximum likelihood $\mathbf{\Omega}_{\text{ML}}$. As a result, this regularization method is called Gaussian sparsification.

With our goal of finding this sparse Gaussian representation, we must define what it means for the distribution implied by $\mathbf{\Omega}_{\text{S}}$ to be close to the maximum likelihood distribution. One obvious measure of closeness is based on the entries of the precision matrix. If we want to remove n parameters, we just choose the n entries of $\mathbf{\Omega}_{\mathrm{ML}}$ with the smallest values, and set them to zero. However, Euclidean distance in parameter space often does not correspond to distance between distributions.

Therefore, we would like to directly minimize the distance between the distributions, subject to our desired sparsity level. We can measure the distance between two Gaussian distributions using the *KL divergence* [74]. For general continuous distributions p and q, the KL divergence between them is given by:

$$D(p\|q) = \int_x p(x) \log \frac{p(x)}{q(x)} dx. \tag{2.47}$$

In the case of two GRVs of dimension N, with the same mean and different covariance matrices, distributed according to $(\mu, \mathbf{\Sigma}_1)$ and $(\mu, \mathbf{\Sigma}_2)$, respectively, the KL divergence is given by:

$$D((\mu, \mathbf{\Sigma}_1)\|(\mu, \mathbf{\Sigma}_2)) = \frac{1}{2}\left[\log\left(\frac{|\mathbf{\Sigma}_1|}{|\mathbf{\Sigma}_2|}\right) + tr(\mathbf{\Sigma}_2^{-1}\mathbf{\Sigma}_1) - N\right] \tag{2.48}$$

$$\frac{1}{2}\left[\log|\mathbf{\Sigma}_1| - \log|\mathbf{\Sigma}_2| + tr(\mathbf{\Sigma}_2^{-1}\mathbf{\Sigma}_1) - N\right], \tag{2.49}$$

and if we rewrite it in terms of the precision matrices:

$$D((\mu, \mathbf{\Omega}_1)\|(\mu, \mathbf{\Omega}_2)) = \frac{1}{2}\left[-\log|\mathbf{\Omega}_1| + \log|\mathbf{\Omega}_2| + tr(\mathbf{\Omega}_2\mathbf{\Omega}_1^{-1}) - N\right], \tag{2.50}$$

Now that we have a concrete distance metric between two distributions, we can pose our search for the sparse Gaussian parameters as an optimization problem:

$$\min_{\mathbf{\Omega}} D((\mu, \mathbf{\Omega})\|(\mu, \mathbf{\Omega}_{\mathrm{ML}})) \tag{2.51}$$

$$\text{s.t.}\mathbf{\Omega}_{ij} = 0 \quad \forall i, j \in S, \tag{2.52}$$

where S is the set of entries in the precision matrix that we wish to drive to zero. This problem can easily be solved using the projected gradient descent method [19]. This method alternates two steps until convergence:

1. **Gradient step**: Take a small gradient step to decrease the objective.

2. **Projection step**: Set the entries of $\mathbf{\Omega}$ that are in S to zero.

This projected gradient descent approach will be used in Chapter 6 to sparsify Gaussian shape models in order to allow for more efficient discrete inference with part of the energy function given by the Gaussian shape probability.

2.7 Chapter Summary

In this chapter we have provided an overview of some of the probabilistic models and methods that will be used in this thesis. We began with some background on how to represent probability distributions, which led us to define the notion of independence of sets of variables. We described Bayesian networks and Markov networks as two types of PGMs that seek to encode the independencies between the variables in a graph structure.

Following this, we discussed probabilistic inference, and showed two existing algorithms for solving certain inference problems. Belief propagation is an algorithm for inference based on passing messages between clusters of variables, and Gibbs sampling is an algorithm based on drawing samples from the posterior to be inferred.

We then described the problem of learning Bayesian networks, including learning both the parameters of the model and the graph structure of the model. We defined the EM algorithm for learning parameters in the case of hidden or missing data, and structural EM for the case of learning the graph structure with missing data.

The final section of the chapter dealt with Gaussian random variables for the modeling of continuous distributions. We described how to learn the maximum-likelihood parameters for a multivariate Gaussian distribution, as well as how to learn a sparse representation that still faithfully models the data. In the next chapter, we will explore the basics of computer vision that will be used in the later development of this thesis.

Chapter 3

Fundamentals of High-Level Computer Vision

Computer vision (CV) can most easily be defined as the study of machines that attempt to "see." Sometimes these systems are targeted towards practical goals. For example, systems in the postal system allow for automatic mail sorting by optically recognizing the zip code on each envelope. Other times "seeing" may be the goal in itself for these systems, as in the task of object recognition, where the desired output is a list of objects present in the image.

In this chapter we will present an overview of some of the tasks that computer vision has been applied to, with varying degrees of success. We begin with a generic architecture that many CV systems follow, and then discuss a few of the tasks within the context of this architecture. The goal of this chapter is to familiarize the reader with the computer vision tools that will serve as the building blocks for the methods developed in this thesis.

3.1 Stages of Processing in a CV system

Although CV systems may differ in both their input/output functionality as well as the underlying algorithm, many follow a common architecture based on a pipeline of processing. Figure 3.1 shows a flow chart of the typical stages in a CV algorithm. The

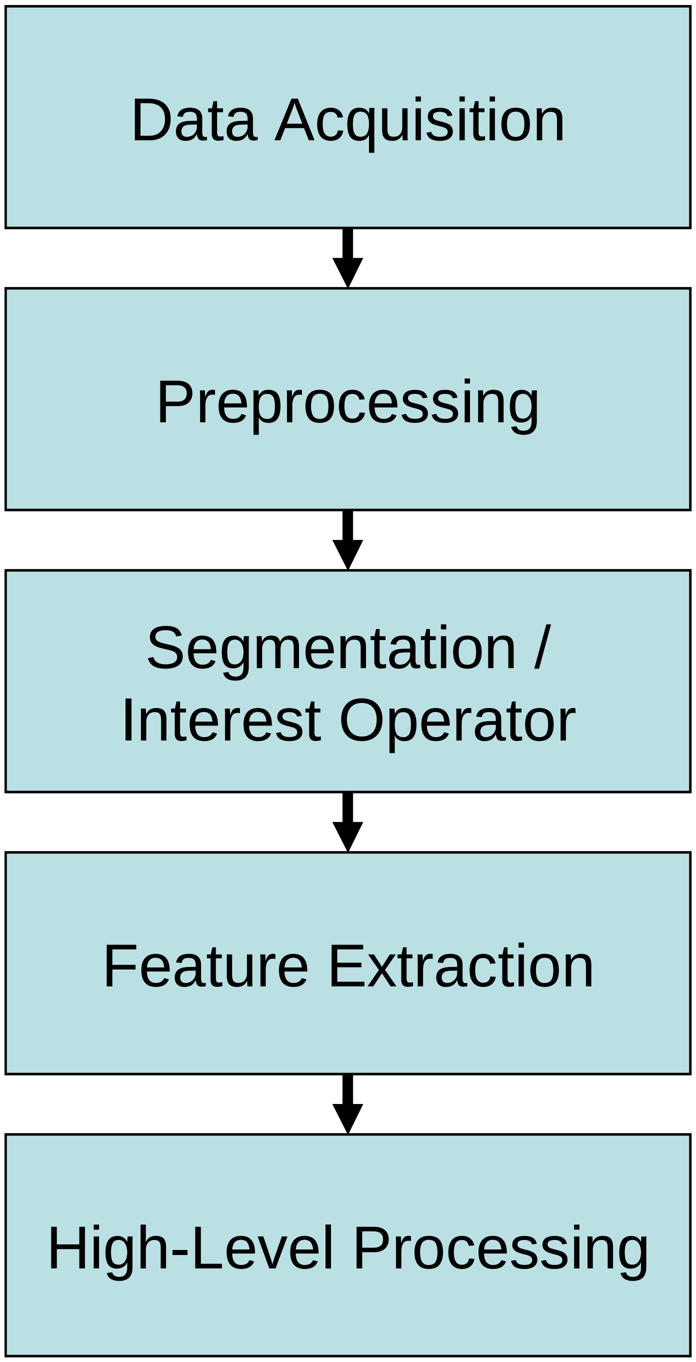


Figure 3.1: A typical processing pipeline for a computer vision system.

first step involves acquisition of the data. In many cases, this might involve simply collecting digital photographs from a personal collection, or crawling the web to obtain a set of digital photographs taken by others. In a robotics or more application-driven scenario, this may mean capturing a new digital photograph that will be further processed. With the development of new and varied sensors, computer vision is no longer limited to working only on digital image data. Other data sources that have been used in various applications include laser range sensors for sensing depth [57], infrared sensors for imaging heat or infrared radiation [117], and volumetric imaging techniques such as magnetic resonance imaging [84]. For the purposes of this thesis, we will assume that the input data to our system is a set of digital images containing intensities for each pixel as well as color information in some cases.

The second stage of many systems is to preprocess the data to put it into a format that is better for computing features. Because digital cameras today capture images at a resolution of 10 megapixels or more, often there is too much raw data to process. As a result, one of most common pre-processing steps is to downsample the image to a resolution that is easier to operate on. Other common preprocessing steps include converting color images into grayscale, or filtering an image to remove noise and high-frequency content. Preprocessing might also involve cropping the image to a region of interest, or adjusting the histogram of an image to somewhat correct for the lighting conditions.

Following preprocessing of the image, the next step is to segment the image into units that will be used in further stages. There are two predominant categories of segmentations: regions and interest points. The first type groups pixels into regions that cover the entire image. These regions might be disjoint "over-segmentations" of the image, where each pixel belongs to one region that attempts to preserve some coherence between the pixels in the region, or they might be overlapping patches of the image, typically square patches laid out in a grid-like fashion. An example of this form of segmentation is shown in Figure 3.2(b).

The other type of segmentation attempts to produce a sparse representation of the image, based on "interest points," or places in the image where it is likely that something important is happening. Many interest operators have been proposed,

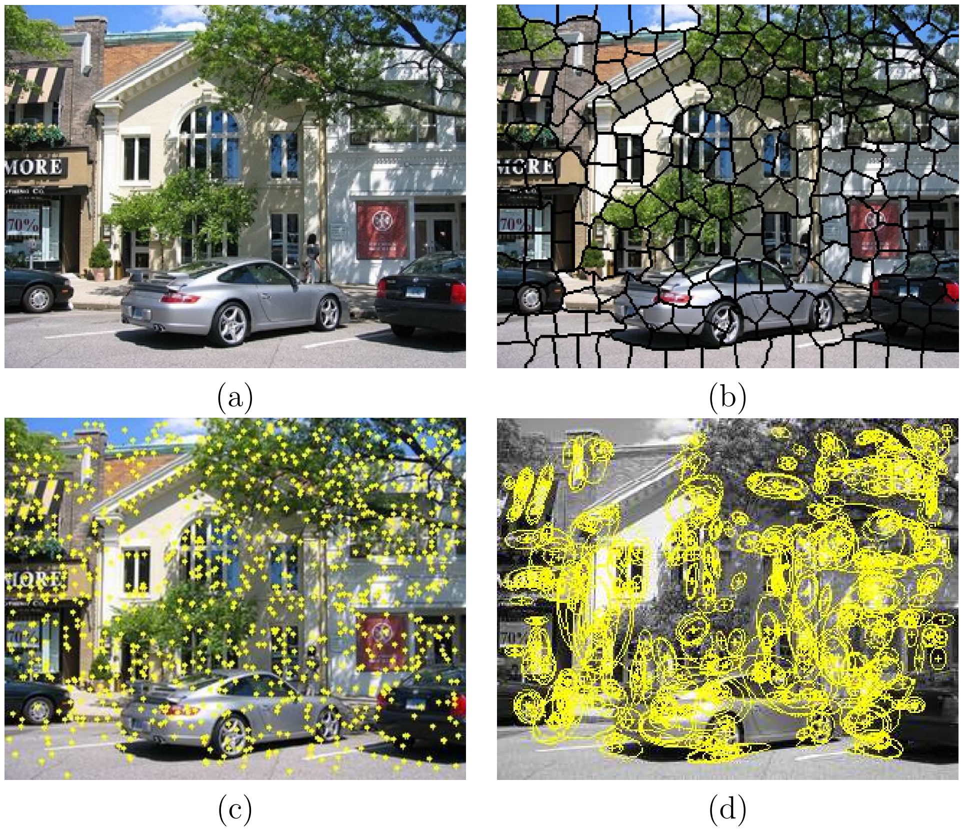
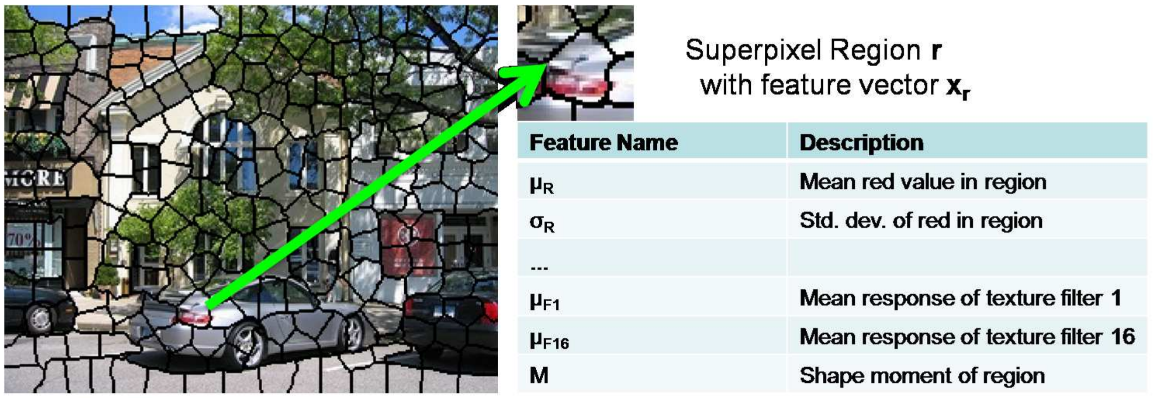


Figure 3.2: (a) An image of a street scene. (b) An oversegmentation of the image into coherent regions that are often called “superpixels.” (c) A sparse segmentation of the image into interest points that are selected using the SIFT keypoint detector. (d) keypoints extracted by the Harris-Affine keypoint detector of Mikolajczyk and Schmid [91]. It is often assumed that these keypoints will represent the important locations in the image, such as objects and boundaries between substructures.

generally based on the idea of finding either corners [91], areas of high gradients [85], or other “salient” regions in the image. The extracted keypoints often include a scale or orientation, which allows us to determine which pixels contribute to this point. Examples of the output from this type of segmentation are illustrated in Figure 3.2(c) and (d).

Once the image has been segmented, determining the fundamental processing unit for the image, we must attach a set of features to each unit. This set of image features often takes the form of a vector of real numbers computed from the pixels



Feature Name	Description
μ_R	Mean red value in region
σ_R	Std. dev. of red in region

μ_{F1}	Mean response of texture filter 1
μ_{F16}	Mean response of texture filter 16
M	Shape moment of region

Figure 3.3: Example feature vector for an image region.

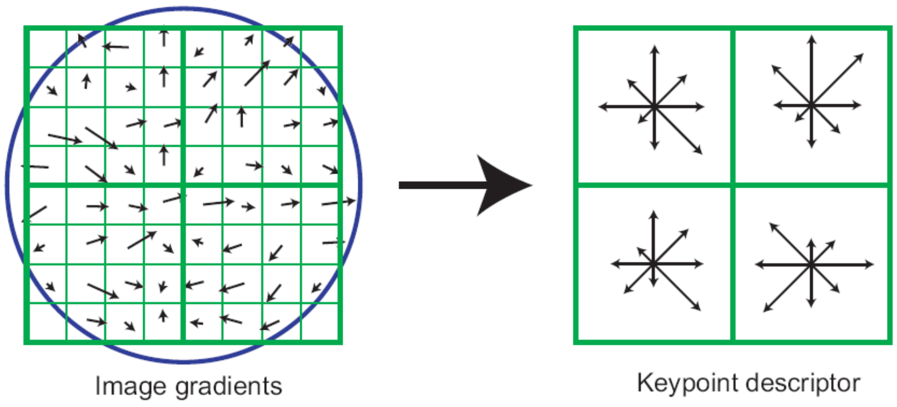


Figure 3.4: Illustration of the SIFT feature descriptor (Figure courtesy of [85]).

in the neighborhood of the segmentation unit. In the case of superpixels or regions, this might include the mean and standard deviations of several quantities across the pixels in the region. An example feature vector for regions is illustrated in Figure 3.3. For interest points, the feature vector is usually computed from a small region around the keypoint, based on the scale and orientation of the point. The well-known scale-invariant feature transform (SIFT) feature is a good example of this type of feature, and is illustrated in Figure 3.4.

In certain cases, we may instead compute features at every pixel location. This is typical in certain applications where we need a decision at every pixel. Examples of this include dense stereo, certain object detection methods, and the problem of

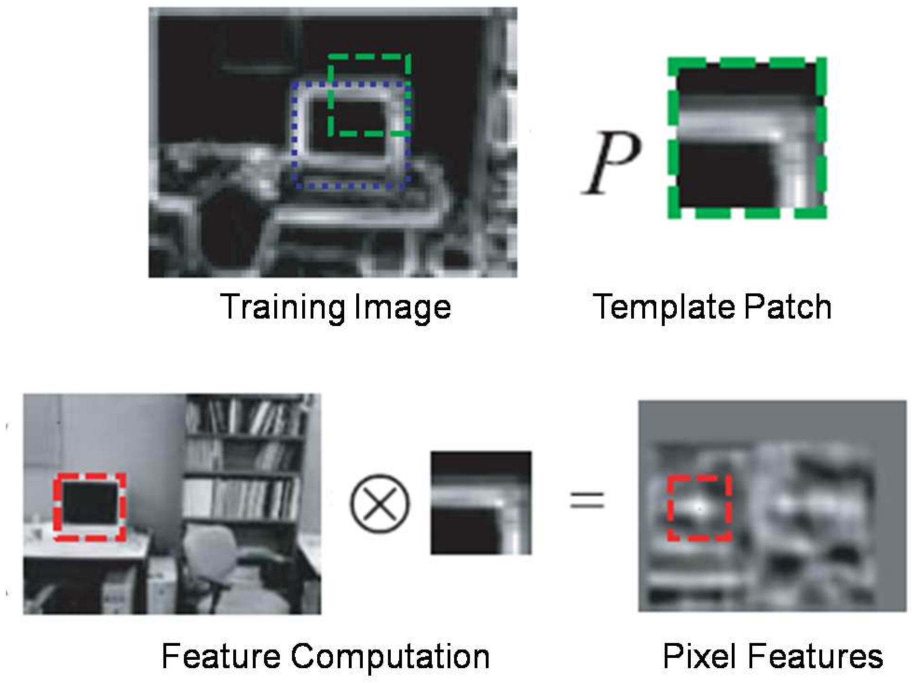


Figure 3.5: Patch-based image features (Images courtesy of Murphy et al. [94])

producing the oversegmentation in the first place. Dense feature vectors (one for every pixel) are often computed by passing a set of filters across the image. These filters can be small patches that pick up certain types of edges or textures, or may be larger patches that are "matched" at every pixel location to determine how much this pixel location "looks like" the template. This process produces a feature vector for every pixel. An example of the computation of such a feature is shown in Figure 3.5.

Because even small images have many pixels, computing these features at every location in the image can be very expensive. One common solution to this problem is to employ a multiresolution approach, where features are computed for a low resolution version of the image, some decisions are made, and then more features are computed for parts or all of the higher resolution versions. An illustration of this idea is shown in Figure 3.6.

The final step of the computer vision pipeline is the "high-level processing" that might detect objects, classify regions, estimate 3D structure, stitch multiple images, or perform some other task that produces usable output. In this thesis, we focus

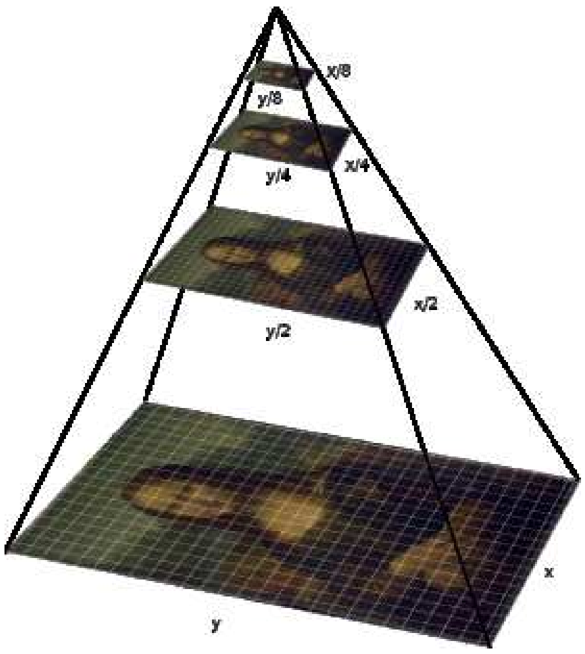


Figure 3.6: An image pyramid. Invariance to scale is achieved by running the algorithm on all images in the pyramid. (Figure from http://www.codeproject.com).

primarily on the last stage, the "high-level processing." In the following sections, we focus on three tasks that will be featured in later chapters. The first is object recognition, including categorization, detection, and localization of objects. The second is multiclass segmentation, where regions of the image are labeled by what type of stuff they contain. The final task we will introduce is that of 3D reconstruction, where the system estimates a depth for every pixel in the image.

3.2 Object Recognition

One of the most common goals of the computer vision processing pipeline is the recognition of *objects* in an image. This problem comes in two flavors. In a *categorization* task, the system is typically asked whether or not the image is a photo of an object from the target class. For example, we might show a series of photographs of animals, and ask the system whether or not each image is of a horse or not. A *detection* task, on the other hand, involves identifying both which objects are present in the

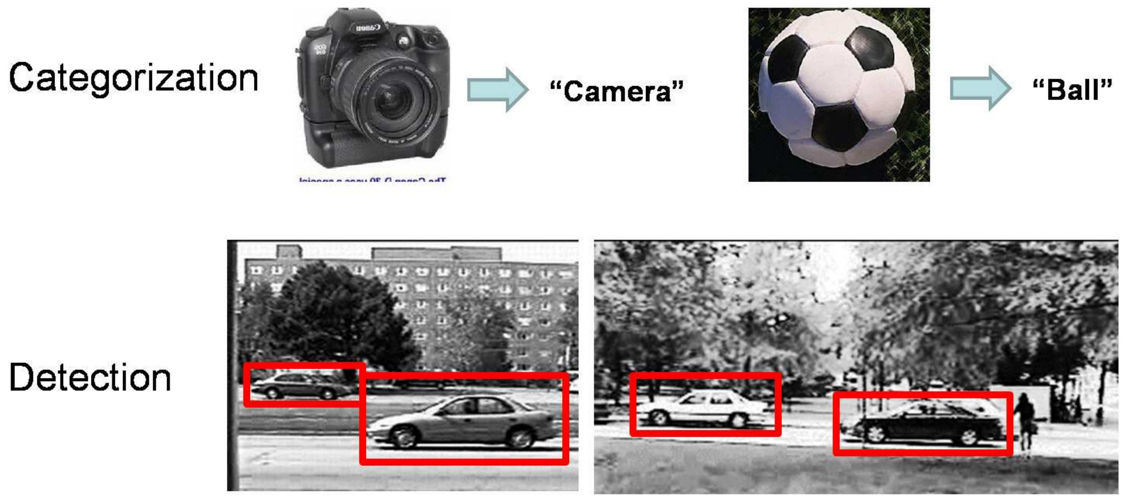


Figure 3.7: Example object categorization and detection outputs.

image and where they are. This second component, often called *localization*, can be represented in a number of ways. One common representation for localization is the so-called bounding box. This is a rectangle that tightly surrounds the object in the image. Figure 3.7 shows example outputs for the categorization and detection tasks.

Object recognition has become an important benchmark for computer vision models in recent years. Indeed, we have seen the creation of several large datasets as well as a series of "challenges" that evaluated several state-of-the-art methods on the same data to determine the relative merits of each. These datasets generally provide a set of images along with a set of groundtruth locations for the objects in the images.

Recognition methods can be broadly categorized based on the features that they use from the image and the model that they use over these features to determine the category and/or location of each object. In the following sections, we will explore some of the variation in these choices, focusing on the methods that we will use in the later chapters of this thesis.

3.2.1 Image Features for Recognition

Image features come in many shapes and sizes. In this section, we describe some of the image features that have been used in object recognition words. Suppose that we have already selected a set of feature points that include sparse or dense locations in the image for which we want to compute a feature vector. We now consider two categories of features.

Intensity-based Features

The first set of image features that we consider come directly from the intensity values in the neighborhood of the feature point. The simplest thing we can do is to take a fixed-size patch (say 11×11 pixels) around the point, and put the intensities into a large vector $\mathbf{x} \in \Re^{121}$. Representing the features directly this way is problematic for two reasons. First, most of the signal in the patch is in the lower-frequency content, while most of the noise or unimportant details are at higher frequencies. Second, if we wish to use a discrete method, this is far too large a domain to quantize in a straightforward way.

One common method for reducing the dimensionality of the feature is to use Principal Component Analysis (PCA), to project the vector of patch intensities down into a smaller space. This projection represents our original patch $\mathbf{x}$ as a mean value, plus a linear combination of "principal" modes of variation:

$$\hat{\mathbf{x}} = \mu + \sum_i a_i \mathbf{v}_i, \qquad \mathbf{x} = \hat{\mathbf{x}} + \textit{residual noise}$$

We then convert our feature vector from the values in $\mathbf{x}$ to the coefficients a_i, stacked into a vector $\mathbf{a}$. This vector has fewer dimensions and has much of the noise removed, and tends to be well modeled by a Gaussian distribution.

If we desire a discrete representation of our features, we can use the concept of *visual words*. Suppose that we extract N patches from an image, where each is represented by a vector $\mathbf{x}_n \in \Re^d$. The first problem is to map these features into a dictionary, so that we have a relatively small (tens to thousands) number of discrete values that a feature can take. One standard solution to this problem is run K-means clustering on the full training set, to obtain K cluster centroids. Following this, each feature vector $\mathbf{x}_n$ is assigned to the cluster k_n with the closest centroid. At this point, we introduce the visual word variables w, which take the value $w_n = k_n$ for feature

n. These are called visual words, because it is reminiscent of modeling a document as the set of words that it contains, where w_n represents the entry in our dictionary corresponding to word n. The words for an image are stacked into a vector $\mathbf{w}$.

For object recognition, we are often interested in determining whether a region in the image has a patch that "looks like" a part of the object. For example, if we are looking for cars, we might expect the bottom right of the car window to have a patch that looks like a wheel. In order to obtain features like this, we can create a patch dictionary of templates that come from the training data. We can then represent each feature point as a vector of "responses" to these templates, where a response is computed using the correlation between the target patch and the template. For a dictionary with K templates, this will produce a vector $\mathbf{x} \in [0,1]^K$ for each feature point. Sometimes these dictionary templates are augmented with an "offset" vector, allowing templates to produce predictions for objects that are not centered at the center of the patch. A more detailed description of this is included in the boosted detections section below.

If our keypoints also give us some information about the scale or orientation of the feature point, we can use features that are invariant to these affine transformations. In the work of Lazebnik et al. [78], for example, elliptical regions of interest are warped into circles of a fixed scale, which are then encoded using a rotation invariant descriptor such as the intensity-domain spin image. The affine-invariant nature of these descriptors is important for recognizing a specific object from various viewpoints, and has been shown to produce better results at recognizing similar textures in natural images.

Gradient or Edge-based Features

The second set of features that are commonly used in the object recognition literature are edge- or gradient-based features. Rather than modeling the actual image intensities, they seek to capture the consistency of changes in intensity.

Probably the most well-known such feature is the SIFT feature of Lowe [85]. This method includes both an interest point detector and a feature descriptor. The detector produces repeatable, affine-invariant keypoints. Each feature is represented

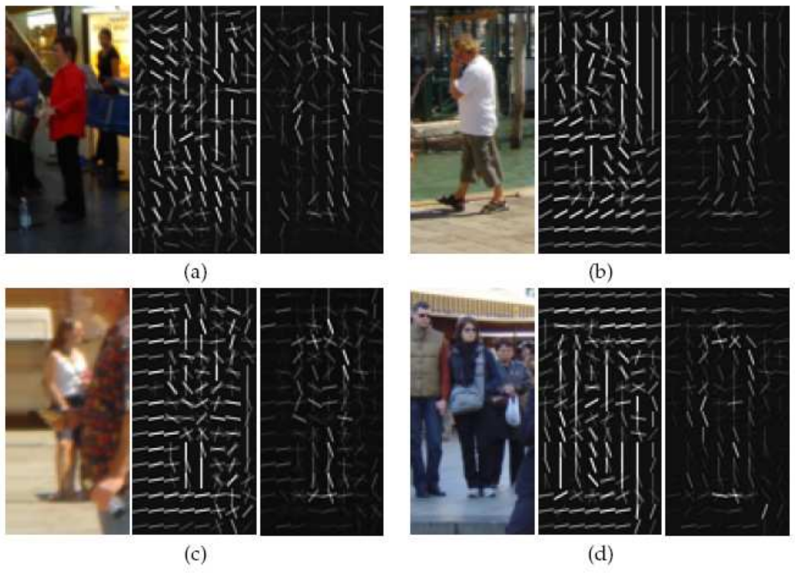


Figure 3.8: Example images and their computed HOG features. (Image courtesy of Dalal [31])

by a descriptor vector that looks in small 4×4 subwindows within the keypoint capture range, and computes histograms of the gradient orientations. This allows the descriptor to differentiate between vertical edges, horizontal edges, and corners, as well as feature points with striped or dotted patterns.

Another very related feature is the Histogram of Oriented Gradients (HOG) feature, developed by Dalal and Triggs [32]. Like SIFT, it is based on constructing a histogram of the gradient orientations in a small window. Figure 3.8 illustrates the HOG feature for a few small image windows. The original window is shown on the left, with a visualization of the HOG descriptor in the middle, and on the right are emphasized the dominant features used by the person detector of Dalal [31]. Unlike SIFT, however, the HOG features are computed in a dense grid over the image, and are not re-oriented according to the dominant direction, nor scale normalized.

While these descriptors use the gradient to encode shape, some features use the edge information directly. These methods begin by extracting an edge map from an image. An example image and its edgemap are shown in Figure 3.9. From this edge map, chains of edges are extracted, and built into features of various types. One

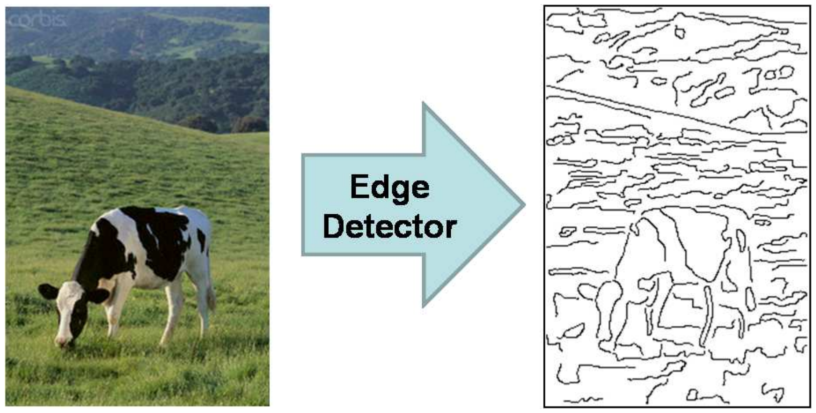


Figure 3.9: Edge detection extracts all pixels from an image that lie on an image edge.

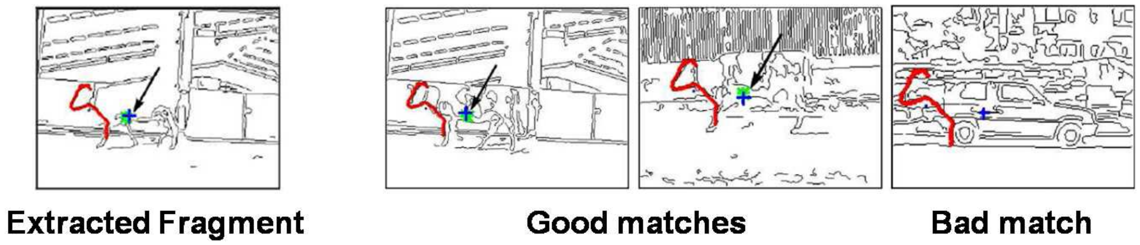


Figure 3.10: The boundary fragment model extracts small edge chains that correlate with the presence of certain object classes (Image courtesy of Opelt et al. [98]).

primary example of such a feature is the *boundary fragment* feature of Opelt et al. [98]. In this method, edge chains of various lengths are extracted from the edgemap of a training image (within the bounding box of the object). These chains are then "matched" to other images using the *Chamfer distance* metric of Borgefors [14]. These boundary fragments can be thought of as a sparse set of feature that can be matched to the edge chains of a novel image. Figure 3.10(left) shows an example boundary fragment extracted from the edge map of a cow object, and Figure 3.10(right) shows some example strong and weak matches to a test image.

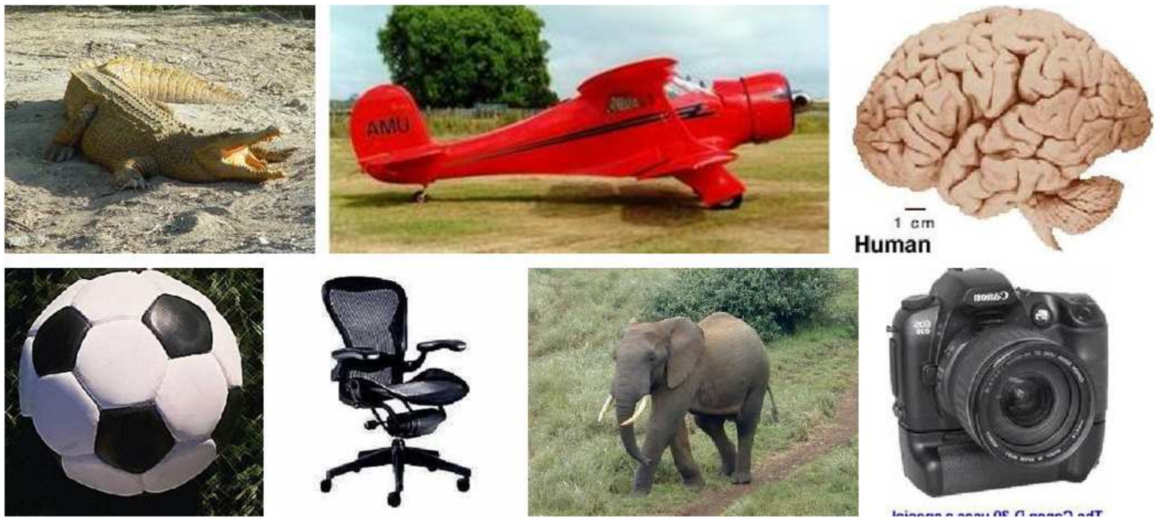


Figure 3.11: Example images from the Caltech 101 dataset, obtained from http://www.caltech.edu/caltech101.

3.2.2 Object Categorization Models

The categorization task is the problem of determining, for a previously unseen test image, what object class it is a picture of. One of the standard benchmark datasets for this problem is the Caltech 101 Classes dataset [42]. Some examples from this dataset, and the list of 101 classes, are given in Figure 3.11. Typically, methods for categorization are evaluated on their accuracy at selecting the correct single class that the test image belongs to.

One of the most straightforward methods for solving this problem is with the so-called "bag of words" model. Csurka et al. [30] give a good description of this method. Briefly, this method represents an image as a document composed of *visual words*, which come from mapping image features into a discrete dictionary. From this representation, it uses techniques from the natural language processing (NLP) community to learn models that classify these images.

Let's suppose we extract N patches from an image, and the corresponding visual word vector $\mathbf{w}$. Each category can now be modeled using a simple *Naive Bayes* model. Suppose that the image belongs to category c, with a prior probability $P(c)$. For this model, we assume that each visual words is generated independently according a multinomial distribution that depends on the category of the object. The full distribution over the category and the words is given by:

$$P(c, \mathbf{w}) = P(c) \prod_n P(w_n \mid c). \tag{3.1}$$

Given fully supervised data (known c for each image), training such a model is very easy, and we can perform classification of a new test image by finding the category c^* that maximizes the posterior:

$$c^* = \text{argmax}_c P(c \mid \mathbf{w}) = \text{argmax}_c P(c) \prod_n P(w_n \mid c). \tag{3.2}$$

While this model is efficient, quite intuitive, and has been shown to give good results in some cases, there are also some drawbacks. Perhaps the most glaring shortcoming of this method is that it does not take into account the *geometry* of an object, which seems to be an important characteristic of many object categories. For example, the bag of words model might have a visual word that corresponds to wheel-like things, which is likely to be a strong signal for the car category. Unfortunately, this model is equally happy to find a wheel-like feature at any location in the image. It would be great if we could find a way to capture the information that wheels tend to occur at the bottom right and bottom left of a (side view) car, rather than anywhere at all.

In order to address this problem, Perona and his colleagues proposed the *Constellation Model* Fergus et al. [45]. Each model for an object class contains a set of "parts" which match to features in the image. The probability of an assignment from parts to image features is determined both by the appearance likelihood (how well the features matches the appearance model for the part), and the shape likelihood, which measures how well the spatial layout of parts matches the distribution of part locations.

We will now describe the model as presented in Fergus et al. [45]. A set of keypoints are extracted using the Kadir and Brady [73] keypoint detector. The patches surrounding these keypoints are extracted and represented in PCA space as described above. We denote by A_i the n dimensional PCA coefficient vector for keypoint i. In addition to this, the Kadir-Brady detector gives us a characteristic scale for each point as well, which we denote by S_i. The pixel location in the image is represented as $X_i \in \Re^2$.

This approach uses a generative model over these quantities. Supposing there are P parts in the model, and N keypoints extracted from an image. The constellation model assumes that each part manifests as a single feature point, and we use a P-vector $\mathbf{h}$ to denote the assignment of parts to features. In order to classify a novel image, we marginalize out this assignments. In the presence of an object from class c, the generative probability is given by:

$$\begin{aligned} P(\mathbf{X}, \mathbf{S}, \mathbf{A}; \theta_c) =& \sum_{\mathbf{h}} P(\mathbf{X}, \mathbf{S}, \mathbf{A}, \mathbf{h}, \theta_c) \\ =& \sum_{\mathbf{h}} P(\mathbf{A} \mid \mathbf{h}, \theta_c) P(\mathbf{X} \mid \mathbf{S}, \mathbf{h}, \theta_c) P(\mathbf{S} \mid \mathbf{h}, \theta_c) P(\mathbf{h} \mid \theta_c) \end{aligned} \tag{3.3}$$

Categorization is performed by choosing the class c^* with largest posterior probability:

$$\begin{aligned} c^* =& \text{argmax} P(c \mid \mathbf{X}, \mathbf{S}, \mathbf{A}) \\ =& \text{argmax} P(\mathbf{X}, \mathbf{S}, \mathbf{A}, \theta_c) P(c) \end{aligned} \tag{3.4}$$

Figure 3.12 shows a visualization of a motorcycle model learned using this method.

3.2.3 Object Detectors

The typical datasets for categorization problems are close-up photographs containing only the object in question. Most natural images used in real-world applications, however, contain a number of different objects at different locations in the picture. In such images, it is more natural to define the question, “where are all the objects of type X in this dataset?” This it the object detection problem. A standard output for this task is a set of windows or *bounding boxes* that the algorithm believes contain objects from the category. Figure 3.13 shows some examples of successful detection outputs for images from the 2005 PASCAL Visual Object Classes Challenge [40].

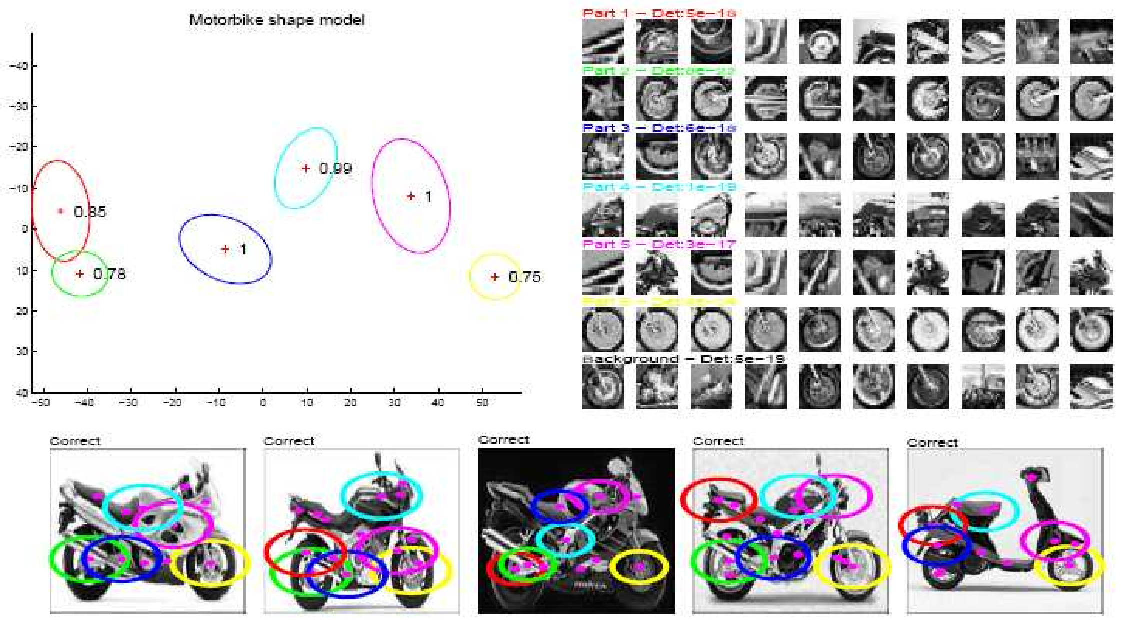


Figure 3.12: A visualization of the motorcycle model learned by the constellation method (Image courtesy of Fergus et al. [45]).

Many of the existing techniques for object detection rely on a "sliding window" approach [131, 94]. Under this paradigm, object detection is posed as a binary classification problem, where each "window" or region of the image is classified as to whether it contains an instance of the target object class. Each of these methods produces a score $f(W)$ for each window W, which may be defined by a center, width, and height. The large number of methods following this paradigm vary in the types of features used for each window, and the classification algorithm used. Because they consider all windows, they are guaranteed to "visit" every positive instance, and the hard part is just to find an effective scoring function. While these methods can be computationally expensive, recent work has shown how to make them more efficient [77].

One of the best known examples of this type of classifier is provided in Murphy et al. [94]. Features are generated by creating a large dictionary of patch "templates." Each template is extracted from a positive instance of the target class, and is defined by the triple (f_i, P_i, g_i). f_i is a filter channel applied to the original image, such as

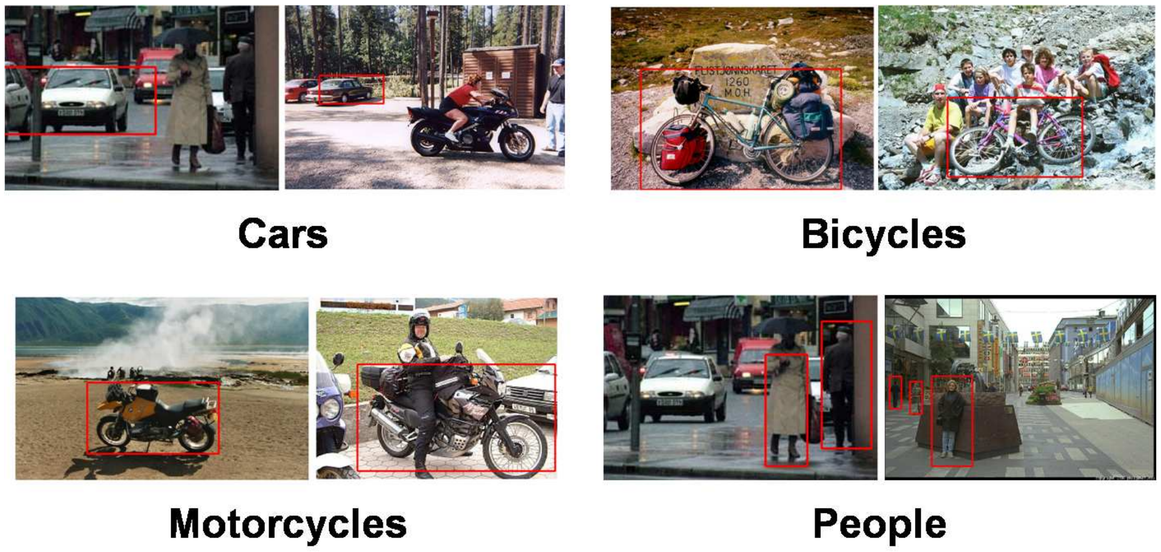


Figure 3.13: Example true detections of object in the 2005 PASCAL VOC Challenge dataset.

a Gabor filter, Laplacian of Gaussian, or other similar filter. Some examples of such filters are shown in Figure 3.14. P_i represents a template response patch, which is a small square patch extracted directly from the filtered training example. The final piece, g_i, is a spatial mask, indicating which location within the bounding box of the object the patch was extracted from. An example of such a dictionary entry is provided in Figure 3.15. The i^{th} feature, x_i, for a pixel is generated by convolving the image with f_i, correlating with P_i, and shifting (convolving) by the spatial mask g_i:

$$x_i = g_i * [(f_i * I) \oplus P_i] .$$

This feature provides a weak signal that the test bounding box has a patch that "looks like" the training patch in a similar location relative to the center of the box. Once these feature vectors have been collected for each pixel in the training instances, a boosted classifier is learned with decision stumps on the individual features. This detector serves as the basis for our detectors in Chapter 6. More details can be found there.

Another prominent example from this family of algorithms is the HOG detector of

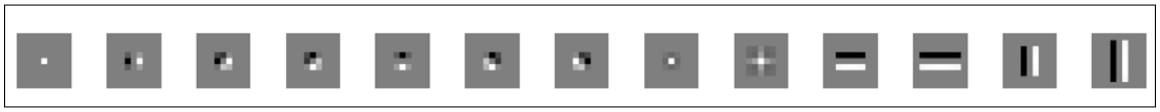


Figure 3.14: Texture filters used for object detection (Image courtesy of Murphy et al. [94]).

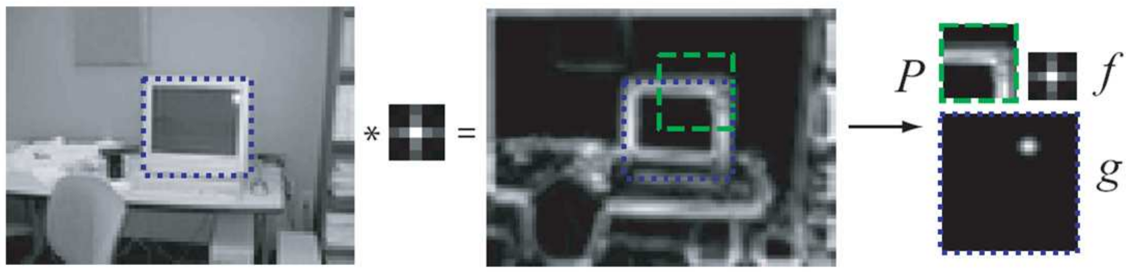


Figure 3.15: Filter-patch-based weak detector (Image courtesy of Murphy et al. [94]).

Dalal and Triggs [32]. This detector uses the HOG feature (described earlier). Because they are dense, HOG features tend to do good job of encoding the entire shape of an object, as opposed to other approaches that target the "salient" edges or patches of an object. Dalal and Triggs [32] showed that an SVM over the the HOG descriptor outperformed the same classifier over SIFT descriptors for pedestrian detection, and further work has shown good performance on a number of other object classes [31]. The HOG detector of Dalal and Triggs [32] is used as the base state-of-the-art detector in many of the models developed in this thesis.

3.2.4 Evaluating Object Detectors

Evaluating the effectiveness of an object categorization system is straightforward: we can report an overall classification accuracy, indicating how often we predict the correct class. In the case of object detection, things are not quite so simple. Consider a typical $N \times N$ image that contains two cars. If we fix the aspect ratio $(2 : 1)$ of a car, we still have N^3 possible windows (N^2 bounding box centers at N scales). This means that only 2 out of N^3 windows are positive, and the rest are negative. For any reasonable N, our accuracy is going to be effectively perfect by guessing negative for all windows.

This is solved by considering the precision-recall curve. In the case of object detection, we have two somewhat competing goals. The first is to find as many of the objects in the target class as possible, and the second is to minimize the number of false positives, or windows that we classify as the object when it is actually not present. We measure the first using the *recall rate*, which is given by:

$$REC = \frac{TP}{NP}, \tag{3.5}$$

where TP indicates the number of true positives (detection windows classified as positive that are truly positive), and NP indicates the number of groundtruth positives in the dataset. The second goal is measured using the detector *precision*, which is given by:

$$PREC = \frac{TP}{TP + FP}, \tag{3.6}$$

where FP indicates the number of false positives returned by the detector. For a given detector, we can report both of these numbers, and our goal is to maximize both.

Generally, most object detectors calculate a score for each window in the image, and return all windows with scores above some threshold as positive detections. We have thus far not discussed how to set the threshold. If we set the threshold low enough, we will only get the very confident objects, in which case our precision is likely to be relatively high. However, if we set the threshold very low, then we will return almost all of the windows as positive, and we are more likely to get all of the true positives. This will lead to a high recall, while suffering from a low precision.

In order to visualize this tradeoff, we can vary the threshold to produce a curve that maps out the feasible (recall,precision) pairs. This curve is called the *precision-recall curve* (PR curve), and an example is shown in Figure 3.16(a). The ideal point on this curve is in the top right, and a better method achieves a higher precision at all or most levels of recall. In order to summarize this curve in a single number, we consider the area under the curve. However, these PR curves tend to be very noisy, so

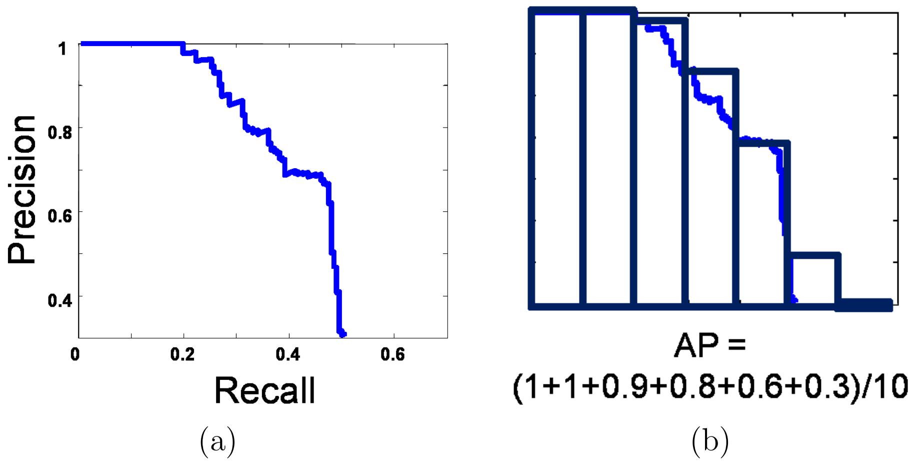


Figure 3.16: (a) An example precision-recall curve for evaluating an object detector. (b) A robust variant of the area under the curve (AUC) known as average precision (AP).

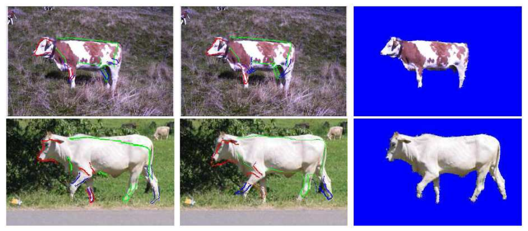
the standard measure is the so-called *average precision*, a robust variation on the area under the curve. Figure 3.16(b) illustrates how the average precision is computed.

3.2.5 Refined Localization: Beyond the Bounding Box

While for some applications producing a bounding box for each detected object is sufficient, there are times when we require a more refined localization. For example, if we want to "cut out" an object from an image, to be used in a new image or figure, for instance, we want to know exactly which set of pixels correspond to the object. In Chapter 6 we will explore more cases where a refined localization of the object is useful.

Many early approaches to this problem used deformable templates [144], or active shape models [24]. More recently, several methods have been developed that take a more global approach.

Yu and Shi [143] use a graph-based approach, and Kumar et al. [75] used a model based on pictorial structures, to segment a small set of known objects from cluttered backgrounds. Borenstein et al. [13] segment objects of known classes from the background by combining "bottom-up" cues such as texture and intensity similarity with


(a)

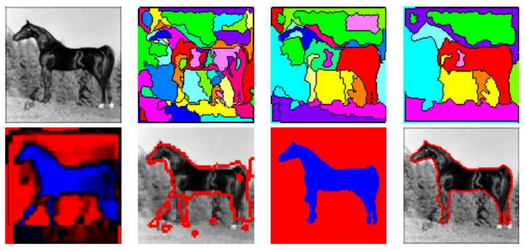

(b)

Figure 3.17: Existing object-based segmentation methods. (a) The OBJ CUT algorithm of Kumar et al. [75] uses pictorial structures to match a model to the image (Image from Kumar et al. [75]). (b) Borenstein et al. [13] combines top-down cues with bottom-up features to produce an object-specific segmentation (Image from Borenstein et al. [13])

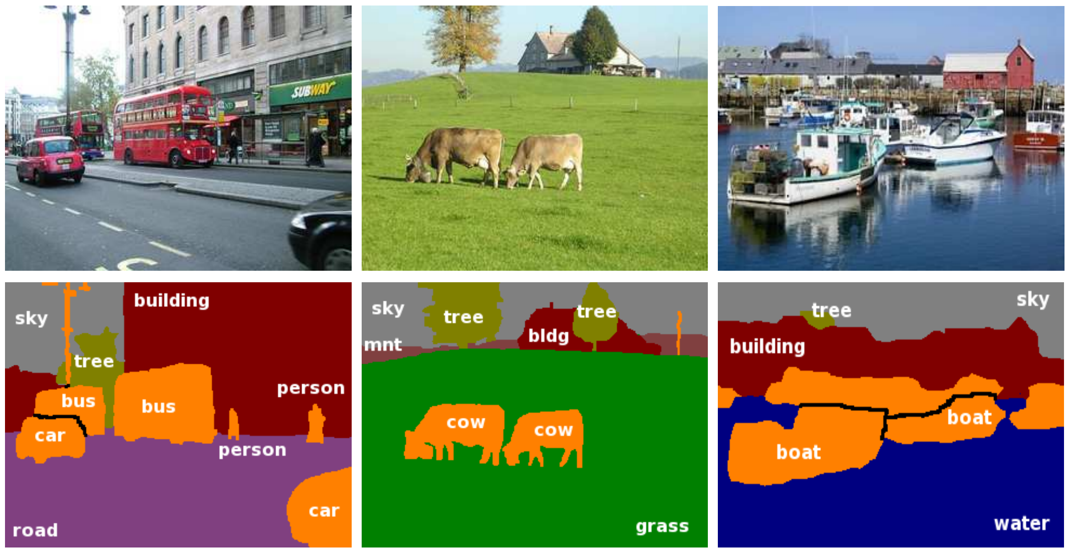
"top-down" cues based on class-specific image fragments. Leibe et al. [80] use an object detector similar to the patch-based detector described above. In addition to the appearance of each patch, they also store a local foreground/background segmentation mask for each patch, and "back-project" the best matching patches to obtain a soft segmentation map of the object.

Figure 3.17 shows example inputs and outputs for this task. Typically these results have been evaluated either qualitatively (based on how it looks), or on pixel accuracy (i.e., the fraction of pixels correctly classified as foreground or background). In Chapter 6, we will use outline-based metrics, and explore what we can do once we have such a refined localization.

3.3 Multi-class Image Segmentation

Another problem that the high-level vision community has made great progress on in the past several years is the problem of multiclass image segmentation. For this task, the goal is to label every pixel in the image with the class that it belongs to. Figure 3.18 shows three example images and true labels for this task.

Suppose that we have selected a set of classes for a particular dataset, denoted by $\mathcal{C}$ (e.g., we might have $\mathcal{C} = \{grass, road, building, cow, etc.\}$). We introduce a label variable C_i for each pixel i, which indexes into the set of classes, indicating which



(a) Urban (b) Rural (c) Harbor

Figure 3.18: Example (a) urban, (b) rural and (c) harbor scenes that we would like to label. Figure shows original image, and the semantically labeled regions (groundtruth).

class this pixel belongs to. We can evaluate a labeling based on its accuracy:

$$acc(\mathbf{C}) = \frac{\sum_i 1\{C_i = C_i^*\}}{P},$$

where P is the number of pixels in the image, and C_i^* is the groundtruth (manually labeled) class of pixel i. We can also look at a confusion matrix, where entry j, k indicates the number of pixels with true class j that are classified as belonging to class k. Figure 3.19 shows an example confusion matrix.

3.3.1 CRF Models for Multi-class Segmentation

One common model used to solve this problem is a conditional Markov Random Field (CRF) over pixels or superpixels in the image (e.g., [119], [82], [62]). In this discussion, our description is based on the model of Gould et al. [58], but other methods are quite similar in many respects. Each region (pixel or superpixel) is assigned a variable S_i (as above), and the probabilistic model consists of two types of terms. Figure 3.20

Pred / True	Sky	Grass	Building	Water
Sky	1823	13	116	374
Grass	24	3782	156	308
Building	99	168	701	257
Water	337	87	139	1178

Figure 3.19: An example confusion matrix for the multiclass segmentation problem.

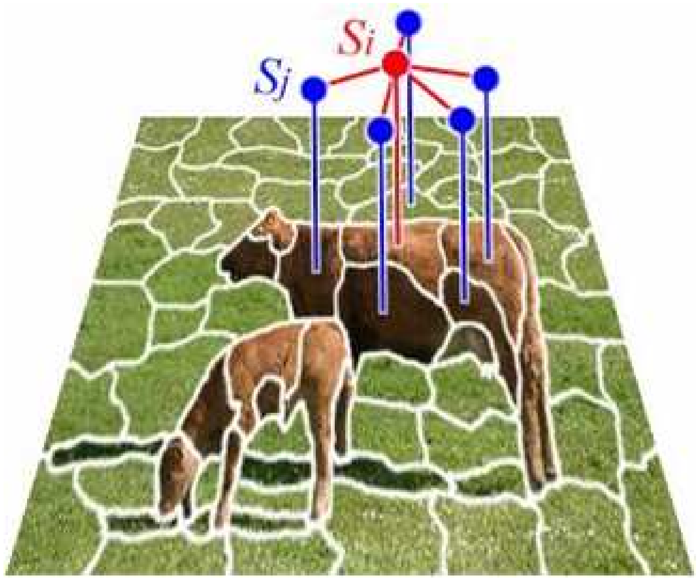


Figure 3.20: A visualization of the multi-class segmentation CRF.

illustrates the setup of the multi-class segmentation CRF.

The first set of potentials models the correlation between the local appearance of the region and the class, and is denoted by $\psi(S_i; \mathbf{A}_i, \mathbf{X}_i)$. This potential conditions on $\mathbf{A}_i$, the appearance of region i, and $\mathbf{X}_i$, the location of region i. In Gould et al. [58], the raw region features include the mean and standard deviation within the region of the various color and texture components, as well as several geometric features based on the shape of the region. A boosted classifier for each class is learned on top of these features, and the output of the boosted detector for each region is used as features in a log-linear potential ψ. Similarly, Shotton et al. [119] uses three singleton terms, based on color, texture, and location in the image. He et al. [62] use a multi-layer perceptron classifier as the local potential term.

The second type of potential models the smoothness between nearby regions. Intuitively, if two regions are neighbors, we expect them to have the same class. Also, if region i is above region j, and region j is a building region, we believe that region i is more likely to be a sky region than a road region (roads generally don't occur above buildings). We can model this using pairwise potentials between neighboring regions, denoted by $\psi(S_i, S_j; \mathbf{A}_i, \mathbf{X}_i, \mathbf{A}_j, \mathbf{X}_j)$, which can in general depend on the appearance of both regions, and their locations.

With these two types of potentials, we create a joint distribution over all of the labels, given by:

$$P(\mathbf{S} \mid \mathcal{I}) = \frac{1}{Z(\mathcal{I})} \prod_i \psi(S_i; \mathbf{A}_i, \mathbf{X}_i) \prod_{ij} \psi(S_i, S_j; \mathbf{A}_i, \mathbf{X}_i, \mathbf{A}_j, \mathbf{X}_j). \tag{3.7}$$

These models are typically learned with fully supervised data: each pixel in the training images is manually annotated with the class label. Because CRF learning is in general intractable, however, the parameters for these models are learned in a variety of different approximate ways. In particular, Shotton et al. [119] use piecewise training, where the various terms are learned independently, and hand-engineered weights are used to balance between the different terms. He et al. [62] use contrastive divergence training, and Gould et al. [58] use the pseudolikelihood training objective, which conditions the likelihood of each variable on the values of all of its neighbors.

At inference time, most of these methods are interested in the most likely labeling of the image, or the MAP assignment to the class variables. Again, because of the complexity of the models, exact inference is intractable, and approximate inference must be used. He et al. [62] use a Gibbs sampling based approach, while Gould et al. [58] use a belief propagation based inference. Shotton et al. [119] use the α-expansion algorithm of Boykov et al. [16] to perform inference on their model.

3.3.2 Other Approaches

While the CRF approaches are the most common in the literature, other methods have been developed for solving the multi-class segmentation problem. Rather than use explicit pairwise constraints, Schroff et al. [113] look at pixel features of larger spatial extent to help impose the smoothness constraint. Their features involve histograms of textons in a large window around the target pixel. This allows very efficient classification, as each pixel can be classified independently, once the large feature has been computed.

Yang et al. [140] use region and keypoint features to classify each of the superpixels in the first round of inference. Following this step, a shape model for the object is used to greedily update the segmentation so that a high score is achieved both for the object shape and the region appearances.

3.4 3D Reconstruction from a Single Image

There is a great deal of work in the computer vision community on extracting the 3D shape of a scene. Most often, this is done through the use of sophisticated sensors, such as time-of-flight lasers [36], or through the use of multiple cameras (e.g., stereo methods [86]) or multiple views of the scene (e.g., structure from motion [34]). More recently, researchers are attempting to reconstruct the 3D shape of the scene from a single natural image.

This task of 3D reconstruction from a single image has several interesting characteristics. It is inherently much more difficult than the other approaches described above. In sensor-based, stereo, or structure from motion, the mathematics of the reconstruction are perfect, and the difficulty in designing these systems lies in the noise and ambiguity of the image features. In the case of a single image reconstruction, however, the true answer is inherently ambiguous. For an image, there is an infinite set of 3D structures that will produce the same 2D image.

Despite this, however, humans are still often able to produce qualitatively accurate reconstructions of most natural image scenes. This is because we have a very strong

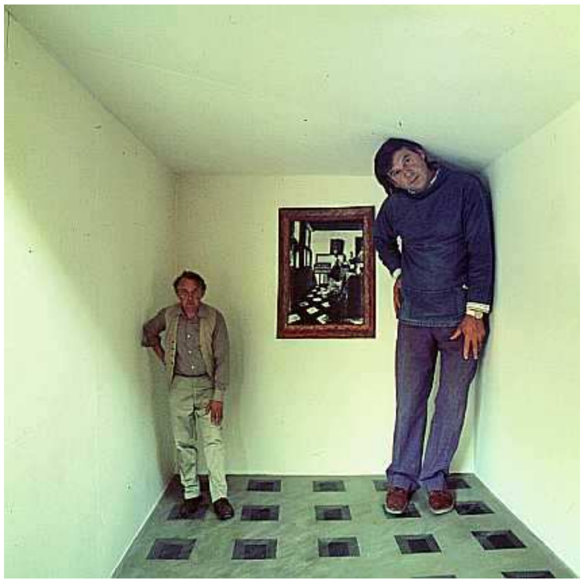


Figure 3.21: The Ames room illusion (image courtesy of http://illusionism.org/Depth-Perception/Ames+Room/).

prior about the types of structures that occur in the world, and we have seen many images of these structures in the past. Indeed, when we look at the common illusion image of Figure 3.21, the reason it looks so strange is because it violates the priors that we have developed over many years.

Furthermore, natural images taken by humans tend to have the camera placed in relatively the same position and orientation relative to the ground. Because of this, and the ability of humans to somewhat solve this problem, we can plausibly believe that developing an algorithm to solve this problem is possible.

Another important characteristic of this problem is that many of the cues that humans use are based on relatively low-level image primitives. For example, when we see a straight intensity-edge in the image, we are likely to believe that this corresponds to a straight line in the 3D world. This leads us to believe that we can use approaches similar in principle to the multi-class segmentation methods described above to solve this problem.

Formally, the goal of this problem is to determine the 3D coordinates of the point in space that generates each pixel in the image, in a camera-centric coordinate system.

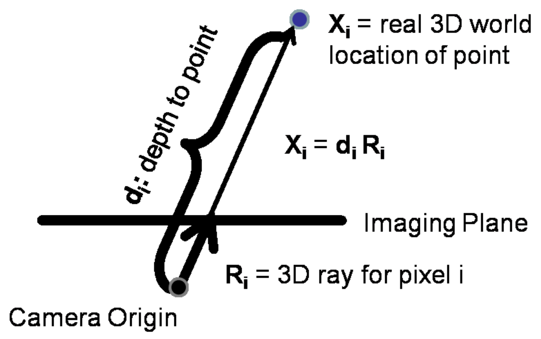


Figure 3.22: The physical setup for the depth reconstruction problem.

We introduce random variable $\mathbf{X}_i = (x_i, y_i, z_i)$ for each pixel i in the image to indicate the 3D coordinates of the point. Assuming a known camera (we know the focal length, resolution, etc.), we can compute the real-world ray $\mathbf{R}_i$ that corresponds to each pixel i. If the object that emitted the light received at pixel i is at a distance of d meters from the camera, then the corresponding point is given by: $\mathbf{X}_i = d\mathbf{R}_i$. Figure 3.22 illustrates this idea. Thus our goal is to estimate the single real number d for each pixel.

Once we have produced a reconstruction, we can evaluate it both qualitatively and quantitatively. Qualitatively, we might visualize the resulting 3D scene using a virtual world viewer (see Figure 3.23) to see if it matches a human intuition. For a quantitative evaluation, we require a groundtruth measurement of the depth for each pixel. This can be obtained using a laser sensor that is aligned with the camera coordinate system. Figure 3.24(a) shows a system developed for the STanford AI Robot (STAIR) to collect this data. Using this, we can measure our reconstruction error in absolute terms (e.g., root-mean-squared meters) on test data.

3.4.1 CRF Models for 3D Reconstruction

One of the primary methods for solving the problem of 3D reconstruction from a single image was introduced by Saxena et al. [111]. This approach uses a CRF model over superpixels. Each superpixel region r gets a random vector $\alpha_r \in \Re^3$ indicating the plane that the region lies on. If the ray corresponding to some pixel i in region r

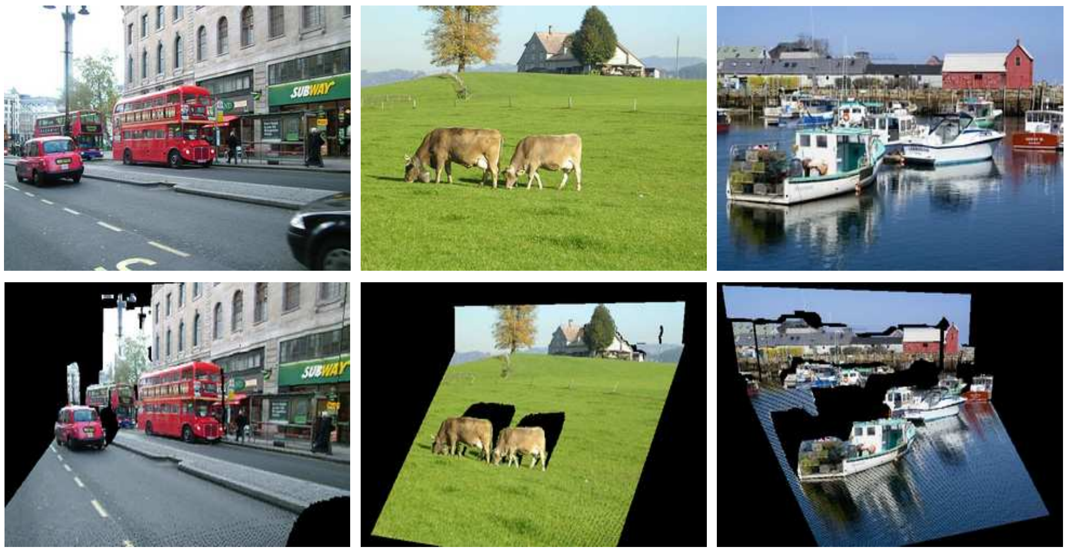


Figure 3.23: Example images and a visualization of their 3D reconstruction. From this image we can qualitatively evaluate whether our reconstruction is accurate.

is $\mathbf{R}_i$, then the depth of that point is given by $d_i = \frac{1}{\mathbf{R}_i^T \alpha_r}$.

This model attempts to estimate the depth using a linear function of a set of features of the region. Denote by $\mathbf{f}_r$ the feature vector for region r, and the linear parameters by $\mathbf{w}$. Then we have: $\hat{d}_r = \mathbf{w}^T \mathbf{f}_r$, where $\hat{d}_r$ is the estimated depth of region r. With this estimated depth, we can evaluate the relative error using:

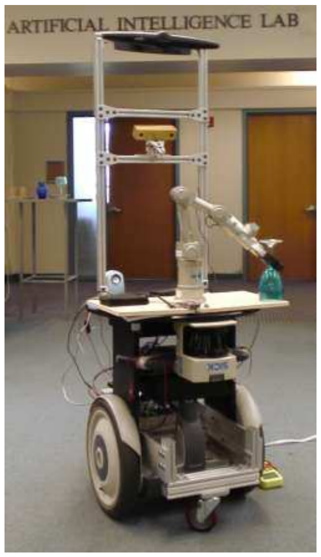
$$error = \frac{(\hat{d} - d)}{d} = \frac{\hat{d}}{d} - 1 \tag{3.8}$$

This error is used to construct CRF potentials of the form:

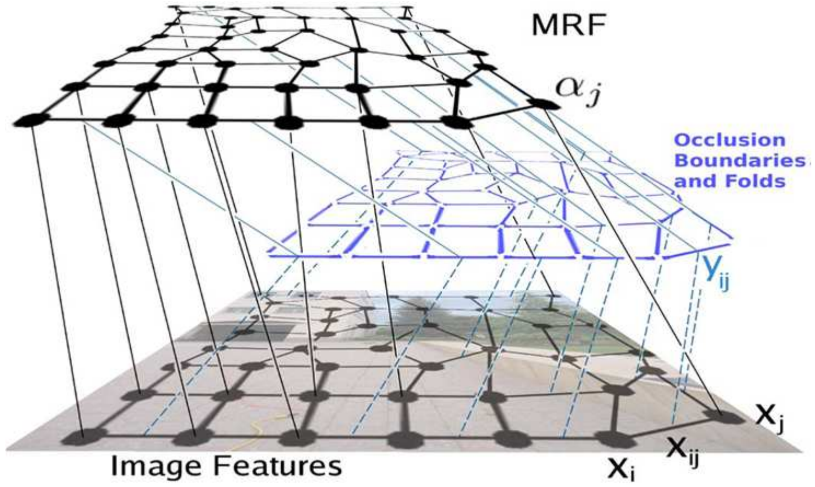
$$\psi(\alpha_i; \mathcal{I}) = \exp\left[-\sum_i \left| \mathbf{R}_i^T \alpha_{r(i)} (\mathbf{w}^T \mathbf{f}_{r(i)}) - 1 \right| \right], \tag{3.9}$$

where $r(i)$ is the region that pixel i belongs to.

This method similarly constructs pairwise potentials over neighboring regions to enforce connectedness, and nearby and distant regions that are believed to lie on the same plane. The connected structure constraint selects points i and j on the boundary of neighboring superpixels as points that should be close (i.e., if the structure is



(a) STAIR Robot



(b) Depth CRF

Figure 3.24: (a) The STAIR robot with an laser rangefinder aligned to a camera for simultaneous measurement of an image and the depths corresponding to each pixel. (b) The CRF model for the depth reconstruction problem.

connected, they should be neighbors). The potential over these two points penalizes the distance, and takes the form:

$$\psi(\alpha_i, \alpha_j; \mathcal{I}) = \exp\left[-y_{ij}\left|(\mathbf{R}_i^T\alpha_{r(i)} - \mathbf{R}_j^T\alpha_{r(j)})\sqrt{\hat{d}_i\hat{d}_j}\right|\right]. \tag{3.10}$$

For co-planarity, we assume that point k, which is in the center of the same region as j, should be close to the plane defined for point i. This potential takes the form:

$$\psi(\alpha_i, \alpha_j; \mathcal{I}) = \exp\left[-y_{ij}\left|(\mathbf{R}_i^T\alpha_{r(i)} - \mathbf{R}_i^T\alpha_{r(j)})\hat{d}_k\right|\right]. \tag{3.11}$$

Parameters of the model are learned using piecewise training, where each type of term learns parameters independently. The final (unnormalized) CRF probability is convex in the unknown variables (the α_i's). This allows MAP inference to be performed relatively efficiently using convex optimization methods (a linear program (LP) in this case). Figure 3.24(b) illustrates the CRF that is used for 3D reconstruction of an image.

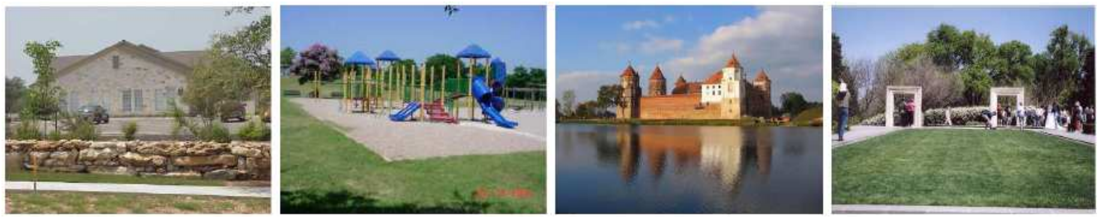


Figure 3.25: Some example images used in the work of Saxena et al. [111] for reconstructing the 3D scene geometry from a single still image.

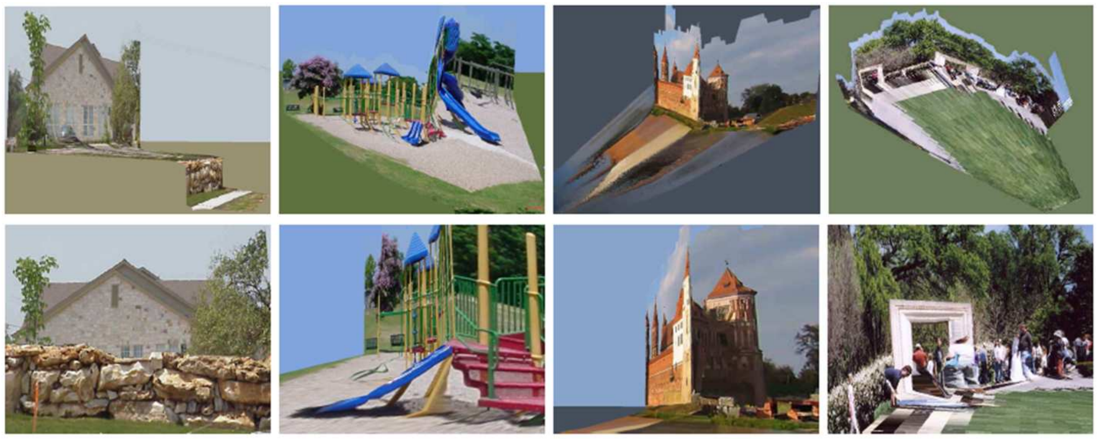


Figure 3.26: Some example results of Saxena et al. [111]. The images from Figure 3.25 are reconstructed, and shown from two novel viewing angles.

Figure 3.25 shows some of the images that Saxena et al. [111] used to test their algorithm. The results are visualized in Figure 3.26 by showing the reconstructed scene from two novel viewing angles. The views from these new angles are surprisingly realistic, despite the algorithm having no actual 3D information about the scene.

3.5 Chapter Summary

In this chapter we introduced many important fundamental concepts in high-level computer vision. We presented a "typical" processing pipeline for a computer vision system, describing each of the stages.

In section Section 3.2 we described some common approaches for detecting objects in images. We detailed various types of image features that are used toward object recognition, including both intensity- and edge-based features. We illustrated a set of categorization models that included extremely simple methods as well as some methods that attempt to use shape. We outlined sliding window object detectors and how these detectors are evaluated, and briefly discussed the small set of works that seek to produce a more refined localization than just a bounding box.

Following this, we gave an overview of multi-class image segmentation in Section 3.3. We described the problem, briefly outlined a solution based on a CRF and touched on some alternate solutions. Section 3.4 introduced the problem of reconstructing the 3D geometry of a scene from a single still image. We quickly covered a solution based on a CRF and showed some results.

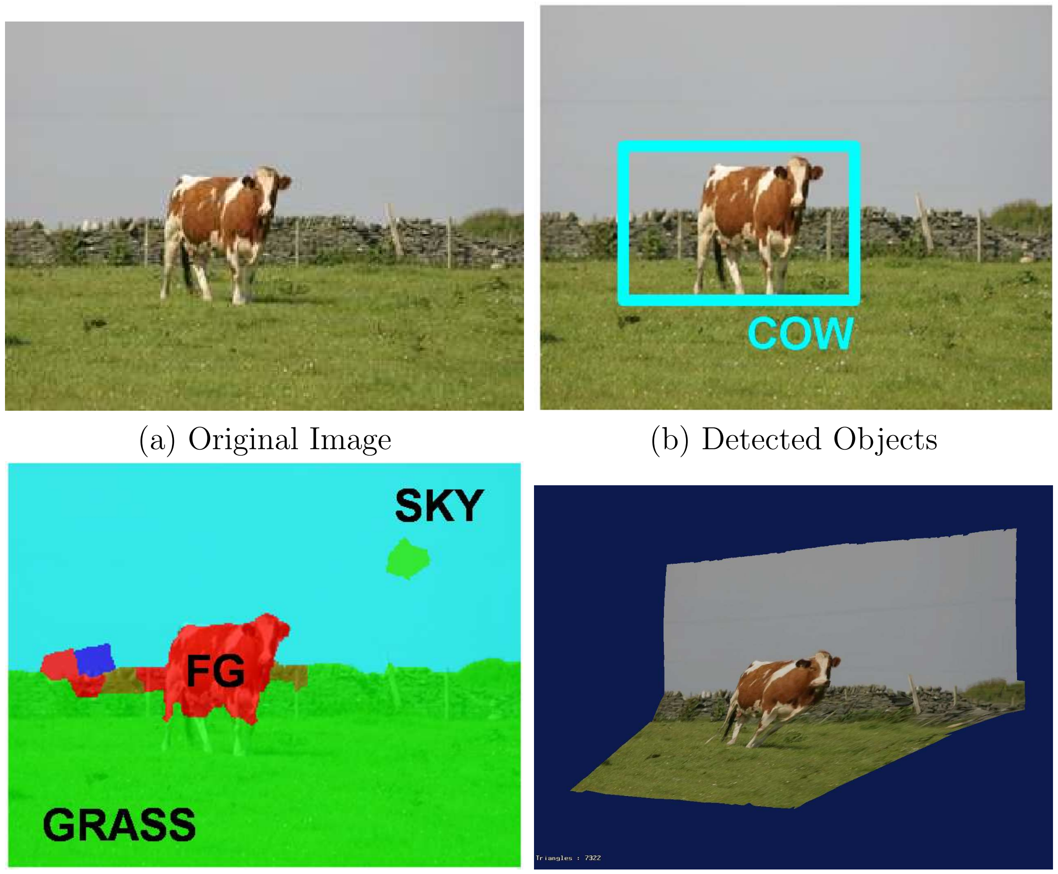
In the next three chapters, we will use the tools described in this chapter and Chapter 2 to develop three novel methods that address problems in high-level computer vision.

Chapter 4

Integrating Vision Models: Cascaded Classification Models

In this chapter, our goal is to develop a system that identifies the primitives that make up a scene. Presented with a previously unseen image, we want to process the image in order to answer a number of questions, including: (i) What type of scene is it (e.g., urban, rural, indoor)? (ii) What meaningful regions compose the image? (iii) What objects are in the image? (iv) What is the 3D structure of the scene? As discussed in Chapter 1, many of these questions are coupled—e.g., a car present in the image indicates that the scene is likely to be urban, which in turn makes it more likely to find road or building regions. Figure 4.1 shows an example image with some of the outputs that we hope to capture. With this in mind, our goal is to produce a framework for integrating models for these component sub-tasks in such a way that exploits the dependencies between them.

We can consider two alternative approaches to achieving this goal. In the first approach, we might build a single probabilistic model over the output variables that correspond to each of the sub-tasks. In a single large probabilistic graphical model, these variables will be coupled through CPDs in the case of directed models (Bayesian networks), or potential functions in the case of undirected MRFs. Within this model, we will have the flexibility to use whichever image features are appropriate for each



(a) Original Image (b) Detected Objects

(c) Classified Regions (d) 3D Structure

Figure 4.1: Some properties of a scene required for holistic scene understanding that we seek to unify using a cascade of classifiers.

task, and we can construct our potential functions to reflect the sophisticated relationships present in our domain. Unfortunately, taking such an approach requires a great deal of expertise on the part of the designer. While state-of-the-art models already exist for many of the tasks of interest in high-level vision, these models have been carefully engineered over several years by experts in that particular problem. These models are often tricky to modify, or even simply to re-implement from available descriptions. Furthermore, once we have designed a single model to encompass all of the tasks we have chosen, introducing an additional set of variables corresponding to a new task might be a nightmare, ruining our learning and inference algorithms for the old model.

A second possible approach to model integration involves a simpler interfacing of the existing methods. Because of the complexity of some of these algorithms, it is sometimes desirable to treat them as “black boxes,” where we have we have access only to a very simple input/output interface. In this case, we can build a system over top of these methods that tie the output of each component into the input for the other components, allowing contextual signals to be sent in a way that does not break the original models. In order to do this, we can require only a simple interface to the methods, including the ability to train on data and produce classification outputs for each data instance. While this affords a great deal of flexibility, we pay for this with a loss of modeling power. Because we are constrained by the input/output interfaces of existing methods, we cannot design the probabilistic connections between these components in any way we choose. We cannot construct an MRF by directly including undirected potentials between the variables, for example. The range of interactions that we can model, and the tools we can bring to bear are significantly limited by this approach.

In this chapter, we take the second approach. We present the framework of Cascaded Classification Models (**CCM**s), where state-of-the-art “black box” classifiers for a set of related tasks are combined to improve performance on some or all tasks. Specifically, the **CCM** framework creates multiple instantiations of each classifier, and organizes them into tiers, where models in the first tier learn in isolation, processing the data to produce the best classifications given only the raw instance features. Subsequent tiers accept as input both the features from the data instance and features computed from the output classifications of the models at the previous tier. While only demonstrated in the computer vision domain, we expect the **CCM** framework have broad applicability to many applications in machine learning.

We begin this chapter with a motivation for integrating the sub-tasks of scene understanding. We show various examples where contextual signals should intuitively provide improved results, and we illustrate the types of contextual signals that might provide that improvement. Following this, we describe the **CCM** approach in a general setting of multiple classifiers for different variable sets in the same domain.

With our generic **CCM** framework in hand, we then apply our model to specific problem of scene understanding by combining scene categorization, object detection, multi-class segmentation, and 3d reconstruction. Through the example sub-tasks used, we demonstrate how “black-box” classifiers can be easily integrated into our framework. We empirically validate our approach through extensive experiments on large databases, which show that our combined model yields superior results on each of the tasks considered. We conclude with a discussion of our relationship to existing work in the field, and of the implications and conclusions to be drawn from this work.

4.1 Motivation for Integration: Detectors on Their Own

Recall from the background on object detectors in Chapter 3 that such methods tend to look only inside the candidate window to determine whether it contains the object in question. In this chapter, we use the HOG detector of Dalal and Triggs [32] as our base detection model. In general, across a wide variety of classes and levels of clutter, this detector performs very well, and is considered one of the state-of-the-art methods for object detection. Despite this, it still makes a large number of mistakes in any particular dataset.

To explore the error modes of the detector, we have run the HOG detector for 6 classes (cars, people, motorcycles, boats, sheep, and cows) on a large dataset of images (the DS2 dataset described below), and selected the highest scoring detection in each image. Figure 4.2 shows some of the mistakes made by the base detector. In the top left we see a case where the person detector found legs that could plausibly be part of a person (this is one of the most common mistakes in our dataset). The bottom left image shows a boat detection on the front of the car. In this case, the black stripe below the license plate has a shape very similar to a boat hull, which might explain the mistake.

While most of these mistakes are “understandable,” many are extremely strange to a human observer. Part of the reason for this is that these detections occur very much

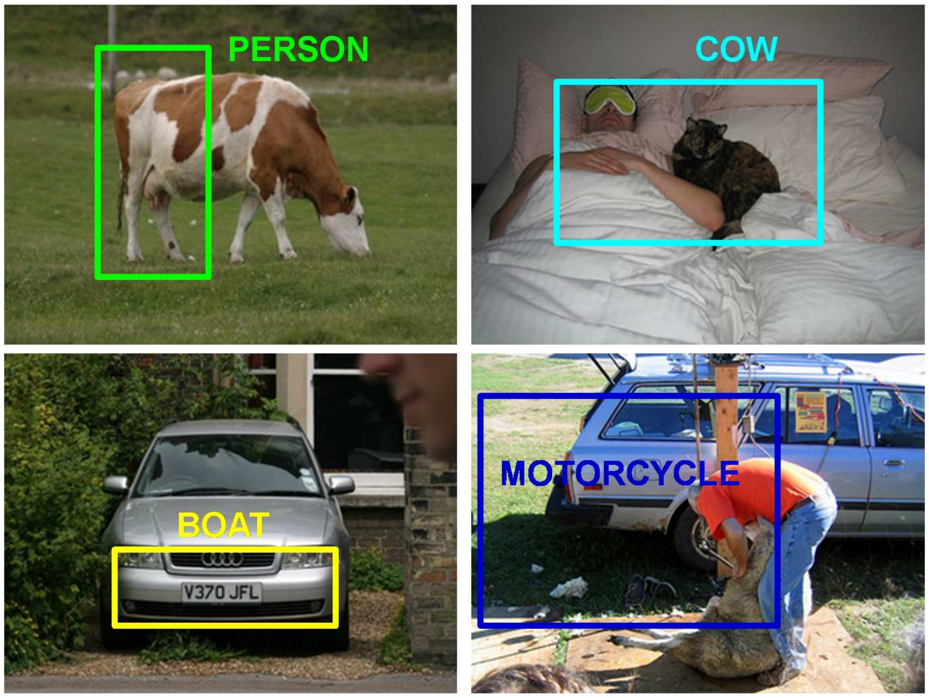


Figure 4.2: Some mistakes made by the base HOG detector.

out of context. Consider the examples in Figure 4.3. In both of these images, the detection boxes are around actual objects, but the objects are categorized incorrectly. This is a common mistake, as foreground objects tend to have coherent edge features that might fire multiple detectors. In both of these cases, however, context can provide a strong cue about the identity of the object. Motorcycles tend to occur on roads rather than fields of grass, and most people do not walk on water. If we could classify the regions surrounding the objects, we can expect to learn a strong contextual signal that will improve our detection results.

Similarly, the examples of Figure 4.4 show some mistakes made by the object detector that are out of context. In this case, however, the contextual mismatch comes from a problem in the scales of the objects. While people do walk on roads, in the first case the bounding box is far too small to contain a person in this image, and in the second case it is too large. If we had some estimate of the scale of the scene, we could expect to learn that neither of these are feasible people. Indeed, the work of

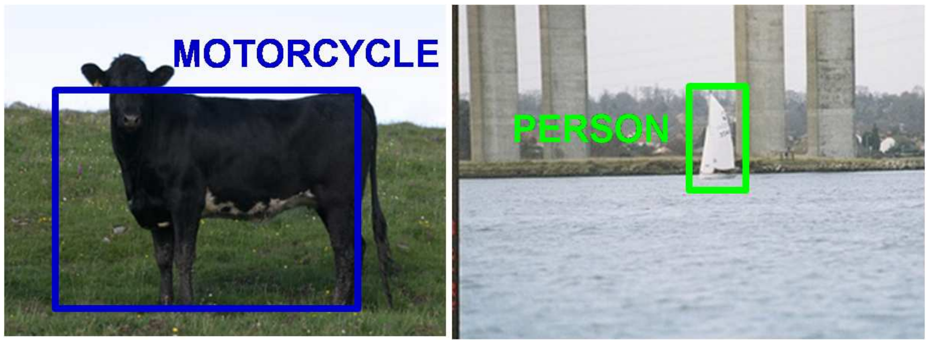


Figure 4.3: Cases where the HOG detector identifies the object incorrectly.

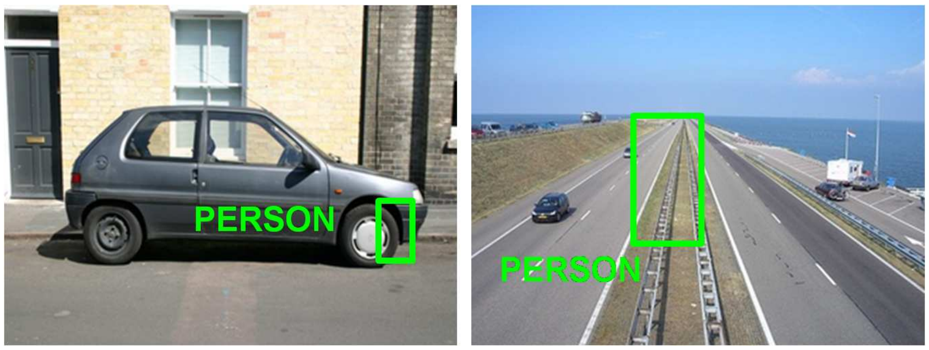


Figure 4.4: Cases where the HOG detector detects an object at a scale that is impossible, given the context.

Hoiem et al. [70] suggests that this type of context can produce a large increase in the effectiveness of object detectors. This analysis confirms our intuitions that combining object detectors with classifiers for other components of the scene provides a good opportunity for improving our detectors.

While these examples only consider the mistakes made by the detector, it seems natural that similar problems will arise with the other tasks. Figure 4.5 shows one example mistake made by the multi-class segmentation model. In this case, the region at the bottom of the image is labeled as road, even though the object in the image is a sail boat. If we knew that this was a boat, that should provide a strong signal that the region at the bottom should be water instead.

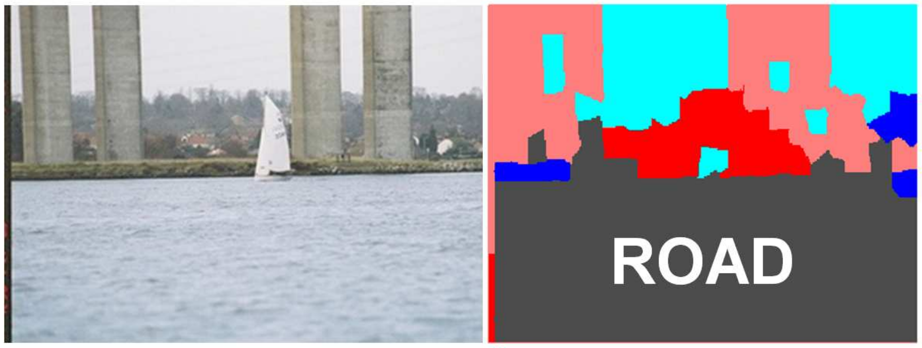


Figure 4.5: Segmentation mistake that can be corrected by the contextual cue from the boat.

With this motivation, we now turn to the question of how to combine these components so that the contextual cues can be shared between them.

4.2 Cascaded Classification Models

Our goal is to classify various characteristics of our data using state-of-the-art methods in a way that allows the each model to benefit from the others' expertise. We are interested in using proven "off-the-shelf" classifiers for each sub-task. As such these classifiers will be treated as "black boxes," each with its own (specialized) data structures, feature sets, and inference and training algorithms.

To fit into our framework, we only require that each classifier provides a mechanism for including additional (auxiliary) features from other modules. Many state-of-the-art models lend themselves to the easy addition of new features. In the case of "intrinsic images" [6], the output of each component is converted into an image-sized feature map (e.g., each "pixel" contains the probability that it belongs to a car). These maps can easily be fed into the other components as additional image channels. In cases where this cannot be done, it is trivial to convert the original classifier's output to a log-odds ratio and use it along with features from their other classifiers in a simple logistic model.

A standard setup has, say, two separate models that predict the variables $\mathbf{Y}_D$ and

$\mathbf{Y}_S$, respectively, for the same input instance $\mathcal{I}$. For example, $\mathcal{I}$ might be an image, $\mathbf{Y}_D$ could be the locations of all cars in the image, and $\mathbf{Y}_S$ could be a probability map indicating which pixels are likely to be road. Most algorithms begin by processing $\mathcal{I}$ to produce a set of features, and then learn a function that maps these features into a predicted label (and in some cases also a confidence estimate). Cascaded Classification Models (**CCM**s) is a joint classification model that shares information between tasks by linking component classifiers in order to leverage their relatedness. Formally:

Definition 4.2.1: An *L-tier* Cascaded Classification Model (L-**CCM**) is a cascade of classifiers of the target labels $\mathcal{Y} = \{\mathbf{Y}_1, \ldots, \mathbf{Y}_K\}^L$ (L "copies" of each label) consisting of **independent** classifiers $f_{k,0}(\phi_k(\mathcal{I}); \theta_{k,0}) \rightarrow \hat{\mathbf{Y}}_k^0$ and a series of **conditional** classifiers $f_{k,\ell}(\phi_k(\mathcal{I}, \mathbf{y}_{-k}^{\ell-1}); \theta_{c,\ell}) \rightarrow \hat{\mathbf{Y}}_k^{\ell}$, indexed by ℓ, indicating the "tier" of the model, where $\mathbf{y}_{-k}$ indicates the assignment to all labels *other than* $\mathbf{y}_k$. The labels at the final tier $(L-1)$ represent the final classification outputs. ∎

A **CCM** uses L copies of each component model, stacked into tiers, as depicted in Figure 4.6. One copy of each model lies in the first tier, and learns with only the image features, $\phi_k(\mathcal{I})$, as input. Subsequent tiers of models accepts a feature vector, $\phi_k(\mathcal{I}, \mathbf{y}_{-k}^{\ell-1})$, containing the original image features and additional features computed from the outputs of models in the preceding tier. Given a novel test instance, classification is performed by predicting the most likely (MAP) assignment to each of the variables in the final tier.

We learn our **CCM** in a feed-forward manner. That is, we begin from the top level, training the independent $(f_{k,0})$ classifiers first, in order to maximize the classification performance on the training data. Because we assume a learning interface into each model, we simply supply the subset of data that has ground labels for that model to its learning function. For learning each component k in each subsequent level ℓ of the **CCM**, we first perform classification using the $(\ell-1)$-tier **CCM** that has already been trained. From these output assignments, each classifier can compute a new set of features and perform learning using the algorithm of choice for that classifier.

For learning a **CCM**, we assume that we have a dataset of fully or partially

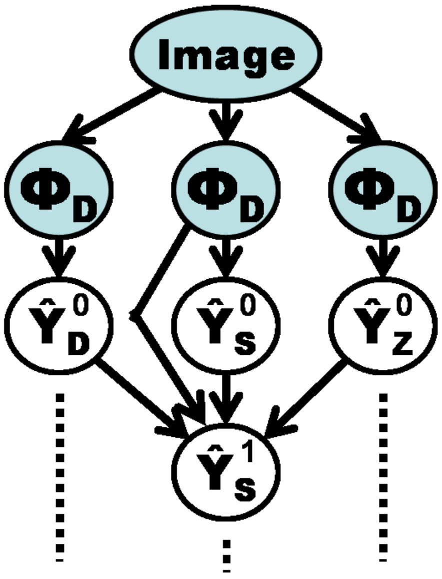


Figure 4.6: The **CCM** framework for jointly predicting each of these label types.

annotated instances. It is not necessary for every instance to have groundtruth labels for every component, and our method works even when the training sets are disjoint. This is appealing since the prevalence of large, volunteer-annotated datasets (e.g., the LabelMe dataset [109] in vision or the Penn Treebank [88] in language processing), is likely to provide large amounts of heterogeneously labeled data.

4.3 CCM for Holistic Scene Understanding

We now return to the specific problem of scene understanding. Our scene understanding model uses a **CCM** to combine various subsets of four computer vision tasks: scene categorization, multi-class image segmentation, object detection, and 3d reconstruction. We first introduce the notation for the target labels and then briefly describe the specifics of each component. Consider an image $\mathcal{I}$. Our scene categorization classifier produces a scene label C from one of a small number of classes. Our multi-class segmentation model produces a class label S_j for each of a predefined set of regions j in the image. The base object detectors produce a set of scored windows

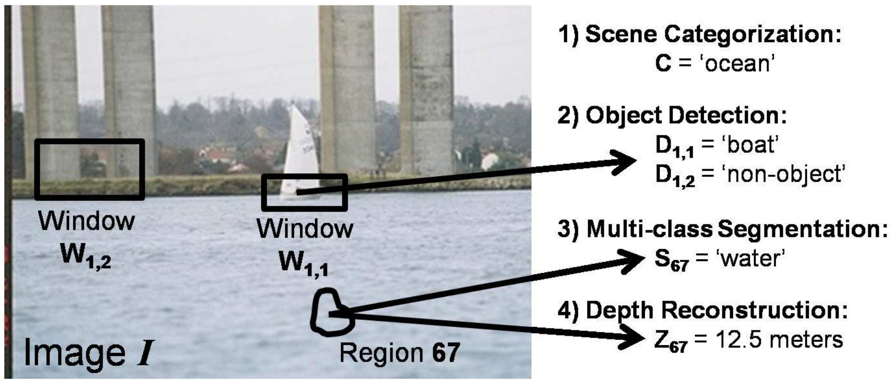


Figure 4.7: The setup for the scene understanding problem, with the variables that we wish to predict.

($W_{c,i}$) that potentially contain an object of type c. We attach a label $D_{c,i}$ to each window, that indicates whether or not the window is predicted to contain the object. Our last component module is monocular 3D reconstruction, which produces a depth Z_i for every pixel i in the image. Figure 4.7 illustrates the problem setup, with the various variables that we wish to predict.

4.3.1 Scene Categorization

Our scene categorization module is a simple multi-class logistic model (defined in Eq. (2.7)) that classifies the entire scene into one of a small number of classes. While many sophisticated image categorization models have been developed, the work of Torralba and Oliva [128] indicates that robust statistical models applied to simple low-level image statistics may be sufficient, "taking advantage of the regularities found in the statistical distribution of features per scene category" described in their paper [128]. The specific features used differ from this paper, but the concepts are the same.

With this motivation, we develop a base model over a 13 dimensional feature vector $\phi(\mathcal{I})$ with elements based on mean and variance of RGB and YCrCb color channels over the entire image, plus a bias term. Table 4.1(left column) lists the features for the independent model. In the context-aware (cascaded) model, we include features that indicate the relative proportions of each region label (a histogram of S_j values) in the

Independent Features	Context-Aware Features
1 (bias term)	% of tree pixels in I
Mean red (R) value	% of road pixels in I
Std. dev. red (R) value	% of grass pixels in I
Mean green (G) value	% of water pixels in I
Std. dev. green (G) value	% of sky pixels in I
Mean blue (B) value	% of building pixels in I
Std. dev. blue (B) value	% of foreground pixels in I
Mean luminance (Y) value	# of cars detected in I
Std. dev. luminance (Y) value	# of people detected in I
Mean blue chrominance (Cb) value	# of motorcycles detected in I
Std. dev. blue chrominance (Cb) value	# of boats detected in I
Mean red chrominance (Cr) value	# of sheep detected in I
Std. dev. red chrominance (Cr) value	# of cows detected in I

Table 4.1: Features used by the scene categorization model.

image, obtained from the output variables of the region labeling model, plus counts of the number of objects of each type detected, obtained from the object detection model. This produces a final feature vector of length 26, including the independent features. Table 4.1(right column) lists the features for the context-aware categorizer.

4.3.2 Image Segmentation

The segmentation module aims to assign a label to each pixel. We base our model on the work of Gould et al. [58] who make use of relative location—the preference for classes to be arranged in a consistent configuration with respect to one another (e.g., cars are often found above roads). Each image is pre-partitioned into a set $\{S_1, \ldots, S_N\}$ of superpixel regions and the pixels are labeled by assigning a class to each region S_j. The method employs a pairwise conditional Markov random field (CRF) constructed over the superpixels with node potentials based on appearance features and edge potentials encoding a preference for smoothness.

In this model, each superpixel S_i has a feature vector $\mathbf{f}_i$, which contains 213 features reminiscent of the features of [5]. The features are listed in Table 4.2. The

Singleton Region Features	# of Features
(x, y) location of region centroid	2
r^2 squared radius from center of image	1
area, perimeter, 3 moments of inertia	5
ratio of perimeter to area	1
mean, std. dev., skewness, kurtosis of *RGB*, *Lab*, *YCbCr* values	36
mean, std. dev., skewness, kurtosis of 42 filter responses	168

Table 4.2: Singleton features used by the Multi-class image segmentation model.

CRF pairwise potentials between neighboring regions bias the labels to be more likely to be the same than different.

In the context-aware model, we wish to model the relative location between detected objects and region labels. We accomplish this with the use of *relative location maps*, which are defined as:

Definition 4.3.1: Relative Location Map: For a particular detected object class d and a particular region label r, a relative location map $M^{d,r}$ is a matrix of probabilities, indexed by pixel offsets in $[-W/2, W/2]$, where W is the width of the map. Suppose that we have detected an object of type d, and we let (i_c, j_c) be the center pixel of the detection, while s is the “scale” at which the detection occurs. The entry $M^{d,r}_{ij}$ gives the probability that the pixel at $(i_c + si, j_c + sj)$ is a pixel from a region of label r. ▮

Intuitively, these maps represent an affinity for certain pixels to be assigned certain region labels, conditioned on the locations of the objects in the image. Figure 4.8 shows the relative location maps (visualized as heat map images) learned by our model for the 7 region labels and 6 object classes considered in experiments below. Because the detection model provides probabilities for each detection, we actually use the relative location maps multiplied by the probability that each detection is a true detection. Preliminary results showed an improvement in using these soft detections over hard (thresholded) detections. The scale s is the width of the (fixed-aspect-ratio) detection window. Relative location maps are stored using a normalized width of 100 pixels and are scaled separately for each detection. This encoding of scale was not

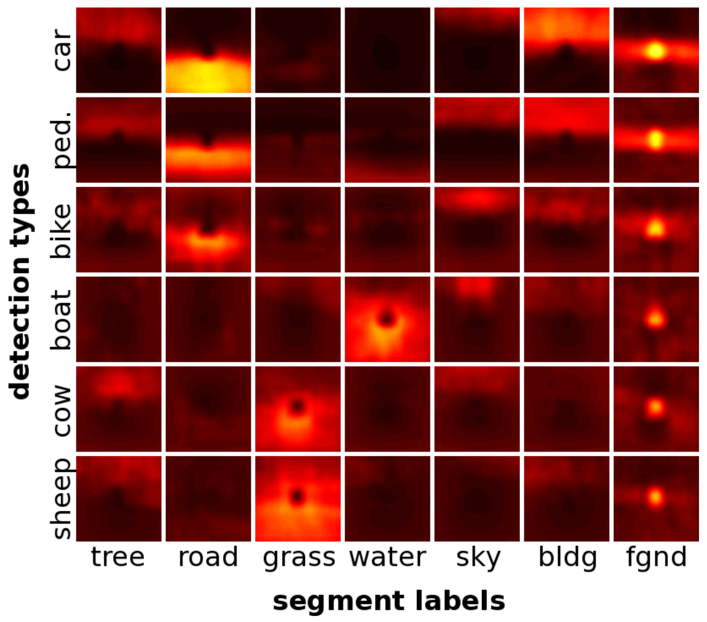


Figure 4.8: Relative location maps showing the relative location of regions (columns) to objects (rows). Each map shows the prevalence of the region relative to the center of object. For example, the top row shows that cars are likely to have road beneath and sky above, while the bottom rows show that cows and sheep are often surrounded by grass.

possible in [58].

For each detectable object type and region label, we average the scaled and weighted relative location maps to produce a relative location map for the entire image. Following this, we construct new singleton features for each region by taking the mean and standard deviation of this map across pixels in the region. This produces two features for each of six object types and seven region labels, for an additional 84 singleton features.

4.3.3 Object Detectors

Our detection module builds on the HOG detector of Dalal and Triggs [32]. For each class, the HOG detector is trained on a set of images disjoint from our datasets below. This detector is then applied to all images in our dataset with a low threshold that

Independent Features	Context-Aware Features
1 (bias term)	% of each region type in, above, and below window ($3 \times 7 = 21$ features)
HOG detector score	% of each region type in same row, column, image as window (21)
$\tilde{x} = \frac{x}{W} - \frac{1}{2}$	Scene type marginals (4)
$\tilde{y} = \frac{y}{H} - \frac{1}{2}$	Indicator vector of MAP scene type (4)
$\tilde{w} = \frac{w}{W}$	Indicator of whether this window was a detection at the previous tier (1)
$\tilde{h} = \frac{h}{H}$	% overlap with closest previous positive detection (1)
$\tilde{x}^2$, $\tilde{y}^2$, $\tilde{x}\tilde{y}$, $\tilde{w}^2$, $\tilde{h}^2$	Indicator of region label one bounding box diagonal away at 8 different angles (56)
Indicator of window hanging off image	Indicator of most likely object one bounding box diagonal away at 8 different angles (48)

Table 4.3: Features used by the object detection model. (left) Independent model features for detection window $\mathbf{W}$ at image location (x, y) of size (w, h) in an image of size $W \times H$.

produces an overdetection. For each image $\mathcal{I}$, and each object class c, we typically find 10-100 candidate detection windows $W_{c,i}$. Our independent detector model learns a logistic model over a small feature vector $\phi_{c,i}$ that can be extracted directly from the candidate window. The left column of Table 4.3 describes the set of features used in the independent logistic regression model.

Our context-aware conditional classifier seeks to improve the accuracy of the HOG detector by using image segmentation (denoted by S_j for each region j), 3D reconstruction of the scene, with depths (Z_j) for each region, and a categorization of the scene as a whole (C), to improve the results of the HOG detector. Thus, the output from other modules and the image are combined into a feature vector $\phi_k(\mathcal{I}, C, \mathbf{S}, \mathbf{Z})$. A visualization of some of the features used are shown in Figure 4.9, and the specific features used are described in the right column of Table 4.3. This augmented feature vector is used in a logistic model as in the independent case. Both the independent and context aware logistics are regularized with a small ridge term to prevent overfitting. Specific values used are given in experiments below.

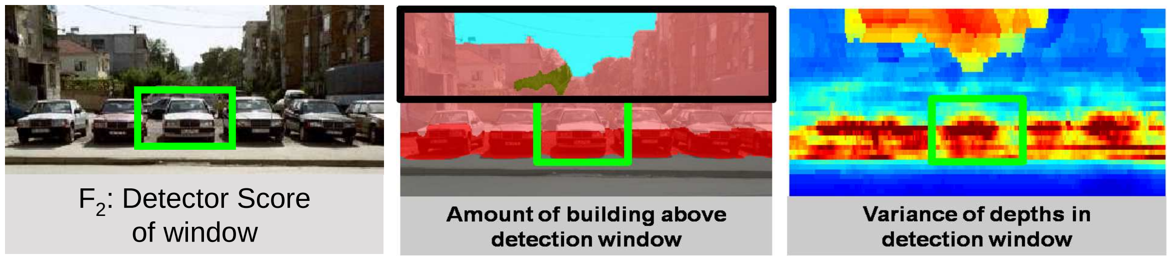

F_2: Detector Score of window
Amount of building above detection window
Variance of depths in detection window

Figure 4.9: Three example features used by the “context” aware object detector.

4.3.4 3D Reconstruction Module

Our reconstruction module is based on the work of Saxena et al. [111]. Our Markov Random Field (MRF) approach models the 3D reconstruction (i.e., depths Z at each point in the image) as a function of the image features and also models the relations between depths at various points in the image. For example, unless there is occlusion, it is more likely that two nearby regions in the image would have similar depths.

More formally, our variables are continuous, i.e., at a point i, the depth $Z_i \in \Re$. Our baseline model consists of two types of terms. The first terms model the depth at each point as a linear function of the local image features, and the second type models relationships between neighboring points, encouraging smoothness. Our conditional model includes an additional set of terms that models the depth at each point as a function of the features computed from an image segmentation $\mathbf{S}$ in the neighborhood of a point. By including this third term, our model benefits from the segmentation outputs in various ways. For example, a classification of grass implies a horizontal surface, and a classification of sky correlates with distant image points. While detection outputs might also help reconstruction, we found that most of the signal was present in the segmentation maps, and therefore dropped the detection features for simplicity.

4.4 Experiments

We perform experiments on two subsets of images. The first subset **DS1** contains 422 fully-labeled images of urban and rural outdoor scenes. Each image was assigned a category in (*urban*, *rural*, *water*, *other*) by a volunteer not associated with the project. We hand label each pixel as belonging to one of: *tree*, *road*, *grass*, *water*, *sky*, *building* and *foreground*. The foreground class captures detectable objects, and a *void* class (not used during training or evaluation) allows for the small number of regions not fitting into one of these classes (e.g., mountain) to be ignored. This is standard practice for the pixel-labeling task (e.g., see [29]). We also annotate the location of six different object categories (*car*, *pedestrian*, *motorcycle*, *boat*, *sheep*, and *cow*) by drawing a tight bounding box around each object. We use this dataset to demonstrate the combining of three vision tasks: object detection, multi-class segmentation, and scene categorization using the models described above.

Our much larger second dataset **DS2** was assembled by combining 362 images from the **DS1** dataset (including either the segmentation or detection labels, but not both), 296 images from the Microsoft Research Segmentation dataset [29] (labeled with segments), 557 images from the PASCAL VOC 2005 and 2006 challenges [40] (labeled with objects), and 534 images with ground truth depth information. After removing duplicates, this results in 1745 images with disjoint labelings (no image contains groundtruth labels for more than one task). Combining these datasets results in 534 reconstruction images with groundtruth depths obtained by laser range-finder (split into 400 training and 134 test), 596 images with groundtruth detections (same 6 classes as above, split into 297 train and 299 test), and 615 with groundtruth segmentations (300 train and 315 test). This dataset demonstrates the typical situation in learning related tasks whereby it is difficult to obtain large fully-labeled datasets. We use this dataset to demonstrate the power of our method in leveraging the data from these three tasks to improve performance.

	Car	Pedes.	Bike	Boat	Sheep	Cow	**Mean**
HOG	0.39	0.29	0.13	0.11	0.19	0.28	0.23
Independent	0.55	0.53	0.57	0.31	0.39	0.49	0.47
2-CCM	0.58	0.55	**0.65**	**0.48**	**0.45**	0.53	**0.54**
5-CCM	**0.59**	**0.56**	0.63	0.47	0.40	**0.54**	0.53
Ground	0.49	0.53	0.62	0.35	0.40	0.51	0.48
Ideal Input	0.63	0.64	0.56	0.65	0.45	0.56	0.58

Table 4.4: Numerical evaluation of our detection performance for various training regimes for the **DS1** dataset. We show average precision (AP) for the six classes, as well as the mean.

4.4.1 DS1 Dataset

Experiments with the **DS1** dataset were performed using 5-fold cross validation, and we report the mean performance results across folds. We compare five training/testing regimes. **Independent** learns parameters on a 1-tier (independent) **CCM**, where no information is exchanged between tasks. We compare two levels of complexity for our method, a **2-CCM** and a **5-CCM** to test how the depth of the cascade affects performance. The last two training/testing regimes involve using groundtruth information at every stage for training and for both training and testing, respectively. **Groundtruth** trains a 5-**CCM** using groundtruth inputs for the feature construction (i.e., as if each tier received perfect inputs from above), but is evaluated with real inputs. The **Ideal Input** experiment uses the **Groundtruth** model and also uses the groundtruth input to each tier *at testing time*. We could do this because, for this dataset, we had access to fully labeled groundtruth. Obviously this is not a legitimate operating mode, but does provide an interesting upper bound on what we might hope to achieve with this approach to combining classifiers.

To quantitatively evaluate our method, we consider metrics appropriate to the tasks in question. For scene categorization, we report an overall accuracy for assigning the correct scene label to an image. For segmentation, we compute a per-pixel accuracy, where each pixel in the region is assigned the label inferred for the segment. Following standard procedure, we ignore all pixels with groundtruth label 'void'. For detection, we consider a particular detection correct if the overlap score is larger

	Segment	**Category**
Independent	79.5%	70.6%
2-CCM	82.2%	**77.3%**
5-CCM	**83.3%**	76.8%
Ground	80.7%	69.9%
Ideal Input	85.9%	86.7%

Table 4.5: Evaluation of segmentation and scene categorization accuracy for various training regimes for the **DS1** dataset.

than 0.2 (overlap score equals the area of intersection divided by the area of union between the detected bounding box and the groundtruth). We plot precision-recall (PR) curves for detections, and report the average precision of these curves.

We present the detection results in Table 4.4 and the segmentation and categorization results in Table 4.5. The best performing method for each column is bolded. The corresponding graphs are given in Figure 4.10 and Figure 4.11. The PR curves compare the HOG detector results to our **Independent** results and to our **2-CCM** results. It is interesting to note that a large gain was achieved by adding the independent features to the object detectors. While the HOG score looks at only the pixels inside the target window, the other features take into account the size and location of the window, allowing our model to capture the fact that foreground objects tend to occur in the middle of the image and at a relatively small range of scales. On top of this, we were able to gain an additional benefit through the use of context in the **CCM** framework. For the categorization task, we gained 7% using the **CCM** framework, and for segmentation, **CCM** afforded a 3% improvement in accuracy. Furthermore, for this task, running an additional three tiers, for a 5-**CCM**, produced an additional 1% improvement.

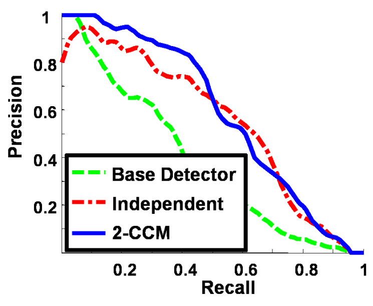
Interestingly, the **Groundtruth** method performs little better than **Independent** for these three tasks. This shows that it is better to train the models using input features that are closer to the features it will see at test time. In this way, the downstream tiers can learn to ignore signals that the upstream tiers are bad at capturing, or even take advantage of consistent upstream bias. Also, the **Ideal Input** results show that **CCMs** have made significant progress towards the best we can hope for from these models.

In addition to this quantitative analysis, we can get a qualitative feel for the results from looking at a few examples. For this chapter, the legend of labels for the detection and segmentation tasks is shown in Figure 4.12. Objects are shown with boxes in the appropriate color, and segmentation labels are shown as color maps of the region label.

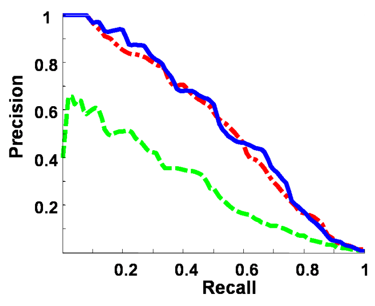
In Figure 4.13, we show an example image where the scene category label changed between the independent and **CCM** model. The image was initially categorized as an ‘urban’ scene, because the grass in this image is similar in color to the road regions across the dataset. In addition, the large amount of white sky is more indicative of urban scenes in this dataset. The later tiers of categorization models, however, got to see that these grass regions were indeed labeled as ‘grass’, and that many of the top detections in this image were ‘cows’ (it has mistaken the sheep for cows, a very common mistake). Based on this information, the most likely category was changed to ‘rural.’

In other images, the detections are improved, particularly when the segmentation result is considered. In Figure 4.14, the texture of the concrete produces a pattern that looks like a sheep (and is in approximately the correct location and scale in the image for sheep). This results in the top independent detection being the sheep. However, the concrete slab and people are identified as foreground objects, and the water is (mostly) correctly labeled. This results in a change of the top detection to a person, which fits better with this image layout.

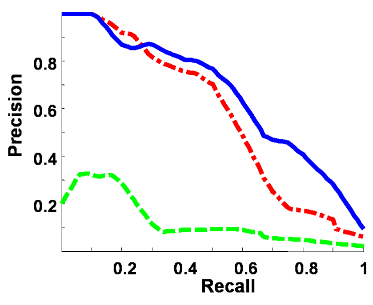
The final component, the multi-class segmentation, also benefits from looking at the detection results. In Figure 4.15, because of the clutter in the middle of the image, the independent segmentation model assigns the label ‘foreground’ to many of the building regions. However, once the car is detected, along with the person, the context-aware segmentation model determines the labels for these superpixels correctly.



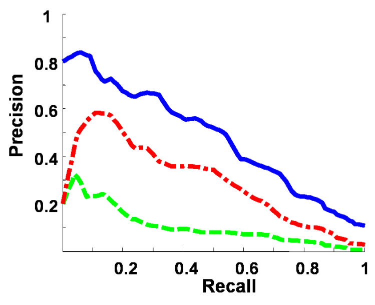
(a) Cars



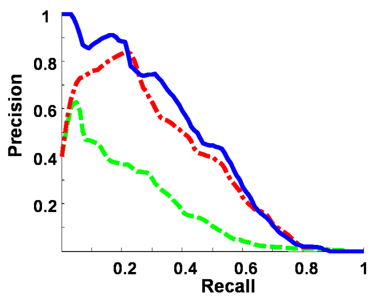
(b) Pedestrians



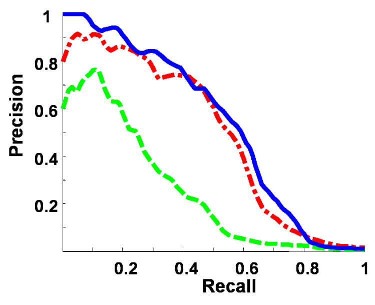
(c) Motorcycles



(d) Boats

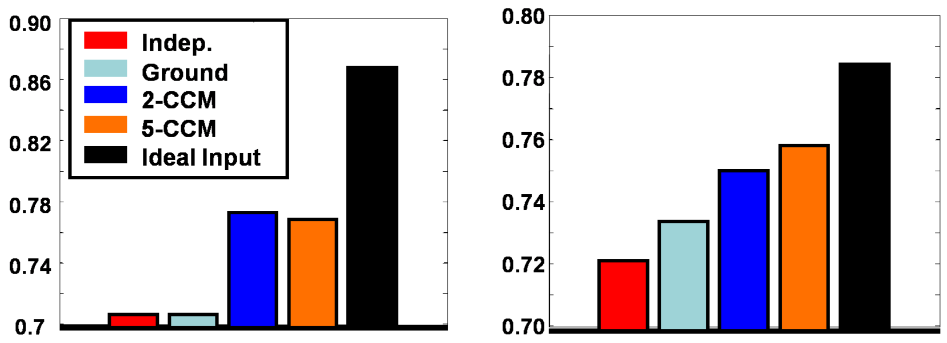


(e) Sheep



(f) Cows

Figure 4.10: Detection results for the DS1 dataset. We show precision-recall curves for the six object classes that we consider.



(a) Categorization (b) Segmentation

Figure 4.11: Segmentation and categorization results for the DS1 dataset. (a) shows our accuracy on the scene categorization task and (b) our accuracy in labeling regions in one of seven classes.

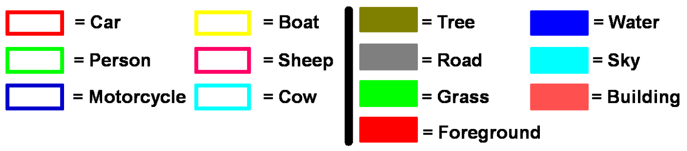


Figure 4.12: Legend of object and region labels for this chapter.

4.4.2 DS2 Dataset

For the second dataset (DS2) we combine the three sub-tasks of reconstruction, segmentation, and object detection. Furthermore, as described above, the labels for our training data are disjoint. We trained an **Independent** model and a 2-**CCM** on this data. Quantitatively, 2-**CCM** outperformed **Independent** on segmentation by 2% (75% vs. 73% accuracy), on detection by 0.02 (0.33 vs. 0.31 mean average precision), and on depth reconstruction by 1.3 meters (15.4 vs. 16.7 root mean squared error). Table 4.6 shows the detection results per class and Table 4.7 shows the per-class accuracy of the multi-class segmentation.

Figure 4.16 and Figure 4.17 show example outputs from each component. Figure 4.16 shows images where all components improved over the independent model. In the top left our detectors removed some false boat detections which were out of

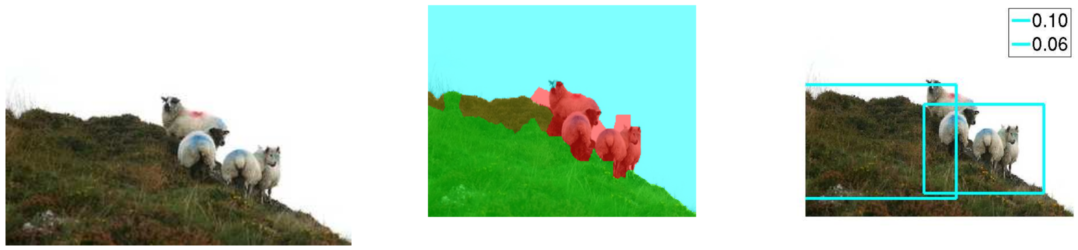
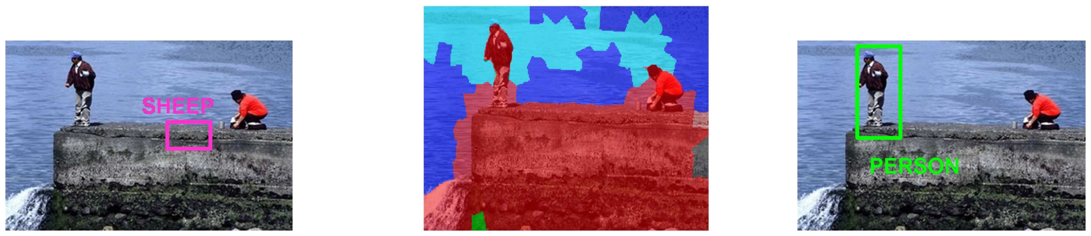


Figure 4.13: Example image from the DS1 dataset, where the independent categorization model predicted ‘urban’, while the context-aware categorization determined that it should be ‘rural’, based on the region labels and objects found.

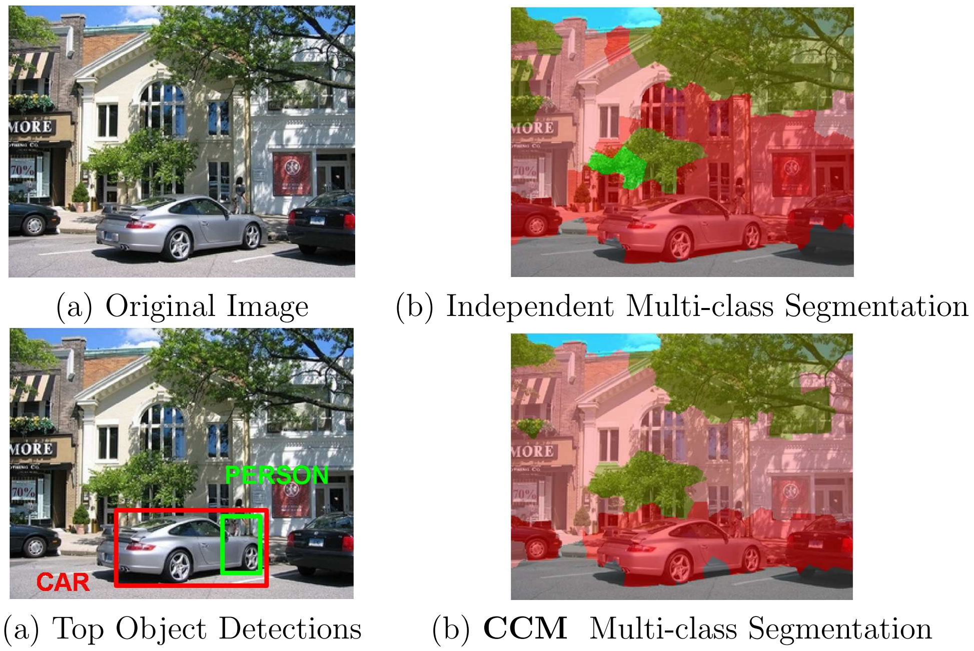


(a) Independent Detections (b) Multi-class Segmentation (c) **CCM** detections

Figure 4.14: Example image from the DS1 dataset, where the original top detection (a) is a sheep, but after considering the segmentation (b), resulting a correct final detection (c) of the person.

context and determined that the watery appearance of the bottom of the car was actually foreground. Also by providing a sky segment, our method allowed the 3D reconstruction model to infer that those pixels must be very distant (red). The next two examples show similar improvement for detections of boats and water.

The remaining examples of Figure 4.17 show how separate tasks improve by using information from the others. In each example we show results from the independent model for the task in question, the independent contextual task and the 2-**CCM** output. The first four examples show that our method was able to make correct detections whereas the independent model could not. The last examples show improvements in multi-class image segmentation.



(a) Original Image (b) Independent Multi-class Segmentation

(a) Top Object Detections (b) **CCM** Multi-class Segmentation

Figure 4.15: Example image from the DS1 dataset (a), where the independent multi-class segmentation (b) puts too much 'foreground' in the middle of the image (where the building is). Once it sees the detections (c), however, the final segmentation correctly identifies the building and the trees above the car.

4.5 Related Work

A number of works in various fields aim to combine classifiers to improve final output accuracy. These works can be divided into two broad groups. The first is the combination of classifiers that predict the *same* set of random variables. Here the aim is to improved classifications by combining the outputs of the individual models. Boosting [112], in which many weak learners are combined into a highly accurate classifier, is one of the most common and powerful such schemes. In computer vision, this idea has been very successfully applied to the task of face detection using the so-called Cascade of Boosted Ensembles (CoBE) [131, 18] framework. While similar to our work in constructing a cascade of classifiers, their motivation was computational efficiency, rather than a consideration of contextual benefits. Tu [129] learns context cues by cascading models for pixel-level labeling. However, the context is,

	CAR	PEDES.	BIKE	BOAT	SHEEP	COW	**Mean**
HOG	0.151	0.185	0.068	0.164	0.198	0.200	0.161
Independent	0.357	0.267	**0.410**	0.096	**0.319**	0.395	0.307
2-CCM	**0.364**	**0.272**	**0.410**	**0.412**	0.289	**0.415**	**0.326**

Table 4.6: Numerical evaluation of our detection performance for various training regimes for the **DS2** dataset. We show average precision (AP) for the six classes, as well as the mean.

	Tree	**Road**	**Grass**	**Water**	**Sky**	**Building**	**Foreground**
Independent	0.529	**0.709**	0.860	0.440	0.928	0.423	**0.790**
2-CCM	**0.572**	0.695	**0.865**	**0.558**	**0.933**	**0.472**	0.772

Table 4.7: Evaluation of segmentation accuracy for the various classes in the **DS2** dataset.

again, limited to interactions between labels of the same type. The idea of "stacked generalization" from Wolpert [138] also cascaded classifiers, in order to perform something like cross validation. They introduce a first level of classifiers that might all be different (e.g., one is a SVM, one is a neural network, etc.), and use a second layer of classifiers with a different training set (the validation set) to classify based on the outputs of the first layer classifiers. While the classifiers in this system are all predicting the same variables, this idea of looking at the data in different ways is somewhat similar to our approach. Also in this vein is the work on Hierarchical Mixtures of Experts (HME) by Jordan and Jacobs [72]. In this framework, a tree of classifiers is learned, where each classifier uses a different subset of the training data and the classification outputs of its children.

The other broad group of works that combine classifiers is aimed at using the classifiers as components in large intelligent systems. Kumar and Hebert [76], for example, develop a large MRF-based probabilistic model linking multiclass segmentation and object detection. Such approaches have also been used in the natural language processing literature. For example, the work of Sutton and McCallum [122] combines a parsing model with a semantic role labeling model into a unified probabilistic framework that solves both simultaneously. While technically-correct

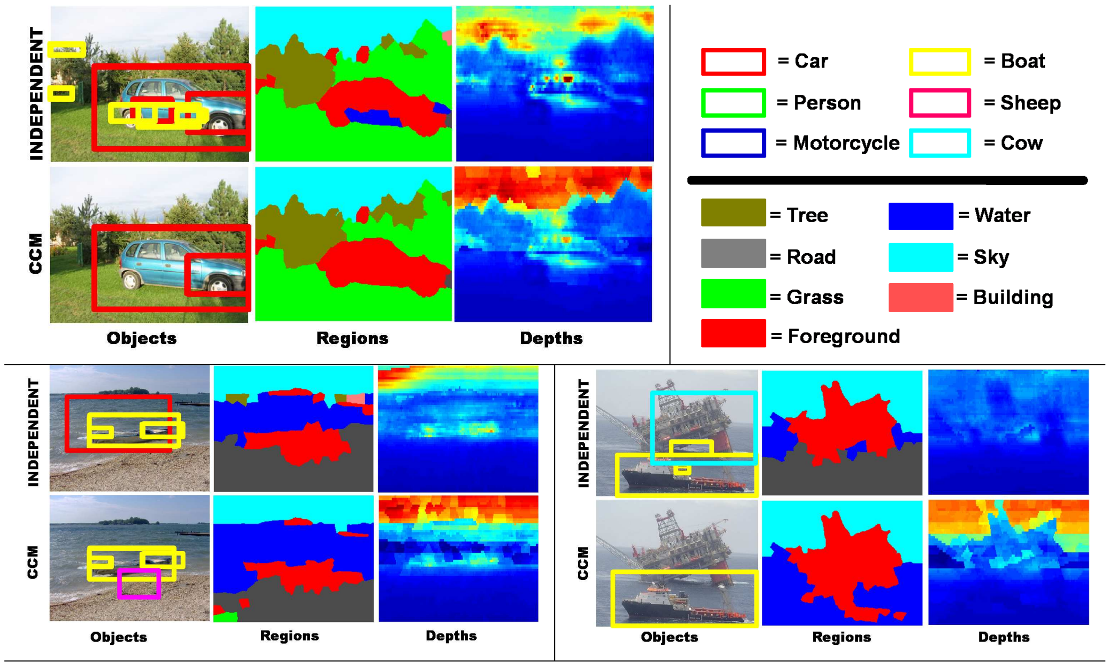


Figure 4.16: Three cases where **CCM** improved results for all tasks. In the first, for instance, the presence of grass allows the **CCM** to remove the boat detections.

probabilistic representations are appealing, it is often painful to fit existing methods into a large, complex, highly interdependent network. By leveraging the idea of cascades, our method provides a simplified approach that requires minimal tuning of the components.

As discussed in Section 1.3, the goal of holistic scene understanding dates back to the early days of computer vision. Over the last few decades, however, researchers have instead targeted isolated computer vision tasks, with considerable success in improving the state-of-the-art. For example, in our work, we build on the prior work in scene categorization of Li and Perona [83], object detection of Dalal and Triggs [32], multi-class image segmentation of Gould et al. [58], and 3D reconstruction of Saxena et al. [111]. Recently, however, researchers have returned to the question of how one can benefit from exploiting the dependencies between different classifiers.

Torralba [126] use context to significantly boost object detection performance,

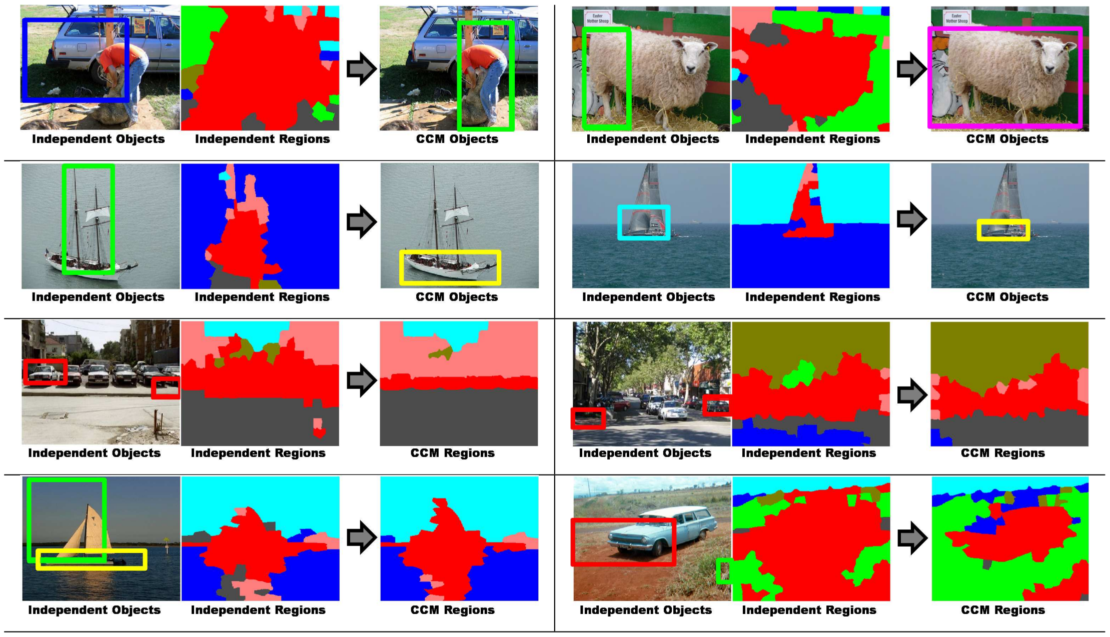


Figure 4.17: The four rows show four examples where detections are improved and four examples where segmentations are improved.

and Sudderth et al. [121] use object recognition for 3D structure estimation. In independent contemporary work, Hoiem et al. [71] propose an innovative system for integrating the tasks of object recognition, surface orientation estimation, and occlusion boundary detection. Like ours, their system is modular and leverages state-of-the-art components. However, their work has a strong leaning towards 3D scene reconstruction rather than understanding, and their algorithms contain many steps that have been specialized for this purpose. Their training also requires intimate knowledge of the implementation of each module, while ours is more flexible allowing integration of many related vision tasks regardless of their implementation details. Furthermore, we consider a broader class of images and object types, and label regions with specific classes, rather than generic properties.

4.6 Discussion

In this chapter, we have presented the Cascaded Classification Models (**CCM**) method for combining a collection of state-of-the-art classifiers toward improving the results of each. We demonstrated our method on the task of holistic scene understanding by combining scene categorization, object detection, multi-class segmentation and depth reconstruction, and improving on all. Our results are consistent with other contemporary research, including the work of Hoiem et al. [71], which uses different components and a smaller number of object classes.

Importantly, our framework is very general and can be applied to a number of machine learning domains. This result provides hope that we can improve by combining our complex models in a simple way. The simplicity of our method is one of its most appealing aspects. Cascades of classifiers have been used extensively within a particular task, and our results suggest that this should generalize to work between tasks. In addition, we showed that **CCM**s can benefit from the cascade even with disjoint training data, e.g., no images containing labels for more than one sub-task.

In our experiments, we passed relatively few features between the tasks. Due to the homogeneity of our data, many of the features carried the same signal (e.g., a high probability of an ocean scene is a surrogate for a large portion of the image containing water regions). For larger, more heterogeneous datasets, including more features may improve performance. In addition, larger datasets will help prevent the overfitting that we experienced when trying to include a large number of features.

The features that are derived from the outputs at one tier and passed to the next can be thought of as “messages” that contain the beliefs of the earlier tier about the variables in question. In this sense, our method is reminiscent of the belief propagation (BP) algorithm described in Chapter 2. Thinking of each type of classifier in our framework as a node in the loopy graph, we can think of a **CCM** as unrolling the BP message passing, with each tier corresponding to one round of (synchronous) updates. The primary difference is that the **CCM**approach uses a *different* set of parameters at each tier. Indeed, we train these parameters to be the best (on the training set) for this tier, given the results of previous tiers.

The other difference is that each of classifiers is trained for their particular task. As a result, the "message" that are being passed are conditioned on the other variables, rather than based on the joint "beliefs" of BP. This aspect of our method recalls the dependency network representation of Heckerman et al. [63]. Dependency networks are an alternative to Bayesian networks, where each variable has as parents all variables that directly probabilistically influence it (not through an intervening variable). Such a network is directed, but not acyclic, and allows for fast inference where each variable conditions only on its parents at inference time. Indeed, this suggests that a more probabilistically principled form of **CCM**s might take the form of a dependency network.

It is an open question how deep a **CCM** is appropriate in a given scenario. Overfitting is anticipated for very deep cascades. Furthermore, because of limits in the context signal, we cannot expect to get unlimited improvements. Unfortunately, we do not have any convergence guarantees, or even a coherent function that we are optimizing at inference time. This even further complicates the question of cascade depth. In practice, we often get improvement from one tier to the next, but this is only an empirical observation. Figure 4.18 graphs the segmentation accuracy and categorization accuracy at each level of the **CCM**. While the segmentation continues to improve at each level, the categorization accuracy actually goes down at the last tier. Further exploration of the cascade depth, and determination of cases where such a framework is appropriate are important future directions.

Another exciting avenue is the idea of feeding back information from the later classifiers to the earlier ones. Intuitively, a later classifier might encourage earlier ones to focus its effort on fixing certain error modes, or allow the earlier classifiers to ignore mistakes that do not hurt "downstream." This also should allow components with little training data to optimize their results to be most beneficial to other modules, while worrying less about their own task.

While our focus has been mostly on image understanding, the goal of combining related classifiers is relevant to many other machine learning domains where several related tasks operate on the same (or related) raw data and provide correlated outputs. In the area of natural language processing, for instance, we might want to

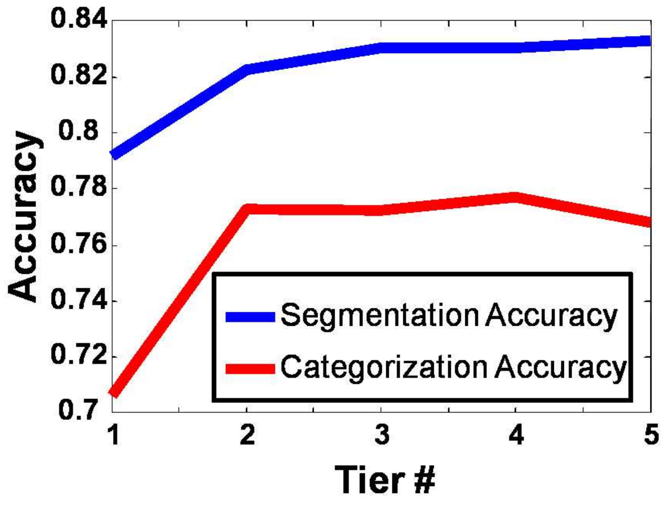


Figure 4.18: How deep should a **CCM** be? The segmentation accuracy continues to improve at each tier, but the categorization accuracy actually gets worse at the last tier.

process a single document and predict the part of speech of all words, correspond the named entities, and label the semantic roles of verbs. In the area of audio signal processing, we might want to simultaneously do speech recognition, source separation, and speaker recognition.

In the introduction, we discussed the possibility of construction a single, large probabilistic model that encompasses all of these tasks. While in this chapter, with the **CCM** approach, we were able to achieve a boost in performance across the tasks, the idea of a unified approach is still attractive. However, because of the difficulties of including arbitrary components into such a model, we might begin with a simpler combination. One of the strongest interactions in this chapter occurred between the labeling of image regions (in the multi-class segmentation), and the identification of the image objects (object detection). In the next chapter, we will develop a unified model that links these two types of primitives through their spatial relationships in the image.

Chapter 5

Learning Spatial Context: Using Stuff to Find Things

Consider again the problem of object detection. Intuitively, recognizing objects in an image requires combining many different signals from the raw image data. Figure 5.1 shows an example satellite image of a street scene, where we may want to identify all of the cars. From a human perspective, there are two primary signals that we leverage. The first is the local appearance in the window near the potential car. The second is our knowledge that cars appear on roads. This second signal is a form of contextual knowledge, and is an example of one of the chief cues that allowed us to achieve improved detection performance in Chapter 4.

In this chapter we will probe this particular contextual signal in greater depth. Rather than employing a simple method that is able to incorporate many computer vision tasks, as in the **CCM** approach of the previous chapter, we will instead construct a probabilistic model that strives to capture the probabilistic relationship between regions and objects. We hope that by sacrificing the ease of design of **CCM**s, we can gain better performance and more importantly, a better understanding of how this relationship works. In addition, based on the discussion in Chapter 1, we also believe that these relationships themselves are important for scene understanding. Indeed, recognizing that a person is walking on a road vs. walking on the grass can help answer certain questions about the scene that require deeper understanding.

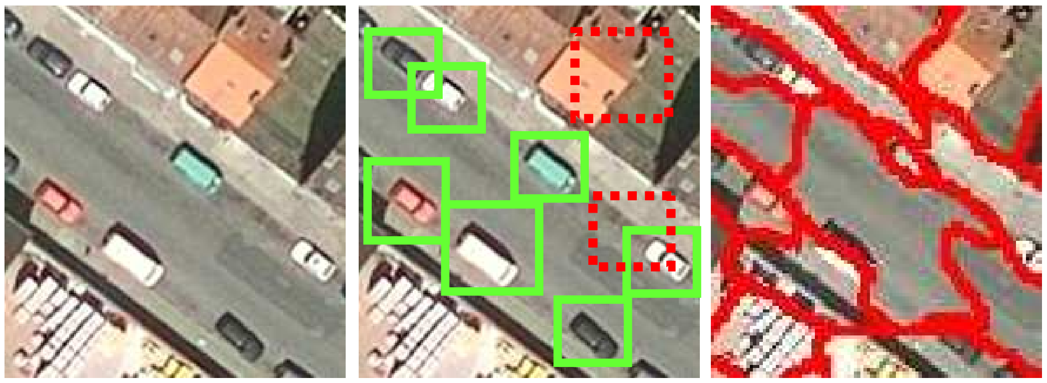


Figure 5.1: (Left) An aerial photograph. (Center) Detected cars in the image (solid green = true detections, dashed red = false detections). (Right) Finding "stuff" such as buildings, by classifying regions, shown delineated by red boundaries.

While **CCM**s were able to learn to use this signal, the relationship was not explicitly encoded anywhere in the model. In this chapter, we seek to remedy that.

Recent papers have demonstrated that boosted object detectors (described in Chapter 3) can be effective at detecting monolithic object classes, such as cars [126] or faces [131]. Similarly, several works on multiclass segmentation (described in Chapter 3) have shown that regions in images can effectively be classified based on their color or texture properties [119]. These two lines of work have made significant progress on the problems of identifying "things" and "stuff," respectively. The important differentiation between these two classes of visual objects is summarized in Forsyth et al. [53] as:

> The distinction between materials — "stuff" — and objects — "things" — is particularly important. A material is defined by a homogeneous or repetitive pattern of fine-scale properties, but has no specific or distinctive spatial extent or shape. An object has a specific size and shape.

In addition to our work in Chapter 4, other recent work has also shown that classifiers for both things or stuff can benefit from the proper use of contextual cues. For the purposes of this discussion, we will split the use of context into a number of categories. *Scene-thing context* allows scene-level information, such as the scale or the "gist" [93], to determine location priors for objects. *Stuff-stuff context* captures the

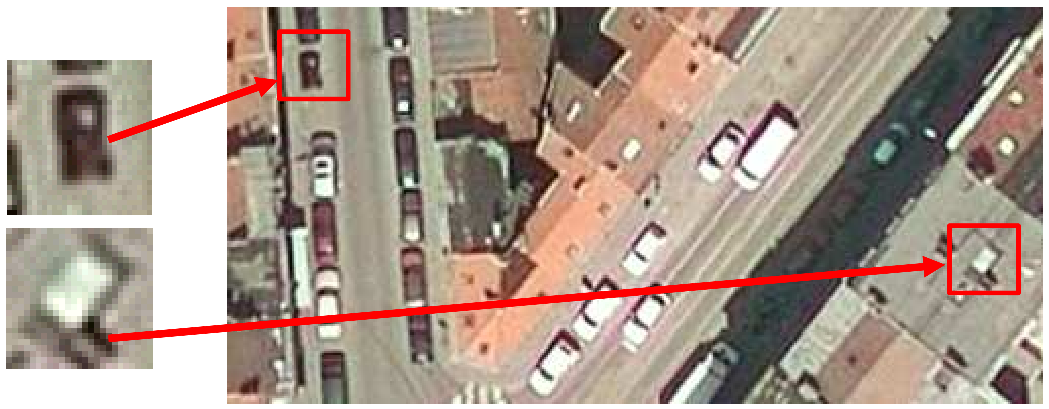


Figure 5.2: Example detections from the satellite dataset that demonstrate context. Classifying using local appearance only, we might think that both windows at left are cars. However, when seen in context, the bottom detection is unlikely to be an actual car.

notion that sky occurs above sea and road below building [120]. *Thing-thing context* considers the cooccurrence of objects, encoding, for example, that a tennis racket is more likely to occur with a tennis ball than with a lemon [103]. Finally *stuff-thing context* allows the texture regions (e.g., the roads and buildings in Figure 5.1) to add predictive power to the detection of objects (e.g., the cars in Figure 5.1). We focus on this fourth type of context. Figure 5.2 shows an example of this context in the case of satellite imagery.

In this chapter, we present a probabilistic model linking the detection of things to the unsupervised classification of stuff. Our method can be viewed as an attempt to cluster "stuff," represented by coherent image regions, into clusters that are both visually similar and best able to provide context for the detectable "things" in the image. Cluster labels for the image regions are probabilistically linked to the detection window labels through the use of region-detection "relationships," which encode their relative spatial locations. For example, a gray, flat-texture region below a candidate window in the image is likely to be a road region, and therefore provides a contextual signal that the window probably does contain a car. On the other hand, a green, heavily-textured region might indicate that a cow or sheep should be found above it.

We call this model the things and stuff (TAS) context model because it links things and stuff into a coherent whole. The graphical representation of the TAS model is depicted in Figure 5.3. At training time, our approach uses supervised (ground-truth) detection labels, without requiring supervised labels for the "stuff"-regions, and learns the model parameters using the Expectation-Maximization (EM) algorithm. At test time, only the relationships are observed, and both the region labels and detection labels are inferred.

In addition to learning and inference with a known set of relationships, we demonstrate a principled method for learning the set of active relationships (those used by the model). In general, the system designer may have some ideas about what types of relationships will provide context, but he or she might also want to consider additional candidate relationships. Furthermore, in cases where we do not know what to expect, it would be nice to "try" a large set of relationships to find what works best. Based on this motivation, we develop a mechanism for selecting good relationships from a large set of candidates. Relationship selection is accomplished through a variant of structural EM [54]. This novel algorithm allows the determination of which types of relationships are most appropriate without costly hand-engineering or cross-validation.

Compared to the **CCM** approach of Chapter 4, the TAS model focuses more directly on modeling and using the spatial relationships between regions and detectable objects. This is evident in the explicit representation of relationships and the isolation of these two vision sub-tasks. The other primary difference is the lack of supervision in the region labels in TAS. In Chapter 4, we assumed that classifiers for specific semantic region classes were available. In this chapter, we do not label the regions, but rather learn the important clusters (labels) from the data. These two shifts in focus allow the TAS model to concentrate on a single type of interaction and to mine the data for the most salient examples of this interaction.

We present results of the TAS method on diverse datasets. Using the Pascal Visual Object Classes challenge datasets from 2005 and 2006, we utilize one of the top competitors as our baseline detector [32], and demonstrate that our TAS method improves the performance of detecting cars, bicycles and motorbikes in street scenes and cows and sheep in rural scenes (see Figure 5.7). In addition, we consider a very different dataset of satellite images from Google Earth, of the city and suburbs of Brussels, Belgium. The task here is to identify the cars from above; see Figure 5.2. For clarity, the satellite data is used as the running example throughout the chapter, but all descriptions apply equally to the other datasets. We finish our experimental evaluation with a comparison between TAS and **CCM**, including a discussion of the relative tradeoffs of the two methods.

5.1 Things and Stuff (TAS) Context Model

Our probabilistic context model builds on two standard components that are commonly used in the literature. The first is sliding window object detection and the second is unsupervised image region clustering. As discussed in Section 3.2, a common method for finding "things" is to slide a window over the image, score each window's match to the model, and return the highest matching such windows. We denote the features in the i^{th} candidate window by W_i, the presence/absence of the target class in that window by T_i (T for "thing"), and assume that what we learn in our detector is a conditional probability $P(T_i \mid W_i)$ from the window features to the probability that the window contains the object; this probability can be derived from most standard classifiers, such as the highly successful boosting approaches [126]. The set of windows included in the model can, in principle, include all windows in the image. We, however, limit ourselves to looking at the top scoring detection windows according to the detector (i.e., all windows above some low threshold, chosen in order to capture most of the true positives).

The second component we build on involves clustering coherent regions of the image into groups based on appearance. We begin by segmenting the image into regions, known as superpixels, using the normalized cut algorithm of Ren and Malik [107]. For each region j, we extract a feature vector $\mathbf{F}_j$ that includes color and texture features. For our stuff model, we will use a generative model where each region has a hidden class, and the features are generated from a Gaussian distribution with parameters depending on the class. Denote by S_j (S for "stuff") the (hidden) class label of the j^{th} region. We assume that the features are derived from a standard mixture of

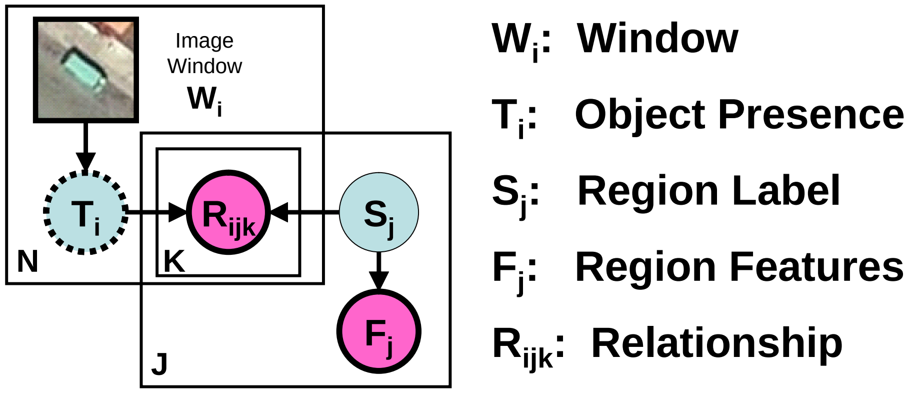


Figure 5.3: The TAS model. The plate representation gives a compact visualization of the model. Nodes with dotted lines represent variables that are observed during training; only the pink nodes (which represent image features) are observed during testing.

Gaussians model, where we have a probability distribution $P(\mathbf{F}_j \mid S_j)$ of the image features given the class label.

In order to relate the detector for "things" (T's) to the clustering model over "stuff" (S's), we need to develop an intuitive representation for the relationships between these units. Human knowledge in this area comes in sentences like "cars drive on roads," or "cars park 20 feet from buildings." We introduce indicator variables R_{ijk} that indicate whether candidate detection i and region j have relationship k. The different k's represent different relationships, such as: "detection i is *in* region j" ($k = 1$), or "detection i is about 100 pixels away from region j" ($k = 2$). For now we assume the set of relationships (the meaning of R_{ijk} for each k) is known, and in Section 5.2 we describe how this set of relationships is learned from the data.

We can now link our component models into a single coherent probabilistic things and stuff (TAS) model, as depicted in the plate model of Figure 5.3. Probabilistic influence flows between the detection window labels and the image region labels through the v-structures that are activated by observing the relationship variables. For a particular input image, this plate model unrolls into a "ground" network that encodes a distribution over the detections and regions of the image. Figure 5.4 shows

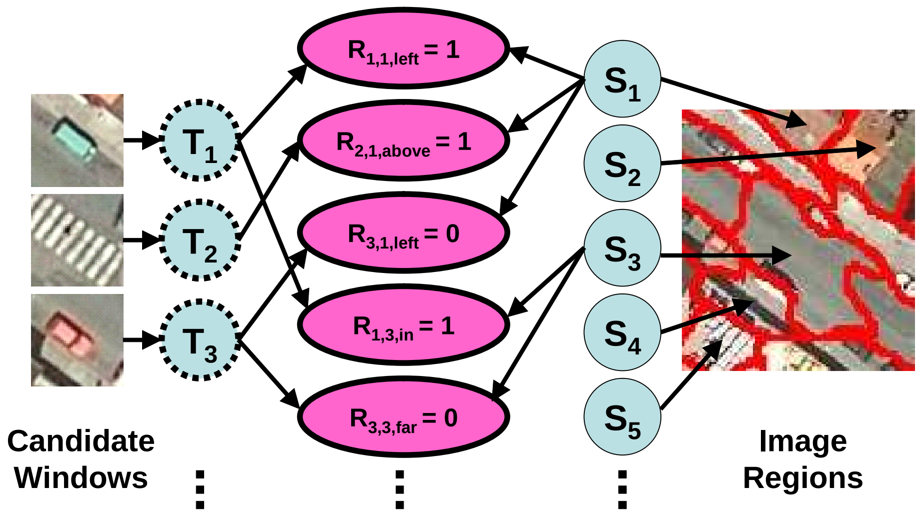


Figure 5.4: An example of the the TAS model unrolled into a "ground" network for some particular image.

a toy example of a partial ground network for the image of Figure 5.1. It is interesting to note the similarities between TAS and the MRF approaches in the literature. In effect, the relationship variables link the detections and the regions into a probabilistic web where signals at one point in the web influence a detection in another part of the image, which in turn influences the region label in yet another location. Although similar to an MRF in its ability to define a holistic view of the entire image, our model is generative, allowing us to train very efficiently, even when the S_j variables are unobserved. In addition, while MRFs have pairwise terms between regions, encouraging smoothness, our construction does not include these terms. While such terms have been shown to improve segmentation performance, this is not our goal, and as a result we have used this simpler formulation.

All variables in the TAS model are discrete except for the feature variables $\mathbf{F}_j$. This allows for simple table conditional probability distributions (CPDs) for all discrete nodes in this Bayesian network. The probability distribution over these variables decomposes according to:

$$P(\mathbf{T}, \mathbf{S}, \mathbf{F}, \mathbf{R} \mid \mathbf{W}) = \prod_i P(T_i \mid W_i) \prod_j P(S_j) P(\mathbf{F}_j \mid S_j) \prod_{ijk} P(R_{ijk} \mid T_i, S_j).$$

As the $\mathbf{W}$, $\mathbf{R}$ and $\mathbf{F}$ variables are always observed, the conditional probability distribution over the target variables is merely a renormalized version of this equation.

We note that TAS is very modular. It allows us to "plug in" any detector and any generative model for regions (e.g., [119]).

5.2 Learning and Inference in the TAS Model

Because TAS unrolls into a Bayesian network for each image, we can use standard learning and inference methods. In particular, we learn the parameters of our model using the Expectation-Maximization (EM) [35] algorithm and perform inference using Gibbs sampling, a standard variant of MCMC sampling [55]. Furthermore, we show how to learn the set of active relationships from a large candidate relationship pool using a structure search interleaved with the EM [54].

5.2.1 Learning the TAS Parameters with EM

At learning time, we have a set of images with annotated labels for our target object class(es). We first train the base detectors using this set, and select as our candidate windows all detections above a threshold. This produces a set of candidate detection windows $W_1 \dots W_N$ along with their ground-truth labels $T_1 \dots T_N$. We also have a set of regions and a feature vector F_j for each, and an R_{ijk} relationship variable for every window-region pair and relationship type, which is observed for each pair based on their relative spatial relationship. Our goal is to learn parameters for the TAS model.

Perhaps the simplest option is to learn in two phases: we first use an existing clustering method (such as K-means or Mixture-of-Gaussian clustering) to assign the S_j variables in the training data; we then learn parameters for the TAS model with fully observed data. This approach is attractive as it allows the first part of the learning to be fully unsupervised, enabling us to use more images than exist in our labeled dataset. We call the model learned with this method **Pre-Clustered TAS**. But, while the resulting clusters are likely to be visually coherent, they do not exploit the spatial relationships between the regions and the object detections.

We therefore propose a joint training method where we perform joint learning in the full model. Because our variables are discrete and have relatively small cardinality, the EM algorithm is a natural fit. EM iterates between using probabilistic inference to derive a soft completion of the hidden variables (E-step) and finding maximum-likelihood parameters relative to this soft completion (M-step). The E-step here is particularly easy: at training time, only the S_j's are unobserved; moreover, because the T variables are observed, the S_j's are conditionally independent of each other. Thus, the inference step turns into a simple computation for each S_j separately, a process which can be performed in linear time. Suppose that we have a current estimate of the parameters, given by Θ. In the E-step, our goal is to compute the expected counts:

$$\begin{aligned}\bar{M}[t,s,r_k] =& E_\Theta\left[\sum_m\sum_i\sum_j 1\{T_i[m]=t, S_j[m]=s, R_{ijk}[m]=r_k\}\right]\\ =& \sum_{m,i,j} P\left(T_i[m]=t, S_j[m]=s, R_{ijk}[m]=r_k \mid (\mathbf{F},\mathbf{W})[m],\Theta\right)\\ =& \sum_{m,i,j} P\left(S_j[m]=s \mid (\mathbf{T},\mathbf{R},\mathbf{F},\mathbf{W})[m],\Theta\right) 1\{T_i[m]=t, R_{ijk}[m]=r_k\}.\end{aligned}$$

As noted, the distribution over $S_j[m]$ conditioned on all of the observed variables is independent of all other $S_k[m]$. This distribution can be computed from:

$$P(S_j \mid \mathbf{T},\mathbf{F},\mathbf{R},\mathbf{W}) \propto P(S_j)P(\mathbf{F}_j \mid S_j)\prod_{ik} P(R_{ijk} \mid T_i, S_j). \tag{5.1}$$

The M-step for table CPDs can be performed easily in closed form, given these counts. The parameters for the learned CPDs are given by:

$$P(R_{ijk} = 1 \mid T_i = t, S_j = s) = \frac{\bar{M}[t, s, r_k = 1]}{\bar{M}[t, s, r_k = 0] + \bar{M}[t, s, r_k = 1]} \tag{5.2}$$

$$P(S_j = s) = \frac{\sum_{t,r_k} \bar{M}[t, s, r_k]}{\sum_{t,r_k,s'} \bar{M}[t, s', r_k]}. \tag{5.3}$$

The Gaussian parameters for the region feature distributions are given by:

$$\begin{aligned}\mu_s =& E[\mathbf{F}_j \mid S_j = s] = \frac{\sum_{m,j} P(S_j[m] \mid \mathbf{T}[m], \mathbf{F}[m], \mathbf{R}[m], \mathbf{W}[m])\mathbf{F}_j[m]}{\sum_{m,j} P(S_j[m] \mid \mathbf{T}[m], \mathbf{F}[m], \mathbf{R}[m], \mathbf{W}[m])} \\ \Sigma_s =& E[(\mathbf{F}_j - \mu_s)(\mathbf{F}_j - \mu_s)^T \mid S_j = s] \\ =& \frac{\sum_{m,j} P(S_j[m] \mid \mathbf{T}[m], \mathbf{F}[m], \mathbf{R}[m], \mathbf{W}[m])(\mathbf{F}_j[m] - \mu_s)(\mathbf{F}_j[m] - \mu_s)^T}{\sum_{m,j} P(S_j[m] \mid \mathbf{T}[m], \mathbf{F}[m], \mathbf{R}[m], \mathbf{W}[m])}.\end{aligned}$$

These parameters can efficiently be computed during the E-step by accruing the moments during the computation of the posterior distribution over the S_j's.

To provide a good starting point for EM, we initialize the cluster assignments using the K-means algorithm. EM is guaranteed to converge to a fixed point (usually a local maximum) of the likelihood function of the observed data.

5.2.2 Learning the TAS Relationships

So far, we have assumed a known set of relationships. However, because different data may require different types of contextual relationships, we would like to learn which to use. We begin by defining a large set $\mathcal{C}$ of "candidate relationships" (i.e., all possible relationships that we want to consider for inclusion). For instance, we may include both "above by 100 pixels," and "above by 200 pixels" in $\mathcal{C}$, even though we believe only one of these will be chosen. Our goal is to search through $\mathcal{C}$ for the subset of relationships that will best facilitate the use of context. We denote this "active" set by $\mathcal{R}$.

Intuitively, if a particular type of relationship (e.g., , "above by 100 pixels") is "inactive" then we want to force the value of the corresponding R_{ijk} variables to be independent of the value of the T_i's and S_j's. In the language of Bayesian networks, we can achieve this by removing the edges from all T_i and S_j variables to the R_{ijk} variables for this particular k. With this view of "activating" relationships by including the edges in the Bayesian network, we can formulate our search for $\mathcal{R}$ as a structure learning problem. To learn this network structure, we turn to the method of structural EM [54]. In particular, if we are considering K possible relationships, there are 2^K possible subsets to consider (each relationship can be "active" or "inactive"). We search this space using a greedy hill-climbing approach that is interleaved with the EM parameter learning. The hill-climbing begins with an empty set of relationships ($\mathcal{R} = \emptyset$), and adds or removes relationships one at a time until a local maximum is reached.

Standard structural EM scores each candidate graph structure $\mathcal{G}$, using a penalty plus the expected log likelihood of the data (hidden and observed), given the best soft completions to the missing variables at the previous step, and the best resulting parameters for the candidate structure. This quantity is easily computed for each structure based on the output of the E-step above. In our case, this score (without the penalty) is given by:

$$score_{ll}(\mathcal{G}) = E\left[\sum_m \log P(\mathbf{T}[m], \mathbf{S}[m], \mathbf{R}[m], \mathbf{F}[m] \mid \mathbf{W}[m], \Theta_{\mathcal{G}})\right] \tag{5.4}$$

where $\Theta_{\mathcal{G}}$ are the ML parameters for the structure $\mathcal{G}$ given the expected counts. Because all variables other than the S's are observed, this only requires summing over the values for each S_j independently. This scoring function allows for efficient scoring of the candidate networks.

In our case, however, the final goal is to correctly classify the "things." The above score awards the more accurate modeling of all of the variables, so that a structure that allows for much better likelihood of R's can outshine a model that has slightly better likelihood for the T's. In preliminary experiments, this scoring function chose to activate relationships that are easy to predict from the parent T and S, rather than relationships that helped to predict the T's. In order to overcome this problem, we would rather score each structure using the log probability of the T_i's. This "posterior" score is given by:

$$score_{post}(\mathcal{G}) = \sum_m \log P(\mathbf{T}[m] \mid \mathbf{R}[m], \mathbf{F}[m], \mathbf{W}[m], \Theta_{\mathcal{G}}) \tag{5.5}$$

While this requires us to perform inference (described below) for each candidate structure, it is both theoretically more appropriate and produced far better results than the standard score. In order to avoid overfitting, we could use a standard BIC penalty of the form:

$$penalty_{BIC} = \frac{\log M}{2} Dim(\mathcal{G}), \tag{5.6}$$

where $Dim(\mathcal{G})$ is the number of free parameters in the graph structure $\mathcal{G}$, and M is the number of data instances. In initial experiments, we found that this resulted in too few "active" relationships. This problem of underfitting is common when the BIC score is used [139], and motivates the use of a weaker penalty. In experiments below, we instead include a Gaussian prior over the number of active relationships:

$$penalty_A = \log \mathcal{N}(A; \mu_A, \sigma_A^2), \tag{5.7}$$

where A is the number of active relationships. The prior is set to encourage a small number of relationships (typically $\mu_A = 0$ and $\sigma_A^2 \in [1, 10]$). This leads to a final scoring function given by:

$$score_{TAS} = \sum_m P(\mathbf{T}[m] \mid \mathbf{R}[m], \mathbf{F}[m], \mathbf{W}[m], \Theta_{\mathcal{G}}) - \log \mathcal{N}(A; \mu_A, \sigma_A^2) \tag{5.8}$$

Our learning process outputs an active set of relationships, $\mathcal{R}$, and the parameters of the TAS model for that set, $\theta_{\mathcal{R}}$. The algorithm is outlined in Figure 5.5.

5.2.3 Inference in the TAS model

At test time, our system must determine which windows in a new image contain the target object. We observe the candidate detection windows (W_i's, extracted

```
Algorithm LearnTAS
Input:  Candidate relationships $\mathcal{C}$, Dataset $\mathcal{D} = \{(\mathbf{W}[m], \mathbf{T}[m], \mathbf{F}[m], \mathbf{R}[m])\}$
    $\mathcal{R} \leftarrow \emptyset$ (all relationships "inactive")
    Repeat until convergence
        Repeat until convergence (EM over Parameters)
            E-step: $Q[m] \leftarrow P(\mathbf{S}[m] \mid \mathbf{T}[m], \mathbf{F}[m], \mathbf{R}[m]; \theta_{\mathcal{R}}) \quad \forall m$
            M-step: $\theta_{\mathcal{R}} \leftarrow \text{argmax} E_Q\left[\sum_m \ell(\mathbf{S}[m], \mathbf{T}[m], \mathbf{F}[m], \mathbf{R}[m] \mid \mathbf{W}[m]; \theta_{\mathcal{R}})\right]$
        Repeat until convergence (Greedy Structure Search)
            Forall $k$, $score_k = score_{TAS}(\mathcal{R} \oplus k)$ (score with $k$ "activated")
            $\mathcal{R} \leftarrow \mathcal{R} \oplus k^*$      where $k^* = \text{argmax } score_k$
Return  Set $\mathcal{R}$ of "active" relationships, TAS parameters $\theta_{\mathcal{R}}$
```

Figure 5.5: Learning a TAS model. Here ℓ represents the log-likelihood of the data, and $\oplus$ represents the set exclusive-or operation.

by thresholding the base detector output), the features of each image region ($\mathbf{F}_j$'s), and the relationships (R_{ijk}'s). Our task is to find the probability that each window contains the object:

$$P(\mathbf{T} \mid \mathbf{F}, \mathbf{R}, \mathbf{W}) = \sum_{\mathbf{S}} P(\mathbf{T}, \mathbf{S} \mid \mathbf{F}, \mathbf{R}, \mathbf{W}) \tag{5.9}$$

Unfortunately, this expression involves a summation over an exponential set of values for the $\mathbf{S}$ vector of variables. We solve the inference problem approximately using a Gibbs sampling MCMC method [55]. We begin with some assignment to the variables. Then, in each Gibbs iteration we first resample all of the S's and then resample all the T's according to the following two probabilities:

$$P(S_j \mid \mathbf{T}, \mathbf{F}, \mathbf{R}, \mathbf{W}) \propto P(S_j) P(F_j \mid S_j) \prod_{ik} P(R_{ijk} \mid T_i, S_j) \tag{5.10}$$

$$P(T_i \mid \mathbf{S}, \mathbf{F}, \mathbf{R}, \mathbf{W}) \propto P(T_i \mid W_i) \prod_{jk} P(R_{ijk} \mid T_i, S_j). \tag{5.11}$$

These sampling steps can be performed efficiently, as the T_i variables are conditionally independent given the S's and the S_j's are conditionally independent given the T's. In the last Gibbs iteration for each sample, rather than resampling $\mathbf{T}$, we compute the

Features	Description	Number
location	(x,y) pixel location of region centroid	2
mean and std of R	Image red channel	2
mean and std of G	Image green channel	2
mean and std of B	Image blue channel	2
mean and std of L	Image Lab luminance channel	2
mean and std of a	Image Lab ‘a’ channel	2
mean and std of b	Image Lab ‘b’ channel	2
size	# pixels in region / # pixels in image	1
ap ratio	# pixels in region / (region perimeter)^2	1
moment	moment of inertia of region shape	1
aspect	region area / region bounding box area	1
mean and std of responses to 13 texture filters	convolution with Gabor-like filters	26

Table 5.1: Features computed for each region.

posterior probability over **T** given our current **S** samples, and use these distributional particles for our estimate of the probability in Eq. (5.9). Determining the convergence of Gibbs inference is a tricky question, which we explore a bit in experiments below.

5.3 Experimental Results

In order to evaluate the TAS model, we perform experiments on three datasets. The first two are from the PASCAL Visual Object Classes challenges 2005 and 2006 [40]. The scenes are urban and rural, indoor and outdoor, and there is a great deal of scale variation amongst the objects. The third is a set of satellite images acquired from Google Earth, with the goal of detecting cars. Because of the impoverished visual information, there are many false positives when a sliding window detector is applied. In this case, context provides a filtering mechanism to remove the false positives. Because these two applications are different, we use different detectors for each.

We allow S a cardinality of $|S| = 10$, having determined that results are generally robust to a range of $|S|$ between 5 and 20. We use 44 features for the image regions

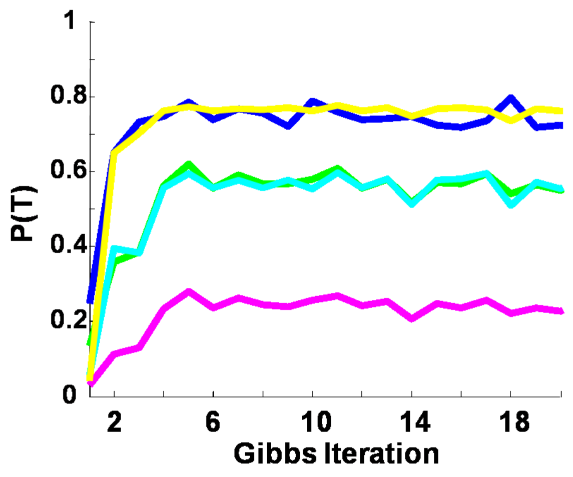


Figure 5.6: Gibbs sampling trajectories for five thing variables (candidate cars) for a random satellite image. These five variable have the most variation from the beginning to the end of the Gibbs inference, but still converge after a small number (4-5) of iterations.

(color, texture, shape, etc.), based on the features developed by Barnard et al. [5]). The particular features are described in Table 5.1. We learned our model with a set of 25 candidate relationships that included regions within the window, at offsets in eight directions (every 45 degrees) at two different distances, and the union of various combinations of features (e.g., R_{24} indicates regions to the right *or* left of the window by one bounding box). Table 5.2 lists the relationships used in experiments below.

5.3.1 Determination of Experimental Parameters

For our MCMC inference, we found that the Markov chain tended to converge very rapidly. In many cases, the appearance distribution of the clusters was strong enough such that, for most of the regions, each of the samples assigned it to the same cluster. This resulted in a very fast convergence of the posterior distribution over the thing (T_i) variables. To illustrate this, Figure 5.6 looks at a few sampling trajectories for T_i variables in the satellite data. We randomly selected an image and here we plot the Gibbs inference estimate of $P(T_i)$, based on 50 samples, for the five thing variables

Relationship Names	Description
IN	Center of window (C_i) IN region j area A_j
NEAR	C_i within T=50 pixels of A_j
ABOVE	One bounding box height ABOVE center of window (C_i + (0,H_{bb}) in A_j
BELOW	(C_i - (0,H_{bb}) in A_j
RIGHT	One bounding box width to the RIGHT of window (C_i + (W_{bb},0) in A_j
LEFT	(C_i - (W_{bb},0) in A_j
ABOVE LEFT	(C_i + (-$\sqrt{1/2}W_{bb}$, $\sqrt{1/2}H_{bb}$) in A_j
ABOVE RIGHT	(C_i + ($\sqrt{1/2}W_{bb}$, $\sqrt{1/2}H_{bb}$) in A_j
BELOW LEFT	(C_i + (-$\sqrt{1/2}W_{bb}$, -$\sqrt{1/2}H_{bb}$) in A_j
BELOW RIGHT	(C_i + ($\sqrt{1/2}W_{bb}$, -$\sqrt{1/2}H_{bb}$) in A_j
FAR ABOVE	Two bounding box heights ABOVE center of window (C_i + (0,2H_{bb}) in A_j
FAR BELOW	(C_i - (0,2H_{bb}) in A_j
FAR RIGHT	Two bounding box widths to the RIGHT of window (C_i + (2W_{bb},0) in A_j
FAR LEFT	(C_i - (2W_{bb},0) in A_j
FAR ABOVE LEFT	(C_i + (-$\sqrt{2}W_{bb}$, $\sqrt{2}H_{bb}$) in A_j
FAR ABOVE RIGHT	(C_i + ($\sqrt{2}W_{bb}$, $\sqrt{2}H_{bb}$) in A_j
FAR BELOW LEFT	(C_i + (-$\sqrt{2}W_{bb}$, -$\sqrt{2}H_{bb}$) in A_j
FAR BELOW RIGHT	(C_i + ($\sqrt{2}W_{bb}$, -$\sqrt{2}H_{bb}$) in A_j
ANY CARDINAL	ABOVE, BELOW, RIGHT, or LEFT
ANY OFF-CARDINAL	ABOVE RIGHT, BELOW RIGHT, ABOVE LEFT, or BELOW LEFT
ANY FAR CARDINAL	FAR ABOVE, FAR BELOW, FAR RIGHT, or FAR LEFT
ANY FAR OFF-CARDINAL	FAR ABOVE RIGHT, FAR BELOW RIGHT, FAR ABOVE LEFT, or FAR BELOW LEFT
LEFT OR RIGHT	LEFT or RIGHT
FAR LEFT OR RIGHT	FAR LEFT or FAR RIGHT
ALL LEFT OR RIGHT	LEFT, RIGHT, FAR LEFT, or FAR RIGHT

Table 5.2: Set of relationships considered by the relationships learning algorithm.

with the largest variation from the beginning of the Gibbs iterations to the end. The x-axis represents which sampling iteration, and we plot the posteriors out to 20 iterations. Even for these cases, where the variation was the largest, the estimates appear to settle after 4-5 sampling iterations. Because the final evaluation of the model will be based on this posterior, this indicates that we probably do not need much more than this many iterations during inference. Samples for the VOC data tended to converge even faster than those for the satellite data, because there were far fewer detection candidates per image.

5.3.2 PASCAL VOC Datasets

For these experiments, we used four classes from the VOC2005 data, and two classes from the VOC2006 data. The VOC2005 dataset consists of 2232 images, manually annotated with bounding boxes for four image classes: cars, people, motorbikes, and bicycles. We use the "train+val" set (684 images) for training, and the "test2" set (859 images) for testing. The VOC2006 dataset contains 5304 images, manually annotated with 12 classes, of which we use the cow and sheep classes. We train on the "trainval" set (2618 images) and test on the "test" set (2686 images). To compare with the results of the challenges, we adopted as our detector the HOG (histogram of oriented gradients) detector of Dalal and Triggs [32]. This detector uses an SVM and therefore outputs a score $\text{margin}_i \in (-\infty, +\infty)$, which we convert into a probability by learning a logistic regression function for $P(T_i \mid \text{margin}_i)$. We also plot the precision-recall curve using the code provided in the challenge toolkit.

Our learning, which included ten random restarts of EM at each parameter learning iteration, took between 3 and 28 hours on an Intel Dual Core 1.9 GHz machine with 2 GB of memory. Because we have so many images in the testing set (859 or 2686), we desired fast inference per image. Results above indicate that only a small number of Gibbs iterations are needed, and as a result, we used only 10 samples with 5 block-Gibbs iterations each, which required 0.5 seconds per image. We ran separate experiments for each class, though in principle it would be possible to learn a single joint model over all classes. By separating the classes, we were able to isolate

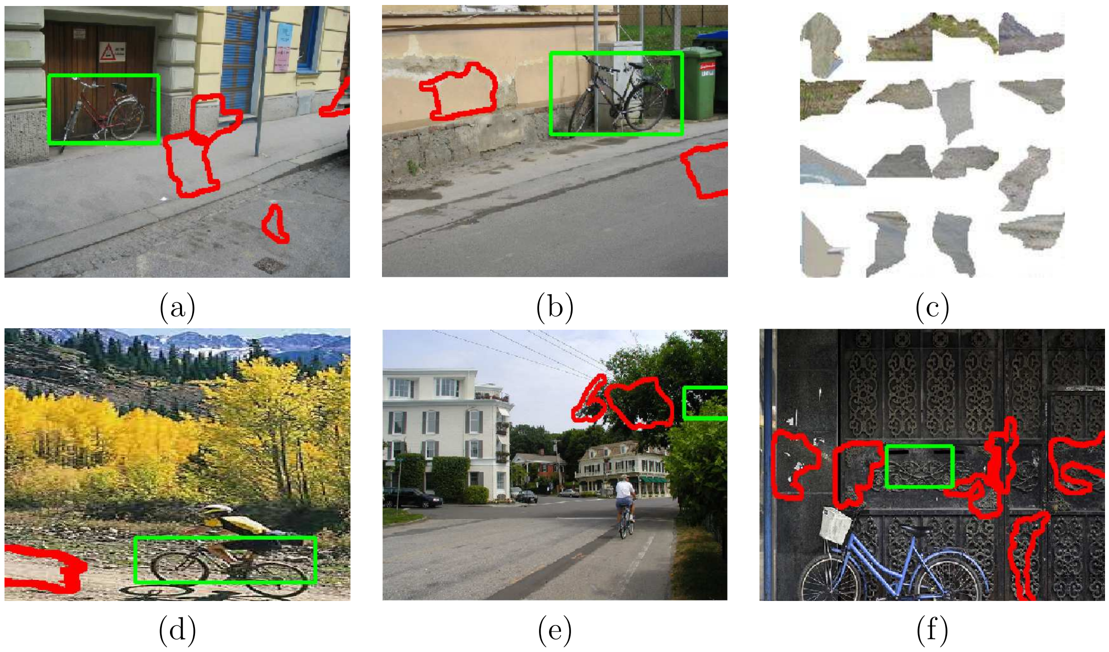
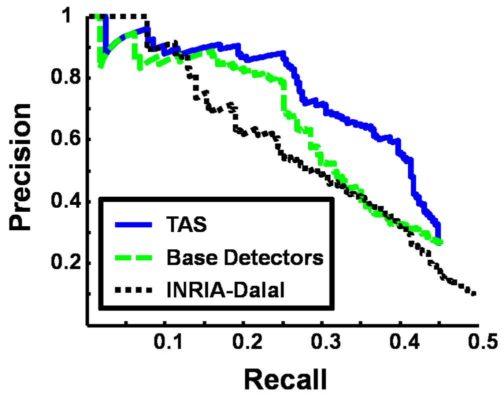


Figure 5.7: (a,b) Example training detections from the bicycle class, with detection windows outlined by the green rectangles. The image regions with active relationships to the detection window are outlined in red. (c) 16 of the most representative regions for cluster #3. This cluster corresponds to “roads” or “bushes” as things that are gray/green and occur near cars. (d) A case where context helped find a true detection. (e,f) Two examples where incorrect detections are filtered out by context.

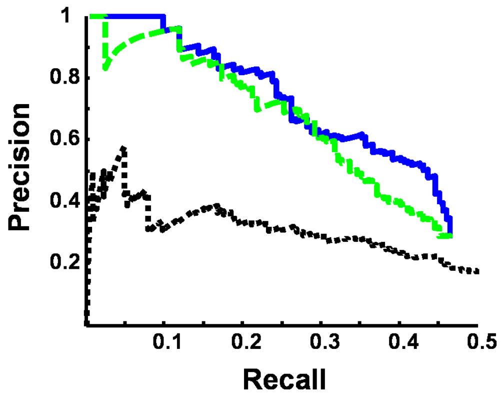
the contextual contribution from the stuff, rather than between the different types of things present in the images.

We begin with a qualitative assessment of the results. Figure 5.7 (top row) shows example bicycle detection candidates, and the related image regions, suggesting the type of context that might be learned. For example, the region beside and below both detections (outlined in red) belongs to cluster #3, which looks visually like a road or bush cluster (see Figure 5.7(c)). The learned values of the model parameters also indicate that being to the left or right of this cluster increases the probability of a window containing a bicycle (e.g., by about 33% in the case where $R_{ijk} = 1$ for this relationship).

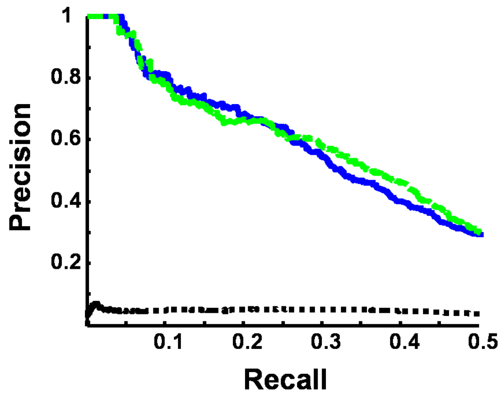
The bottom row of Figure 5.7 shows some detections that were corrected using



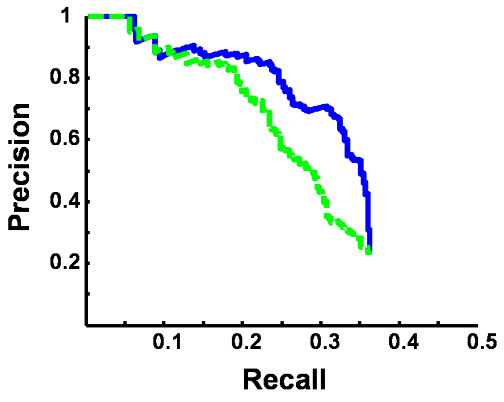
(a) Cars (2005)



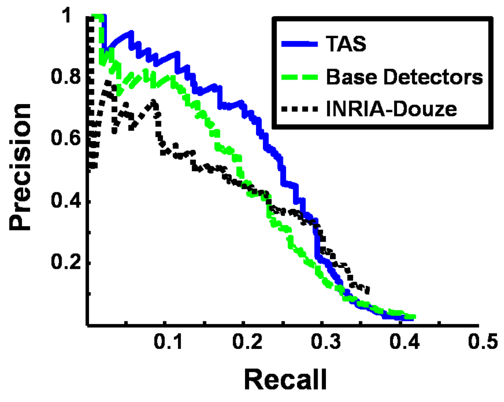
(b) Motorbikes (2005)



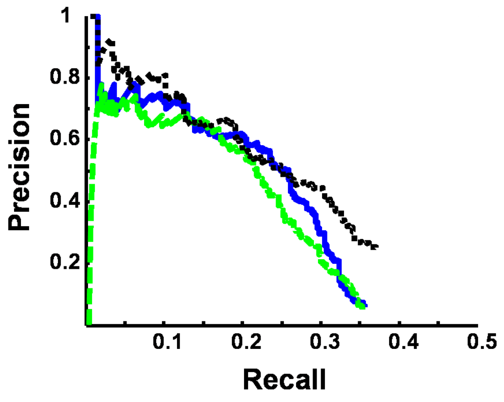
(a) People (2005)



(b) Bicycles (2005)



(a) Cows (2006)



(b) Sheep (2006)

Figure 5.8: Precision-recall (PR) curves for the VOC classes.

Object Class	Base AP	TAS AP (Fixed R)	TAS AP (Learned R)	Improvement (TAS - Base)
Cars	0.325	0.360	**0.370**	0.045
Motorbikes	0.341	**0.390**	0.360	0.019
People	**0.346**	0.346	0.340	-0.006
Bicycles	0.281	0.310	**0.317**	0.036
Cows	0.224	0.241	**0.256**	0.032
Sheep	0.206	0.233	**0.240**	0.034

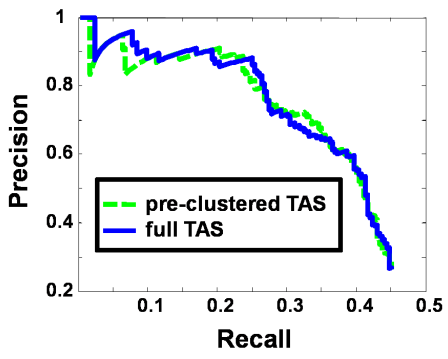
Table 5.3: Average precision (AP) scores for each experiment. AP is a robust variant of the area under the PR curve. We show the AP score for TAS with hand-selected relationships as well as with learned relationships, with results from the best performing model in bold.

context. We show one example where a true bicycle was discovered using context, and two examples where false positives were filtered out by our model. These examples demonstrate the type of information that is being leveraged by TAS. In the first example, the dirt road to the left of the window gives a signal that this detection is at ground level, and is therefore likely to be a bicycle.

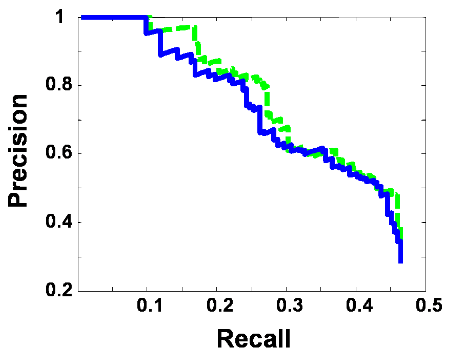
Figure 5.8 shows the full recall-precision curve for each class. For (a-d) we compare to the 2005 **INRIA-Dalal** challenge entry, and for (e,f) we compare to the 2006 **INRIA-Douze** entry, both of which used the HOG detector. We also show the curve produced by our **Base Detector** alone. [1] Finally, we plot the curves produced by our **TAS Model**, trained using full EM, which scores windows using the probability of Eq. (5.9). From these curves, we see that the TAS model provided an improvement in accuracy for all but the "people" class. We believe the lack of improvement for people is due to the wide variation of backgrounds in these images, providing no strong context cues to latch onto. Furthermore, the base HOG detector was in fact originally optimized to detect people.

Figure 5.9 compares learning with pre-clustered TAS to full TAS. The curves are very similar, and the only apparent difference is that the pre-clustered model seems

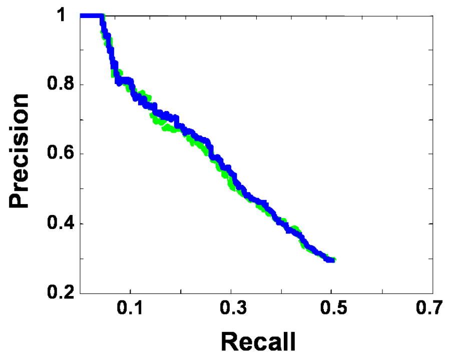
[1]Differences in PR curves between our base detector and the **INRIA-Dalal/INRIA-Douze** results come from the use of slightly different training windows and parameters. **INRIA-Dalal** did not report results for "bicycle."



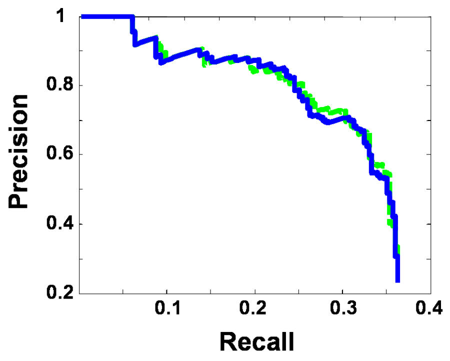
(a) Cars (2005)



(b) Motorbikes (2005)



(a) People (2005)



(b) Bicycles (2005)

Figure 5.9: Precision-recall (PR) curves for the VOC 2005 classes, comparing pre-clustered TAS to full TAS. There is no clear difference between these learning strategies for this data.

to slightly outperform the full TAS for the motorbike class. This is likely due to overfitting of the contextual component of the clusters to the training data. For other classes, the performance of the two models is basically indistinguishable.

5.3.3 Satellite Images

The second dataset is a set of 30 images extracted from Google Earth. The images are color, and of size 792×636, and contain 1319 manually labeled cars. The average car window is approximately 45×45 pixels, and all windows are scaled to these dimensions for training. We used 5-fold cross-validation, and results below report the mean performance across the folds.

Here, we use a patch-based boosted detector very similar to that of Torralba [126]. We use 50 rounds of boosting with two level decision trees over patch cross-correlation features that were computed for 15,000–20,000 rectangular patches (intensity and gradient) of various aspect ratios and widths of 4-22 pixels. As above, we convert the boosting score into a probability using logistic regression. Because the size of this dataset is significantly smaller than the VOC datasets used above, we can afford to do more rounds of EM in between each structure step. For training the TAS model on this data, we used ten random restarts of EM, selecting the parameters that provided the best likelihood of the observed data. For inference, we can also afford to use more Gibbs samples and iterations. We utilize 20 samples per image, where each sample undergoes 20 iterations.

Figure 5.10 shows some learned "stuff" clusters. Eight of the ten learned clusters are shown, visualized by presenting 16 of the image regions that rank highest with respect to $P(\mathbf{F} \mid S)$. These clusters have a clear interpretation: cluster #4, for instance, represents the roofs of houses and cluster #6 trees and water regions. With each cluster, we also show the odds-ratio of a candidate window containing a car given that it is in this region. Clusters #7 and #8 are road clusters, and increase the chance of a nearby window being a car by a factor of 2 or more. Clusters #1 and #6, however, which represent forest and grass areas, decrease the probability of nearby candidates being cars by factors of 9 or more.

Figure 5.11 shows an example with the detections of the detector alone and of the TAS model, which filters out many of the false positives that are not near roads. Because there are many detections per image, we plot the recall versus the number of false detections per image in Figure 5.11(c). The **Base Detectors** are compared to the **TAS Model**, verifying that context indeed improves our results, by filtering out many of the false positives.

Finally, we compare the pre-clustered TAS model (described above) to the full TAS setup. Figure 5.12 shows the recall vs. false-positives-per-instance (fppi) curves, zoomed in to a portion of the graph between 10 and 50 false positives per image. False-positives-per-instance measures the average number of false positives detected in each

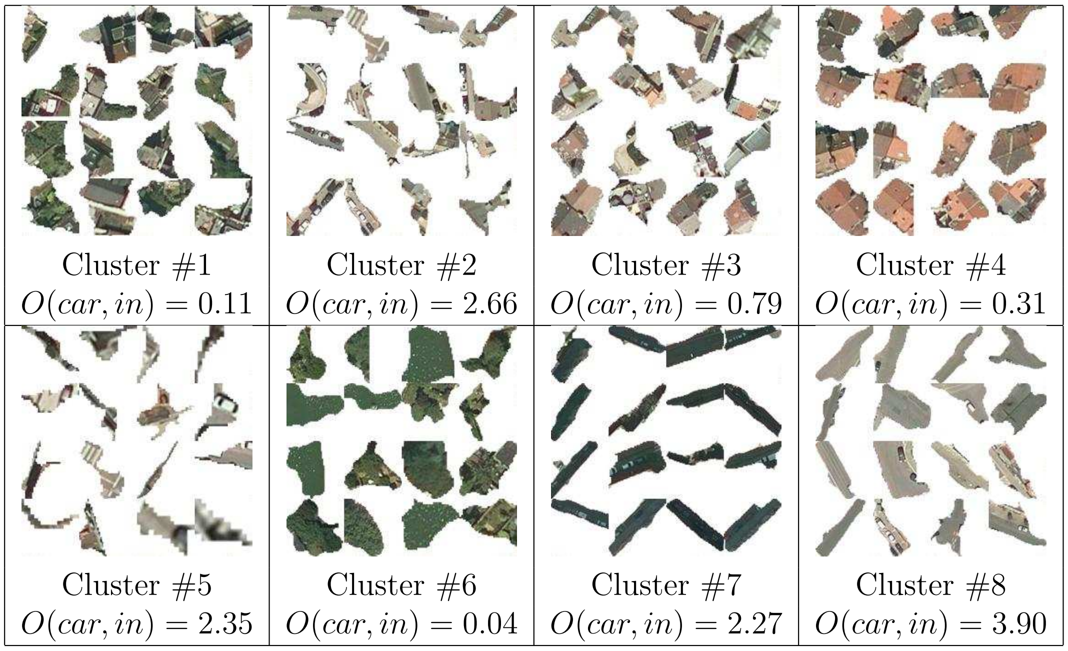
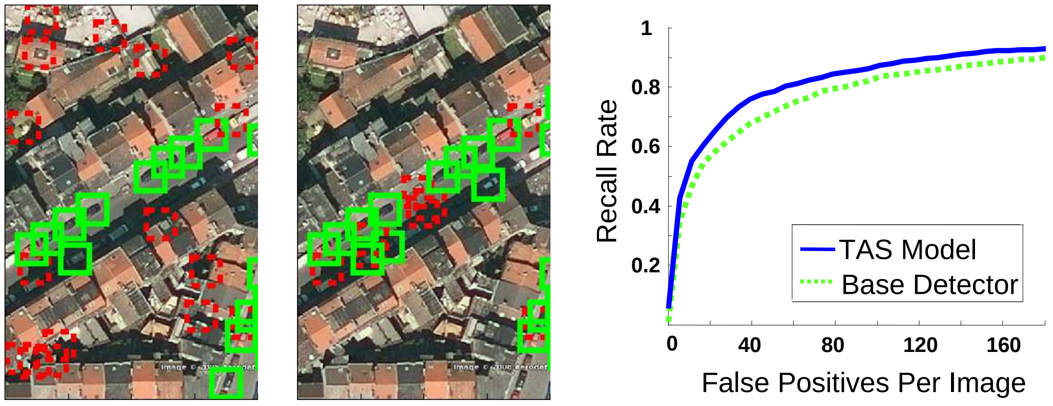


Figure 5.10: Example clusters learned by the context model on the satellite dataset. Each cluster shows 16 of the training image regions that are most likely to be in the cluster based on $P(\mathbf{F} \mid S)$. For each cluster, we also show the odds-ratio of a window "in" a region labeled by that cluster containing a car ($O(car, in) = P(in|car, q)/P(in|no\ car, q)$). A higher odds-ratio indicates that this contextual relationship increases the model's confidence that the window contains a car.

image. This area is where the greatest deviation between the methods occurs. Averaged across all precisions (0-200 fppi), the full TAS outperformed the pre-clustered by about 1% in recall rate. The maximum deviation was a 3% improvement over pre-clustered. This result shows that learning in the full setup gives a modest, but consistent improvement over learning the clusters by appearance only.

5.4 Comparison: TAS vs. CCM

Recall that one of the primary goals of the development of the TAS model was the desire to more accurately represent the contextual relationship between things and stuff that we began exploring with the **CCM** approach in Chapter 4. Indeed, the TAS



(a) Base Detectors (b) TAS Detections (c) PR Curve

Figure 5.11: Example image, with detections found by the base detector (a), and by the TAS model (b) with a threshold of 0.15. The TAS model filters out many of the false positives far away from roads. (c) shows a plot of recall rate vs. false positives per image for the satellite data. The results here are averaged across 5 folds, and show a significant improvement from using TAS over the base detectors.

and **CCM** models share many similarities. Both methods leverage contextual cues, in the form of region and object labels, to update beliefs about the classification results. Both methods represent objects as candidate bounding boxes, and both represent regions as pre-segmented superpixels with a small number of discrete labels.

The two most significant differences between these methods are the way that contextual information is shared between the components, and the meaning of the region labels. In **CCM**, contextual information is shared when output *classifications* from the regions produced input *features* for the detectors. In TAS, the *beliefs* over the region labels influence the *beliefs* over the object labels through their spatial relationships.

The difference in meaning between the labels is also a significant difference between the two. In the **CCM** work, the region labels indicated specific "semantic" classes of stuff, for which we used a classifier trained specifically to identify that type of stuff (the boosted detectors that served as singleton features in the CRF model). The training data was manually annotated with labels for each training region, allowing the classifiers to learn what "road" means, even though it might have several different

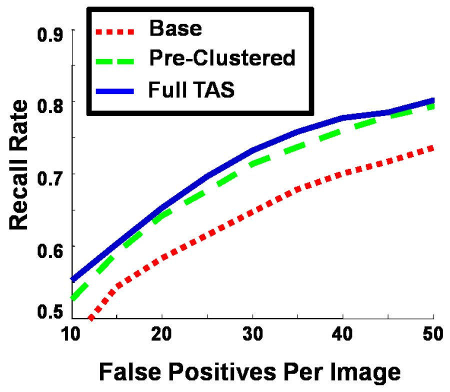


Figure 5.12: Recall rate vs. false positives per image for the satellite dataset. We compare the base detector to the pre-clustered TAS and to the full TAS.

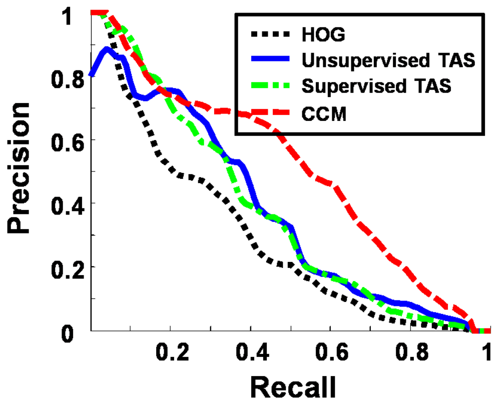
appearance modes. In the TAS work, the region "labels" are cluster indices, indicating to which contextual cluster type the region belongs. These clusters were learned in an unsupervised way, using only the appearance and contextual information. Indeed, as we saw in Figure 5.10, the TAS model learned multiple clusters that corresponded to road, and multiple clusters that corresponded to buildings. As a result, we cannot hope to learn as strongly tuned a segmentation model as we did in Chapter 4.

While the first difference is an inherent distinction between the models, the second was just a choice. Indeed, if we had labeled training data for the regions, we could train the TAS model in a supervised manner, with the clusters corresponding to the same semantic classes that we used in Chapter 4. In order to explore this possibility, we ran on the DS1 dataset of Chapter 4, and compared the results between the TAS model and the **CCM** approach. We ran TAS in two settings. The first is standard unsupervised TAS, trained in the same manner as the previous experiments. The second is supervised TAS, where at learning time, we fix the cluster labels for the superpixels based on the hand-labels of the training data. In particular, during the E-step of the EM algorithm, rather than predicting the soft assignment of the S_j variables, we assign each to the label of the plurality of pixels in that region.

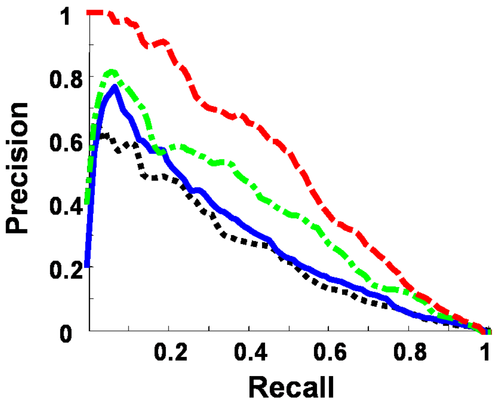
In these experiments, we use the same 5-fold cross validation splits as in the previous chapter, and our base detector is the same HOG detector for each. For comparison purposes, we train the independent **CCM** detectors with only the base detector score and bias as features (to make it the same as the TAS detection component). Figure 5.13 shows the results of these experiments, comparing **CCM** to the supervised and unsupervised versions of TAS. The first thing to notice is that including supervision does in fact improve the results of TAS on four out of six classes. Only in the boat class does the performance degrade. This is likely due to the multimodal nature of the appearance of water, which cannot be well modeled by the TAS region appearance model. In many cases, the benefit of supervision is slight, however, which supports the notion that much of the contextual signal can be discovered from the data.

The second observation from Figure 5.13, is that the **CCM** approach performs best, with the highest average precision in all classes except sheep. The **CCM** detectors perform particularly well for the motorcycle and boat classes. As the training data was exactly the same, this indicates that the state-of-the-art classifiers of the **CCM**, together with a simple cascade of output classifications to input features better informs the context-aware detectors than the joint modeling of the TAS approach.

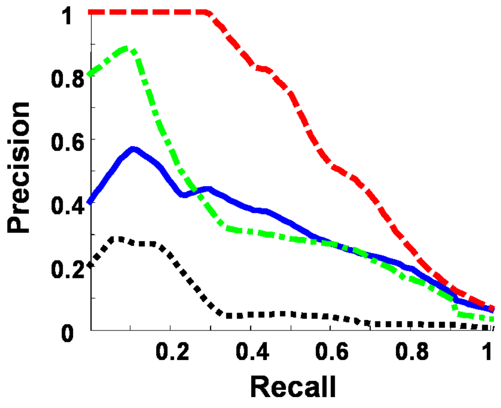
This performance gap comes from one (or more) of three possible sources. The first is that a mixture of Gaussians is not the appropriate model for semantically classifying the regions. Indeed, most segmentation models use cross-edges between the regions to enforce some notion of global smoothness. Figure 5.14 shows two example predictions of the region label using both the TAS and **CCM** models. The lack of smoothing in the TAS model is clearly evident, and the overall lower accuracy is also apparent. Across the full dataset, TAS predicts the correct pixel label only 56% of the time, compared to 74% for the **CCM** segmentation model. In both models, the contextual influence of the regions on the detections requires accurate region labeling (if you mistake road for water, you are likely to mistake a car for a boat). We can directly compare this component by using the CRF segmentation model from **CCM** to predict the regions, while maintaining the rest of the TAS model as is. Table 5.4 shows average precision numbers for the six classes, comparing this hybrid approach (TAS + **CCM** Regions) to both supervised TAS and **CCM**. While using the regions from the **CCM** segmentation model does tend to improve results (in four



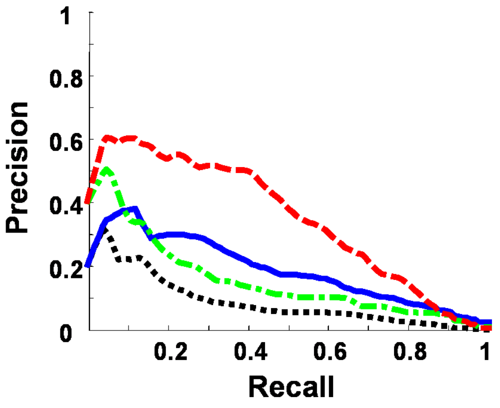
(a) Cars



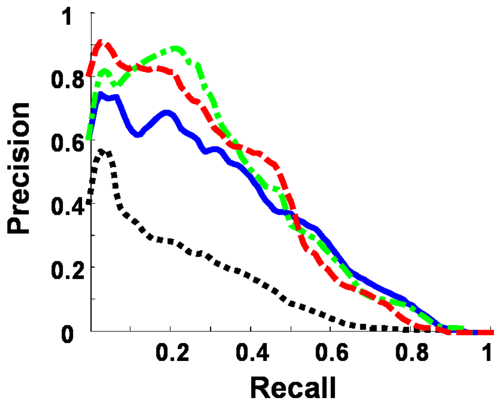
(b) Pedestrians



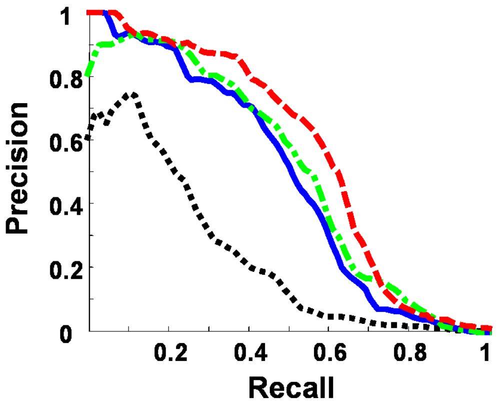
(c) Motorcycles



(d) Boats

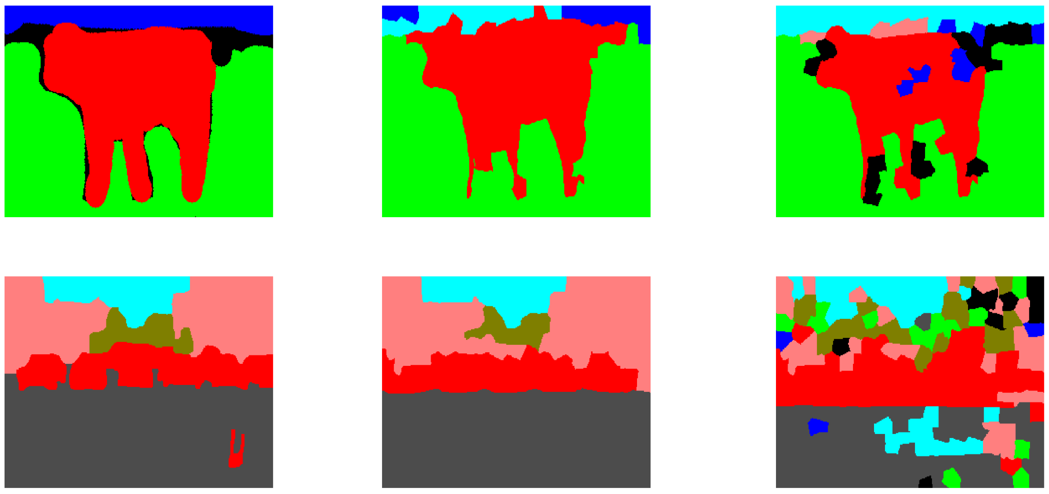


(e) Sheep



(f) Cows

Figure 5.13: Comparison of detection results on the DS1 dataset of Chapter 4 between the **CCM** method and the TAS approach.



(a) Groundtruth (b) CCM Output (c) Supervised TAS Output

Figure 5.14: Comparison of region labeling results on the DS1 dataset of Chapter 4 between the **CCM** method and the supervised TAS approach.

out of six classes), the improvement is not significant compared to the performance of the full **CCM** approach.

The second possible source of **CCM**'s dominance is the discriminative training of the **CCM** detection model. Because the model is trained directly to optimize detection performance on the training set, this might allow it to tune more appropriately than the TAS model, which is trained in a generative manner, such that the relationships, things, and stuff are all well-explained by the model. In order to explore this possibility we can run the **CCM** method with detection features derived from the TAS relationships. For each relationship that is active in the TAS model, we create an indicator vector for the class label of the related region. This feature is then used in place of the context features described above in Chapter 4. In Figure 5.15 we show the resulting precision-recall curves, comparing this approach (**CCM** + TAS features) to the base TAS and base **CCM** approaches. The TAS features are clearly inferior to the features developed for the **CCM** context-aware detectors, even with the discriminative training. Interestingly, in four out of the six classes, the supervised TAS outperforms the **CCM** with the same features. This seems to indicate that TAS

Object Class	Supervised TAS AP	**CCM** P	TAS + **CCM** Regions
Cars	0.38	**0.50**	0.43
Pedestrians	0.37	**0.51**	0.36
Motorbikes	0.37	**0.63**	0.41
Boats	0.17	**0.35**	0.20
Sheep	0.42	0.41	**0.44**
Cows	0.50	**0.55**	0.50

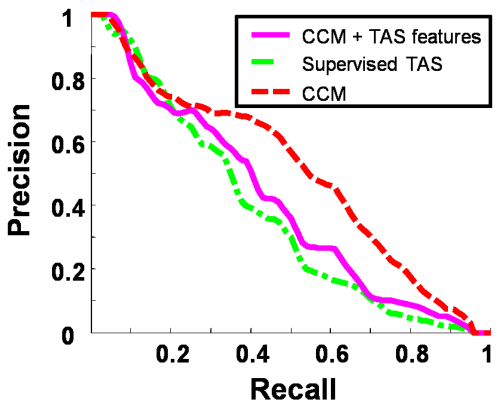
Table 5.4: Average precision detection scores for our six DS1 classes, comparing the hybrid approach that uses the **CCM** region labeler together with the TAS model to the supervised TAS and **CCM** approaches.

is (at least) as good as **CCM**s with this set of features, but still not as good as **CCM** with the features of Chapter 4.

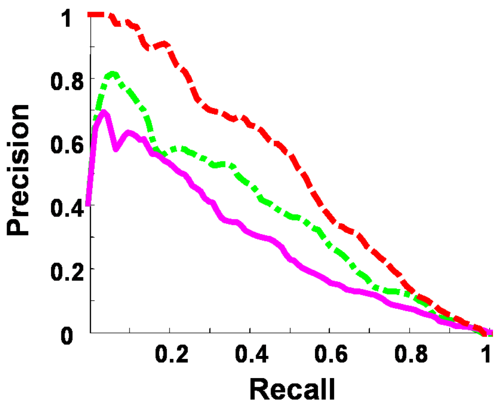
The final source of the gap is how the relationships are encoded. In TAS, the variables are directly coupled through CPDs for the relationship variables. In CCMs, the related regions are aggregated into various statistics. For example, for many of the objects, one of the most salient features in the **CCM** detector is the percent of 'foreground' in the object's bounding box. While 100% foreground likely indicates that the bounding box is too small (it is entirely contained in the foreground), a number between 60-80% makes it very likely that the bounding box contains a single monolithic object. Unfortunately, this cannot be captured by TAS as currently formulated.

Based on the above analysis it seems that the aggregation of the region labels and the statistics derived from this process produces the biggest context contribution. Unfortunately, it is not straightforward how to include these statistics in the TAS model. Furthermore, while computing a slew of features appears to allow for better precision, it does not provide insight into the probabilistic nature of the contextual signal, which is available in the TAS model.

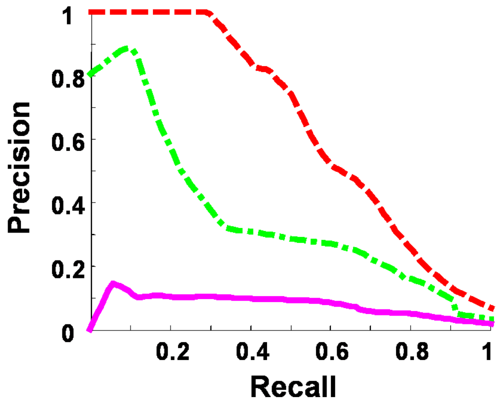
From a broader perspective, because the signals captured by the two methods are different (appearance similarity vs. semantic similarity), and because both provided improvements in the models' ability to detect objects, perhaps the best model would capture both of these signals. In one sense, the **CCM** paradigm of semantic context



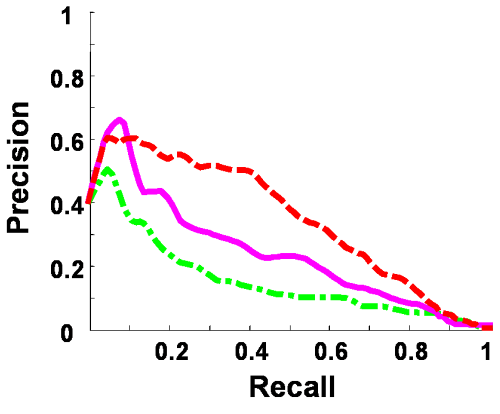
(a) Cars



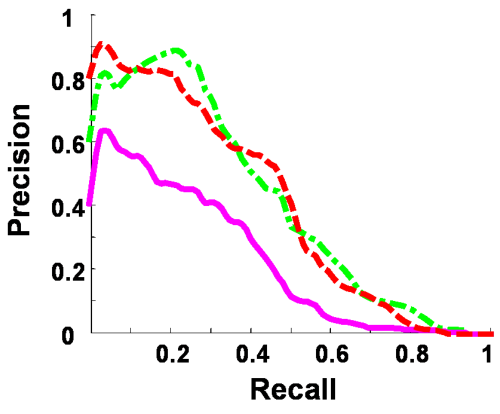
(b) Pedestrians



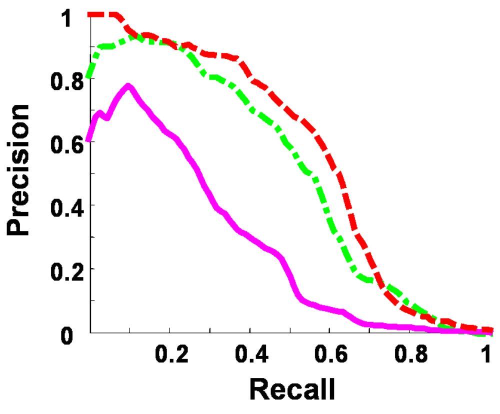
(c) Motorcycles



(d) Boats



(e) Sheep



(f) Cows

Figure 5.15: Comparison of detection results on the DS1 dataset of Chapter 4 between the original **CCM** method, the supervised TAS method, and **CCM** detection with features derived from the TAS relationships.

is correct, because, for instance, cars occur on roads, not on patches of gray color. However, in order to leverage this, we require a significant number of images annotated with groundtruth region labels. On the other hand, with enough data, a TAS-like approach might be able to capture most of this signal. Even though the appearance of roads, for example, is multi-modal, given enough examples our models can potentially learn contextual clusters for each type of road.

Further exploration of the tradeoffs between these methods and the possibility of a new approach which takes the best parts of each is an exciting and important future research direction.

5.5 Related Work

The role of context in object recognition has become an important topic, due both to the psychological basis of context in the human visual system [97] and to the striking algorithmic improvements that “visual context” has provided [125].

The word “context” has been attached to many different ideas. One of the simplest forms is co-occurrence context. The work of Rabinovich et al. [103] demonstrates the use of this context, where the presence of a certain object class in an image probabilistically influences the presence of a second class. The context of Torralba et al. [125] assumes that certain objects occur more frequently in certain rooms, as monitors tend to occur in offices. While these methods achieve excellent results when many different object classes are labeled per image, they are unable to leverage unsupervised data for contextual object recognition.

In addition to co-occurrence context, many approaches take into account the spatial relationships between objects. At the descriptor level, Wolf et al. [137] detect objects using a descriptor with a large capture range, allowing the detection of the object to be influenced by surrounding image features. Because these methods use only the raw features, however, they cannot obtain a holistic view of an entire scene. This is analogous to addressing image segmentation using a wider feature window rather than by using a Markov random field (MRF). Similarly, Fink and Perona [51] use the output of boosted detectors for other classes as additional features for the detection of a given class. This allows the inclusion of signal beyond the raw features, but requires that all "parts" of a scene be supervised. Murphy et al. [93] use a global feature known as the "gist" to learn statistical priors on the locations of objects within the context of the specific scene. The gist descriptor is excellent at predicting large structures in the scene, but cannot handle the local interactions present in the satellite data, for example.

Another approach to modeling spatial relationships is to use a Markov Random Field (MRF) or variant (CRF,DRF) [76, 20] to encode the preferences for certain spatial relationships. These techniques offer a great deal of flexibility in the formulation of the affinity function and all the standard benefits of a graphical model formulation (e.g., well-known learning and inference techniques). Singhal et al. [120] also use similar concepts to the MRF formulation to aggregate decisions across the image. These methods, however, suffer from two drawbacks. First, they tend to require a large amount of annotation in the training set. Second, they put things and stuff on the same footing, representing both as "sites" in the MRF. Our method requires less annotation and allows detections and image regions to be represented in their (different) natural spaces.

Perhaps the most ambitious attempts to use context involves the attempt to model the scene of an image holistically. Torralba [126], for instance, uses global image features to "prime" the detector with the likely presence/absence of objects, the likely locations, and the likely scales. The work of Hoiem and Efros [70] takes this one level further by explicitly modeling the 3D layout of the scene. This allows the natural use of scale and location constraints (e.g., things closer to the camera are larger). Their approach, however, is tailored to street scenes, and requires domain-specific modeling. The specific form of their priors would be useless in the case of satellite images, for example.

5.6 Discussion

In this chapter, we have presented the TAS model, a probabilistic framework that captures the contextual information between “stuff” and “things”, by linking discriminative detection of objects with unsupervised clustering of image regions. Importantly, the method does not require extensive labeling of image regions; standard labeling of object bounding boxes suffices for learning a model of the appearance of stuff regions and their contextual cues. We have demonstrated that the TAS model improves the performance even of strong base classifiers, including one of the top performing detectors in the PASCAL challenge.

The flexibility of the TAS model provides several important benefits. The model can accommodate almost any choice of object detector that produces a score for candidate windows. It is also flexible to any generative model over any type of region features. For instance, we might pre-cluster the regions into visual words, and then use a multinomial distribution over these words [5]. Additionally, because our model discovers which relationships to use, our method has the ability to discover spatial interactions that are not already known to the modeler. Indeed, automated structure-learning such as the one we employ here can provide a valuable complement to the laborious process of manual feature construction that is necessary for engineering a computer vision system.

Because the image region clusters are learned in an unsupervised fashion, they are able to capture a wide range of possible concepts. While a human might label the regions in one way (say trees and buildings), the automatic learning procedure might find a more contextually relevant grouping. For instance, in the case of our satellite data, we saw in Figure 5.10 that our learning procedure chose to split the roads into two clusters. Cluster #7 represents roads in shadow, while cluster #8 represents clusters in sunlight. It turns out that in our data, cars are slightly less likely to occur in shaded roads, which is indicated by the lower odds-ratio for cluster #7. In this case the model has made a contextual grouping that is probably different from what a person would choose.

As discussed in Section 5.5, recent work has amply demonstrated the importance of context in computer vision. The context modeled by the TAS framework is a natural complement for many of the other types of context in the literature. In particular, while many other forms of context can relate known objects that have been labeled in the data, our model can extract the signals present in the unlabeled part of the data. However, a major limitation of the TAS model is that it captures only 2D context. This issue also affects our ability to determine the appropriate scale for the contextual relationships. It would be interesting to integrate a TAS-like definition of context into an approach that attempts some level of 3D reconstruction, such as the work of Hoiem and Efros [70] or of Saxena et al. [111], allowing us to utilize 3D context and address the issue of scale.

In addition, while the relative locations of objects is an important characteristic of a scene, much of the information is also present in the actions, poses, or articulations of the objects. Because both the TAS model and the **CCM** models consider only the class and rough bounding box of the monolithic objects, they will generally have no ability to capture these properties. In the next chapter we explore models for classifying these characteristics of objects by detecting the object's outline.

Chapter 6

Descriptive Querying of Images

The basic questions about objects that we have considered thus far, such as "What is it?" (categorization) and "Where is it?" (detection), are central to machine vision and have received most of the attention in recent years. In many cases, however, we are also interested in more refined descriptive questions about objects. We define a descriptive query to be one that considers characteristics of an object that vary between instances within the object category. Examples include questions such as "Is the giraffe standing upright or bending over to drink?", "Is the cheetah running?", or "Find me all lamps that have a beige, rectangular lampshade." These questions relate to the object's pose, articulation, or localized appearance.

In principle, it is possible to convert such descriptive queries into discriminative classification tasks given an appropriately labeled training set. Once we have done this, we can use the methods of the previous chapters to detect or classify these objects. In order to take such an approach, however, we would need to construct a specialized training set (often a large one) for each descriptive question we want to answer, at significant human effort. Intuitively, it seems that we should be able to avoid this requirement. After all, to a person, we can convey the difference between a standing and bending giraffe using one or two training instances of each. The key, of course, is that the person has a good representational model of what giraffes look like, one in which the salient differences between the different descriptive labels are easily captured and learned.

In this chapter, we focus on the aspect of object shape, an important characteristic of many objects that can be used as the basis for many descriptive distinctions [43], including the examples described above. We address the dual goals of finding *precise*, *corresponded* localizations of object classes in cluttered images while allowing for large deformations, and answering descriptive queries based on the discovered localizations. We introduce a method that automatically learns a probabilistic characterization of an object class shape and appearance, allowing instances in that class to be encoded in this new representational space. We provide an algorithm for converting new test images containing the object into this representation, and show that this ability allows us to answer difficult descriptive questions, such as the ones above, using a very small number of training instances. Importantly, the specific descriptive questions are not known at the time the probabilistic model is learned, and the same model can be used for answering multiple questions.

Concretely, we propose an approach called LOOPS: **L**ocalizing **O**bject **O**utlines using **P**robabilistic **S**hape. Our LOOPS model combines two components: a model of the object's deformable shape, encoded as a joint probability distribution over the positions of a set of automatically selected *landmark* points on the outline of the object; and a set of appearance-based boosted detectors that are aimed at individually localizing each of these landmarks. Both components are automatically learned from a set of labeled training contours, of the type obtained from the LabelMe dataset (*http://labelme.csail.mit.edu*). The location of these landmarks in an image, plus the landmark-localized appearance, provide an alternative representation from which many descriptive tasks can be easily answered. The primary technical challenge, which we address in this chapter, is to correspond this model to a novel image that often contains a large degree of clutter and object deformation or articulation.

Contour-based methods such as active shape/appearance models (AAMs) [23, 116] were also developed with the goal of localizing an object shape model to an image. However, these methods typically require good initial guesses and are applied to images with significantly less clutter than real-life photographs. As a result, AAMs have not been successfully used for class-level object recognition/analysis. Some works use some form of geometry as a means toward an end goal of object classification or detection [45, 68, 99, 118]). Since, for example, a misplaced leg has a negligible effect on classification, these works neither attempt to optimize localization nor evaluate their ability to do so. Other works do attempt to accurately localize objects in cluttered photographs but only allow for relatively rigid configurations (e.g., [9, 49]), and cannot capture large deformations such as the articulation of the giraffe's neck (see Figure 6.1).

To allow robust localization of landmarks in cluttered images with significant deformation, we propose a hybrid method, which combines both discrete global and continuous local search steps. In the discrete phase, we use a discrete space defined by a limited set of candidate assignments for each landmark; we use a Markov random field (MRF) to define an energy function over this set of assignments, and effectively search this multi-modal, combinatorial space by applying state-of-the-art probabilistic inference methods over this MRF. This global search step allows us to find a rough but close-to-optimal solution, despite the many local optima resulting from the clutter and the large deformations allowed by our model. This localization can then be refined using a continuous hill-climbing approach, allowing a very good solution to be found without requiring a good initialization or multiple random restarts. Preliminary investigations showed that a simpler approach that does a purely local search, similar to the active appearance models of Cootes et al. [23], was unable to deal with the challenges of our data.

To evaluate the performance of our method, we consider a variety of object classes in cluttered images and explore the space of applications facilitated by the LOOPS model in two orthogonal directions. The first concerns the machine learning task: we present results for classification, search (ranking), and clustering. The second direction varies the components that are extracted from the LOOPS outlines for these tasks: we show examples that use the entire object shape, a sub-component of the object shape, and the appearance of a specific part of the object. Given enough task-specific data, each of these applications might be accomplished by other methods. However, we show that the LOOPS framework allows us to approach these tasks with a single class-based model, without the need for task-specific training.

In this chapter, we begin with a description of the LOOPS method, including the

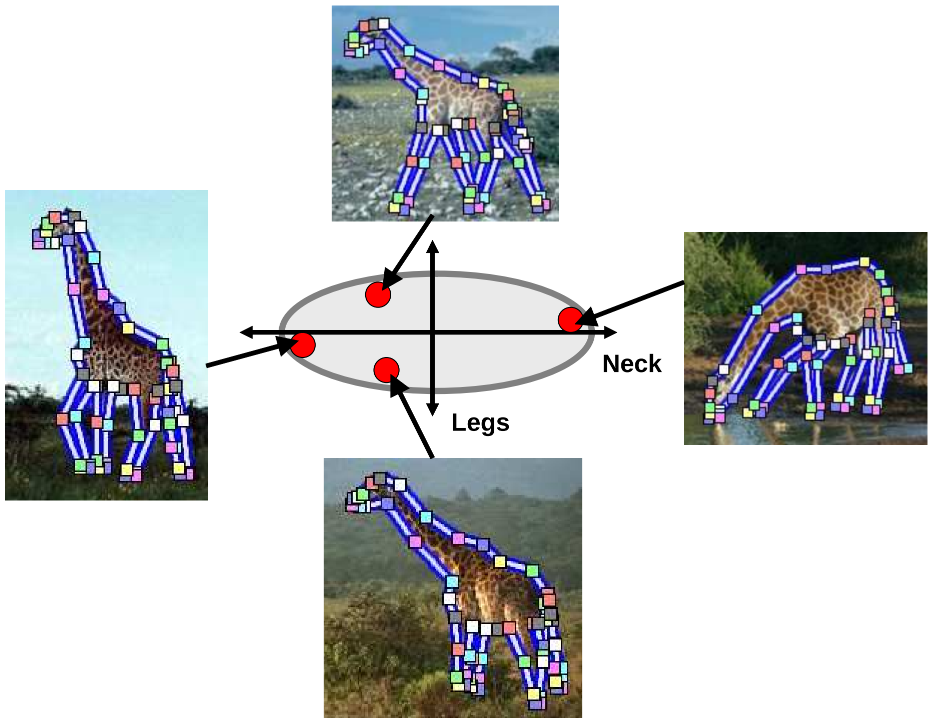


Figure 6.1: Example deformations of the giraffe object class. The axis shows the location of instances along the two principal components in our dataset. The horizontal axis corresponds to articulation of the neck, while the vertical axis corresponds to articulations of the legs. The ellipse indicates the set of instances within one standard deviation of the mean. We show four typical examples of giraffes that illustrate the outer extremes along the first two modes of variation.

automatic selection of landmarks in Section 6.2 and the learning of the shape and appearance models in Section 6.3. Following this, we describe how to localize a novel object in an image using the LOOPS model in Section 6.4. We then evaluate the LOOPS method as an object detector and as an object-based segmentation method in Section 6.5. Finally, in Section 6.6, we turn to the descriptive queries that are our goal.

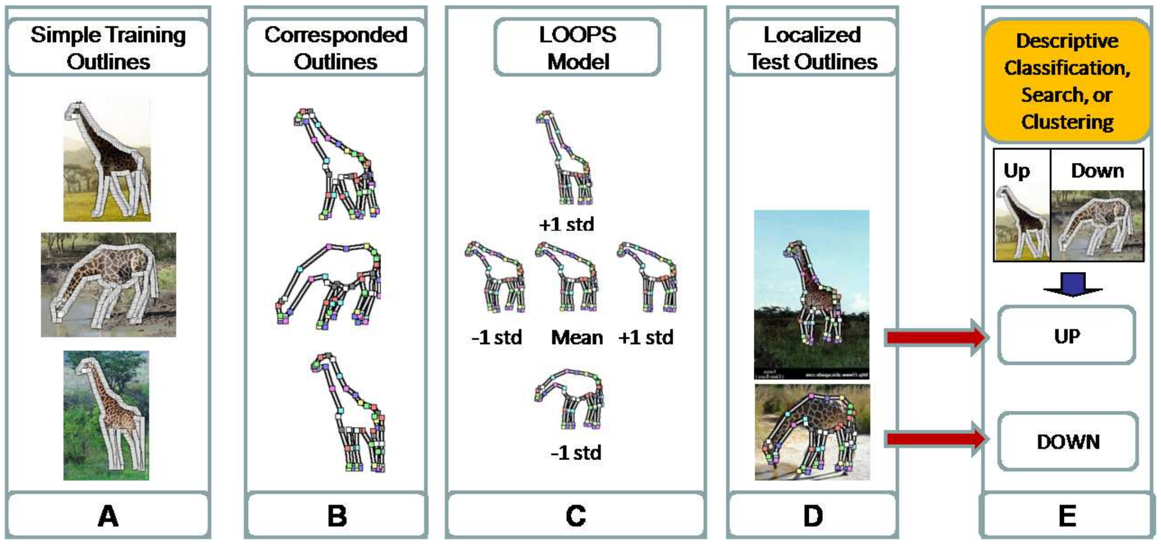


Figure 6.2: A flowchart depicting the stages of our LOOPS method.

6.1 Overview of the LOOPS Method

As discussed, having a representation of the constituent elements of an object should aid in answering descriptive questions. For example, to decide whether a giraffe is standing upright or bending down to drink, we might look at the characteristics of the model elements that correspond to the head, neck, or legs. In this section we briefly describe our representation over these automatically learned elements (landmarks). We also provide an overview of the LOOPS method for learning an object class model over these landmarks and using it to localize corresponded outlines in test images. A flowchart of our method is depicted in Figure 6.2.

We start by representing the shape of an object class via an ordered set of N landmark points that together constitute a piecewise linear contour (see Figure 6.1). It is important that these landmarks be *corresponded* across instances — landmark '17' in one instance should correspond to the same meaningful point (for example, the nose) as landmark '17' in another instance. Obtaining training instances labeled with corresponded landmarks, however, requires painstaking supervision. Instead, we would like to use simple outlines such as those in the LabelMe dataset (*http://labelme.csail.mit.edu*), which requires much less supervision and for which many examples are available from a variety of classes. We must therefore *automatically* augment the simple training outlines with a corresponded labeling. That is, we want to specify a relatively small number of landmarks that still represent the shape faithfully, and consistently position them on all training outlines. This stage transforms arbitrary outlines into useful training data as depicted in Figure 6.2(box B). We describe our method for performing this automatic correspondence in Section 6.2. Once we have our set of training outlines, each with N corresponded landmarks, we can construct a distribution of the geometry of the objects' outline as depicted in Figure 6.2(box C) and augment this with appearance based features to form a LOOPS model, as described in Section 6.3. With a model for an object class in hand, we now face the real computational challenge that our goal poses: how to localize the landmarks of the model in test images in the face of clutter and large deformations and articulations (box D in Figure 6.2). Our solution involves a two-stage scheme: we first use state-of-the-art inference from the field of graphical models to achieve a coarse solution in a discretized search space, and then refine this solution using greedy optimization. The localization of outlines in test images is described in detail in Section 6.4. Finally, once the localization is complete, we can readily perform a range of descriptive tasks (classification, ranking, clustering), based on the predicted location of landmarks in test images as well as appearance characteristics in the vicinity of those landmarks. We demonstrate how this is carried out for several descriptive tasks in Section 6.5.

6.2 Automatic Landmark Selection

In this section we describe our method for transforming simple outlines into corresponded outlines over a relatively small and consistent set of landmarks, as in Figure 6.3. Many works have studied the problem of automatic landmark selection for point distribution models (PDMs) for use in ASMs or AAMs. In particular, Hill and Taylor [67] developed a robust method based on optimization of a cost function over

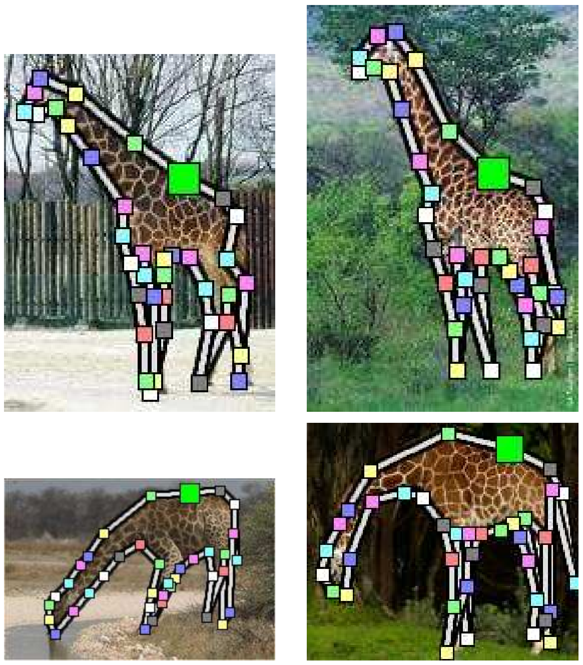


Figure 6.3: Example training instances with an outline defined by a piecewise linear contour connecting 42 landmarks. Note that the landmarks are not labeled in the training data but automatically learned using the method described in Section 6.2. These instances demonstrate the effectiveness of the correspondence method (e.g., the larger green landmark in the middle of the back is landmark #7 in all instances) as well as imperfections (e.g., misalignments near the feet). The most systematic problems occur in articulated instances, such as these. While the three landmarks at the lower back of the giraffe are consistent across these instances, some of the landmarks along the head must "slide" in order to keep the overall alignment correct.

a sparse polygonal representation of the outlines. Any robust method for correspondence between training outlines and salient landmark selection should work for our purposes. For completeness, we present our method, which is simple and was robust and accurate for all classes that we considered. Our method builds on an intuition similar to that of Hill and Taylor [67].

Intuitively, a good correspondence trades off between two objectives. The first is that corresponding points should be "equally spaced" around the boundary of the object, and the second is that the nonrigid transformation that aligns the landmark points should be small, in some meaningful sense. In our case, we achieve the correspondence using a simple two-step process: find a correspondence between high-resolution outlines then prune down to a small number of "salient" landmarks.

6.2.1 Arc-length Correspondence

Our correspondence algorithm searches for a correspondence between pairs of instances that achieves the lowest cost. Suppose we have a candidate correspondence between two outlines, represented by the vectors of points $\mathbf{O}^{(1)}$ and $\mathbf{O}^{(2)}$ (vectors are constructed by stacking the x and y coordinates of the landmarks in order around the contour). We first analytically determine the affine transformation $\mathbf{T}$ of $\mathbf{O}^{(2)}$ that minimizes its least-mean-square distance to $\mathbf{O}^{(1)}$. Applying $\mathbf{T}$ to each landmark in $\mathbf{O}^{(1)}$ produces the vector of landmarks denoted by $\tilde{\mathbf{O}}^{(2)}$. Our score penalizes changes in the offset vectors between distant landmarks on these two outlines. In particular, we compute our cost as:

$$Cost(\mathbf{O}^{(1)}, \tilde{\mathbf{O}}^{(2)}) = \sum_{i,j} \|\delta_{ij}^{(1)} - \tilde{\delta}_{ij}^{(2)}\|^2$$

where $\delta_{ij}^{(m)}$ is the vector offset between landmark i and landmark j on contour m. Landmarks i, j are pairs of landmarks that include each landmark i, together with the landmark j that has largest geodesic distance (distance along the first contour $\mathbf{O}^{(1)}$), from i. We note that our correspondence is scale-invariant since the target contour is affine transformed before the cost is computed.

In our search for the best correspondence according to this score, we leverage the fact that our outlines are ordered one-dimensionally. We begin by sampling a large number (500) of equally spaced landmarks along each of the training outlines. We fix a "base" contour (randomly selected), and a landmark numbering scheme for that base, by choosing a random landmark to serve as 0, and a direction to count in. We can now correspond the entire dataset by corresponding each "target" outline to our base. Figure 6.4 shows the score assigned by our algorithm for two sample correspondences from the target image to the base image.

Because we are dealing with outlines, or points that lie on a one-dimensional surface, we can define a fixed ordering of these points around the outline (clockwise or counterclockwise). Using this ordering assumption and our fixed choice of landmarks (the 500 equally spaced points), for each pair of contours there are only 500 possible

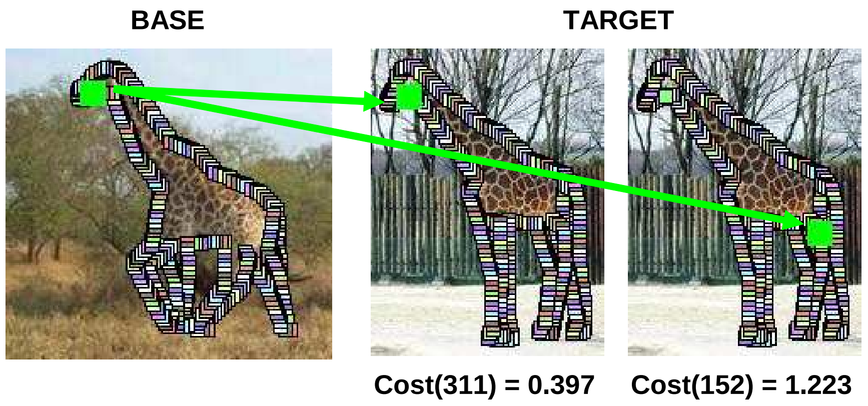


Figure 6.4: Our arc-length correspondence procedure. (a) The first outline, $\mathbf{O}^{(1)}$, with landmark 0 marked. (b,c) The second outline, $\mathbf{O}^{(2)}$, with two different choices for landmark 0, along with the correspondence cost for that choice.

correspondences. This can be seen because landmark 0 in the base outline only has 500 choices for a corresponding landmark in each target outline. We can thus simply compute the cost for all 500 choices, and select the best one. Once we have selected the correspondence for each target contour, we have a fully corresponded dataset.

The entire process is simple and takes less than a minute to correspond each target instance to the base. In order to increase the robustness of our algorithm further, we repeat this process for a random selection of 10 base instances, and choose the base instance that produces the lowest average correspondence cost over the rest of the training set. This entire process takes only a few minutes.

Note that one of the primary assumptions in this method is that the desired corresponding points are "equally spread" across the outline. Clearly this is strictly not true, as a longer neck will require that the outline distance between the front of the neck and back of the neck is a higher percentage of the total outline. However, in our data this method was sufficient for learning our LOOPS model. A more sophisticated variant of our method might borrow some ideas from the Dynamic Time-Warping approach of Salvador and Chan [110]. In a correspondence method based on this idea, we might allow individual landmarks to slide along the contour in certain instances to improve a score that rewards similarity of curvature rather than arc-length.

6.2.2 Landmark pruning

The next step is to reduce the number of landmarks used to represent each outline in a way that takes into account *all* training contours. Our goal is to remove points that contribute little to the outline of any instance. Toward this end, we greedily prune landmarks one at a time. At any given time in this process, each landmark can be scored by the error its removal would introduce into the dataset. Suppose that our candidate for removal is landmark i at location l_i. If i is removed from instance m, the outline in the vicinity of landmark i will be approximated by the line segment between landmarks $i-1$ and $i+1$ at locations $l_{i-1}^{(m)}$ and $l_{i+1}^{(m)}$. We define $dist_{\mathbf{O}^{(m)}}(l_{i-1}^{(m)}, l_{i+1}^{(m)})$ to be the mean segment-to-outline squared distance, and let the cost of removing landmark i be the average of these distances across all M instances in the entire dataset:

$$C_i = \frac{1}{M}\sum_m dist_{\mathbf{O}^{(m)}}(l_{i-1}^{(m)}, l_{i+1}^{(m)}).$$

At each step, we remove the landmark whose cost is lowest, and terminate the process when the cost of the next removal is above a fixed threshold (2 pixels^2).

Figure 6.3 shows an example of the correspondence of giraffe outlines. Note that the fully automatic choice of landmarks is both sparse and similar to what a human labeler might choose. In the next section we show how to use corresponded outlines to learn a shape and appearance model that will later be used (see Section 6.4) to detect objects and precisely localize these landmarks in cluttered images.

6.3 The LOOPS Model

Once we have corresponded the training outlines so that each is specified by N (corresponded) landmarks, we are ready to construct a distribution over possible assignments of these landmarks to pixels in an image. Toward this end, the LOOPS model combines two components: an explicit representation of the object's shape (2D silhouette), and a set of image-based features. We define the shape of a class of objects via the locations of the N object landmarks, each of which is assigned to one of the pixels in the image. We represent such an assignment as a $2N$ vector of image coordinates which we denote by $\mathbf{L}$. Using the language of Markov random fields [101], the LOOPS model defines a conditional probability distribution over $\mathbf{L}$:

$$\begin{aligned}
P(\mathbf{L} \mid \mathcal{I}, \mathbf{w}, \mu, \Sigma) = & \\
\tfrac{1}{Z(\mathcal{I})} P_{\text{Shape}}(\mathbf{L}; \mu, \Sigma) \quad & \prod_i \exp\left(w_i F_i^{\text{det}}(l_i; \mathcal{I})\right) \\
& \prod_{i,j} \exp\left(w_{ij} F_{ij}^{\text{grad}}(l_i, l_j; \mathcal{I})\right),
\end{aligned} \tag{6.1}$$

where $\mu, \Sigma, \mathbf{w}$ are model parameters, and i and j index landmarks of the model. P_{Shape} encodes the distribution over the object shape (described in Section 6.3.1), $F^{\text{det}}(l_i)$ is a landmark specific detector (described in Section 6.3.2, and $F_{ij}^{\text{grad}}(l_i, l_j; \mathcal{I})$ encodes a preference for aligning outline segments along image edges (described in Section 6.3.3).

The shape model and the detector features are learned independently, as shown in Figure 6.2(b). Below, we describe these features and how they are learned. The weights $\mathbf{w}$ are set as follows: $w_i = 5$, $w_{ij} = 1$, for all i, j. In principle, we could learn these weights from data, for example using a standard conjugate gradient approach. This process, however, requires an expensive inference step at each iteration. Our preliminary experiments indicated that our outlining results are relatively robust to a range of these weights and that learning the weights provides no clear benefit. We therefore avoid this computational burden and simply use a fixed setting of the weights.

We note that our MRF formulation is quite general, and allows for both flexible weighting of the features, and for the incorporation of additional features. For instance, we might want to capture the notion that internal line segments (lines entirely contained within the object) should have low color variability. This can naturally be posed as a pairwise feature over landmarks on opposite sides of the object.

6.3.1 Object Shape Model

We model the shape component of Eq. (6.1) as a multivariate Gaussian distribution over landmark locations with mean μ and covariance $\mathbf{\Sigma}$. The Gaussian parametric form has many attractive properties, and has been used successfully to model shape distributions in a variety of applications (e.g., [24, 2]). In our context, one particularly useful property is that the Gaussian distribution decomposes into a product of quadratic terms over pairs of variables. Defining $\mathbf{\Omega} = \mathbf{\Sigma}^{-1}$ to be the inverse covariance or "precision" matrix, we decompose this distribution according to:

$$\begin{aligned} P_{\text{Shape}}(\mathbf{L} \mid \mu, \mathbf{\Sigma}) &= \frac{1}{Z}\exp\left(-\frac{1}{2}(\mathbf{x}-\mu)\mathbf{\Omega}(\mathbf{x}-\mu)\right) \\ &= \frac{1}{Z}\prod_{i,j}\exp\left(-\frac{1}{2}(x_i-\mu_i)\mathbf{\Omega}_{ij}(x_j-\mu_j)\right) \\ &= \frac{1}{Z}\prod_{i,j}\phi_{i,j}(x_i, x_j; \mu, \mathbf{\Omega}), \end{aligned} \tag{6.2}$$

where Z is the normalization factor for the Gaussian density. We can see from Eq. (6.2) that we can specify potentials $\phi_{i,j}$ over only singletons and pairs of variables and still manage to reconstruct the full likelihood of the shape. This allows Eq. (6.1) to take an appealing form in which all terms are defined over at most a pair of variables.

As we discuss below in Section 6.4, the procedure to locate the model landmarks in an image first involves discrete global inference using the LOOPS model, followed by a refinement stage that takes local steps. Even if we limit ourselves to pairwise terms, performing discrete inference in a densely connected MRF may be computationally impractical. Unfortunately, a general multivariate Gaussian includes pairwise terms between all landmarks. Thus, during the discrete inference stage, we limit the number of pairwise elements by approximating our shape distribution with a sparse multivariate Gaussian. During the final refinement stage, we use the full multivariate Gaussian distribution.

The sparse shape approximation can be selected in various ways, trading off accuracy of the distribution with computational resources. In our implementation, we include only two types of terms: terms correlating neighboring landmarks along the outline, and a linear number (one per landmark) of terms encoding long-range dependencies that promote stability of the shape. We greedily select the latter terms over pairs of landmarks for which the relative location has the least variance (averaged across the x and y coordinates).

Estimation of the maximum likelihood parameters of the full Gaussian distribution may be solved analytically using the corresponded training instances. If we desire a sparse Gaussian distribution, however, there are multiple options for which approximation to use. As described in Chapter 2, we use a standard iterative projected gradient approach ([15]) to minimize the Kullback-Leibler divergence [25] between the sparse distribution and the full maximum likelihood distribution:

$$(\mu_S, \mathbf{\Omega}_S) = \text{argmin}_{\mu, \mathbf{\Omega} \in \Delta} KL\left(\mathcal{N}(\mu, \mathbf{\Omega}) \| \mathcal{N}_D(\mu_D, \mathbf{\Omega}_D)\right)$$

where μ_D and $\mathbf{\Omega}_D$ are the (dense) ML parameters, and Δ is the set of all inverse covariances that meet the constraints defined by our choice of included edges. Note that a removal of a potential (edge) between a pair of landmarks is equivalent to constraining the corresponding entry in the precision matrix $\mathbf{\Omega}$ to be zero.

6.3.2 Landmark Detector Features

To construct our detector features F^{det}, we build on the demonstrated efficacy of discriminative methods in identifying salient regions (parts) of objects (e.g., [45, 68, 127, 99]). The extensive work in this area suggests that the best results are obtained by combining a large number of features. However, incorporating all of these features directly into Eq. (6.1) is problematic because setting such a large number of weights corresponding to these features requires a significant amount of tuning. If we chose to learn the weights, this problem would not be alleviated: training a conditional MRF requires that we run inference multiple times as part of the gradient descent process; models with more parameters require many more iterations of gradient descent to converge, making the learning cost prohibitive.

Our strategy builds on the success of boosting in state-of-the-art object detection methods [99, 127]. Specifically, we use boosting to learn a strong detector (classifier), H_i for each landmark i. We then define the feature value in the conditional MRF for the assignment of landmark i to pixel l_i to be:

$$F_i^{\text{det}}(l_i; \mathcal{I}) = H_i(l_i).$$

Any set of image features can be used in this approach; we use features that are based on our shape model as well as other features that have proven useful for the task of object detection (see Figure 6.5 for examples of each):

- **Shape Templates**: The first feature we consider is a feature aimed at capturing the shape directly. We use a template of the edge points surrounding the landmark. While a common approach is to consider all edge points in a ball or a patch centered at the landmark, we construct our template using regularly sampled points along the contour, up to some "geodesic" distance away from the landmark (measured along the contour itself). Our shape template is defined by a set $\mathbf{D}$ of offsets $\mathbf{d}_i = (d_{x,i}, d_{y,i})$ from the landmark to each sampled point along the contour. In Figure 6.5(top left) we show a shape template for the nose of the airplane in red. In an ideal image of the object, we expect each point along the outline to generate a Canny-detected edge pixel in the image. In reality, we have to allow for missing edge pixels and edge detection noise. Because we expect the edges in the image to correspond only roughly to the object outline, we match a shape template to actual edges in the image using chamfer distance matching [14]. This feature is entirely shape dependent and can only be used in the case where training outlines are provided.

- **Boundary Fragments**: This feature is made of randomly selected edge chains (contiguous groups of edge pixels) within the bounding box of the training objects, and are matched using chamfer distance. We construct these using a protocol similar to Opelt et al. [99], except that we neither cluster fragments nor combine pairs of fragments into a single feature.

- **Filter Response Patches**: This feature is represented by a patch from a filtered version of the image (filters include bars and oriented Gaussians). Patches are randomly extracted from within the object bounding boxes in the training images. The patch is matched to a new image using normalized cross-correlation. We construct the patches similarly to Torralba et al. [127].

- **SIFT Descriptors**: Interesting keypoints are extracted from each image, and a SIFT descriptor [85] is computed for each keypoint. In order to limit the number of SIFT features, each descriptor is scored based on how common it is in the training set, and the best scoring descriptors are kept. Some properties of SIFT features suggest that they are ill-suited to this use. While the features are rotation invariant, our objects do not undergo significant rotation, so this should only hurt us. Also, the SIFT keypoints tend to occur at the boundary of the objects, so they will also learn something about the background, which may not be desirable. However, despite these shortcomings, this feature was included and found to improve our overall results.

Each landmark detector is trained by constructing a strong boosted classifier from a set of the weak single feature detectors described above. To build such detectors, we rely on boosting and adapt the protocol of Torralba et al. [127]. We now provide a brief description of the essentials of the process.

Boosting provides a straightforward way to perform feature selection and learn additive classifiers of the form

$$H_i(p) = \sum_{t=1}^{T} \alpha_t h_i^t(p),$$

where each $h_i^t(p)$ is a weak feature detector for landmark i (described below), T is the number of boosting rounds, and $H_i(p)$ is a strong classifier whose output is proportional to the log-odds of landmark i being at pixel p.

A weak detector h_i is a feature of one of the types listed above (e.g., a filter response patch) providing information on the location of landmark i. It is defined in terms of a feature vector of the relevant type and an offset v between the pixel p at

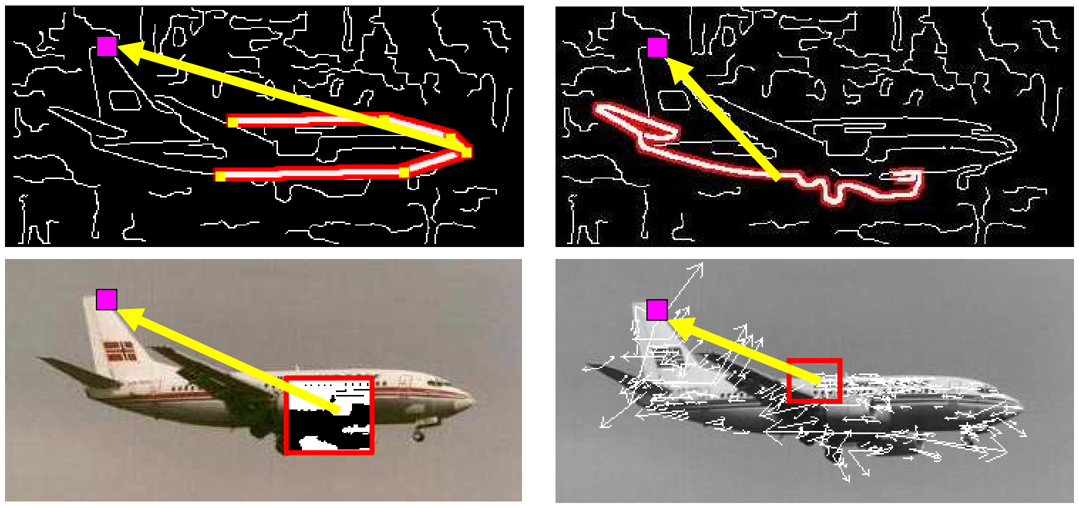


Figure 6.5: Examples of weak detectors (clockwise from the top left): a shape template feature, representing a segment of the mean contour near the nose of the plane; a boundary fragment generated from an edge chain; a SIFT feature; a filter response patch extracted from a random location within the object bounding box. All features are shown along with their offset vector (yellow arrow) that "votes" for the location of a landmark at the tail of the airplane (pink square).

which the feature is computed and the landmark position. Thus, if a weak detector h_i produces a high response on a test image at pixel p, then this detector will "vote" for the presence of the landmark at $p + v$. Figure 6.5 illustrates each type of weak detector together with a sample landmark offset.

A weak detector is applied to an image $\mathcal{I}$ in the following manner. First, the detector's feature is evaluated for each pixel in the image and shifted by the offset for the corresponding landmark i to produce the response image R_i. For our patch features, we make R_i sparse by zeroing out all but the top K (50) responses and blur it to provide a softer voting mechanism, in which small perturbations are not penalized. The sparsification of the response map allows only the most confident feature matches. A more detailed explanation of this process can be found in Murphy et al. [94]. In all experiments, we blur the response image using a Gaussian blur with a standard deviation of 10 pixels. We now let $h_i(p)$ be a regression stump over the response $R_i(p)$ that indicates the probability of the landmark occurring at this pixel.

Once we have computed the responses for each weak detector, we learn the strong detector $H_i(p)$ using Real Adaboost [112] with regression stumps (the h_i's) as in [127, 94]. For each landmark i, our positive training set consists of the response vectors corresponding to assignments of that landmark in each of the training instances. In practice, because our landmark locations are noisy (due to labeling and correspondence error), we add vectors from a small (3x3) grid of pixels around the true assignment. Our negative set is constructed from the responses at random pixels that are at least a minimum distance (10 pixels) away from the true answer.

6.3.3 Gradient Features

Our model also includes terms that estimate the likelihood of the edges connecting the landmarks. In general, we expect that the image will contain high gradient magnitudes at the object boundary. For neighboring landmarks i and j with pixel assignments (l_i, l_j), we define the feature:

$$F_{ij}^{\text{grad}}(l_i, l_j; \mathcal{I}) = \sum_{r \in \overline{l_i l_j}} |\mathbf{g}(r)^T \mathbf{n}(l_i, l_j)|,$$

where r varies over all of the points along the line segment from l_i to l_j, $\mathbf{g}(r)$ is the image gradient at point r, and $\mathbf{n}(l_i, l_j)$ is the normal to the edge (between l_i and l_j). High values of the feature encourages the boundary of the object to lie along segments of high gradient.

6.4 Localization of Object Classes

We now address our central computational challenge: assigning the landmarks of a LOOPS model to test image pixels while allowing for large deformations and articulations. Recall that the conditional MRF defines a distribution (Eq. (6.1)) over assignments of model landmarks to pixels. This allows us to outline objects by using probabilistic inference to find the most probable such assignment:

$$\mathbf{L}^* = \text{argmax}_{\mathbf{L}} P(\mathbf{L} \mid \mathcal{I}, \mathbf{w}).$$

Because, in principle, each landmark can be assigned to any pixel, finding $\mathbf{L}^*$ is computationally prohibitive. One option is to use an approach analogous to active shape models, using a greedy method to deform the model from a fixed starting point. However, unlike most applications of active shape/appearance models (e.g., [23]), our images have significant clutter, and such an approach will quickly get trapped in an inferior local maximum.

A possible solution to this problem is to consider a series of starting points. Figure 6.6 shows two example images, with the results of running a greedy shape search from a series of starting points. On the left, we see the result of starting from the assignment of each landmark to the pixel with the highest landmark detector score. Here the search is unable to escape the terrible original shape. In the next panel, we initialize our shape model from the mean shape at the center of the image, and search from there. In this case, we do okay on the elephant, which is indeed near the center of the image, but we fail on the giraffe, never moving the points sufficiently far over to the true object. In the third panel, we repeat this search from 20 random starting points of the mean. This time, we capture the body of the giraffe, but fail to correctly find the articulation of the head. The conclusion of this analysis was that such an approach requires a computationally prohibitive number of starting points. The final panel of Figure 6.6 shows the results of our inference procedure, which we will now describe.

To overcome the limitations of a greedy search, we propose an alternative two step method, depicted in Figure 6.7: we first approximate our problem and find a coarse solution using discrete inference; we then refine our solution using continuous optimization and the full objective defined by Eq. (6.1).

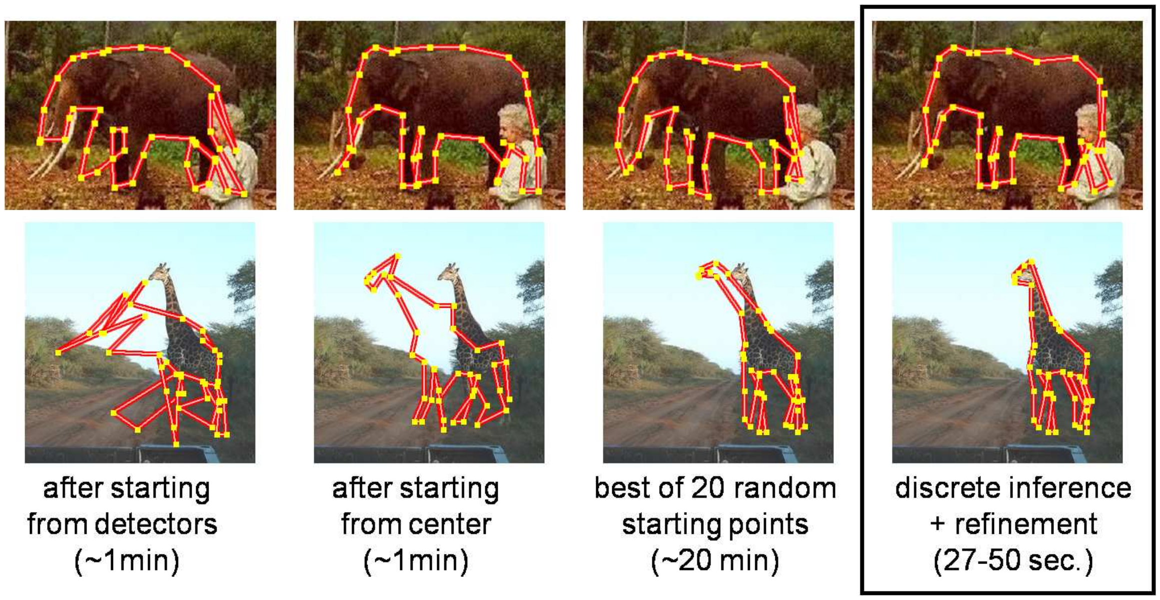

after starting from detectors (~1min)
after starting from center (~1min)
best of 20 random starting points (~20 min)
discrete inference + refinement (27-50 sec.)

Figure 6.6: Results of localizing objects using a greedy search procedure. On the left we initialize from the best detection for each landmark, and the shape never recovers to something reasonable during our search. In the next two panels, we start from the mean shape at the center, then at 20 random starting points. With enough starting points, we can eventually capture most of the shape. However, this is computationally prohibitive, and once you include articulation because basically intractable. On the right, we show the results of our LOOPS inference method.

6.4.1 Discrete Inference

Obviously, we cannot perform inference over the entire space as even a modestly sized image (200×300 pixels) results in a search space of size $N^{60,000}$ where N is the number of model landmarks. Thus, we start with pruning the domain of each landmark to a relatively small number of pixels. To do so both effectively and efficiently (without requiring inference), we first assume that landmarks will fall on "interesting" points in the image, and consider only points found by the SIFT interest operator [85]. We adjust the settings of the SIFT interest operator to produce between 1000-2000 descriptors per image (we set the number of scales to 10), allowing us to have a large number of locations to choose from. Following this step, we make use of the MRF appearance based feature functions F_i^{det} to identify the most promising

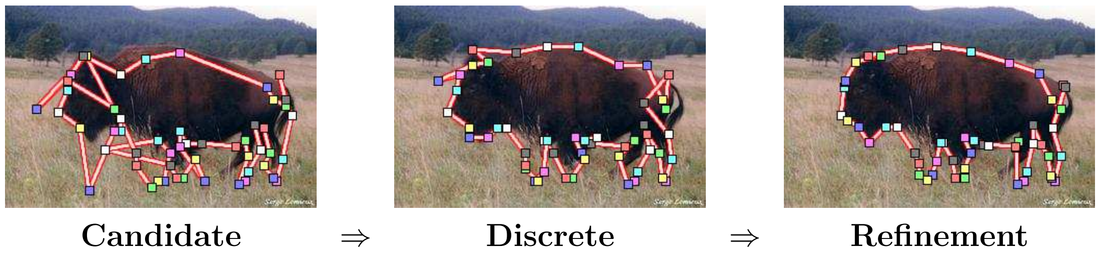


Figure 6.7: (a) An outline defined by the top detection for each landmark independently, (b) an outline inferred by inference over our discrete conditional MRF, and (c) a refined outline after coordinate-ascent optimization.

candidate pixel assignments for each landmark. While we cannot expect these features to identify the single correct pixel for each landmark, we might expect the correct answer to lie towards the higher end of the response values. Thus, for each landmark, we use the corresponding F_i^{det} to score all the SIFT keypoint pixels in the image and choose the top K local optima as candidate assignments. Figure 6.7(a) shows the top assignment for each landmark according to the detectors for a single example test image.

One important choice we face is the choice of K, the number of candidates to use. Figure 6.8 explores this choice for four object classes. We consider a landmark "found" in a set of K candidates if at least one of the candidates is within some threshold distance T of the true (hand-labeled) location. The graphs in Figure 6.8 plot the percent of "found" landmarks as a function of K candidates for four different object classes at three different selections for the threshold T. We measure the distance as a fraction of the object size (defined as the length of the bounding box diagonal). Thus $T = 1\%$ means that, for an object with a bounding box diagonal of 200 pixels, a landmark must be within 2 pixels of the correct location to be considered "found." These graphs show two important things. First, we can see that across these classes, we can generally find more than 90% of the landmarks if we use a threshold of 5%. This means that if we can select the best candidate, we are likely to get good localizations. Second, while increasing the number of candidates seems to help, most of the improvement is at the beginning, and for most classes we measured, selecting

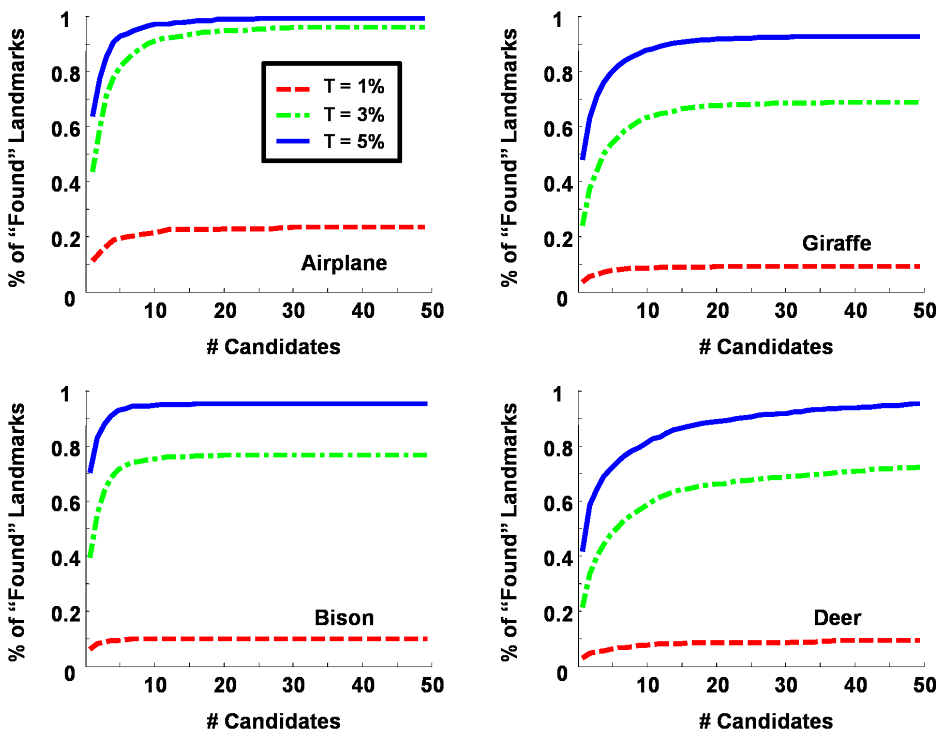


Figure 6.8: Graphs of the fraction of "found" landmarks for four classes, varying the number of candidates used (K). We can see that for these classes, we can expect to have a landmark within 5% of the correct location in the candidates more than 90% of the time. Furthermore, as these curves tend to level off relatively early, we can get a good accuracy from using only $K = 25$ candidates.

$K = 25$ gives a good tradeoff between complexity and accuracy. We therefore run all experiments below with $K = 25$.

Given a set of candidates for each landmark, we must now find a single, consistent joint assignment $\mathbf{L}^*$ to all landmarks together. However, even in this pruned space, the inference problem is quite daunting. Thus, we further approximate our objective by sparsifying the multivariate Gaussian shape distribution to include only terms corresponding to neighboring landmarks plus a linear number of terms (see Section 6.3.1). The only pairwise feature functions we use are over neighboring pairs of landmarks (as described in Section 6.3.3), which does not add to the density of the MRF construction, thus allowing the inference procedure to be tractable.

Finally, we find $\mathbf{L}^*$ by performing approximate max-product inference, using the empirically successful Residual Belief Propagation (RBP) of Elidan et al. [39]. Figure 6.7(b) shows an example of an outlines found after the discrete inference stage.

6.4.2 Refinement

Given the best coarse assignment $\mathbf{L}^*$ predicted in the discrete stage, we perform a refinement stage in which we reintroduce the entire pixel domain and use the full shape distribution. This allows us to more accurately adapt to the image statistics while also considering shapes that fit a better distribution than was available at the discrete stage. Refinement is accomplished using a greedy hill-climbing algorithm in which we iterate across each landmark, moving it to the best candidate location using one of two types of moves, while holding the other landmarks fixed. In a **local** move, each landmark can pick the best pixel location in a small window around its current location. In a **global** move, each landmark can move to its mean location given all the other landmark assignments. Determining the conditional mean location is straightforward given the Gaussian form of the joint shape model. If we seek the conditional mean of the location of landmark n, denoted x_n, given the locations of landmarks $1 \ldots n-1$, denoted by $\mathbf{x}^-$, we begin with the joint distribution:

$$\begin{bmatrix} \mathbf{x}^- \\ x_n \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu^- \\ \mu \end{bmatrix}, \begin{bmatrix} \Sigma^- & \beta_n^- \\ \beta_n^{-T} & \sigma_n^2 \end{bmatrix} \right).$$

In this case, the conditional mean is given by:

$$E[x_n \mid \mathbf{x}^-] = \mu + \beta_n^{-T} \left(\Sigma^-\right)^{-1} \left(\mathbf{x}^- - \mu^-\right).$$

Interestingly, in a typical refinement, the **global** moves dominate the early iterations, correcting large mistakes made by the discrete stage and that resulted in an unlikely shape. In the later iterations, **local** moves do most of the work by carefully adapting to the local image characteristics. Figure 6.7(c) shows an example of an outline found after the refinement stage.

6.5 Experimental Evaluation of LOOPS Localization

We begin our experimental evaluation with a quantitative evaluation of the localization ability of the LOOPS model. We begin with an exploration of the use of LOOPS as an object detector, and follow this with an analysis of the outline localization produced by the method. In the following section, we will demonstrate the range of capabilities of the LOOPS model by showing how these correspondences can be used for a series of descriptive tasks.

In all experiments, we trained our loops model on 20 randomly chosen training images and tested on the others. We report average results over 5 random train/test partitions. The features used to construct the boosted classifier for each landmark (see Section 6.3.2) include shape templates of lengths $3, 5, 7, 10, 15, 20, 30, 50, 75$ for each landmark as well as 400 boundary fragments, 400 SIFT features and 600 filter features that are shared between all landmarks.

6.5.1 Object Detection with the LOOPS Method

Intuitively, if the object is not in the image, it is unlikely that any landmark-to-pixel assignment will have a probable shape *and* rank high with the discriminative detectors. We therefore pose the problem of determining whether the object is present as a likelihood ratio test. Using our LOOPS model, we consider an object to be detected when:

$$\log \frac{P(\mathbf{L}^* \mid Obj = 1; \mathcal{I})}{P(\mathbf{L}^* \mid Obj = 0; \mathcal{I})} = \log \frac{\frac{1}{Z_1} P'(\mathbf{L}^* \mid Obj = 1; \mathcal{I})}{\frac{1}{Z_0} P'(\mathbf{L}^* \mid Obj = 0; \mathcal{I})} > 0.$$

where $P'() = Z \cdot P()$ denotes the unnormalized probability. Estimating this log-odds ratio appears to be intractable, as evaluating $P()$ requires that we compute the normalization constants Z_0 and Z_1, which involves summing over all pixels in the image. We overcome this using two assumptions. The first is that the denominator is uniform across all assignments of $\mathbf{L}$. Second, we further assume that Z_0 and Z_1

do not depend on the image size. In the case of discrete inference, this is reasonably given that we consider a fixed size domain (25 pixels) for each landmark independent of the image size. Using both assumptions, our log-odds ratio comparison becomes:

$$\log P'(\mathbf{L}^* \mid Obj = 1; \mathcal{I}) > C \tag{6.3}$$

where C is some constant. Thus, evaluating the hypothesis that the object appears in the image becomes a thresholding problem on the unnormalized probability of Eq. (6.1), which is easily evaluated.

For the detection task, we consider a set of object classes that represent a range of shape variability and difficulty. Specifically, we consider two groups of classes:

1. **Mammals**: Six classes from the dataset of Fink and Ullman [50] including bison, deer, elephant, giraffe, llama, and rhino. While no published detection results exist for these datasets, severe clutter, lack of color separability, and object articulation all make these classes significantly more challenging both for detection and accurate localization. All images in these datasets have been rescaled so that the diagonal of the object's bounding box is of length 200 pixels.

2. **ETHZ Shape classes**: The five classes from the ETHZ dataset of Ferrari et al. [47], which exhibit large scale variation and zero or multiple objects in some of the images. The objects in these images have bounding boxes that range in diagonal length from 140 pixels up to 756 pixels.

Mammals

For these datasets, we assume a single instance of the object class per image, and perform a LOOPS localization, as described in Section 6.4. We evaluate the resulting detections using the bounding box overlap metric of [127]. The overlap between two bounding boxes ($\mathbf{B}_1$ and $\mathbf{B}_2$) is defined as:

$$O(\mathbf{B}_1, \mathbf{B}_2) = \frac{Area(\mathbf{B}_1 \cap \mathbf{B}_2)}{Area(\mathbf{B}_1 \cup \mathbf{B}_2)}.$$

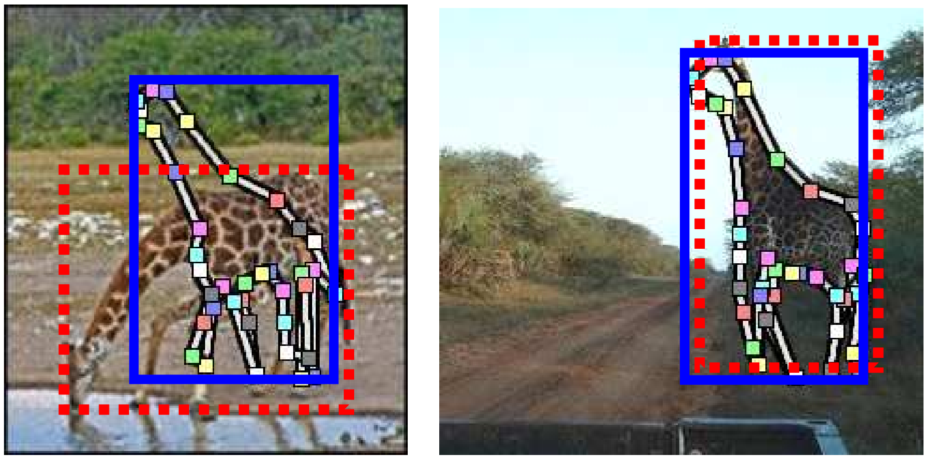

(a) overlap = 0.48 (b) overlap = 0.87

Figure 6.9: Example localizations for the giraffe class along with the overlap of the predicted localization relative to the human-labeled ground-truth. In the left image, LOOPS mistakenly produces an outline of a standing giraffe, and achieves an overlap score of 0.49. In the right image, we produce an accurate outline that is scored as a 0.87. We can see from this example that overlap scores near 0.9 are very good.

To get a sense of the overlap measure, Figure 6.9 shows two example detections from the giraffe class along with their overlap scores, relative to a human-labeled ground truth.

For a baseline comparison on these classes, we create a single boosted **Centroid** detector. This detector uses the same types of features as our landmark detectors, but all votes are for the centroid of the object as in Torralba et al. [127]. A bounding box based on the mean shape is placed around the detected centroid.

Figure 6.10 shows the mean of the overlap score between the ground truth and the **Centroid** or LOOPS outline for each of the Mammal classes. Not surprisingly, as all test images contain a single relatively large object, absolute overlap score at the bounding box level is relatively high. Still, the advantage of the bounding box predicted by LOOPS over the **Centroid** baseline is clear. We note that the continuous refinement step of our method did not seem to improve over the result of discrete inference on this metric. This is not surprising, as the continuous refinement generally makes only small local changes to the object shape; while these changes

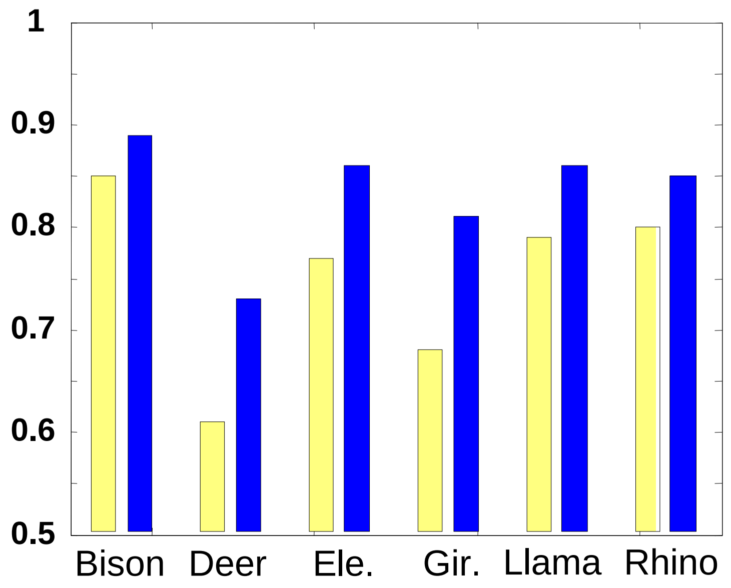


Figure 6.10: Mean overlap scores for the six mammal classes, comparing the **Centroid** method to the LOOPS approach. As demonstrated in Figure 6.9, overlap scores above 0.80 or so are visually very good. For all classes except Deer we achieve means at or above 0.8. In the Deer class, the foreground is often very similar to the background in texture and luminance. The blending in of the deer with the background makes this class clearly the most difficult of those used.

can greatly improve both the fine-scale landmark position and the appearance of the object, they often do not significantly change the bounding box. As we will see below, the continuous refinement does provide improvements in precise landmark-level localization.

ETHZ Shape Classes

In this dataset, there are 87 giraffe images, 40 apple logo images, 32 swan images, 48 mugs, and 48 bottle images. For each class, we consider all images from the other classes as the negative set. For these images, where objects appear at multiple scales, we use a simple yet effective search over scales, a common approach in the literature (e.g., [131]). Since our multivariate Gaussian shape model is able to capture scale deformations to a limited extent, we search over a small number (7) of image

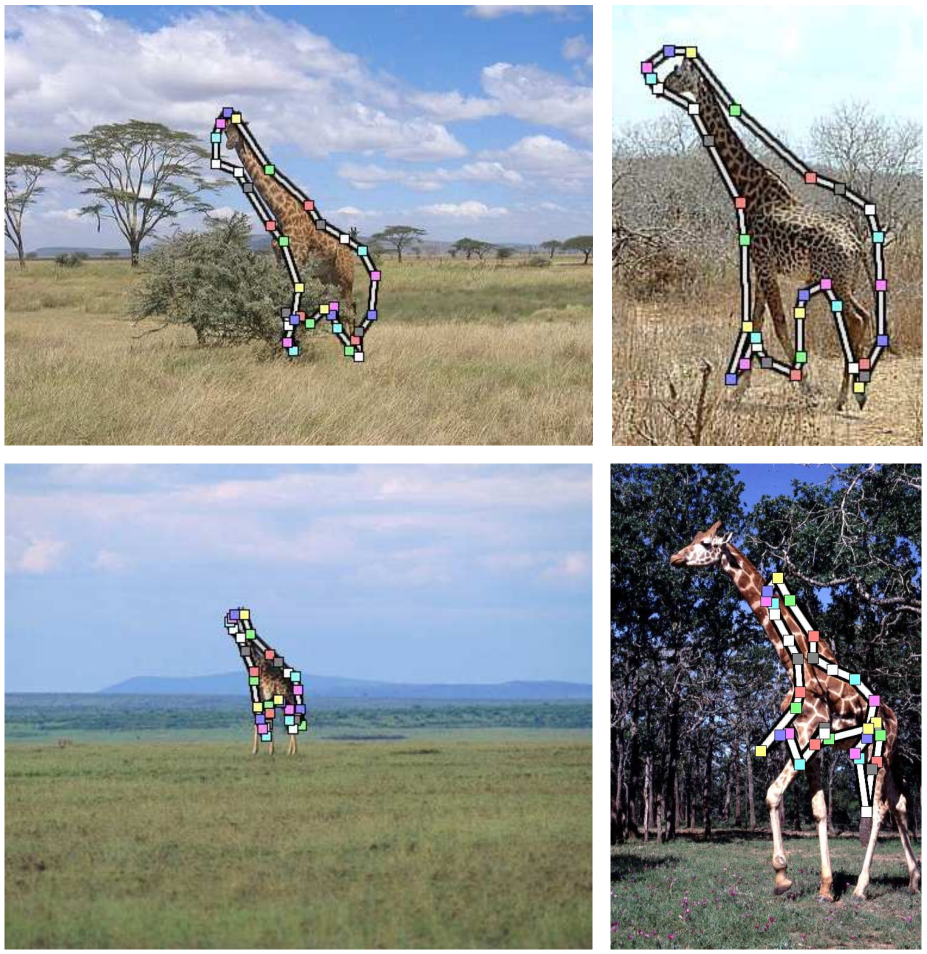


Figure 6.11: Examples from the ETHZ giraffes experiment.

scales $(1.0, 0.77, 0.59, 0.46, 0.35, 0.27, 0.21)$. Amongst all scales, we pick the one that produces the best unnormalized MRF score.

To make a fair comparison to the results of Ferrari et al. [49] on this dataset, we use *only* the boundary fragment and shape template features in our boosted detectors so that our model only uses edge information. Furthermore, we train our LOOPS model on half of the images in the object class, and include the other half in the test set together will all other images in the dataset. We also use the same evaluation protocol: a detection is considered successful if it overlaps by at least 20% with the groundtruth bounding box, and vice versa; and, as in their work, we report recall rate at the threshold that produces 0.4 false positives per image.

Figure 6.11 shows some example giraffe localizations from this dataset. Note that LOOPS was able to localize giraffes at a wide range of scales. Table 6.1 shows our detection performance on these three classes, compared to the **kAS Detector** of Ferrari et al. [49]. Their approach, developed over a series of papers, combines groups of contour segments into a powerful feature, and uses Hough-style voting across scales using these features plus a non-rigid matching algorithm to accurately localize object boundaries. Their voting mechanism provides an elegant and systematic solution to the scale variation problem. Furthermore, the features are specifically geared toward detection from edge data alone. Despite the limitations of the LOOPS model on both these counts, LOOPS significantly outperforms their method on the giraffe class. For swans, LOOPS achieves recall rates comparable to their models learned from real images, but not as good as those learned from hand-drawn models. For the Apple logos, mugs, and bottles classes, LOOPS obtains inferior recall rates.

While Ferrari et al. [49] developed the method on this dataset over several papers to optimize detection performance, the LOOPS method is designed more for deformable objects, and we compare on these datasets in the following sections. The detection results indicate that the benefit obtained from the LOOPS models depends on the extent to which the shape provides strong cues for the particular class. For datasets where the objects undergo a lot of scale changes but little articulation or deformation, the kAS detector is likely to perform best, whereas for classes with large deformations but low scale variables, LOOPS might be expected to perform better as a detector.

The results also suggest that an integration of the strengths of the two methods might provide considerable benefits. Perhaps the inclusion of such a detector as a pre-processing stage to the LOOPS outlining is a promising direction for data where identification of the region containing the object is more difficult.

Class	Hand Drawn [49]	Learned [49]	**LOOPS**
ETHZ Apple Logos	**86%**	83%	75%
ETHZ Bottles	**93%**	83%	61%
ETHZ Giraffes	70%	59%	**83%**
ETHZ Mugs	83%	**84%**	70%
ETHZ Swans	**94%**	75%	83%

Table 6.1: Recall rates at 0.4 false positives per image (FPPI) for the three classes from the ETHZ dataset. The mean recall rate across 5 randomizations is shown. We show both the results using a hand constructed model (second column), and a model learned from real images (third column) from the work of Ferrari et al. [49].

6.5.2 Accurate Corresponded Localization

In order for a LOOPS model to achieve its goals of classification, search and clustering based on characteristics of the shape or shape-localized appearance, it is necessary for our localization to be accurate at a more refined level than the bounding box prediction that is typical in the literature. In this section we evaluate the ability of our model to produce accurate outlines in which the model's landmarks are positioned consistently across test images.

As mentioned in related work, there are existing methods that also seek to produce an accurate outline of object classes in cluttered images. In order to evaluate the LOOPS outlines, we compare to two such state-of-the-art methods. The first is the **OBJCUT** model of Prasad and Fitzgibbon [102]. This method is based on the work of Kumar et al. [75], which produces an uncorresponded segmentation of the object. Briefly, this method uses an exemplar-based shape model of the object class together with a texture model to find the best match of an exemplar to the image. The second method we compare to is the **kAS Detector** of Ferrari et al. [49]. This approach uses adjacent contour segments as features for a detector. Both methods were updated to fit our data with help from the authors (P. Kumar, V. Ferrari; personal communications). Note that unlike both **OBJCUT** and our LOOPS method, kAS only requires bounding box supervision for the training images rather than full outlines. The matched contour segments for each detection are returned as the detected shape of the object. We compare LOOPS to these two methods on four

Class	LOOPS	OBJCUT	kAS Detector
Airplane	2.0	6.0	3.9
Cheetah	5.2	12.7	11.9
Giraffe	2.9	11.7	8.9
Lamp	2.9	7.5	5.8

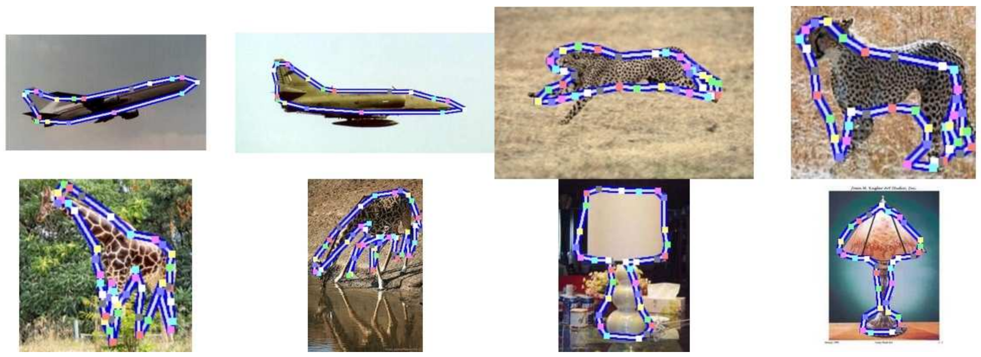
Table 6.2: Normalized symmetric root mean squared (rms) distance between the produced outline and the hand-labeled groundtruth, for the three competitors. Outlines are converted into a high-resolution point set, and the number reported is the rms of the distance from each point on the outline to the nearest point on the groundtruth (and vice versa), as a percentage of the groundtruth bounding box diagonal. Note that while the LOOPS and OBJCUT methods require full object outlines for the training images, the kAS Detector only requires bounding boxes.

object classes: 'giraffe', 'cheetah', 'airplane', and 'lamp'. Randomly selected example outlines are shown in Figure 6.12.

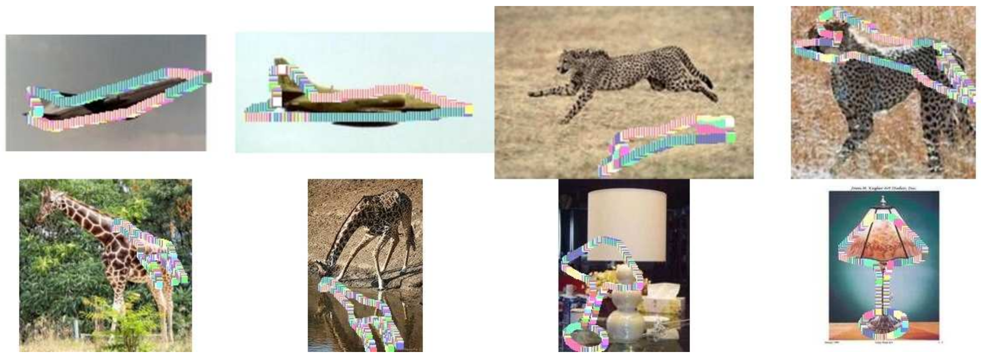
In order to evaluate these results quantitatively, we measured the symmetric root mean squared (rms) distance between the produced outlines and the hand-labeled groundtruth. These numbers are reported in Table 6.2. Based on this metric, LOOPS clearly produces more accurate outlines than both competing methods. In addition to these evaluations, we include a quantitative analysis for the descriptive experiments in Section 6.6.

While in some cases producing outlines is sufficient, there are also cases where we care about the precision of the localized landmarks. In the next section we show some experiments that use the localized landmarks for classification. Toward this goal, we now evaluate the accuracy of the model landmarks in test images. We consider the 'airplane', 'bass', 'buddha' and 'rooster' classes from the Caltech 101 dataset [42] as well as the significantly more challenging 'bison', 'deer', 'elephant', 'giraffe', 'llama' and 'rhino' classes from the mammal dataset of [50].[1] These images have more cluttered backgrounds than the Caltech images as well as foreground objects that blend into these backgrounds.

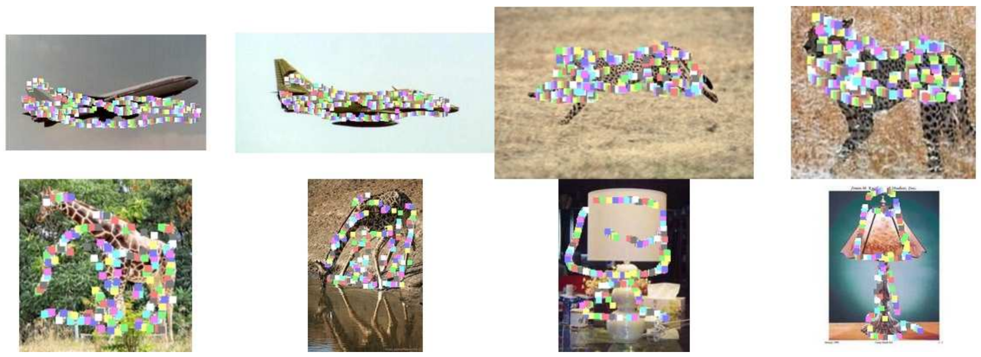
[1]Classes were selected based on the number of images available. For classes with almost enough images, we augmented the dataset with additional ones from Google images. Images used in this chapter can be found at `http://ai.stanford.edu/~gaheitz/Research/Loops`.



(a) LOOPS outlines



(b) OBJCUT segmentations



(c) kAS Detector detections

Figure 6.12: Example outlines for the three methods. The full set of resulting outlines for each method can be found at `http://ai.stanford.edu/~gaheitz/Research/Loops`.

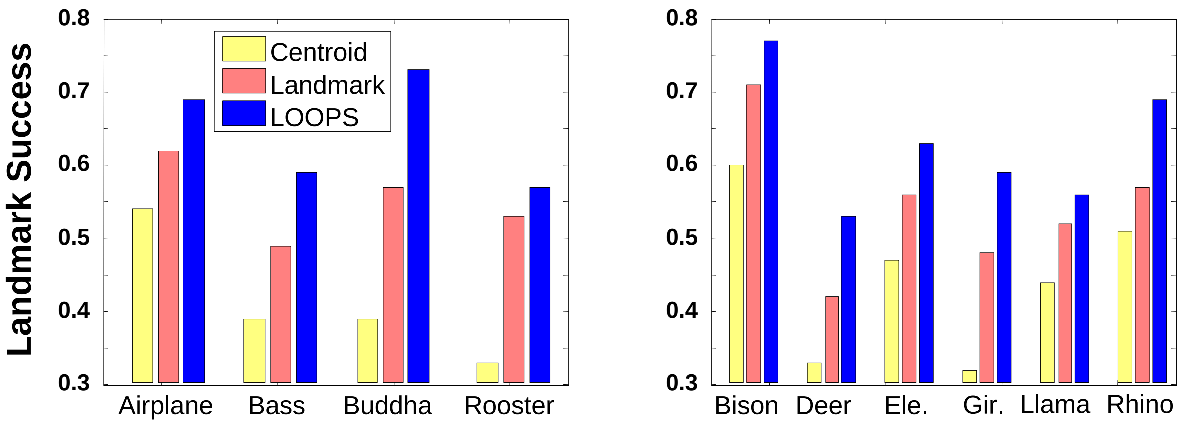


Figure 6.13: Landmark success rates for the Caltech and Mammal classes. Shown is the average fraction of landmarks that were localized within 5% of the bounding box diagonal from the groundtruth for our **LOOPS** method as well as the **Centroid** and **Landmark** baselines.

To measure our ability to accurately localize landmarks, we need a groundtruth labeling for the landmarks that is corresponded with our model. Thus, for the purposes of *this evaluation only*, we did not use our automatic method for corresponding training outlines, but rather labeled the Caltech and Mammal classes with manually corresponded landmarks. We then train a LOOPS model on these highly-supervised instances and evaluate our results using a scale-independent landmark success rate: a landmark is successfully localized if its distance from the manually labeled location is less than 5% of the diagonal of the object's bounding box in pixels.

Figure 6.13 compares the landmark success rates of our LOOPS model to two baseline approaches. In the first, we pick each landmark location using its corresponding detector by assigning the landmark to the pixel location in the image with the highest detector response. This **Landmark** approach uses no shape knowledge beyond the vote offset that is encoded in the weak detectors. As a second baseline, we create a single boosted **Centroid** detector that attempts to localize the center of the object, and places the landmarks relative to the predicted centroid using the mean object shape.

The evaluation of error on a per landmark basis is particularly biased in favor of the **Landmark** method, which is trained specifically to localize each landmark,

	Centroid	**Landmark**	**LOOPS**	
Caltech			Discrete	Refined
Airplane	2.6 (1.2)	2.0 (1.0)	1.6 (0.7)	**1.5** (0.7)
Bass	5.9 (3.8)	4.3 (2.3)	**4.0** (2.9)	4.1 (3.1)
Buddha	7.5 (6.4)	4.8 (3.7)	4.1 (3.4)	**4.0** (3.3)
Rooster	6.5 (6.4)	4.7 (5.0)	**3.8** (3.0)	**3.8** (2.8)
Mammals				
Bison	2.7 (0.6)	2.7 (1.7)	**2.0** (0.5)	**2.0** (0.6)
Deer	5.9 (4.0)	6.6 (4.1)	4.5 (3.3)	**4.4** (3.7)
Elephant	3.3 (0.9)	3.3 (1.5)	**2.5** (0.9)	**2.5** (0.9)
Giraffe	7.1 (4.3)	4.5 (3.2)	3.4 (2.3)	**3.3** (2.3)
Llama	3.7 (1.5)	3.2 (1.3)	3.5 (2.5)	**3.1** (1.3)
Rhino	3.3 (1.0)	3.1 (1.2)	2.6 (0.8)	**2.4** (1.0)

Table 6.3: Normalized symmetric root mean squared (rms) distance between the LOOPS outline and the hand-labeled groundtruth. The score for the class is the mean rms distance across instances, and the standard deviation across instances is given in parentheses.

without attempting to also fit the shape of the object. Nevertheless, the relative advantage of the LOOPS model, which takes into account the global shape, is evident. We note that while other works quantitatively evaluate the accuracy of their *outlines* (e.g., Ferrari et al. [49]), our evaluation explicitly measures the accuracy of the localized correspondences.

To get an absolute sense of the quality of our corresponded localizations, Table 6.3 shows the symmetric root mean squared distance between the predicted and groundtruth outlines. The improvement over the baselines is obvious and most errors are on the order of 10 pixels or less, indicating that we have produced sufficiently precise localizations for images that are 300 pixels in width. We note that our worst performance for the Buddha class is in part due to inconsistent human labeling that includes the base of the statue in some images but not in others. Figure 6.14 and Figure 6.15 provide a qualitative sense of the outlines predicted by LOOPS, showing several examples from each of the Mammal classes and Figure 6.16 shows several examples from each of the Caltech classes.

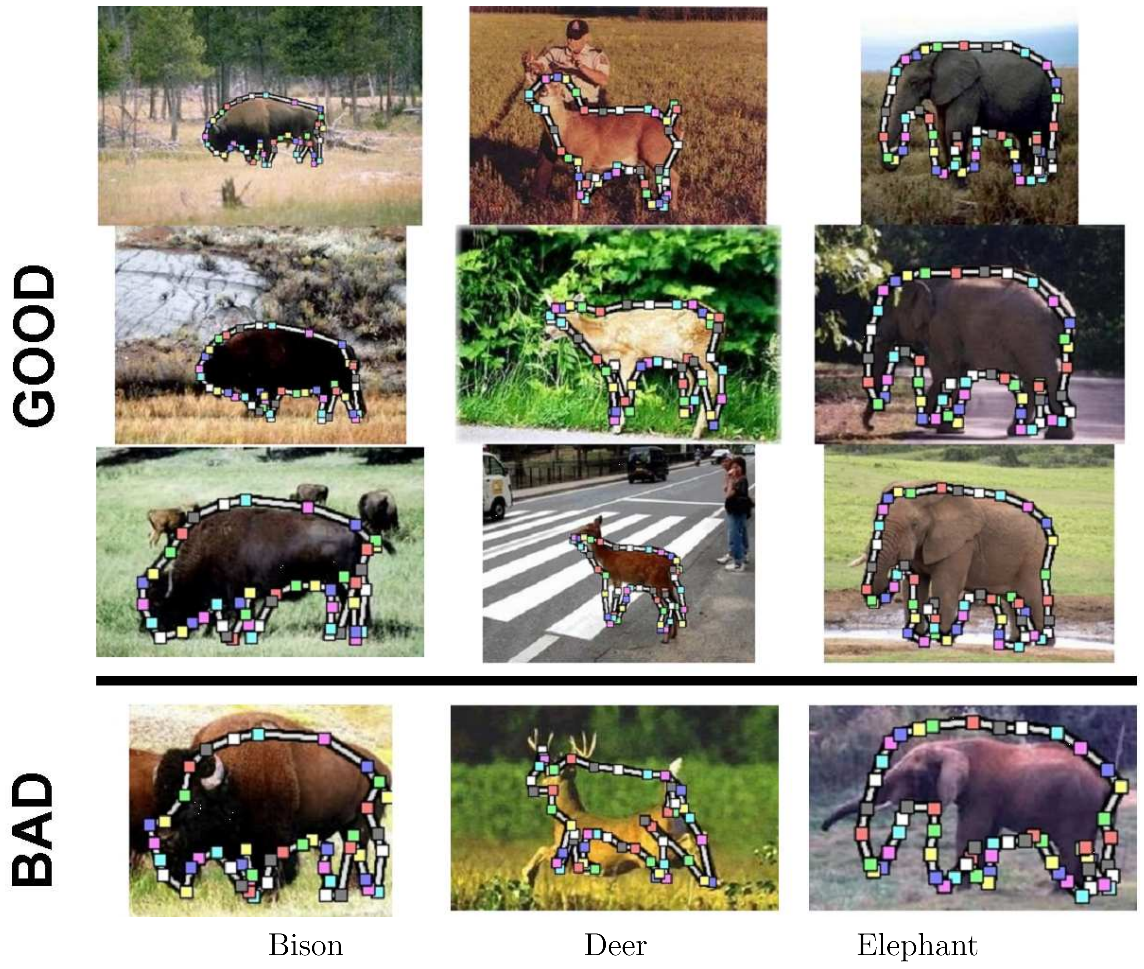


Figure 6.14: Example LOOPS localizations from the Mammal classes. The top three for each class are successful localizations, while the fourth is a failure. The full set of outlines can be found at `http://ai.stanford.edu/~gaheitz/Research/Loops`.

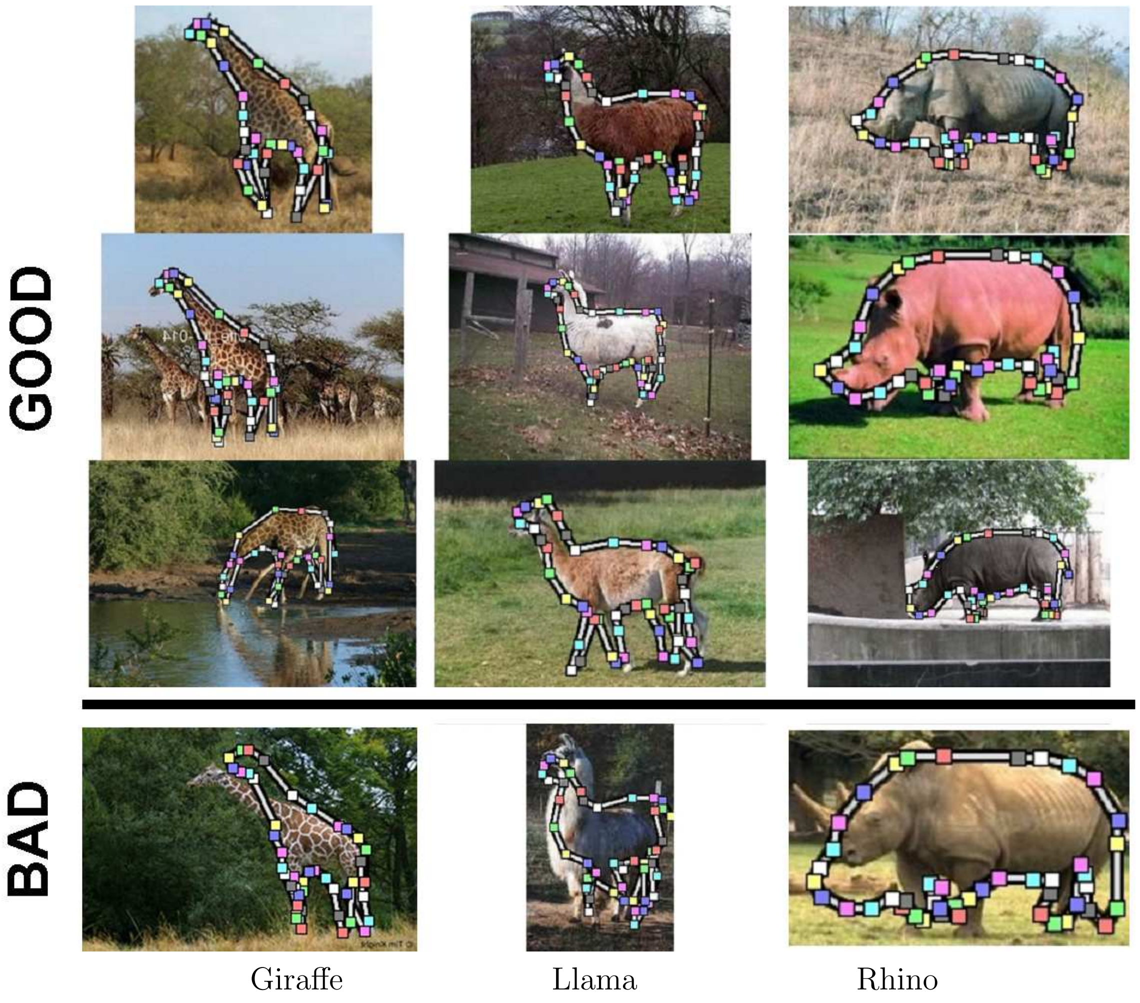


Figure 6.15: Example LOOPS localizations from the Mammal classes. The top three for each class are successful localizations, while the fourth is a failure. The full set of outlines can be found at `http://ai.stanford.edu/~gaheitz/Research/Loops`.

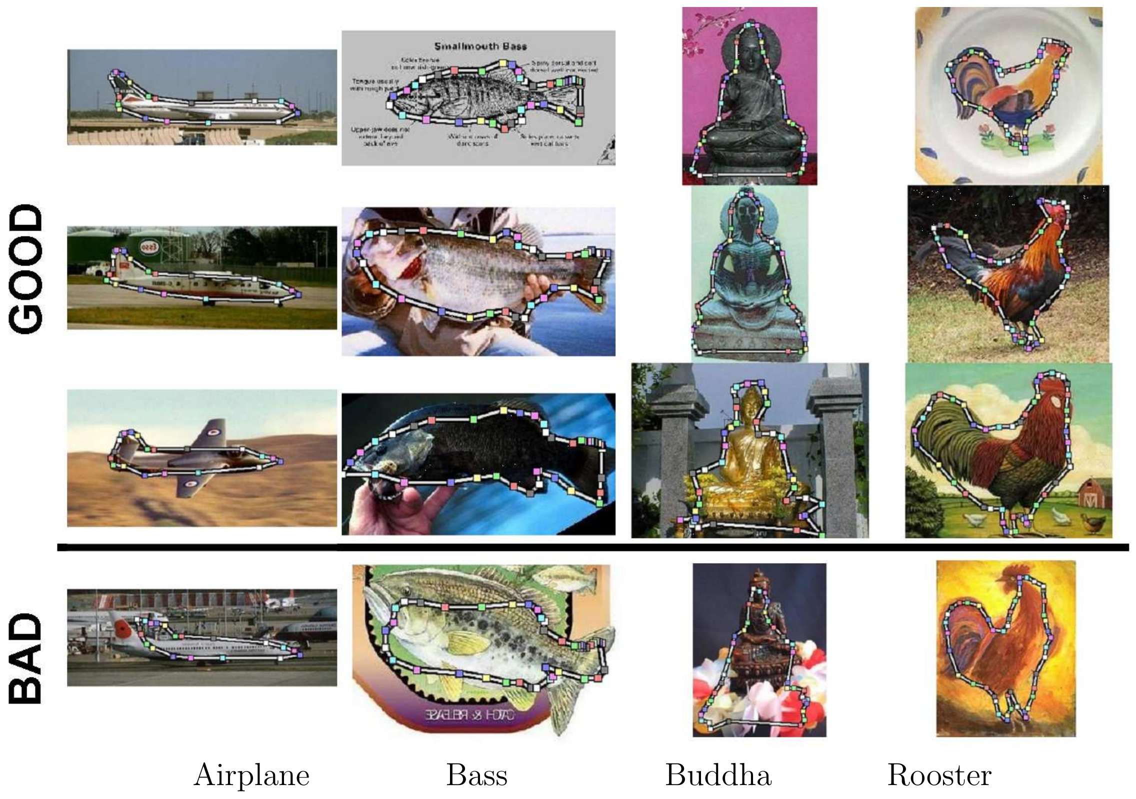


Figure 6.16: Example LOOPS localizations for the Caltech classes. The top three for each class are successful localizations, while the fourth (below the line) is a failure. The full set of resulting outlines can be found at `http://ai.stanford.edu/~gaheitz/Research/Loops`.

6.6 Descriptive Queries with LOOPS Outlines

Recall that our original goal was to perform descriptive queries on the image data. In this section we consider tasks that attempt to explore the space of possible applications facilitated by the LOOPS model. Broadly speaking, this space varies along two principal directions. The first direction concerns variation of the machine learning application. We will present results for classification, search (ranking), and a clustering application. The second direction varies the components that are extracted from the LOOPS outlines for these tasks. We will show examples that use the entire object shape, those that use a part of the object shape, and those that use the local appearance of a shape-localized part of the object. While each particular example shown below might be accomplished by other methods, the LOOPS framework allows us to approach any of these tasks with the same underlying object model.

6.6.1 User-defined Landmark Queries

We begin by allowing the user to query the objects in our images using refined shape-based queries. In particular, we use an interface in which the user selects one or more landmarks in the model and provides a query about those landmarks in test images. For example, to answer the question "Where is the giraffe's head?" the user might select a landmark in the head and ask for its location in test images. The user might also ask for the distance between two landmarks, or the angle between three landmarks. Figure 6.17 shows how this can be used to find images of lamps whose bases vary in width, as well as cheetahs with varying angles of their legs. While one might be able to engineer a task-specific solution for any particular query using another method, the localization of corresponded landmarks of the LOOPS model allows a range of such queries to be addressed in a natural and simple manner.

6.6.2 Shape-based Classification

Our goal is to use the predicted LOOPS outlines to distinguish between configurations of an object. For example, by precisely localizing the landmarks of an automobile,

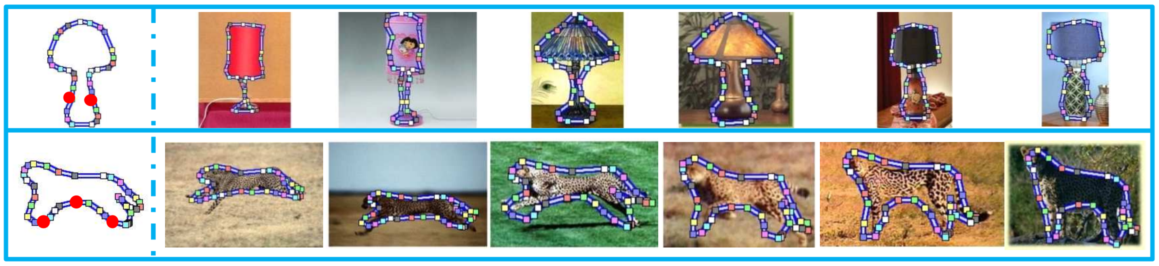


Figure 6.17: Refined object descriptions using pairs and triplets of landmarks. In the top row, the user selects two landmarks from the model on either side of the lamp base (highlighted on the left) and asks for the distance between them in test images, using the LOOPS derived landmark localizations. Results are displayed in order of this distance increasing from left to right. In the bottom row, the user selects a landmark on the front and hind legs of the cheetah as well as a landmark on the stomach in between them and asks for the angle formed by the three points. Test images are displayed in order from widest angle on the left to smallest angle on the right.

we can easily differentiate between an sedan (with the rear significantly below the room), an SUV (rear equally as high as the roof), or a pickup truck (long flat back). To accomplish this type of classification, we first train the joint shape and appearance model and perform inference to localize outlines in the test images, all *without* knowledge of the task at hand or any labels. We then incorporate knowledge of the desired *descriptive* task, including the descriptive labels, and perform the task using the corresponded localizations in the test images.

We now show how to perform descriptive classification with minimal supervision in the form of example images. Rather than specifying relationships between particular landmarks, the user simply provides classification labels for a small subset of the instances used to train the LOOPS model (as few as one per descriptive label) and the entire outline is used. We present three descriptive classification tasks for three quite different object classes (below we will also consider multiple classification tasks applied to the same object class):

1. **Giraffes standing vs. bending down**

2. **Cheetahs running vs. standing**

3. **Airplanes taking off vs. flying horizontally**

The first two of these tasks depend on the *articulation* of a target object, while the third task demonstrates the ability of LOOPS to capture the *pose* of an object. With these experiments we hope to demonstrate two points. The first is that an accurate shape allows for better classification than using image features alone. To show this, we compare to generative and discriminative classifiers that operate directly on the image features. Our second point is that the LOOPS model in particular produces outlines that are better suited to these tasks than existing methods that produce object outlines. We compare to two existing methods and show that the classifications produced by LOOPS are superior.

We begin our comparison with two baseline methods that operate directly on the image features. The first baseline is a generative **Naive Bayes** classifier that uses a SIFT dictionary for constructing features. For each SIFT keypoint for each image, we create a vector by concatenating the location, saliency score, scale, orientation and SIFT descriptor. Across the training images, we cluster these vectors into bins using K-means clustering, thus creating a SIFT-based dictionary (we chose 200 bins as this achieves the best overall classification results). We then train a generative Naive-Bayes model, where, for each label (e.g., standing = 0, bending down = 1), we learn a multinomial over the SIFT descriptor dictionary entries. When all of the training data is labeled, the multinomial for each label may be learned in closed form; when some of the training data is unlabeled, the EM algorithm is used to train the model by filling in the missing labels.

Our second baseline uses a discriminative approach, similar to the detector used by Torralba et al. [127]. First we predict the centroid of the object, using our **Centroid** detector. We then use the vector of feature responses at that location in order to classify the object. Specifically, we train a second boosted classifier to differentiate between the response vector of the two classes using the labeled training instances. We note that, unlike the generative baseline, this discriminative method is unable to make use of the unsupervised instances in the training set.

When using the LOOPS model, we train the joint shape and appearance model independently of the classification tasks and *without* knowledge of the labels. We then use the labels to train a classifier for predicting the label given a corresponded localization. To do this in a manner that is invariant to scaling, translation and rotation, we first align the training outlines to the mean using the procrustes method [37]. (In the airplane task, where the rotation is the goal itself, we normalize only for the translation and scale of the instances, while keeping their original orientations). Once our instances are aligned, we use principal component analysis to represent our outlines as a vector of coefficients that quantify the variation along the most prominently varying axes. We then learn a standard logistic regression classifier over these vectors; this classifier is trained to maximize the likelihood of the training labels given the training outlines. To classify a test image, we align the corresponded outline produced by LOOPS to the mean training outline, compute the PCA coefficients of the shape, and apply the classifier to the resulting PCA vector.

In order to get a sense of what contributes to the errors made by the LOOPS method, we also include classification results using the groundtruth outlines of the test instances. In this method, called **GROUND** in graphs below, the training and test groundtruth outlines are mutually corresponded using our automatic landmark selection algorithm (see Section 6.2). These corresponded outlines are then used in the same way as the LOOPS outputs. This measure shows the signal present in the "ideal" localizations (according to human observers), and serves as a rough indication of how well the LOOPS method would perform if its localizations matched human annotations.

Figure 6.18 (left column) shows the classification results for the three tasks as a function of the number N of supervised training instances per class (x-axis). For all three tasks, **LOOPS + Logistic** (solid blue) outperforms both baselines. Importantly, by making use of the shape of the outline predicted in a cluttered image, we surpass the fully supervised baselines (rightmost on the graphs) with as little as a single supervised instance (leftmost on the graphs). Furthermore, our classification results were of similar quality when using both a nearest-neighbor classifier and support vector machine. This indicates that once the data instances have been projected into the correct space (that of the corresponded outline), many simple machine learning algorithms can easily solve this problem. For the airplane and giraffe classes, the

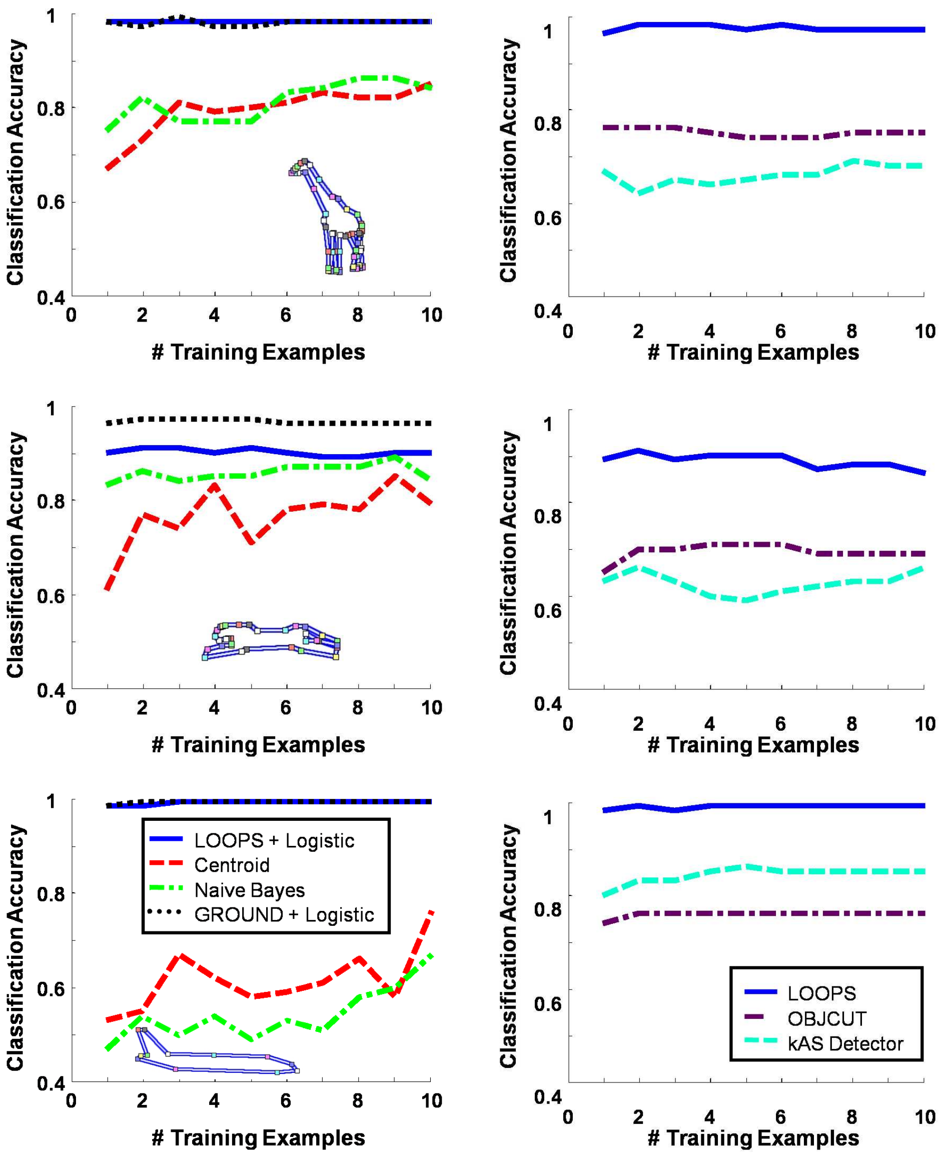


Figure 6.18: (left column) Results for three single-axis descriptive classification tasks. Compared are a logistic regression classifier on the **LOOPS** outlines, a **Naive Bayes** classifier and a discriminative appearance based **Centroid** classifier. On both the giraffe and airplane tasks, the combination of LOOPS outlining and a logistic regression classifier (**LOOPS + Logistic**) achieves near perfect performance, even with a single training instance for each descriptive class. For the cheetah task where the images are blurred, the LOOPS method only comes near to perfect performance and its advantage over the **Naive Bayes** classifier is less pronounced. (right column) A comparison of **LOOPS** to two other shape-based object detectors.

LOOPS + Logistic method perfectly classifies all test instances, even with a single training instance per class. Our advantage relative to the **Naive Bayes** classifier is least pronounced in the case of the cheetah task. This is not surprising as most running cheetahs are on blurry or blank backgrounds, and a SIFT descriptor based approach is expected to perform well.

Convinced that outlines are superior to image features for these descriptive classification tasks, we now turn to the question of how LOOPS compares to existing methods in producing outlines that will be used for these tasks. We compare to the two existing methods described above, **OBJCUT** and the **kAS Detector**. Because these methods have no notion of correspondence between outline points, we will use a classification technique that ignores the correspondences of the LOOPS outline. In particular, for each produced shape (by all three methods) we rescale the shape to have a bounding-box diagonal of 100 pixels, and center the shape around the origin. We can then compute the chamfer distance between any pair of shapes normalized in this manner.

We build nearest-neighbor classifiers for the descriptive task in question using this chamfer distance as our distance metric. Classification accuracy for these three methods as a function of the number N of supervised training instances is shown in Figure 6.18 (right column). We can see that LOOPS significant outperforms both the **OBJCUT** and **kAS Detector** for these tasks.

Once we have our output outlines, an important benefit of the LOOPS method is that we can in fact perform multiple descriptive tasks with the same object model. We demonstrate this with a pair of classification tasks for the lamp object class. The tasks differ in which "part" of the object we consider for classification. In particular, we learn a LOOPS model for table lamps, and consider the following two descriptive classification tasks:

1. **Triangular vs. Rectangular Lamp Shade**.
2. **Thin vs. Fat Lamp Base**.

While we do not explicitly annotate a subset of the landmarks to consider, we show that by including a few examples in the labeled set, our classifiers can learn to consider only the relevant portion of the shape. The setup for each task is the same as described in the previous section. We stress that the test localizations predicted by LOOPS are the same for both tasks. The only things that change are the label set and the learned logistic regression classifier that attempts to predict labels based on the output localization. In this case, the baseline methods must be completely retrained for each task, which requires EM learning in the case of **Naive Bayes**, and boosting in the case of **Centroid**.

As can be seen in Figure 6.19 (left column), for both descriptive distinctions we are able to achieve effective classification, while both feature-based baselines perform barely above random. In addition, we again outperform the shape-based baselines, as can be seen in the right column of Figure 6.19. The consequences of this result are promising: we can do most of the work (learning a LOOPS model and predicting corresponding outlines) once, and then easily perform several descriptive classification tasks. All that is needed for each task is a small number of labels and an extremely efficient training of a simple logistic regression classifier. Figure 6.20 shows several qualitative examples of successes and failures for the above tasks.

We note that in the lamp tasks, the **LOOPS + Logistic** method sometimes outperforms the **GROUND + Logistic** "upper bound" for a small number of training instances. While this seems counter-intuitive, in practice a different choice of landmarks or inaccuracies in localizations can lead to fluctuations in the classification accuracy.

6.6.3 Shape Similarity Search

The second application area that we consider is similarity search, which involves the ranking of test instances on their similarity to a search query. A shopping website, for example, might wish to allow a user to organize the examples in a database according to similarity to a query product. The similarity measure can be any feature that is easily extracted from the LOOPS outline or from the image based on the corresponded outline. We demonstrate similarity using the entire shape, a user-specified component of the shape, and an appearance feature (color) localized to a certain part of the

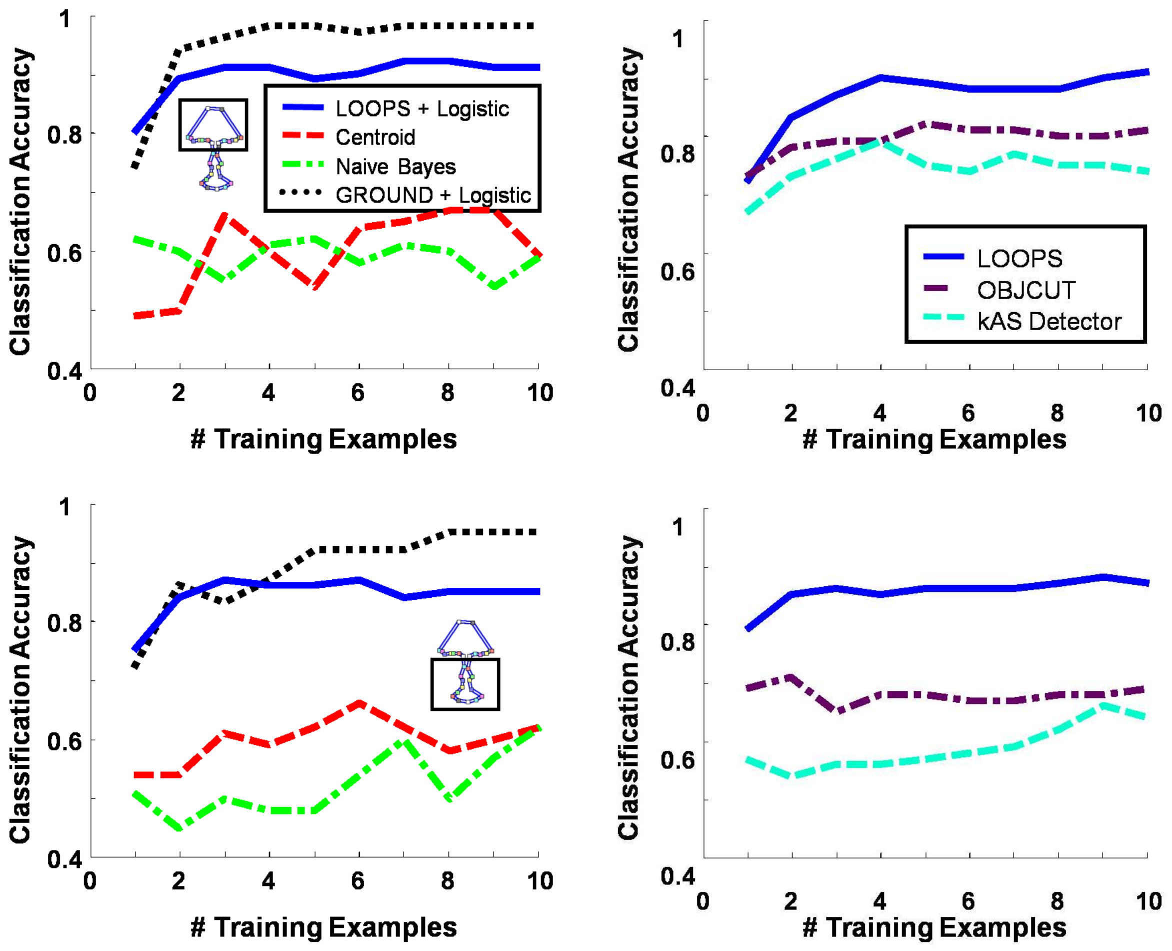


Figure 6.19: (left column) Classification results for the two descriptive tasks for the table lamp object class. Compared are the LOOPS outlining method joined by a logistic regression classifier **LOOPS + Logistic**, a **Naive Bayes** classifier and a discriminative appearance based **Centroid** classifier. We also include the **GROUND + Logistic** results, in which manually labeled outlines from the training and test set are used and that approximately upper bounds the achievable performance when relying on outlines. In both tasks, the LOOPS model is clearly superior to the baselines and is not far from perfect (**GROUND + Logistic**) performance. (right column) (right column) A comparison of **LOOPS** to two other shape-based object detectors. For these tasks, the **LOOPS** outlines are far superior.

object.

The experimental setup is as follows: offline, we train a LOOPS model for the

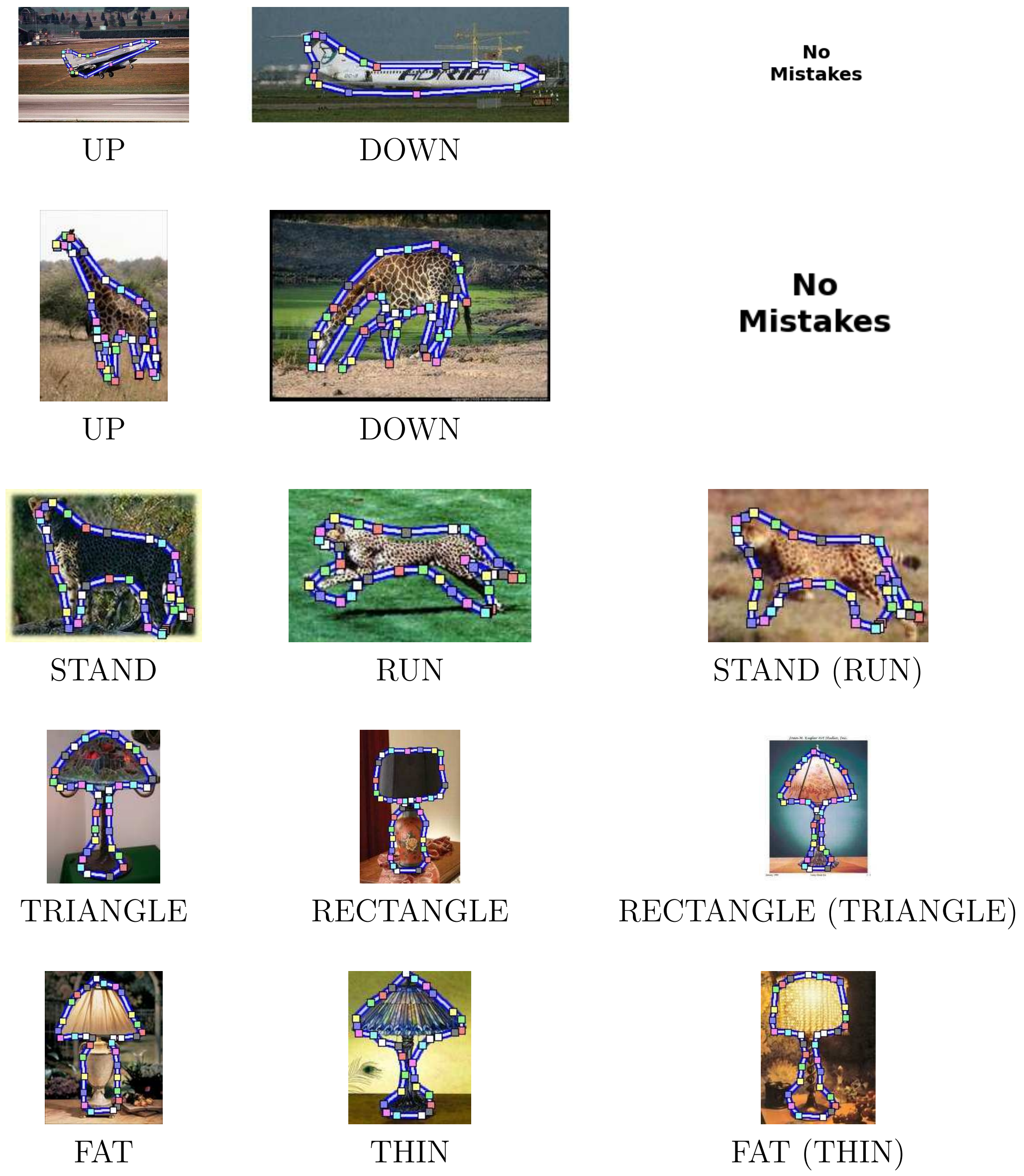


Figure 6.20: Example classifications for our five tasks. Underneath each image we report the label produced by LOOPS + Logistic, and for errors we give the groundtruth label in parentheses. The first two columns show a correct labeling for each label, and the third column shows a mistake for each task (where available). In rows three and five (cheetah and lamp bases), we can see that the mistake is caused by a failure of LOOPS to correctly outline the object. In the fourth row (lamp shades), we see that despite an accurate localization, we still incorrectly classify the lamp shade. This is due to the inherent limitations of the logistic approach with little training data.

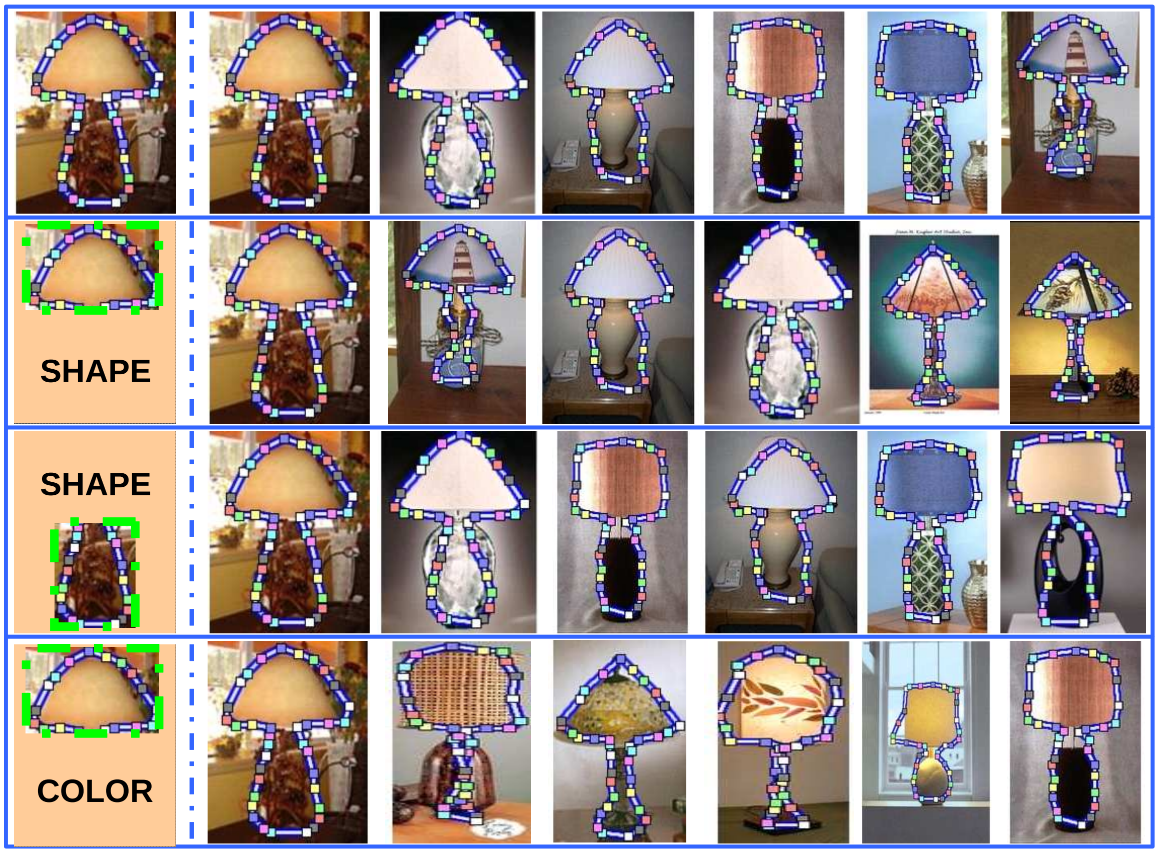


Figure 6.21: Object similarity search using the LOOPS output. In the top row we show the top matches in the test database using the full shape of the query object (left column). Similarity is computed as the negative sum of the squared distances between the corresponding PCA coefficients of the procrustes aligned shapes. In the second row we refine our search by using only the landmarks that correspond to the lamp shade. The results now show only triangular lamp shades. The third row shows a search based on the lamp base, which returns wide bases. In the bottom row, we use color similarity of the lamp shade to rank the search results. The top matches show lamps with faded yellow shades.

object class and localize corresponded outlines in each of the test set images. Recall that in LOOPS this is done once and all tasks are carried out given the *same* localized outlines. Online, a user indicates an instance from the test set to serve as a "query" image and a similarity metric to use. We search our database for the images that are most similar based on the specified similarity metric, and return the ranked list back to the user.

As an example, we consider the case of the lamp dataset used above. Users select a particular lamp instance, a subset of landmarks (possibly all), and whether to use shape or color. Each instance in the dataset is then ranked based on Euclidean distance to the query in shape PCA space. When a subset of landmarks are chosen, the principal component vectors are computed for these landmarks only, and the distance is computed for the PCA vectors in this space. This requires running PCA for each subset, which can be achieved by selecting the covariance submatrix corresponding to the selected landmarks, and computing the eigenvectors of this submatrix. The covariance calculation can be done once and stored, while the eigen-decomposition needs to be computed at search time. As these vectors tend to be very small, this will not be a computational bottleneck.

Figure 6.21 shows some example queries. In the top row we show a full-shape search, where the first (left-most) instance is the query instance. The results are shown left to right in increasing distance from the query instance. For the full-shape query, we see that both triangular lampshades and wide bases are returned. Suppose that the user decides to focus only on the lampshade; the second row shows the results of a search using only the landmarks in the selected region. This search returns lamps with triangular shades, but with bases of varying width. The third row displays the results of a search where the user instead decides to select only the base of the lamp, which returns lamps with wide bases. Finally, the bottom row shows a search based on the color of the shade. In this case we compute the similarity between two instances as the negative distance between the mean LAB color vectors[2] in the region outlined by the selected landmarks. The top-ranked results all contain off-white shades similar in color to the query image.

Similarity search allows users to browse a dataset with the goal of finding instances with certain characteristics. In all of the examples considered above, by projecting the images into LOOPS outlines, similarity search desiderata were easily specified and effectively taken into account. Importantly, the similarity of interest in all of

[2]The Lab color space represents colors as three real numbers, where the L component indicates "lightness" and the A and B dimensions represent the "color". The space is derived from a nonlinear compression of the XYZ color space.

these cases is hard to specify without a predicted outline.

6.6.4 Descriptive Clustering

Finally, we consider clustering a database by leveraging the LOOPS predicted outlines. As an example, suppose we have a large database of airplane images, and we wish to group our images into "similar looking" groups of airplanes. Clustering based on shape might produce clusters corresponding to passenger jets, fighter jets, and small propeller airplanes. In this section, we consider an outline *and* appearance based clustering where the feature vector for each airplane includes the mean color values in the *LAB* color space for all pixels inside the airplane boundary. We cluster the database of examples on this feature vector using K-means clustering. Figure 6.22(a) shows one cluster that results from this procedure for a database of 770 images from the Caltech airplanes image set [42]. Such results might be obtained from any method that produces a precise outline (or segmentation) of the object.

Imagine, however, that instead of clustering using the appearance for the entire airplane, we are instead interested in the appearance of the tail. This may be a useful scenario because patterns and colors on the tail are generally more informative about the airline or country of origin. In order to focus on the airplane tails, we can specify a subset of the landmarks in the LOOPS model to indicate the region of interest (as in the lamp example above). For each image this set of landmarks should correspond to the same "part" of the object. We can look only at the portion of the image around these landmarks, in this case zooming in on the tail of the airplane. In Figure 6.22(b), we show only this portion of the image for the same images that are shown in Figure 6.22(a). Despite the fact that the cluster looks coherent when considering the whole plane, the tails are very heterogeneous in appearance.

The fact that the outlines produced by LOOPS are consistently corresponded allows us to cluster the appearance of the tail itself. That is, LOOPS allows us to rely on the *localized appearance* of different parts of the object. Specifying the tail as the landmark subset of interest, we compute the same feature vectors only for the pixels contained in the hull of the tail. We then re-cluster our database using these

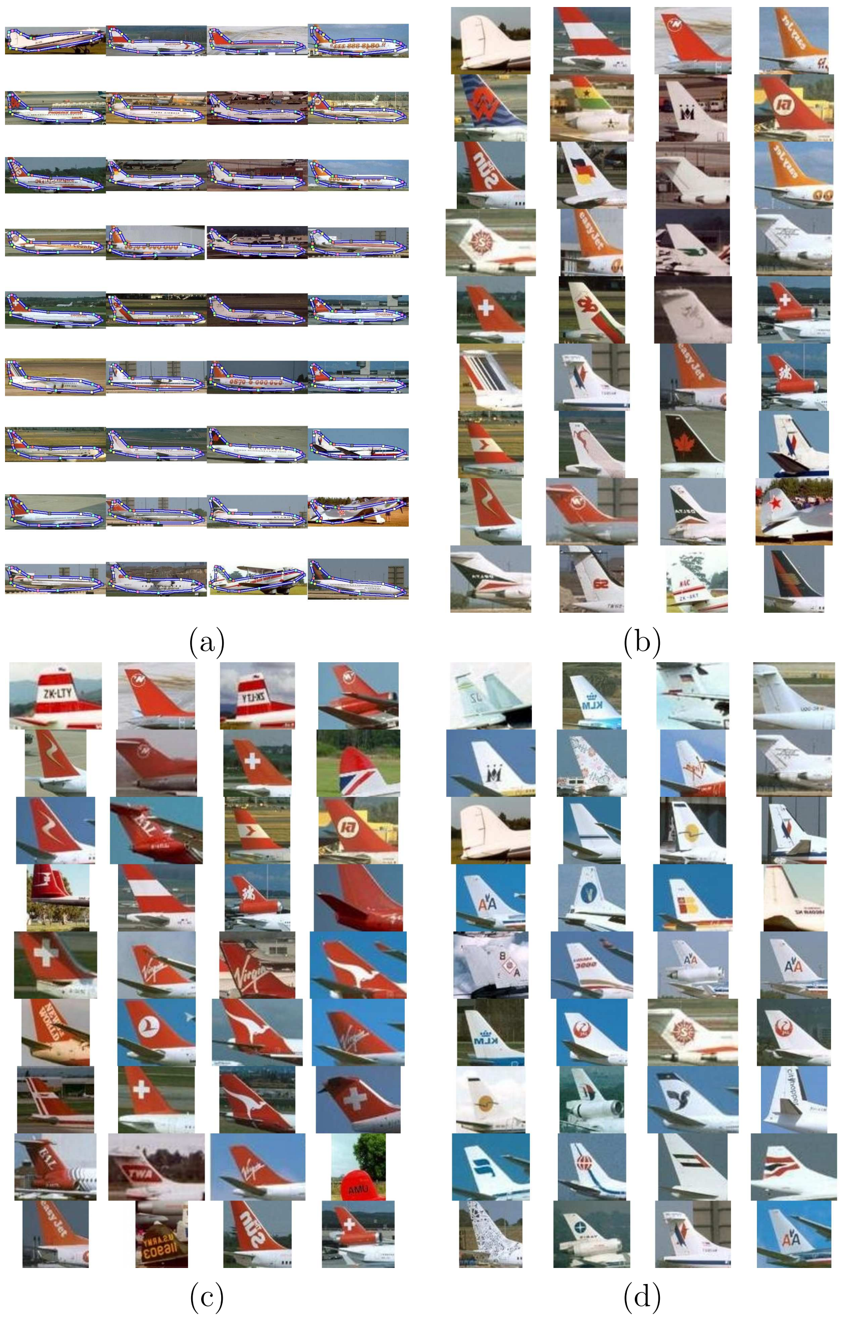


Figure 6.22: Clustering of a large airplane database. For each cluster we show the top 36 instances based on the distance to the cluster centroid. (a) full airplane clusters. (b) tails of clustering in (a). (c) and (d) show the clusters based on tail color that contain the first two examples of (a) and (b).

tail feature vectors. Figure 6.22(c) and (d) shows the zoomed in tails for the two clusters that contain the first two instances from Figure 6.22(a). The coherence of the tail appearance is much more apparent in this case, and both clusters group many tails from the same airlines.

In order to perform such coherent clustering of airplane tails, one needs first to accurately localize the tail in test images. Even more than the table lamp ranking task presented above, this example highlights the ability of LOOPS to leverage *localized appearance*, opening the door for many additional shape and appearance based descriptive tasks.

6.7 Related Work

Our method relies on a joint probabilistic model over landmark locations that unifies boosted detectors and global shape toward corresponded object localization. Our boosted detectors are constructed, for the most part, based on the state-of-the-art object recognition methods of Opelt et al. [99] and Torralba et al. [127] (see Section 6.3.2 for details). In this section, we concentrate on the most relevant works that employ some notion of geometry or "shape" for various image classification tasks.

6.7.1 Shape based classification

Several recent works study the ability to classify the shape of an outline directly [7, 59, 8, 115, 44]. Such techniques often take on the form of a matching problem, in which the cost between two instances is the cost of "deforming" one shape to match the other. Most of these methods assume that the object of interest is either represented as a contour or has already been segmented out from the image. While many of these approaches could aid classification of objects in cluttered scenes, including the classification along the various descriptive dimensions explored above, few of them address the more serious challenge of extracting the object's outline from such an image. One exception is the work of Thayananthan et al. [124], which uses shape-contexts for object localization in cluttered scenes. Their method demonstrates matching of rigid templates and a manually-constructed model of the hand, but does not attempt to learn models from images.

6.7.2 Contour propagation methods

Some of the best-studied methods in machine vision for object localization and boundary detection rely on shape to guide the search for a known object. In medical imaging, active contour techniques such as snakes [28, 21], active shape models [24], or more recently level set and fast marching methods [116] combine simple image features with relatively sophisticated shape models. In medical imaging applications, where the shape variation is relatively low and a good starting point can easily be found, these methods have achieved impressive results.

Attempting to apply these methods to recognizing object classes in cluttered scenes, however, faces two problems. The first is that we generally cannot find a good starting point, and even if we do, localization is complicated by the many local maxima present in real images. Secondly, edge cues alone are typically insufficient to identify the object in question. In an attempt to overcome this problem, active appearance models have been introduced by Cootes et al. [23]. While this method achieved success in the context of medical imaging or face outlining in images with little additional clutter, they have not (to the best of our knowledge) been successfully applied to the kind of cluttered images that we consider in our experimental evaluation. Indeed, our own early experiments confirmed that a contour front propagation approach with several starting points is generally unable to accurately locate outlines in complex images, and simple methods for finding a good starting point — such as first detecting a bounding box — are insufficient.

6.7.3 Contour outlining in real images

Ferrari et al. match test images to a contour-based shape model constructed from a single hand-drawn outline [47] and learn a similar model automatically from a set of training images with supervised bounding boxes [49]. Their algorithm achieves impressive results at multiple scales and in a fair amount of clutter using only detected edges in the image. However, they do not, in these papers, attempt to use the discovered shapes for any descriptive classification task. In experiments above, we compared to this method, and showed that our method produces more useful outlines for the descriptive classification tasks that we target. Furthermore, our method is more appropriate for classes in which shape is more deformable or cluttered images where features other than edge fragments might be useful. For problems where the goal is a rough localization, this method is able to average over the location of the contour fragments, and therefore achieves are more robust localization. If accurate outlining is desired, then the LOOPS method is more appropriate, as it focuses on the accuracy of the landmarks, rather than of the object as a whole.

6.7.4 "Parts" based methods

Following the generative constellation model of Fergus et al. [45], several works make use of explicit shape by relying on a model of parts (patches) and their geometrical arrangement (e.g., [68], [46]). Most of these models are trained for detection or classification accuracy, rather than accurate localization of the constituent "parts", and do not provide evidence that the discovered parts have any consistent meaning or localization. In fact, even with a "correct" set of parts, localization need not be accurate to aid detection/classification. To see this, consider an example where the giraffe abdomen and legs are localized correctly but the head and neck are incorrectly pointing upwards. In this case, classification and bounding box detection are still probably correct but the localization cannot be used to determine the pose of the giraffe. Indeed, instead of trying to localize parts, the constellation approach *averages* over localization. While this is a useful strategy when the goal is detection/classification, it fails to provide a single most-likely assignment of parts that could be used for the further processing that we consider.

The work of Crandall et al. [26] uses a parts-based model with manually annotated parts. They evaluate the accuracy of localizing these parts in test images. Their follow up work [27] automatically learns the parts, but similar to the constellation method, uses these only as cues for object detection. Our work learns object landmarks automatically and explicitly uses them not only for detection, but for precise outlining and further descriptive querying.

6.7.5 Additional use of shape for classification, detection and segmentation

Foreground/background segmentation is another task that is closely related to object outlining. The class-based segmentation works of Kumar et al. [75] and Borenstein and Ullman [12] rely on class-specific image features to segment objects of the relevant class. These approaches, however, aim to maximize pixel (or region) label prediction and do not provide a corresponded outline. Winn and Shotton [136] attempt simultaneous detection and segmentation using a CRF over object parts. They evaluate their segmentation quality, but do not consider the accuracy of their part localization. In experiments above, we used code based on the work of Kumar et al. [75], to produce uncorresponded segmentation outlines that are compared to de-corresponded outlines produced by our method.

Many recent state-of-the-art methods rely on implicit shape to aid in object recognition. For instance, Torralba et al. [127] use image patches, and Opelt et al. [99] and Shotton et al. [118] use edge fragments as weak detectors with offsets from the object centroid that are boosted to produce a strong detector. These works combine features in order to optimize the aggregate decision, and typically do not aim to segment/outline objects. Leordeanu et al. [81] and Berg et al. [9] similarly combine local image features with pairwise relationship features to solve model matching as a quadratic assignment problem. While such models give a correspondence between model components and the image, the training regime optimizes recognition rather than the accuracy of these correspondences. Despite the training regimes of the above models, the implicit shape used by these models can be used to produce both a detection and a localization through intelligent use of the “weak” detectors and their offsets. Indeed, both Leibe et al. [80] and Opelt et al. [100] produce both a detection and soft segmentation. While these methods tend to show appealing anecdotal results, the merit of their localizations (both corresponded and uncorresponded) is hard to measure, as no quantitative evaluation of segmentation quality or "part" localization is provided.

6.8 Discussion and Future Work

In this work we presented the **L**ocalizing **O**bject **O**utlines using **P**robabilistic **S**hape (LOOPS) approach for obtaining accurate, corresponded outlines of objects in test images, with the goal of performing a variety of descriptive tasks. Our LOOPS approach relies on learning a probabilistic model in which shape is combined with discriminative detectors into a coherent model. We overcome the computational challenge of localizing such a model in cluttered images and under large deformations and articulation by making use of a discrete-then-continuous optimization approach. We directly and quantitatively evaluated the ability of our method to consistently localize constituent elements of the model (landmarks) and showed how such a localization can be used to perform descriptive classification, search based on shape and localized appearance, and clustering that focuses on a particular part of the object. Importantly, we showed that by relying on a model-to-image correspondence, our performance is superior to discriminative and generative competitors often with as little as a single labeled example per descriptive class.

In theory, some existing detection methods (e.g., [49, 9, 99, 118]) lend themselves to some of the descriptive tasks described above, by producing outlines during the detection process. However, in experiments above, we demonstrated shortcomings with two state-of-the-art methods in this regard. Because our LOOPS model was targeted towards producing these types of outlines rather than detecting the presence or absence of the object, it was able to obtain significantly more accurate shape-based classifications. Furthermore, in practice, no other work that considered object classes in cluttered images demonstrated a combination of accurate localization and shape analysis that would solve these problems.

Our contribution is thus threefold. First, we design our LOOPS model with the primary goal of corresponded localization in cluttered scenes. The landmarks are chosen automatically so that they will both be salient and appear consistently across instances, and both the training and correspondence steps are geared specifically toward the goal of accurate localization. Second, we present a hybrid global-discrete then local-continuous approach to the computational challenge of corresponding a model to a novel image. This allows us to achieve consistent correspondence in cluttered images even in the face of large deformations that will hinder alternatives such as active shape or appearance models. Third, we demonstrate that accurate localization is of value for a range of descriptive tasks, including those that are based on appearance.

There are several interesting directions in which our LOOPS approach can be extended. We would like to automatically learn coherent parts of objects (e.g., the neck of the giraffe) as a set of landmarks that articulate together, and accurately localize both landmarks and part joints in test images. Intuitively, learning a distribution over part articulation (e.g., the legs of a running mammal are synchronized) can help localization. More importantly, localizing parts opens the door for novel descriptive tasks that consider variation in the number of parts (e.g., ceiling fan with 4 or 5 blades) or in their presence/absence (e.g., cup with and without a handle).

Chapter 7

Conclusions and Future Directions

This thesis has focused on two important sub-problems of the larger scene understanding problem. In this chapter we summarize the contributions that were made concerning the exploitation of context and the power of obtaining a more refined characterization of objects. Following this, we explore some of the open problems and future directions implied by this work.

7.1 Summary

This thesis has addressed two of the prominent problems at the **Interaction Layer** of the scene understanding hierarchy (recall Figure 1.1): contextual modeling between the various sub-components required for image interpretation, and refined characterization of objects based on their shape, pose, or similar properties.

Chapter 4 explored one potential holistic approach to the problem. The **CCM** approach allowed for the flexible combination of multiple state-of-the-art models, making it simple to incorporate a new model into the whole. The interactions between the sub-tasks were in the form of features derived from the outputs of classifiers for the variables associated with each component. Because these features can be defined in any way that makes sense to the implementer, the **CCM** construction can also be tailored to pick up the signals that are applicable for a given dataset. Because the classifiers in the **CCM** framework are trained in a supervised, discriminative manner, we require data with labels for each of the sub-components. While this can be expensive to gather, there already exist many datasets with various sets of labels. Because of the structure of **CCM**s, they are able to use any amount of data with any labels that are available, rather than requiring that each training image have labels for each task.

In contrast to this approach, the TAS model of Chapter 5 offers a more sophisticated modeling of the interactions between object detection and region labeling. This single *joint* probabilistic model over both variables allows for region labels and object labels to directly influence each other at inference time. In addition, the TAS model uses unsupervised region labeling, where the regions are grouped into clusters based on their appearance and contextual properties. This has the attractive property that it allows for a large quantity of unlabeled data for the purpose of learning the region model.

While these two methods differ substantially in how they share the contextual information, experiments with both taught us some important things that can be used in future models for holistic scene understanding. The results of the **CCM** approach indicate that feature based sharing of contextual information is an efficient way of communicating this information between methods. The TAS experiments showed that even without explicit labeling of the semantic entities that participate in the contextual relationships, these relationships can be discovered from the data, and successfully captured at inference time.

Comparison of the two methods showed that the interaction between the semantics region classes and the detections was more useful than using only the contextually clustered region labels. This indicates that a future approach should attempt to capture the semantic classes in the data. Also, limitations of both of these methods show that a better understanding of the 3D structure of the scene (rather than mere 2D relationships) is likely to be necessary for a satisfactory future system.

Chapter 6 addresses the problem of descriptive classification, where refined characteristics, beyond the mere category, are extracted from objects in images. As described in Section 1.3, this level of processing has been pursued for images of humans in various poses and in the medical community, but is relatively new for the computer vision object detection community. The LOOPS model was developed to precisely outline instances from various classes in images, and we showed how these outlines can be used for a range of descriptive tasks, including shape/pose/articulation-based classification, shape search, and localized appearance-based clustering. In addition to facilitating this type of post-processing, the LOOPS model also produces more accurate and precise outlines than competing state-of-the-art methods for a broad range of classes. This indicates that LOOPS is likely to be useful for other tasks that require such refined localization.

7.2 Open Problems and Future Directions

In addition to answering many questions about scene understanding, the work and experiments in this thesis have opened many doors for future exploration. In this chapter, we discuss three of the most interesting questions for future researchers to answer.

7.2.1 Simple and Flexible vs. Unified and Sophisticated

The most sharp contrast between **CCM**s and the TAS approach is in the complexity of the probabilistic interaction between the sub-components. The **CCM** framework is built on the idea of complex building blocks that interact with a simple feature-based interface. This has several advantages. The designer of such a system can spend most of his time and effort in engineering the correct features to pass between the pieces, rather than struggling to implement complex methods correctly. In addition, because each component is state-of-the-art, and can usually easily be replaced by a newer, better method, it is relatively inexpensive to improve performance. The **CCM** framework also has several disadvantages. If the interaction between two components is very tight (think of boats occurring with water), we would rather infer these things together at the same time. Furthermore, for someone with sufficient expertise, allowing only feature-based interactions can constrain the set of options that can be explored.

On the other side of the spectrum is the TAS model, where the probabilistic interactions occur in a single joint model. Also, the region label model is a simple naive Bayes model over multivariate Gaussian appearances, which cannot capture the multimodal appearance distribution over some semantic classes (e.g., water can be bright blue and textureless, or dark and heavily textured). The sophisticated interaction through the relationship variables provides the advantage of allowing the explicit modeling of what is happening in the world (e.g., the car actually is driving 'on' the road). Rather than just capturing a statistical property of the images (like **CCM** does), this is a more realistic model of the underlying interaction. Unfortunately, because of the machinery required to learn and infer the TAS model, substituting in a "better" multi-class segmentation model will force major changes to the model design decisions. While not impossible, it is also not straightforward to integrate additional components into the model, such as depth reconstruction or scene category modeling.

As a result, it is still an open question which of these two approaches will serve as the foundation for a more successful future model. As a result, in the near future, there is value in continuing to pursue both approaches, in order to weigh their relative merits.

7.2.2 Extending TAS to Include More Tasks

If the best solution to the scene understanding problem involves a proper joint probabilistic model over the various scene understanding sub-tasks, then a TAS-like approach will be required. In this case, it is important that the other pieces be introduced in an effective way, with the proper level of modeling machinery. The open question is how these other models can be integrated. Does a directed BN model make sense if there is a large set of different types of variables that interact in complex and subtle ways? Or is it more natural to switch to an undirected MRF-based approach, where the interactions can be learned as arbitrary potentials without the need to consider which variable is the parent of which?

The other important decision for a future extension of these approaches is whether to use a generative or discriminative model (or a combination of both). As discussed above, one of the primary advantages of the **CCM** approach is the ability to use training data without each instance requiring all groundtruth labels. Because the prospect of obtaining a truly large dataset with full labels for all of the possible components we might want to model is unlikely, our eventual model for this domain should allow for partially supervised instances. Based on this requirement, a generative model becomes more attractive, where hidden causes (which may be observed for some instances) affect the presence and layout of the observed parts of the scene.

Furthermore, if one of the goals of our modeling is to extrapolate to new scenes that have the same pieces, but are unlike other images that we have seen before, then generative models might be a more natural fit. If the goal is merely to accurately classify images that fit within the distribution that we have seen at training time, then discriminative methods may work better. All of these decisions must be made for models along this line of work.

7.2.3 Modeling 3D Relationships

One of the common error modes of both the **CCM** and TAS models arose from the limitations of the 2D notion of spatial relationships. For example, both models learned that cars occur "underneath" regions that looks like buildings or regions that are classified as buildings. However, in the real world, the relationship between buildings and cars is more complex. Cars tend to occur near and beside, or in front of, buildings, with both resting on the same ground plane. This relationship is inherently 3D, and what our models learned was merely a 2D approximation to it.

The work in Chapter 4 used a rough 3D reconstruction to begin to capture some of these notions. Nowhere in this thesis, however, did we actually model such relationships. I believe that this is one of the most promising directions for future work. Indeed, the work of Hoiem et al. [70] showed that merely estimating the camera height and angle from the location of the horizon line provided a large improvement in processing of the scene. Imagine what can be done with an accurate framing of the scene and understanding of the 3D layout of the objects.

Using an accurate 3D reconstruction method as a preprocessing step, the TAS model already has the capability to handle 3D spatial relationships, merely by defining relationship variables for the various 3D configurations for a planar superpixel region and the approximately planar visible surface for the various objects. The interaction also goes both ways, as more accurate recognition of the objects in a scene will lead to better 3D reconstructions, as in Chapter 4.

7.3 What comes next?

The problems described here remain as barriers to a satisfactory solution to the scene understanding problem. Despite this, the increasing availability of large amounts of image data, the promising work by many researchers in the field, and steady improvement of the probability tools and image processing tools provide grounds for optimism that progress can be made in this direction. Even once many of these problems at the **Interaction Layer** are solved, however, there is still a whole research thrust for converting from the image primitives and relationships into a symbolic understanding of the visual world. If successful, modeling at this layer will enable many of the applications of computer vision, and will allow computers and robots to interact with the world in ways not yet imagined.

Bibliography

[1] Hirotugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[2] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. In *SIGGRAPH*, volume 24, pages 408–416, 2005.

[3] Alexandru O. Balan and Michael J. Black. The naked truth: Estimating body shape under clothing. In *European Conference on Computer Vision (ECCV)*, pages 15–29, 2008.

[4] Alexandru O. Balan, Leonid Sigal, Michael J. Black, James E. Davis, and Horst W. Haussecker. Detailed human shape and pose from images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[5] Kobus Barnard, Pinar Duygulu, David Forsyth, Nando de Freitas, David M. Blei, and Michael I. Jordan. Matching words and pictures. *JMLR*, 3, 2003.

[6] H.G. Barrow and J.M. Tenenbaum. Recovering intrinsic scene characteristics from images. In *Computer Vision Systems*, pages 3–26, 1978.

[7] R. Basri, L. Costa, D. Geiger, and D. Jacobs. Determining the similarity of deformable shapes. *Vision Researc*, 38:2365–2385, 1998.

[8] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape context: A new descriptor for shape matching and object recognition. In *NIPS*, pages 831–837, 2000.

[9] A.C. Berg, T.L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages I: 26–33, 2005.

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

[11] T.O. Binford. Visual perception by computer. In *IEEE Conference on Systems and Control*, 1971.

[12] E. Borenstein and S. Ullman. Learning to segment. In *European Conference on Computer Vision (ECCV)*, pages Vol III: 315–328, 2004.

[13] Eran Borenstein, Eitan Sharon, and Shimon Ullman. Combining top-down and bottom-up segmentation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, page 46. IEEE Computer Society, 2004. ISBN 0-7695-2158-4.

[14] Gunilla Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(6):849–865, 1988. ISSN 0162-8828.

[15] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge, 2004.

[16] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23, 2001.

[17] J. D. Bremner, P. Randall, T. M. Scott, R. A. Bronen, J. P. Seibyl, S. M. Southwick, R. C. Delaney, G. McCarthy, D. S. Charney, and R. B. Innis. Mri-based measurement of hippocampal volume in patients with combat-related

posttraumatic stress disorder. *American Journal of Psychiatry*, 152(7):973–981, July 1995.

[18] 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. In *Tech report GIT-GVU-05-28*, 2005.

[19] Paul H. Calamai and Jorge J. More:9A. Projected gradient methods for linearly constrained problems. *Math. Program.*, 39(1):93–116, 1987. ISSN 0025-5610.

[20] P. Carbonetto, N. de Freitas, and K. Barnard. A statistical model for general contextual object recognition. In *ECCV*, 2004.

[21] Vincent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. In *ICCV*, pages 694–699, 1995.

[22] Gregory F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks (research note). *Artificial Intelligence*, 42(2-3):393–405, 1990. ISSN 0004-3702.

[23] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *ECCV*, volume 2, pages 484–498, 1998.

[24] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models: their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, 1995. ISSN 1077-3142.

[25] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley & Sons, New York, 1991.

[26] D.J. Crandall, P.F. Felzenszwalb, and D.P. Huttenlocher. Spatial priors for part-based recognition using statistical models. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages I: 10–17, 2005.

[27] D.J. Crandall and D.P. Huttenlocher. Weakly supervised learning of part-based spatial models for visual object recognition. In *European Conference on Computer Vision (ECCV)*, pages I: 16–29, 2006.

[28] Daniel Cremers, Florian Tischhäuser, Joachim Weickert, and Christoph Schnörr. Diffusion snakes: Introducing statistical shape knowledge into the mumford-shah functional. *Int. J. Comput. Vision*, 50(3):295–313, 2002. ISSN 0920-5691.

[29] Antonio Criminisi. Microsoft research cambridge object recognition image database (version 1.0 and 2.0)., 2004.

[30] Gabriela Csurka, Chris Dance, Jutta Willamowski, Lixin Fan, and Cedric Bray. Visual categorization with bags of keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision*, 2004.

[31] Navneet Dalal. *Finding people in images and videos*. PhD thesis, Institut National Polytechnique de Grenoble, july 2006.

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

[33] A. P. Dawid. Present position and potential developments: Some personal views: Statistical theory: The prequential approach. *Journal of the Royal Statistical Society*, 1984.

[34] F. Dellaert, S.M. Seitz, C.E. Thorpe, and S. Thrun. Structure from motion without correspondence. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages II: 557–564, 2000.

[35] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1), 1977.

[36] James Diebel and Sebastian Thrun. An application of markov random fields to range sensing. In *Advances in Neural Information Processing Systems*. MIT Press, 2005.

[37] I. L. Dryden and K. V. Mardia. Statistical shape analysis, 1998.

[38] Gal Elidan, Geremy Heitz, and Daphne Koller. Learning object shape: From cartoons to images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[39] Gal Elidan, Ian McGraw, and Daphne Koller. Residual belief propagation: Informed scheduling for asynchronous message passing. In *Uncertainty in Artificial Intelligence*, 2006.

[40] Mark Everingham, Andrew Zisserman, Christopher K. I. Williams, Luc J. Van Gool, Moray Allan, Christopher M. Bishop, Olivier Chapelle, Navneet Dalal, Thomas Deselaers, Gyuri Dorkó, Stefan Duffner, Jan Eichhorn, Jason D. R. Farquhar, Mario Fritz, Tom Griffiths, Frédéric Jurie, Daniel Keysers, Markus Koskela, Jorma Laaksonen, Diane Larlus, Bastian Leibe, Hongying Meng, Hermann Ney, Bernt Schiele, Cordelia Schmid, Edgar Seemann, John Shawe-Taylor, Amos J. Storkey, Sándor Szedmák, Bill Triggs, Ilkay Ulusoy, Ville Viitaniemi, and Jianguo Zhang. The 2005 pascal visual object classes challenge. In *MLCW*, 2005.

[41] J. f. Mangin, F. Poupon, E. Duchesnay, D. Rivire, and Service Hospitalier Frdric Joliot. Brain morphometry using 3d moment invariants. medical image analysis 8. *Medical Image Analysis*, 8:187–196, 2004.

[42] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[43] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient matching of pictorial structures. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 66–73, 2000.

[44] Pedtro F. Felzenszwalb and Joshua D. Schwartz. Hierarchical matching of deformable shapes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[45] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 264–271, 2003.

[46] R. Fergus, P. Perona, A. Zisserman, R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages I: 380–387, 2005.

[47] V. Ferrari, T. Tuytelaars, and L. Van Gool. Object detection by contour segment networks. In *European Conference on Computer Vision (ECCV)*, 2006.

[48] V. Ferrari and A. Zisserman. Learning visual attributes. In *Advances in Neural Information Processing Systems*, 2007.

[49] Vittorio Ferrari, Frederic Jurie, and Cordelia Schmid. Accurate object detection with deformable shape models learnt from images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2007.

[50] M. Fink and S. Ullman. From aardvark to zorro: A becnhmark for mammal image classification. *International Journal of Computer Vision*, 77:143–156, 2007.

[51] Michael Fink and Pietro Perona. Mutual boosting for contextual inference. In *NIPS*, 2003.

[52] George S. Fishman. *Monte Carlo: Concepts, algorithms, and applications.* Springer Series in Operations Research. Springer-Verlag, New York, 1996.

[53] David A. Forsyth, Jitendra Malik, Margaret M. Fleck, Hayit Greenspan, Thomas K. Leung, Serge Belongie, Chad Carson, and Chris Bregler. Finding pictures of objects in large collections of images. In *Object Representation in Computer Vision*, 1996.

[54] Nir Friedman. Learning belief networks in the presence of missing values and hidden variables. In *International Conference on Machine Learning*, pages 125–133. Morgan Kaufmann, 1997.

[55] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Readings in computer vision: issues, problems, principles, and paradigms*, 1987.

[56] S.N. Goodman. Toward evidence-based medical statistics. 2: The bayes factor. *Annals of Internal Medicine*, 130(12):1005–1013, 1999.

[57] Stephen Gould, Paul Baumstarck, Morgan Quigley, Andrew Y. Ng, and Daphne Koller. Integrating visual and range data for robotic object detection. In *ECCV Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications (M2SFA2)*, 2008.

[58] Stephen Gould, Jim Rodgers, David Cohen, Gal Elidan, and Daphne Koller. Multi-class segmentation with relative location prior. *International Journal of Computer Vision*, 2008.

[59] Kristen Grauman and Trevor Darrell. Pyramid match kernels: Discriminative classification with sets of image features. In *International Conference on Computer Vision*, October 2005.

[60] Abhinav Gupta and Larry S. Davis. Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers. In *European Conference on Computer Vision (ECCV)*, pages 16–29, 2008.

[61] A.R. Hanson and E.M. Riseman. Visions: A computer system for interpreting scenes. In *CVS78*, pages 303–333, 1978.

[62] Xuming He, Richard S. Zemel, and Miguel . Carreira-perpin. Multiscale conditional random fields for image labeling. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 695–702, 2004.

[63] David Heckerman, David M. Chickering, Christopher Meek, Robert Rounthwaite, and Carl Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *J. Mach. Learn. Res.*, 1:49–75, 2001.

[64] G. Heitz, G. Elidan, B. Packer, and D. Koller. Shape-based object localization for descriptive classification. In *Advances in Neural Information Processing Systems*, 2008.

[65] 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*, 2008.

[66] Geremy Heitz and Daphne Koller. Learning spatial context: Using stuff to find things. In *European Conference on Computer Vision (ECCV)*, pages 30–43, 2008.

[67] A. Hill and C. Taylor. A method of non-rigid correspondence for automatic landmark identification. In *Proceedings of the British Machine Vision Conference*, 1996.

[68] Aharon Bar Hillel, Tomer Hertz, and Daphna Weinshall. Efficient learning of relational object class models. In *International Conference on Computer Vision*, pages 1762–1769, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2334-X-02.

[69] A. E. Hoerl and R. W. Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 42(1):80–86, 2000.

[70] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Putting objects in perspective. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2137–2144, 2006.

[71] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Closing the loop on scene interpretation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2008.

[72] Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214, 1994.

[73] Timor Kadir and Michael Brady. Saliency, scale and image description. *International Journal of Computer Vision*, V45(2):83–105, November 2001.

[74] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:49–86, 1951.

[75] M.P. Kumar, P.H.S. Torr, and A. Zisserman. Obj cut. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages I: 18–25, 2005.

[76] Sanjiv Kumar and Martial Hebert. A hierarchical field framework for unified context-based classification. In *International Conference on Computer Vision*, pages 1284–1291, 2005.

[77] Christoph H. Lampert, Matthew B. Blaschko, and Thomas Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[78] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1265–1278, 2005.

[79] Mun W. Lee and I. Cohen. Proposal maps driven mcmc for estimating human body pose in static images. In *IEEE Computer Society Conference on Computer*

Vision and Pattern Recognition (CVPR), volume 2, pages II–334–II–341 Vol.2, 2004.

[80] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *ECCV'04 Workshop on Statistical Learning in Computer Vision*, pages 17–32, Prague, Czech Republic, May 2004.

[81] M. Leordeanu, M. Hebert, and R. Sukthankar. Beyond local appearance: Category recognition from pairwise interactions of simple features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[82] Anat Levin and Yair Weiss. Learning to combine bottom-up and top-down segmentation. In *European Conference on Computer Vision (ECCV)*, pages 581–594, 2006.

[83] Fei-Fei Li and Pietro Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR*, 2005.

[84] Zhi-Pei Liang and Paul C. Lauterbur. *Principles of Magnetic Resonance Imaging: A Signal Processing Perspective.* Wiley-IEEE Press, October 1999. ISBN 0780347234.

[85] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[86] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision (ijcai). In *International Joint Conference on Artificial Intelligence*, pages 674–679, April 1981.

[87] A.K. Mackworth. Interpreting pictures of polyhedral scenes. In *IJCAI*, pages 557–563, 1973.

[88] Mitchell P. Marcus, Beatrice Santorini, and Mary A. Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1994.

[89] D. Marr. Early processing of visual information. *Philosophical Transactions of the Royal Society of London*, B-275:483–524, 1976.

[90] D. Marr. Representing visual information: A computational approach. In *Computer Vision Systems*, pages 61–80, 1978.

[91] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *European Conference on Computer Vision (ECCV)*, page I: 128 ff., 2002.

[92] R. Mohan and R. Nevatia. Perceptual organization for scene segmentation and description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(6):616–635, June 1992.

[93] Kevin Murphy, Antonio Torralba, and William T. Freeman. Using the forest to see the trees: a graphical model relating features, objects and scenes. In *Advances in Neural Information Processing Systems*, 2003.

[94] Kevin P. Murphy, Antonio Torralba, Daniel Eaton, and William 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.

[95] Jurgen Neubauer, Patrick Mergell, Ulrich Eysholdt, and Hanspeter Herzel. Spatio-temporal analysis of irregular vocal fold oscillations: Biphonation due to desynchronization of spatial modes. *The Journal of the Acoustical Society of America*, 110(6):3179–3192, 2001.

[96] R. Nevatia and T.O. Binford. Structured description of complex objects. In *IJCAI*, pages 641–647, 1973.

[97] Aude Oliva and Antonio Torralba. The role of context in object recognition. *Trends Cogn Sci*, 2007.

[98] A. Opelt, A. Pinz, and A. Zisserman. A boundary-fragment-model for object detection. In *European Conference on Computer Vision (ECCV)*, pages II: 575–588, 2006.

[99] Andreas Opelt, Alex Pinz, and Andrew Zisserman. Incremental learning of object detectors using a visual shape alphabet. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 3–10, 2006.

[100] Andreas Opelt, Axel Pinz, and Andrew Zisserman. Fusing shape and appearance information for object category detection. In *Proceedings of the British Machine Vision Conference*, 2006.

[101] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, San Francisco, CA, 1988.

[102] Mukta Prasad and Andrew Fitzgibbon. Single view reconstruction of curved surfaces. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1345–1354. IEEE Computer Society, 2006. ISBN 0-7695-2597-0.

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

[104] Adrian E. Raftery and Steven Lewis. How many iterations in the gibbs sampler. In *Bayesian Statistics*, pages 763–773. Oxford University Press, 1992.

[105] D. Ramanan and D. A. Forsyth. Finding and tracking people from the bottom up. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages II–467–II–474 vol.2, 2003.

[106] D. Ramanan and C. Sminchisescu. Training deformable models for localization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 206–213, 2006.

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

[108] A. Rosenfeld, R.A. Hummel, and S.W. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man, and Cypernetics*, 6(6):420–433, June 1976.

[109] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: A database and web-based tool for image annotation. Technical report, Tech. Rep. MIT-CSAIL-TR-2005-056, Massachusetts Institute of Technology, 2005.

[110] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[111] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Learning 3-d scene structure from a single still image. In *IEEE PAMI*, 2008.

[112] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999. ISSN 0885-6125.

[113] F. Schroff, A. Criminisi, and A. Zisserman. Object class segmentation using random forests. In *Proceedings of the British Machine Vision Conference*, pages xx–yy, 2008.

[114] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[115] Thomas B. Sebastian, Philip N. Klein, and Benjamin B. Kimia. Recognition of shapes by editing their shock graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(5):550–571, 2004. ISSN 0162-8828.

[116] J. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1998.

[117] S. Shah and J. K. Aggarwal. Hierarchical multifeature integration for automatic object recognition in forward looking infrared images. *Lecture Notes in Computer Science*, 1611, 1999.

[118] J. Shotton, A. Blake, and R. Cipolla. Contour-based learning for object detection. In *International Conference on Computer Vision*, 2005.

[119] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conference on Computer Vision (ECCV)*, pages I: 1–15, 2006.

[120] Amit Singhal, Jiebo Luo, and Weiyu Zhu. Probabilistic spatial context models for scene content understanding. In *CVPR*, 2003.

[121] Erik B. Sudderth, Antonio Torralba, William T. Freeman, and Alan S. Willsky. Depth from familiar objects: A hierarchical model for 3d scenes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[122] Charles Sutton and Andrew McCallum. Joint parsing and semantic role labeling. In *In Conference on Natural Language Learning (CoNLL)*, 2005.

[123] M.F. Tappen, W.T. Freeman, and E.H. Adelson. Recovering intrinsic images from a single image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(9):1459–1472, September 2005.

[124] A. Thayananthan, B. Stenger, P. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003.

[125] A. Torralba, K. Murphy, W. Freeman, and M. Rubin. Context-based vision system for place and object recognition. In *International Conference on Computer Vision*, 2003.

[126] Antonio Torralba. Contextual priming for object detection. *International Journal of Computer Vision*, 53(2), 2003. ISSN 0920-5691.

[127] Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Contextual models for object detection using boosted random fields. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1401–1408, Cambridge, MA, 2005. MIT Press.

[128] Antonio Torralba and Aude Oliva. Statistics of natural image categories. In *Network: Computation in Neural Systems*, pages 391–412, 2003.

[129] Zhuowen Tu. Auto-context and its application to high-level vision tasks. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[130] Zhuowen Tu, Xiangrong Chen, Alan L. Yuille, and Song chun Zhu. Image parsing: Unifying segmentation, detection, and recognition. In *International Journal of Computer Vision*, pages 18–25, 2003.

[131] P. Viola and M. Jones. Robust real-time face detection. In *International Conference on Computer Vision*, 2001.

[132] Martin J. Wainwright, Tommi S. Jaakkola, and Alan S. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on Information Theory*, 49:2003, 2003.

[133] D. Waltz. Understanding line drawings of scenes with shadows. In *Psychology of Computer Vision*, pages 19–91, 1975.

[134] Yang Wang and Greg Mori. Boosted multiple deformable trees for parsing human poses. *Human Motion Understanding, Modeling, Capture and Animation*, pages 16–27, 2007.

[135] Yang Wang and Greg Mori. Learning a discriminative hidden part model for human action recognition. In *Advances in Neural Information Processing Systems*, 2008.

[136] J. Winn and J.D.J. Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages I: 37–44, 2006.

[137] Lior Wolf and Stanley Bileschi. A critical view of context. *IJCV*, 69(2), 2006.

[138] David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[139] Tao Xiang and Shaogang Gong. Optimising dynamic graphical models for video content analysis. *Computer Vision and Image Understanding*, 112(3):310–323, 2008. ISSN 1077-3142.

[140] L. Yang, P. Meer, and D.J. Foran. Multiple class segmentation using a unified framework over mean-shift patches. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[141] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Constructing free energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51:2282–2312, 2005.

[142] Jonathan S. Yedidia, Yige Wang, Yige Wang, Juntan Zhang, Juntan Zhang, Marc Fossorier, and Marc Fossorier. Reduced latency iterative decoding of ldpc codes. In *IEEE Conference on Global Telecommunications (GLOBECOM)*, 2005.

[143] Stella X. Yu and Jianbo Shi. Object-specific figure-ground segregation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 39–45. IEEE, 2003.

[144] A. Yuille and P. Hallinan. Deformable templates. *In A. Blake and A. Yuille, editors, Active Vision, MIT Press.*, pages 21–38, 1992.