PART-BASED STATISTICAL MODELS FOR VISUAL OBJECT CLASS RECOGNITION

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

David James Crandall

August 2008

PART-BASED STATISTICAL MODELS FOR VISUAL OBJECT CLASS RECOGNITION

David James Crandall, Ph.D.

Cornell University 2008

Object class recognition is a central problem of computer vision with broad potential application in image understanding, content-based retrieval, and surveillance. Unlike traditional object recognition where the goal is to detect specific objects, object class recognition aims to detect instances of broad object categories like cars, airplanes, bicycles, and people. This task is challenging because members of an object class may vary widely in appearance; for example, the "car" class includes many makes, models, styles and colors. In addition, class recognition must cope with usual sources of visual variation including viewpoint, illumination, and scale changes.

In this thesis we describe a family of part-based probabilistic models for object class recognition. These models allow for efficient exact inference, unlike most other approaches which rely on feature detection and approximating heuristics to make inference tractable. The object models are hierarchical in nature, allowing evidence at multiple image scales to be combined in making recognition decisions. We show how the multiscale models can be augmented to incorporate local context from the surrounding scene. We also present learning algorithms of various degrees of supervision, including a weakly-supervised algorithm that requires only a set of images known to contain the object. Experimental results demonstrate state-of-the-art performance on challenging datasets of unconstrained consumer images.

BIOGRAPHICAL SKETCH

David Crandall received the M.S. and B.S. degrees in Computer Science and Engineering with highest honors from the Pennsylvania State University in 2001. Before joining the Ph.D. program at Cornell University, he was a Senior Research Scientist in the research labs of Eastman Kodak Company in Rochester, NY.

To my parents, Ron and Sally.

ACKNOWLEDGEMENTS

Only a tiny fraction of the world's population has the chance to pursue a doctoral degree. I feel very lucky to have had this opportunity, and I am grateful to all those who made it possible.

I would like to thank my advisor and champion, Dan Huttenlocher, for his wholehearted support throughout the last five years. He has always been ready with gentle guidance, honest feedback, and plenty of ideas. I thank Claire Cardie for her suggestions on my research and on the drafts of this thesis. I am grateful to my minor advisor, John Henderson, for enthusiastically helping to combine my interests in computer science and Maya epigraphy. Pedro Felzenszwalb was like an unofficial fourth committee member; most of the best ideas in this thesis originated in one way or another from him.

Rangachar Kasturi, my master's degree advisor at Penn State, introduced me to computer vision in the first place. Without his kindness and encouragement I would have never discovered the joy of research. I would like to thank my managers and colleagues at Kodak for their support of my career and of my decision to return to graduate school. Jiebo Luo, John Lacek, Bob Gray, and Ed Giorgianni stand out in particular.

I am grateful to the faculty and staff of Cornell for the opportunity to study at one of the world's great universities. Every class I have taken here has been amazing; for their commitment to teaching I thank Ken Birman, Claire Cardie, Rebeca Franqui, John Henderson, Dan Huttenlocher, Dexter Kozen, Andrew Myers, Keshav Pingali, Mary Roldán, Jayavel Shanmugasundaram, and Éva Tardos in particular.

The experiments discussed in this thesis required large amounts of compute resources, typically involving dozens of machines and hundreds of gigabytes of data.

This would have all been impossible without the tireless efforts of the Computing Facilities Support (CFS) group. In particular I thank Jodie Sprouse, who patiently dealt with the overloaded processors, full disks, kernel panics, crashed file servers, and all the other havoc that my programs inadvertently (yet frequently) created.

I am grateful for the financial support that I received throughout my graduate studies. This thesis is based upon work supported in part by the National Science Foundation (NSF) under a Graduate Research Fellowship and grant IIS-0629447, and by a grant from Eastman Kodak Company. Any opinions, findings, and conclusions or recommendations expressed are those of the author and do not necessarily reflect the views of the sponsoring institutions.

The last five years have not been without major hurdles and considerable angst; I am grateful to everyone who helped me through. In particular I would like to thank Dan Cosley, Jon Kleinberg, Xiangyang Lan, Yunpeng Li, Ritch Savin-Williams, Casey Smith, and Sid Suri for many helpful discussions about computer science, research, academia, and life. Saúl Blanco was my mathematical tutor whenever I had notation I couldn't understand or a theorem that I couldn't prove. He was also a friend when I needed one most. Beth Howard cheerfully helped me navigate through Cornell's many administrative mazes. Shawna Crandall was my link to the real world and the fun side of life; I don't know where I would be without her. Susan Crandall was my strength and inspiration more often than she knows. One day, every child will be able to pursue an advanced degree — and it will be because of her.

Finally I thank my parents, who mean more to me than words can possibly express. I dedicate this thesis to you, my two heroes.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Today's computers perceive through simplistic input devices like keyboards and mice, unable to sense the world around them as humans do. The possibilities of *visually perceptive* machines are particularly tantalizing: cars that avoid collisions, photo albums that organize themselves, X-ray machines that detect cancer, search engines that index Internet imagery, security guards that never tire. A key problem in realizing these applications is recognizing the *objects* that appear in an image or a video. The challenge is in the sheer number and diversity of objects in our world: airplanes, televisions, bridges, dogs, mailboxes, colanders, vacuum cleaners — just to name a few.

Object recognition has been studied for decades but much of that work has focused on *specific object detection* [35, 54, 56, 63]. In this problem the goal is to recognize a specific object instance, given a library of possible objects — for example recognizing that the object in Figure 1.1(a) is the same toy plane as in the lower left of Figure 1.1(b). This is a challenging problem because an object's appearance can vary dramatically across different images, due to factors like illumination differences, viewpoint changes, object non-rigidity, occlusion from other objects, complex and confusing backgrounds, image compression artifacts, etc. However the specific object recognition task is simplified by the fact that the algorithm is trying to match exactly the same object — the same toy plane.

A more general problem is *object class recognition* [14, 25, 26, 32, 33, 58], in which the goal is to recognize instances of broad object categories such as cars, bicycles, and airplanes. In addition to sources of variation like illumination and viewpoint, an additional challenge is the large degree of variation in visual appear-

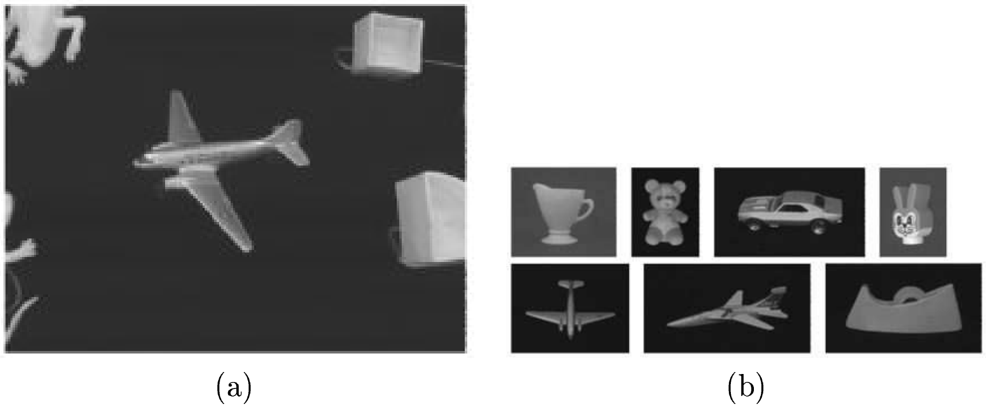


Figure 1.1: An example of the specific object recognition problem, from [71].

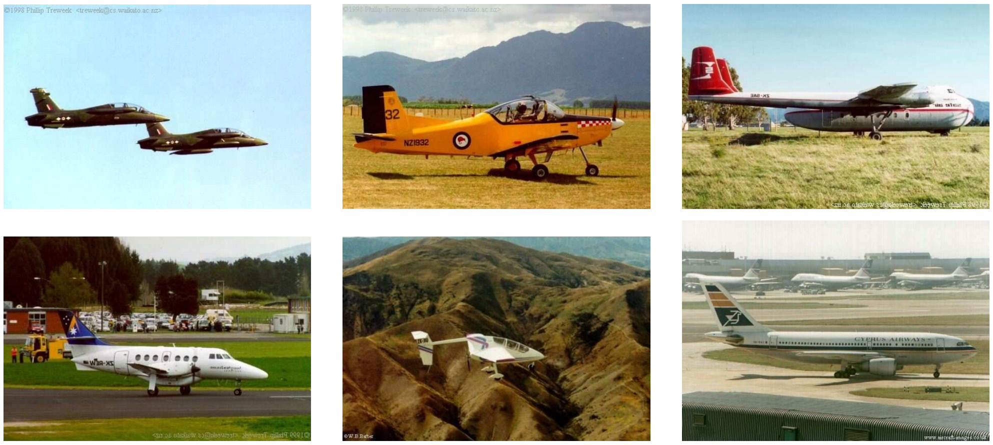


Figure 1.2: Some instances of the "airplane" object class.

ance across specific objects of the same class. For example the "airplane" class encompasses a wide variety of aircraft including turboprops, jumbo jets, experimental aircraft, fighter jets, biplanes, stealth bombers, and many others. Even within each of these types of aircraft there is variation due to manufacturer, model, age, color, etc. Some examples are shown in Figure 1.2.

In many applications of computer vision the ultimate goal is to understand scenes that naturally occur in the world. In this context, object class recognition is often a more relevant problem than specific object recognition. An autonomous vehicle trying to navigate through city streets, for example, needs to detect cars of any type in order to avoid collisions; the ability to detect only white 2002 Subaru Foresters is of very little use. Thus we argue that object class recognition is an important problem in machine perception and make it the focus of this thesis.

1.1 Object category models

Recognition of certain object categories has been studied extensively; in particular, hundreds of methods have been proposed for detecting cars, people, and faces over the years because of the importance of these particular object classes in surveillance and biometrics applications [37, 80, 89]. Most of this work has relied on hand-crafted models that were designed specifically for one particular object category. In these approaches, vision researchers use their intuition about an object class in order to design a set of features and a model.[1] While the performance achieved by these detection methods can be quite good, the human cost of producing these models is enormous: for every object class of interest a human expert must create an object model from scratch. Thus crafting detection systems by hand is not a realistic way of building models for the hundreds of thousands of object categories that exist in our world.

In this thesis we take the alternative approach of developing techniques that are

[1] These approaches often involve some learning component to choose parameter values, however the choice of model and features are typically so specific to one object class that satisfactory parameters cannot be learned for other object classes.

general enough to be applicable to a wide range of object categories. These general models have parameters that are learned for a particular object category; thus the same generic model detects cars when loaded with the car-specific parameters and detects bicycles when loaded with the bicycle parameters. The class-specific parameters can be learned automatically from training images. This approach of generic models with learnable parameters has become very popular within the class recognition community over the last decade [26].

1.1.1 Deformable part-based object models

In choosing among the possible modeling strategies, we need a framework that is general enough to handle a wide variety of object classes, with a parameter space that is rich enough to capture the unique appearance characteristics of each specific class, while also being flexible enough to tolerate the large degree of visual variation typical of most object classes.

Part-based deformable modeling is one approach that has been successfully applied to the category recognition problem [2, 30, 32, 33]. These models arise from the observation that many interesting objects consist of a set of individual parts that are arranged in some characteristic geometry. Faces, for example, consist of eyes, a nose, and a mouth, while airplanes consist of wings, a fuselage, and a tail. While the appearance of an object class might vary dramatically across different instances, the appearance of the small local parts is less variable. Part-based models exploit this observation by decomposing an object into its component parts and then modeling the local appearance of each part individually. These models also include some constraints on the relative spatial configuration of the parts.

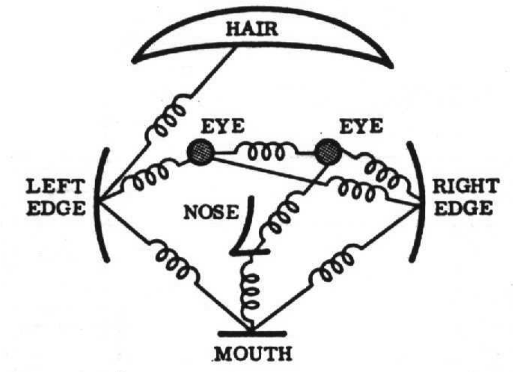


Figure 1.3: Diagram of the part-based deformable models of Fischler and Elschlager [36].

The concept of part-based deformable models dates back at least to 1973 in the work of Fischler and Elschlager [36]. They imagined an appearance model for each individual part, along with a spatial consistency model that intuitively consists of springs connecting some of the parts, as depicted in Figure 1.3. Recently these deformable part models have attracted renewed attention [2, 30, 32, 33]. These approaches differ in the type and amount of spatial constraint they put on the parts — that is, how many "springs" are used and how they are arranged. There is no clear best choice for the form of the spatial model; in general, richer models give a more faithful representation of the object at an increased computational cost during inference and learning.

In this thesis we show that many of the seemingly disparate approaches to part-based flexible models in the literature can be thought of as specific members of the same family, which we call *k-fans*. Studying these different approaches within the same framework will allow us to explicitly consider the trade-off between modeling power and computational complexity.

1.1.2 Statistical object models

As with many problems in computer vision, object recognition involves dealing with large degrees of uncertainty. State-of-the-art object models do not capture all of the nuances of an object and so it is possible for non-objects to fit the model very well, or for actual objects to fit the model poorly. At the same time, photographs contain noise due to factors like image compression and imperfect camera sensors; even an ideal camera would introduce ambiguity because it projects a three-dimensional scene onto the two-dimensional plane of the imaging sensor. Thus in object recognition uncertainty arises because an object model is not an exact representation of an object's visual appearance and a digital image is not an exact representation of a real-world scene.

Probabilistic models (or *statistical models*) are a natural way of handling this uncertainty, and Bayesian statistics in particular has become popular for addressing a wide range of computer vision problems [43, 49, 50, 78, 79, 88, 83]. In a probabilistic model there are random variables that describe the state of the world and a set of probability distributions that captures the correlations between these variables. Some of the random variables are directly observable while other variables are not. Bayesian statistics gives a probabilistic framework for inferring values of (or distributions over) the unobservable (or *hidden*) variables given the values of the observable variables and prior knowledge about the world. In the specific case of object detection, we can think of an image as a random variable that can be observed while an object's location in the image is a *hidden* random variable that cannot be directly observed.

More concretely, consider a part-based model with a set of n parts $V = \{v_1, \ldots, v_n\}$. The location of an object in an image is given by a configuration of its parts $L = (l_1, \ldots, l_n)$, where l_i is the location of the ith part.[2] We can view the image I as a set of observable random variables encoding the color and intensity at each pixel, while the configuration L of the object is not directly observable. Using Bayes' law, the probability that the object is in a particular configuration given an image and a fixed set of model parameters M is,

$$P_M(L|I) = \frac{P_M(I|L)P_M(L)}{P_M(I)}. \tag{1.1}$$

Here, $P_M(I|L)$ is the *likelihood* of observing image I given that a particular configuration of the object occurs in the scene, while $P_M(L)$ and $P_M(I)$ are the *prior* probability distributions on object configuration and images, respectively. Intuitively, Bayesian inference combines what is known about the random variables ahead of time (the prior distribution) with the states of the observable variables (through the likelihood distribution) to produce an estimate of the hidden variables (the posterior distribution).[3]

1.1.3 Inference

Now we can formulate object recognition as a statistical inference problem on this Bayesian framework. Depending on the application, there are two main tasks that are often demanded of an object recognition system: classification and localization.[4] Both of these can be formulated in terms of the statistical framework.

[2] We mean "location" in a very general way; the location space could have arbitrary dimensionality specifying the part's position, orientation, scale, etc. In this thesis we focus on translation-invariant recognition and thus the location space is simply position in the two-dimensional image, $l_i = (x_i, y_i)$, but the statistical framework and the algorithms we discuss generalize to an arbitrary number of dimensions.

[3] The prior over images $P_M(I)$ is notoriously difficult to define [84]; the usual solution to this problem is to assume a uniform prior (i.e. all images are equally likely), and we take that approach here.

[4] We purposely avoid using the terms *detection* and *recognition* here, as their meanings vary across the literature. For example, [19] uses *detection* to describe what we call localization, while detection in [32] is used to refer to what we call classification.

The *classification* problem is to decide if the image has an instance of the object (which we call hypothesis w_1) or if no such instance exists (hypothesis w_0). To make this decision it is natural to consider the Bayes' factor [46],

$$\frac{P_M(I|w_1)}{P_M(I|w_0)}, \tag{1.2}$$

and compare it to a threshold to make the hard "yes" or "no" classification decision.

In *localization* we seek to find the location of the object in an image. From a statistical perspective the task is to find a configuration of the parts that maximizes the posterior probability,

$$L^* = \arg\max_L P_M(L|I).$$

Depending on the application we may require multiple hypotheses of the object's location instead of a single estimate. In this case it is natural to *sample* multiple configurations from the posterior distribution.

1.1.4 Independence assumptions and graphical models

The computational complexity of the inference tasks just described is highly dependent on the form of the likelihood distribution $P_M(I|L)$ and the spatial prior $P_M(L)$. There are two important decisions made when designing a model in this statistical framework that influence the computational cost: the choice of distribution and the independence assumptions.

In choosing the type of distribution, there is a trade-off between practicality and accuracy of the model. Non-parametric distributions allow for arbitrary types of correlations to be captured accurately, but they are difficult to learn and represent due to the large number of parameters; thus parametric distributions such as

Gaussians are often more practical. In this thesis we present two versions of the inference algorithms: one that makes no assumptions on the type of distribution, and another that assumes Gaussian distributions but has a lower computational cost.

The second important decision in designing a statistical model is the set of independence assumptions that are made about the random variables. It is possible to assume that all of the random variables are directly correlated with one another (i.e. no independence assumptions) but exact inference is intractable in that case. Thus one must either settle for an approximate inference algorithm that is computationally tractable but does not give an exact answer, or one must design the model such that some of the random variables are conditionally independent from one another.

For example, most approaches assume that the part appearances are independent of one another; that is, it is assumed that $P_M(I|L)$ factors into a product of functions each of which depends on the location of at most one part,

$$P_M(I|L) = Z(I) \prod_{v_i \in V} P_M(I|l_i), \tag{1.3}$$

where $Z(I)$ is a term that does not depend on the location of any part.[5] On the other hand independence assumptions about the variables in the prior distribution $P_M(L)$ vary considerably across different approaches. These differences reflect different trade-offs that can be made in order to balance computational tractability with expressiveness of the model.

Probabilistic graphical models [51] provide a framework for representing and reasoning about these independence assumptions. A graphical model is a graph

[5] A notable exception is the Patchwork of Parts model of [2] which relaxes this assumption in order to better handle overlapping parts. We discuss the POP model in more detail in Section 4.1.3.

having a node for every random variable in the statistical model. An edge connects two nodes p and q if the model assumes that there is an explicit dependency between the two random variables. A graphical model is called *directed* or *undirected* depending on the type of its edges; directed edges imply causality between variables while undirected edges only imply correlation. In our application to object recognition we have no interest in modeling causal relationships, so the graphical models considered in this thesis are undirected. Undirected graphical models are also called Markov Random Fields [10].

Given a probabilistic graphical model, one can determine the independence relationships of the random variables by examining the edges and paths in the graph. Two random variables p and q are *independent* if there is no path of edges between the two corresponding nodes in the graph; that is, they are independent if p and q are unreachable from one another. Two variables are *conditionally independent* given the values of a set of variables R if all paths in the graph between p and q involve a node in R. In other words, two variables are conditionally independent if they are reachable in the original graph but unreachable if the set of nodes R were to be removed [51].

1.1.5 Inference: unified versus bottom-up

The junction tree algorithm [51] provides a technique for exact inference on probabilistic graphical models. The running time of the algorithm depends on the size of the maximal cliques in the graphical model.[6] In particular the algorithm is linear in the number of nodes but exponential in the size of the largest maximal

[6] A *clique* is a subset of the nodes of a graph such that there is an edge between all pairs of nodes. A *maximal clique* is a clique for which it is not possible to create a larger clique by adding another node.

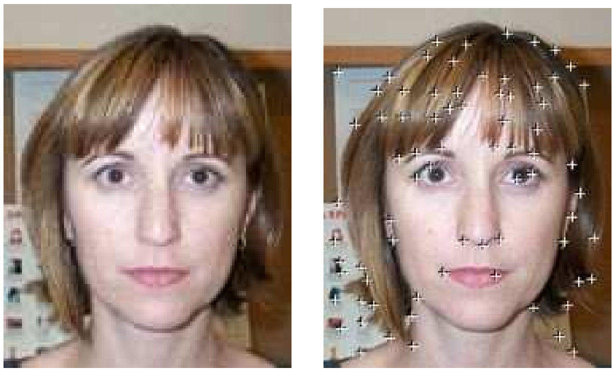


Figure 1.4: Result of feature detection (right) on a sample image (left).

clique. Thus there is a trade-off between the richness of the statistical model and the computational complexity of inference. Richer models have more edges and larger cliques but inference becomes exponentially harder.

One way this problem has been addressed in the context of object recognition is to use rich models but an approximate inference algorithm [32, 33]. A common approach is to first run a *feature detector* on the image to identify salient points of interest such as corners. Then only object configurations that involve this small subset of image points are considered during inference, thus tremendously reducing the computational cost. This can be thought of as a *bottom-up* inference strategy, in which good locations for the individual object parts are estimated individually by finding local maxima of $P_M(I|l_i)$, and then some subset of those high-likelihood parts is combined together to produce an object localization. The output of a feature detector on a sample image is shown in Figure 1.4.

While such an approach makes inference on rich statistical models tractable, a disadvantage is that it makes hard intermediate classification decisions that can lead to poor recognition results. For example, consider an image in which one part of the face, such as the right eye, is occluded by another object (such as a hat). The feature detector might not fire anywhere near the right eye because the local image data there is inconsistent with the expected appearance of an eye. The bottom-up inference approach is likely to fail in this case.

A major difference between this thesis and most other work in object recognition is that we do not employ bottom-up inference or feature detection. Instead we use a *unified inference* approach that postpones hard classification decisions until the last stages of the algorithm, after evidence from the entire model has been combined. Instead of using feature detectors, our approach instead uses feature *operators* that give a response for $P_M(I|l_i)$ at every pixel in the image. Thus we perform *exact* inference; during localization we find a configuration that actually maximizes the posterior $P_M(L|I)$, and during classification we compute the exact value of $P(I|w_1)$. A benefit of this approach is that it naturally handles occlusion. In the above case of an occluded eye, for example, our models would be willing to hallucinate the eye in front of the hat if doing so maximized the posterior probability.

However this exact inference is potentially more expensive depending on the type of object model that is used. To reason about this trade-off between computational cost and representational power, in this thesis we introduce a family of models called k-fans that explicitly parameterizes the richness of spatial information. We find that greater spatial constraint does increase detection performance, but that exact inference on relatively simple spatial models performs as well or better than the more complex models for which inference is not tractable.

1.2 Learning

Given the enormous number of object classes in the world, it is very appealing to learn object models automatically with a minimal amount of human involvement. An automatic learning algorithm is given a set of training images and some amount of ground truth data, where the quantity and type of ground truth depends on the degree of human supervision that is available. In this thesis we present several learning algorithms for our object models, each requiring a different degree of human supervision. We present a fully-supervised algorithm that requires hand-labeled part locations, a partially-supervised algorithm that requires bounding boxes around objects, and a weakly-supervised algorithm that requires nothing except for a boolean value on each image indicating the presence or absence of the object class. Thus the appropriate learning algorithm can be selected based on the resources and requirements of a given application.

In terms of the statistical models framework that we presented above, to learn an object model we wish to find a set of model parameters that maximizes the probability of the training data,

$$M^* = \arg\max_M \prod_{i=1}^{T} P_M(I_i|w_1),$$

where $\{I_1, \ldots, I_T\}$ are images that contain the object of interest. A key difference between our approach and others in the literature is that we do not use feature detection during learning, and we do not assume that the part appearance models are fixed *a priori*. Instead, we learn the part appearance and spatial portions of the model *simultaneously*.

We show that the combination of our weakly-supervised learning algorithm and exact inference algorithms outperform other recent part-based spatial models, such as the constellation models of [32, 33]. In fact, our weakly-supervised algorithm gives consistently better models than the fully-supervised learning approach. Thus our weakly-supervised learning algorithm not only requires less human effort but also produces higher-quality models!

1.3 Hierarchical scene context models

It is well-known that the human visual system integrates contextual cues from an entire *scene* [4] in performing high-level visual tasks. In object recognition, these cues include constraints on the sizes and relative layout of objects in a scene. Other cues arise from physical laws that make some scenes much more likely than others. These constraints differ from object to object. For instance a car is almost always sitting on some horizontal support surface, usually a road, and is often accompanied by pedestrians and other vehicles; an airplane, on the other hand, is often seen in the sky near clouds but almost never appears on the roads of a busy city scene.

Despite its importance in human vision, most object recognition algorithms for computers do not make use of scene-level contextual information, instead modeling only the appearance of an individual object. In this thesis we show how to exploit scene context in order to improve recognition results, by building hierarchical models of objects and scenes. We integrate image information at multiple scales into the same statistical framework that is used for object recognition. This lets us perform unified inference on the entire combined scene and object model simultaneously; that is, we delay making any hard classification decisions until after evidence from both object appearance and scene context have been combined.

The contextual models are learned automatically from training data.

1.4 Experimental evaluations

Given the simplifications and modeling error inherent in computer vision, no discussion of the object class recognition problem is complete without a thorough experimental evaluation. For several years the object class recognition research community has relied on evaluation datasets consisting of photos taken by vision researchers, such as Caltech-4 [32], Caltech-101 [28], UIUC [70], MSR [17], and Graz [58]. These datasets provided a convenient benchmark for testing object detection algorithms on common datasets and are probably to a large degree responsible for the impressive progress on category recognition that has occurred over the last five years. However these datasets were all collected by computer vision researchers and suffer from biases that make them unrepresentative samples of real-world images [62].

Recently more challenging datasets of *consumer images* have become available as part of the PASCAL Visual Object Class (VOC) Challenge competitions [25, 26]. The VOC dataset consists of nearly 15,000 images downloaded from online photo-sharing sites and is thus a large, realistic sample of unconstrained consumer images. Objects in the dataset appear with unconstrained position, scale, viewpoint, illumination, occlusion, and orientation. In addition to the image data, the competition supplies ground truth data and a common experimental protocol, so that results from different papers on this dataset can be directly compared.

The focus of most work in the object category recognition literature has been on the classification task. As an illustration, the 2006 PASCAL VOC challenge had 23 entries in the classification competition but only 6 entries in the localization competition [25]. Thus most papers present results only for the classification task (e.g. [15, 21, 18, 32, 33, 58, 73, 86]). While classification is an important task, localization is also highly important for many applications. In this thesis we present thorough evaluations for both the classification and localization tasks.

We use the VOC data for testing our approach to category recognition because we believe it to be the largest and most challenging image set that is publicly available. We present recognition results on a dozen object categories and compare our results with the teams that entered the competition. The results show that our approach using weakly-supervised models and k-fan spatial models outperforms other recent approaches using deformable part-based models, and gives comparable results to the best object category recognition algorithms published to date.

1.5 Outline of the thesis

The remainder of the thesis is organized as follows. In Chapter 2 we present a family of spatial models called k-fans and show how several recent approaches to part-based object recognition can be explained in terms of this family. We turn to the inference problem in Chapter 3, showing how to perform classification and localization efficiently on k-fan models. The focus of these two chapters is on the spatial portion of the object models and make no assumptions about the appearance model (other than that the image likelihood factors as a product over parts, as discussed above). In Chapter 4 we present several possible choices of feature operators and show how to compute them efficiently. We discuss how to learn our object models in Chapter 5, presenting several algorithms based on the amount of human supervision that is available. In Chapter 6 we show how contextual evidence from the scene can be included in the models, by building hierarchies at multiple image scales. We present detailed experimental results in Chapter 7, and finally conclude in Chapter 8.

CHAPTER 2

STATISTICAL PART-BASED OBJECT MODELS

Probabilistic graphical models are a powerful technique for representing statistical dependencies between random variables. These models have become a popular approach to solving problems in computer vision over the last decade [43, 50, 49, 78, 79, 88, 83]. For object recognition, graphical models have been applied in the context of *part-based object models.* The central idea behind these models is that many objects consist of a small set of well-defined parts. For example, a face consists of eyes, a nose, and a mouth; a bicycle consists of wheels, a seat, and handlebars. Although the appearance of an object class may vary dramatically across different images (due to pose and viewpoint variations as well as appearance variation within the object class), the local appearance of an individual part is usually much more consistent. Part-based object models exploit this observation by modeling the local appearance of each part individually. The overall geometric structure of the object is enforced by a separate spatial model that captures the relative configuration of the parts.

This chapter discusses part-based models for object recognition using probabilistic graphical models to capture inter-part spatial dependencies, using the general statistical framework we presented in Section 1.1.2. This framework is common to a number of recent object recognition approaches, as we show in a literature survey in Section 2.1. We argue that many of these recent approaches to graphical object models are in fact specific instances of a general family which we call *k-fans.* We introduce this family of spatial models in Section 2.2.

We purposely do not discuss the part appearance models in this chapter because the spatial models presented here and the inference techniques in Chapter 3 are largely independent of them. We simply assume that the image likelihood factors as in equation (1.3) and that there is some efficient way of computing $P_M(I|l_i)$ for each part v_i. Thus we present the spatial models and inference algorithms in a general way and delay covering the details of the appearance models until Chapter 4.

2.1 Related work

The statistical framework presented in the last chapter has become popular for part-based object recognition. The major distinguishing characteristic between different approaches is the form of the prior and likelihood distributions that they assume. In this section we review some of these recent approaches.

2.1.1 Bag-of-parts models

Bag-of-parts models [18, 73, 86] treat an object as an unstructured set of parts with no geometric constraints. This approach is analogous to the bag-of-words models in the information retrieval literature [69] which represent a document as an unordered set of words. Bag-of-parts models assume that the part locations are independent of one another (the naïve Bayes assumption), so that in the context of our framework, the prior factors as a product over parts,

$$P_M(L) = \prod_{v_i \in V} P_M(l_i).$$

The graphical model representing this factorization is an empty graph.

The detection and localization problems are particularly easy with this spatial prior. For localization it is only necessary to maximize $P_M(I|l_i)P_M(l_i)$ independently for each part v_i. This can be done in $O(nh)$ time for a model with n parts and an image with h possible locations for each part. But while bag-of-parts models yield computationally-tractable recognition and learning procedures, they are unable to accurately represent multi-part objects since they capture no spatial information. In practice these models have been found to work well for the classification task but poorly on localization, suggesting that spatial models are needed to accurately estimate an object's position and extent [26].

2.1.2 Constellation models

Another option is to make no independence assumptions on the locations of different parts. Constellation models [5, 6, 32, 33] take this approach by using a full multivariate joint Gaussian model for the spatial distribution $P_M(L)$,

$$P_M(L) = \mathcal{N}(L|\mu, \Sigma).$$

where Σ is an arbitrary covariance matrix. The corresponding graphical model in this case is a complete graph, indicating that the position of every node is directly correlated with the position of every other node. Such a graphical model without any conditional independence restrictions is called *saturated* [51].

It is not known how to perform exact inference using this spatial prior efficiently; the best known inference algorithm takes time exponential in the number of parts. Since exact inference with constellation models is therefore intractable, approaches in the literature instead compute an approximate maximization of the posterior by using various heuristics. For example, feature detection is typically used to constrain the set of possible configurations. The idea behind feature detection is to identify a small number of promising *interest points* in the image, and then only consider object configurations that involve those sparse feature points. Various interest point detectors are popular in the literature including Kadir & Brady [48], KLT corners [81], and SIFT [54].

More formally, a feature detector can be thought of as identifying a small set of high-likelihood locations of $P_M(l_i|I)$ for each part. Let S_i denote the set of interest points identified for part v_i, where $|S_i|$ is usually less than a few hundred. Then instead of maximizing the posterior over the complete configuration space I^n, where n is the number of parts in the model, the maximization is instead performed over the small subset $S_1 \times S_2 \times ... \times S_n$.

Thus feature detection is used as a heuristic to reduce the search space and make inference tractable. Because of this heuristic the inference algorithm is not guaranteed to produce a maximization or sum of the posterior that is optimal or even provably approximate. Particularly problematic is the case when a part of an actual object is not detected by the feature detector, because of occlusion for example. To handle this case, the constellation model inference algorithm [32] allows some parts of the model to not be matched anywhere in the image with some small probability. Thus inference with feature detection becomes a combinatorial matching problem: a subset of the parts of the object model are matched with a subset of the detected feature points in the image, so as to find an approximate maximization of the posterior.

2.1.3 Pictorial structures

In pictorial structures models [30] the graphical model for the spatial prior is a tree. Efficient exact inference procedures are known in this case; in particular the detection and localization problems can be solved in $O(nh^2)$ time using dynamic programming. Moreover, in the case when the spatial models are Gaussian, these problems can be solved exactly in $O(nh \log h)$ time and can be closely approximated in $O(nh)$ time — the same asymptotic time required by inference on bag-of-parts models.

The family of spatial models and inference algorithms in this thesis were largely inspired by the pictorial structures models. Our models can be thought of as a generalization of pictorial structures to allow for modeling richer geometric constraints than can be expressed with a tree-structured graphical model.

2.1.4 Patchwork of parts

Patchwork of parts (POP) [2] takes a similar approach to the spatial prior as [30], but takes a unique approach in modeling the image likelihood function. Most other approaches assume that part appearances are independent, thus producing a factorization of the image likelihood into a product over parts as in equation (1.3). This assumption implies that parts do not overlap; if part overlap occurs, the likelihood function effectively weights some pixels more than others because the pixels involved in the overlap contribute multiple terms to the image likelihood.

POP attempts to correct for part overlap by averaging the contributions of overlapping parts. There is no known efficient, exact inference algorithm for models with this overlap-aware likelihood function. To make inference tractable, the simpler factored likelihood is first used to generate object localization hypotheses and then those hypotheses are re-scored using the POP likelihood distribution. We show how this idea can be applied to our models in Section 4.1.3.

2.1.5 Summary of related work

To summarize, approaches to part-based object recognition differ in the conditional independence assumptions they make about the random variables in the statistical model. In fact we can imagine a spectrum of spatial priors, arranged according to the degree of independence assumptions that they make about the relative spatial locations of the parts. At one end of the spectrum, the bag-of-parts models assume that all parts are spatially independent, so that the location of a given part does not constrain the location of *any* other part. Inference in this case is efficient but the object model is weak. At the other end of the spectrum are models that make no spatial independence assumptions, by for example representing the prior as a full-rank multivariate joint Gaussian. This form of spatial prior can capture complex spatial relationships between part locations, but it is not known how to perform exact inference efficiently even for restricted cases. Tree-structured spatial priors fall in between the two extremes, with efficient inference but relatively weak models.

Given these different approaches, it is natural to ask what the best trade-off between computation cost and representation power of the model is. In other words, is it better to use exact inference with a weak model, or approximate inference with a rich model? We set out to answer these questions by proposing a family of spatial priors called k-fans that are explicitly parameterized according to where they fall along the spectrum. The k-fan models are described in the next section, and we present experimental results on object category detection using these models in Chapter 7.

2.2 k-fans

Now we consider a class of spatial priors that lie between the two extremes of the naïve Bayes model and the fully-connected spatial model discussed in the last section. The goal is to find models with recognition performance comparable to a fully-connected model but for which there exist fast procedures for exact inference.

We first consider star graph-structured spatial models. A star graph on n vertices has a central node that is connected to all of the other nodes, but none of the non-central nodes are connected to one another. In other words a star graph is a connected tree with depth one. Let G be a star graph with a set of edges E on a set of vertices V. Denote the central node as v_r. This star-structured graphical model has a simple statistical interpretation: the location of each of the non-central nodes is independent when conditioned on the location of v_r. In other words the prior distribution factors as a product over the non-central parts,

$$P_M(L) = P_M(l_r) \prod_{v_i \in \overline{R}} P_M(l_i|l_r),$$

where $\overline{R}$ is the set of non-central nodes, $\overline{R} = V - \{v_r\}$. We call v_r the *reference part* of this particular spatial prior.

The k-fans family generalizes the star-shaped graphical model to allow for a *set* of reference parts instead of a single part. A k-fan is a graph with k reference parts in which all of the reference parts are connected to one another, while each of the non-reference parts is connected only to the reference parts. More formally, let $R \subseteq V$ be a set of k reference parts and $\overline{R} = V - R$ be the remaining parts of the model. A k-fan is a simple graph $G = (V, E)$ such that the set of edges is,

$$E = R \times R \cup R \times \overline{R}.$$

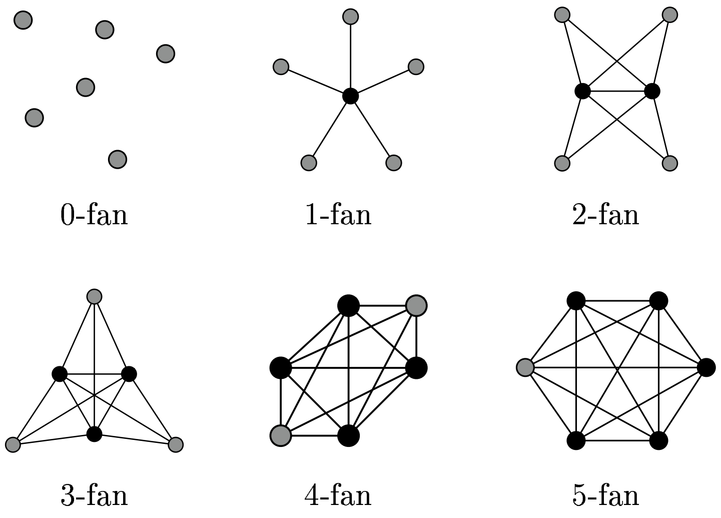


Figure 2.1: Some k-fans on six nodes. Reference nodes are shown in black.

A k-fan can also be seen as a collection of $n - k$ cliques of size $k + 1$ connected together along a common clique of size k. The k nodes in the common clique are the reference parts R. Some examples of k-fans on six nodes are shown in Figure 2.1.

The set of graphs produced as k grows from 0 to $n - 1$ intuitively interpolates between the empty graph and the complete graph on n nodes.[1] Thus we can view k-fans as a family of graphical models in which the degree of expressive power is explicitly parameterized by k.

Without loss of generality assume that the set of reference parts is $R = \{v_1, \ldots, v_k\}$. The prior distribution for a k-fan model is then,

$$P_M(L) = P_M(l_R) \prod_{v_i \in \overline{R}} P_M(l_i | l_R). \tag{2.1}$$

where l_R denotes a particular configuration of the reference parts, $l_R = (l_1, \ldots, l_k)$.

[1] Note that on n nodes, an $n - 1$ fan and an n-fan are equivalent.

From this equation it is clear that the set of k-fan models is exactly the models where the locations of the non-reference parts are conditionally independent given the locations of the reference parts. Alternatively we can write the prior in terms of marginal distributions,

$$P_M(L) = \frac{\prod_{v_i \in \overline{R}} P_M(l_i, l_R)}{P_M(l_R)^{n-(k+1)}}. \tag{2.2}$$

The numerator here is a product over the prior probabilities for each of the $n-k$ maximal cliques in the graph (each consisting of all of the reference parts and one non-reference part), and the denominator involves the prior distribution of just the reference parts. This is a special form of the factorization for a triangulated graph, which is the ratio of a product over maximal cliques and a product over separators [12].

2.2.1 Geometric Interpretation

There is a natural connection between k-fan models and geometric alignment [45]. In a k-fan model the locations of the reference parts can be used to compute a global transformation aligning a geometrical model and the image. This alignment defines an ideal location for each non-reference part, and deviations from these ideal locations can be measured by the conditional distributions $P_M(l_i|l_R)$ [14].

There is also a close connection between k-fan models and object recognition using geometric invariants. Each maximal clique in a k-fan consists of exactly $k+1$ parts, and the location of these parts can be used to define shape constraints that are invariant to certain geometric transformations (see [8]). The number of reference parts controls the type of geometric invariants that can be represented [14].

In a k-fan model the location of a non-reference part can be described in a reference frame defined by the locations of the k reference parts. For example, when $k = 1$ the location of a non-reference part can be described relative to the location of the single reference part. The values $l'_i = l_i - l_r$ are invariant under translations, so 1-fans can be used to define translation invariant models. For the case of $k = 2$ the two reference parts can be used to define models that are invariant to rigid motions and global scaling. When $k = 3$ we can use the three reference parts to define an affine basis in the image plane; if the location of every non-reference part is described in this basis we obtain affine invariant models. These models are important because they capture arbitrary views of planar objects under orthographic projection [14].

To enforce geometric invariants over $k + 1$ parts one could define $P_M(l_i|l_R)$ to be one if the $k + 1$ locations satisfy a geometric constraint and zero otherwise. In general our models capture soft geometric constraints, giving preference to configurations that satisfy relationships on $k + 1$ features as much as possible. The distribution over the reference part locations $P_M(l_R)$ could be uniform in the case where all geometric constrains are defined in terms of $k + 1$ parts. Non-uniform distributions can be used to represent interesting classes of non-rigid objects [14].

2.2.2 Gaussian k-fans

The k-fan models we have presented so far assume that the locations of some parts are conditionally independent from the location of other parts, but otherwise we have made no assumptions on the actual form of the prior distributions. In this section we describe an important specialization of k-fans in which the prior is assumed to be a Gaussian,

$$P_M(L) = \mathcal{N}(L|\mu, \Sigma).$$

This form of the spatial prior allows for more efficient inference, as we will show in Chapter 3. Gaussians have often been used to model the spatial constraints between parts in an object [2, 30, 32, 33].

Any conditional independence assumptions made by the k-fan appear as zero entries in both the covariance matrix Σ and inverse covariance matrix Σ^{-1} [51]. That is, if parts v_i and v_j are conditionally independent, then $\Sigma_{i,j}$, $\Sigma_{j,i}$, $\Sigma^{-1}_{i,j}$, and $\Sigma^{-1}_{j,i}$ are all zero. The marginal distribution of any subset of variables in a multivariate Gaussian is also Gaussian [51]; thus the marginal distribution of just the reference parts $P_M(L_R)$ is Gaussian. Let μ_R and Σ_R denote the mean and covariance of the locations of the reference parts. Now by the same property, the marginal distribution of any subset of parts involving the reference set and a single non-reference part v_i is Gaussian with parameters,

$$\mu_{i,R} = \begin{bmatrix} \mu_i \\ \mu_R \end{bmatrix}, \quad \Sigma_{i,R} = \begin{bmatrix} \Sigma_i & \Sigma_{iR} \\ \Sigma_{Ri} & \Sigma_R \end{bmatrix}. \tag{2.3}$$

These parameters can be used to define the spatial prior in terms of equation (2.2). We will use this for learning Gaussian k-fans, as we will see in Chapter 5.

Another property of multivariate Gaussians is that the conditional probability of any subset of the variables is also Gaussian. In particular, the conditional distribution on the location of a non-reference part given particular locations for the reference parts, $P_M(l_i|l_R)$, has mean and covariance [51],

$$\mu_{i|R}(l_R) = \mu_i + \Sigma_{iR}{\Sigma_R}^{-1}(l_R - \mu_R), \tag{2.4}$$

$$\Sigma_{i|R} = \Sigma_i - \Sigma_{iR}{\Sigma_R}^{-1}\Sigma_{Ri}, \tag{2.5}$$

Note that the covariance $\Sigma_{i|R}$ is independent of the location of the reference parts. This is a non-trivial property that enables the use of convolutions to obtain faster inference algorithms than is possible with non-Gaussian models, as we will show in the next chapter.

In addition to their computational benefits, Gaussian spatial priors have a particularly intuitive interpretation. A model prefers a certain configuration of the model (its mean μ) at which the prior probability is maximum. The probability drops off quickly as the configuration diverges from μ, where the rate of drop-off depends on the covariance Σ. To continue with this intuition it is useful to consider the negative logarithm of the prior,

$$-\log P_M(L) = -(L-\mu)^T \Sigma^{-1} (L-\mu) + Z,$$

where Z is a constant. We can think of this negative log-likelihood as the *cost* or *energy* of a given configuration, and the goal of inference is to minimize this cost. It is easy to see from the above equation that the cost of a given configuration is related to the square of its distance from the mean configuration. This is very similar to the ideal spring model in physics, where the potential energy of a mass connected to a spring is proportional to the square of its displacement from the equilibrium point [74]. Thus we can visualize a Gaussian spatial model as a set of springs connecting some parts of the object, with μ specifying the equilibrium positions of the springs and Σ controlling the spring constants.[2] This is exactly how Fischler and Elschlager imagined part-based deformable models in their 1973 paper, as reprinted in Figure 1.3 [36].

[2] The analogy is not perfect however: our Gaussians are multi-dimensional with arbitrary covariance matrices, which are not easily visualized using one-dimensional springs.

2.2.3 Other graphical models

Although we focus on the k-fan family in this thesis, other families of graphical models exist that may prove useful in object recognition. For example, we have explored using general k-trees [65]; k-fans are a special class of k-trees. To see this, notice that a k-fan can be constructed by starting with a k-clique corresponding to the reference nodes and sequentially adding new nodes by connecting each of them to the reference nodes and nothing else. In particular k-fans are decomposable (also known as triangulated or chordal) graphs. An important difference between k-fans and arbitrary k-trees, however, is that we do not know how to learn k-tree models efficiently. While k-fan models can be learned in time polynomial in n and exponential in k, as we will show in Chapter 5, learning a k-tree is not tractable because the minimum spanning k-tree problem is NP-hard even for small k (the running time is exponential in n) [11].

CHAPTER 3

EFFICIENT INFERENCE

The last chapter introduced a family of statistical models for part-based object recognition. The advantage of these k-fan models over others in the literature is that the degree of spatial structure in the model can be explicitly controlled by the parameter k. In this chapter we turn to the problem of performing efficient inference on these models. We consider in turn the two basic recognition tasks, classification and localization. In each case we show how to perform the task in time linear in the number of parts and exponential in k for a general k-fan model. We then show how the exponent of the computational complexity can be reduced if the spatial prior is a Gaussian distribution. For localization we also present an algorithm based on branch-and-bound search that significantly decreases computation times in practice. All of the inference algorithms presented here are *exact* in the sense that they provably compute the actual maximum or sum of the posterior.

3.1 Classification

In classification the goal is to decide whether or not an image contains an instance of a particular object category. As we described in Section 1.1.3, it is natural when performing classification to compare the ratio of the likelihood that an object appears in the image, $P_M(I|w_1)$, to the likelihood that it does not, $P_M(I|w_0)$. To compute $P_M(I|w_1)$ we must sum over all possible configurations of the parts,

$$P_M(I|w_1) = \sum_L P_M(L) P_M(I|L).$$

Using the likelihood function of equation (1.3) we see that,

$$P_M(I|w_1) = Z(I) \sum_L P_M(L) \prod_{v_i \in V} P_M(I|l_i).$$

where once again $Z(I)$ is a term that does not depend on the part configuration. How $Z(I)$ and $P_M(I|w_0)$ are computed depends on the type of appearance model that is being used. We will describe several appearance models in Chapter 4; for now we simply assume that these distributions exist and can be computed in linear time.

$P_M(I|w_1)$ can be computed by brute force in time $O(h^n)$, where h is the number of possible locations for each part in the image; however we can do much better using the conditional independence assumptions of a k-fan model. For a k-fan model the sum over all configurations L can be factored using the conditional form of the spatial prior in (2.1). As before let V denote the set of parts, R be the set of reference parts, and $\overline{R} = V - R$ be the set of non-reference parts. Now $P_M(I|w_1)$ can be written as,

$$P_M(I|w_1) = Z(I) \sum_{l_R} P_M(l_R) \prod_{v_i \in R} P_M(I|l_i) \prod_{v_i \in \overline{R}} \alpha_i(l_R), \tag{3.1}$$

where $\alpha_i(l_R)$ is defined for each non-reference part v_i,

$$\alpha_i(l_R) = \sum_{l_i} P_M(l_i|l_R) P_M(I|l_i). \tag{3.2}$$

This factorization gives an $O(nh^{k+1})$ algorithm for computing $P_M(I|w_1)$. For any given reference set configuration L_R, $\alpha_i(L_R)$ can be computed in $O(h)$ time; thus each $\alpha_i(\cdot)$ can be computed for all possible configurations in $O(h^{k+1})$ time. The summation in equation (3.1) can be computed using the precomputed values of α in $O(nh^k)$ time.

3.1.1 Efficient classification in Gaussian k-fans

For the case of a Gaussian k-fan we can compute the likelihood ratio even faster using convolutions. For each non-reference part v_i we have,

$$P_M(l_i|l_R) = \mathcal{N}(l_i|\mu_{i|R}(l_R), \Sigma_{i|R}),$$

a Gaussian distribution with mean and covariance given by equations (2.4) and (2.5). Let $\alpha'_i(l_i)$ be the convolution of $P(I|l_i)$ with a Gaussian kernel of covariance $\Sigma_{i|R}$. It is not hard to see that,

$$\alpha_i(l_R) = \alpha'_i(\mu_{i|R}(l_R)).$$

So each α_i can be computed by a convolution in the space of possible part locations. This can be done in $O(h^k + h \log h)$ time (compared to $O(h^{k+1})$ time using brute force) using the Fast Fourier Transform to perform the convolution, as discussed in Appendix A. The overall running time of the classification computation for the case of a Gaussian k-fan model is thus $O(nh^k + nh \log h)$. The $\log h$ dependency can be removed by using linear time methods that approximate Gaussian convolutions, as we discuss in Appendix A. Using that method a good approximation of $P_M(I|w_1)$ can be computed in time $O(nh^k)$. Note that for a 1-fan model this is the same asymptotic cost required if the locations of the parts were completely independent, as in a 0-fan. Thus by using convolutions and dynamic programming, we are able to add spatial constraints to a bag-of-parts model at *no additional asymptotic computational cost.*

3.2 Localization

In localization the goal is to find a configuration of the model having maximum posterior likelihood in a particular image. The posterior distribution for a k-fan model can be written in terms of the likelihood function (1.3) and the spatial prior using the factorization from Baye's law in equation (1.1),

$$P_M(L|I) = Z(I) P_M(L) \prod_{v_i \in V} P_M(I|l_i).$$

A configuration L^* that maximizes this posterior can be found in $O(h^n)$ time by brute force, but we can do better by using the conditional independence assumptions of the k-fan. By manipulating terms and using the conditional form of the prior in equation (2.1) we have,

$$P_M(L|I) = Z(I) P_M(l_R) \prod_{v_i \in R} P_M(I|l_i) \prod_{v_i \in \overline{R}} P_M(l_i|l_R) P_M(I|l_i). \tag{3.3}$$

Notice in equation (3.3) that the ideal location of any non-reference part depends only on the configuration of the reference parts. In other words, if we knew the optimal reference part configuration, then we could easily find the optimal location l_i^* of any non-reference part $v_i \in \overline{R}$ in $O(h)$ time,

$$l_i^* = \arg\max_{l_i} P_M(l_i|l_R) P_M(I|l_i). \tag{3.4}$$

We define $\alpha_i^*(l_R)$ to be the likelihood of non-reference part v_i in its optimal configuration given a particular reference part configuration,

$$\alpha_i^*(l_R) = \max_{l_i} P_M(l_i|l_R) P_M(I|l_i).$$

Using these α_i^* functions and equation (3.3) we can express the probability of an optimal configuration of all of the parts given a particular reference part configuration l_R as,

$$\beta^*(l_R) = Z(I) P_M(l_R) \prod_{v_i \in R} P_M(I|l_i) \prod_{v_i \in \overline{R}} \alpha_i^*(l_R). \tag{3.5}$$

These functions give an algorithm that computes an optimal configuration in time polynomial in the number of parts n and the number of locations h for each part but exponential in k. For each part, $\alpha_i^*(\cdot)$ can be computed by brute force in $O(h^{k+1})$ time, while β^* can be computed in $O(nh^k)$ time. An optimal configuration for the reference parts l_R^* is one maximizing β^*. Finally, we can use equation (3.4) to find the best position for each non-reference part. This takes $O(h)$ time for each non-reference part. The overall running time of this procedure is $O(nh^{k+1})$.

An inference algorithm with a running time of $O(nh^{k+1})$ is computationally tractable only for very small k. Fortunately the running time can be reduced to $O(nh^k)$ when the spatial prior is Gaussian, as we describe in the next section. In Section 3.2.3 we show how the running time can be further reduced in practice using branch-and-bound search.

3.2.1 Efficient localization in Gaussian k-fans

We can speed up localization if the spatial prior is a Gaussian distribution. The technique for doing this is similar to that used for the classification problem in Section 3.1, but it uses *minimum convolutions* instead of regular convolutions. The min-convolution of two sampled functions f and g is defined as,

$$(f \otimes g)(p) = \min_q f(q) + g(p - q).$$

Efficient algorithms exist for computing the min-convolution, as we will describe in Appendix A.

To apply the min-convolution to this problem it is necessary to perform the computations on the logarithms of probabilities instead of on the probabilities themselves. In practice it is also advantageous to use logarithms because they avoid the numerical problems that result from trying to store very small probabilities in a digital computer.[1] In particular we work with the negative logarithms of probabilities because they have a natural intuitive interpretation as energies or *costs.* Thus when performing localization with negative log-likelihoods the goal is to find a configuration that minimizes the cost of placing the model instead of maximizing the probability.

We can think of a "cost distribution" $C(\cdot)$ associated with any probability distribution $P(\cdot)$. The cost distribution corresponding to probability distribution $P(x)$ is,

$$C(x) = -\log P(x),$$

and likewise we denote the cost distribution corresponding to a conditional probability distribution $P(x|y)$ as,

$$C(x|y) = -\log P(x|y).$$

Rewriting equation (3.3) in terms of costs, we have

$$C_M(L|I) = C_M(l_R) - \log Z(I) + \sum_{v_i \in R} C_M(I|l_i) + \sum_{v_i \in \overline{R}} -\log \alpha_i^*(l_R),$$

where

$$-\log \alpha_i^*(l_R) = \min_{l_i} C_M(I|l_i) + C_M(l_i|l_R). \tag{3.6}$$

In the case of a Gaussian k-fan, the distribution of non-reference part v_i conditioned on the reference configuration, $P_M(l_i|l_R)$, is Gaussian and hence the cost function

[1] A back-of-the-envelope calculation is useful to illustrate how small the probabilities in our statistical framework can become. Consider computing the prior probability of a one-megapixel image, assuming a uniform prior. In a typical digital image each pixel has a grayscale intensity value in the range $[0, 255]$. Thus there are $256^{1000000} \approx 10^{2400000}$ possible images, and the prior probability of any given image is about $10^{-2400000}$. This is a fantastically small number; for comparison, consider an experiment in which we draw atoms at random with replacement from the known universe. $10^{-2400000}$ is roughly the probability of drawing the same atom 30,000 times in a row! [40]

$C_M(l_i|l_R)$ is a Mahalanobis distance,

$$C_M(l_i|l_R) = \frac{1}{2}(l_i - \mu_{i|R})^T \Sigma_{i|R}^{-1}(l_i - \mu_{i|R}) + Z_1,$$

where Z_1 is a constant. When written in this form it is clear that equation (3.6) can be computed using a min-convolution. In particular, let $\mathcal{D}_i(l_i)$ be the min-convolution of $C_M(I|l_i)$ and $g(l_i)$, where

$$g(l_i) = \frac{1}{2}(l_i)^T \Sigma_{i|R}^{-1}(l_i) + Z_1.$$

Then the logarithm of α_i^* can be easily computed as,

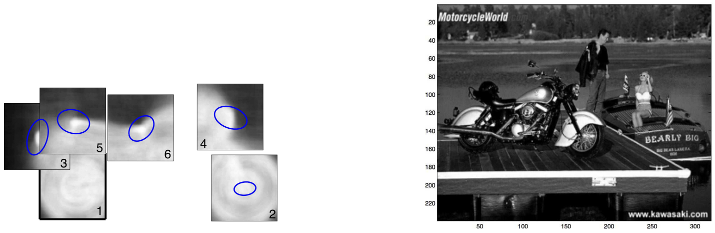
$$-\log \alpha_i^*(l_R) = \mathcal{D}_i(\mu_{i|R}(l_R)).$$

Note that using min-convolution to perform this computation is possible because $\Sigma_{i|R}$ can be written as a function of only the marginal covariances, as in equation (2.5). The computation would not be possible if $\Sigma_{i|R}$ were a function of l_R, for example.

Localization is then completed by computing the negative logarithm of equation (3.5), which is the final cost for each configuration of the reference parts,

$$\begin{aligned} F(l_R) &= -\log \beta^*(l_R) \\ &= C_M(l_R) + \sum_{v_i \in R} C_M(I|l_i) + \sum_{v_i \in \overline{R}} -\log \alpha_i^*(l_R) - \log Z(I), \quad (3.7) \\ & \quad (3.8) \end{aligned}$$

and then finding the minimizing position of each of the non-reference parts using equation (3.4). Performing the min-convolutions takes time $O(h)$ using the linear time algorithm described in Appendix A, so computing $F(l_R)$ for any given value of l_R takes time $O(h)$. Since $F(l_R)$ must be computed for all of the possible h^k configurations of the reference parts, the total running time for localization with a Gaussian k-fan is $O(nh^k)$.



(a) (b)

Figure 3.1: A k-fan model for motorbikes, and a sample input image. The model is a six-part 1-fan with the back wheel as the reference part.

3.2.2 An illustrated example

We have shown mathematically how to perform exact inference with Gaussian k-fan models efficiently without relying on feature detection. It turns out that the inference algorithms are also intuitive and straightforward to implement. In this section we illustrate how the efficient localization algorithm works with a running example showing each step of the process.

Figure 3.1(a) shows a diagram of a six-part 1-fan model for the "motorbike" object class. For visualization purposes the models are drawn as follows: the parts are positioned in their mean locations $\mu_{i|R}$ with respect to the reference part, which is the back wheel, and the conditional covariance $\Sigma_{i|R}$ of each non-reference part location is represented as an ellipse plotted at two standard deviations from the mean. The part appearance models are simple edge-based templates, which will be described in more detail in Chapter 4. We will describe how the localization procedure works using this motorbike model on the sample input image shown in Figure 3.1(b). There are three steps to the procedure, as we now describe.

Step 1: Apply part appearance operators

The first step in performing localization is to compute $C(I|l_i)$ for each part at each possible pixel location. This produces a cost map for each part, indicating how well the part appearance model matches the local image information at each image location. One can visualize this step as template correlation with a template for each part (although the actual operation depends on the specific appearance model used, as we describe in Chapter 4).

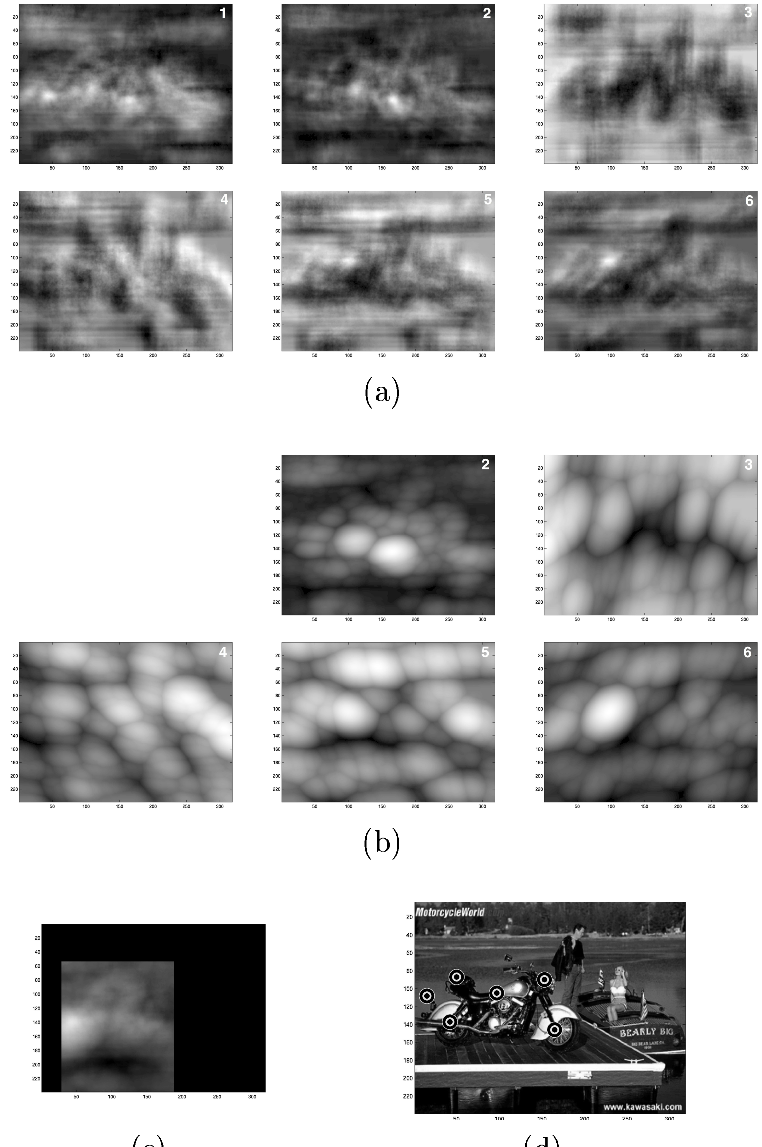
Figure 3.2(a) shows the cost maps that were generated by the motorbike model on the sample input image, with lower-cost locations represented by brighter intensities. To perform 0-fan inference we would simply choose the highest-likelihood location for each part based on these maps alone. The individual quality maps are quite noisy; for example, the front and back wheel are very similar in appearance so there are peaks at the location of the front wheel in the back wheel cost map and vice-versa. This illustrates an advantage of using a spatial model and our unified inference framework: weak evidence from each part is combined together in order to produce better overall localization decisions.

Step 2: Perform min-convolutions

The next step takes into account the spatial dependencies in the model as encoded by the Gaussian prior on each non-reference part with respect to the reference parts. This is done by computing the min-convolution of the quality map for each non-reference part, producing a new quality map $\mathcal{D}_i(l_i)$,

$$\mathcal{D}_i(l_i) = \min_y \; C_M(I|l_i) + \frac{(l_i - y)^T \Sigma_{i|R}^{-1} (l_i - y)}{2}.$$

We show how to compute this min-convolution efficiently in Appendix A.



(a)

(b)

(c)

(d)

Figure 3.2: Illustration of the efficient localization procedure for k-fan models: (a) quality maps indicating the cost of placing each part at each location, with brighter intensity indicating better locations, (b) result of min-convolution applied to the quality maps of the non-reference parts, (c) final quality map showing the cost of placing the reference part at each location, and (d) final result, showing the localized part locations.

The result of this step in the running example is shown in Figure 3.2(b). The transformation effectively "spreads" the cost maps produced by the appearance models, allowing for flexibility in the relative part locations. Intuitively the resulting cost $\mathcal{D}_i(l_i)$ is low near locations where the original part cost was low, with the size and shape of the spreading operation controlled by the conditional covariance parameter $\Sigma_{i|R}$.

Step 3: Combine evidence

The last step in the localization procedure combines the min-convolved cost maps of all of the non-reference parts with the cost maps of the reference parts. The result is a cost for every configuration of the reference parts that takes into account the placement of the whole model. This cost is exactly $F(l_R)$ in equation (3.7), the negative logarithm of $\beta^*(l_R)$ from equation (3.5).

The combining evidence step is particularly simple for the case of a translation-invariant 1-fan model. In this case the reference set is a single part, and the prior on reference set configuration is uniform (since translation invariance implies that all object positions are equally likely). Thus the cost distribution $C_M(l_R)$ is constant and can be ignored. To compute $F(l_R)$, we shift the quality maps $\mathcal{D}_i(l_i)$ by the mean position $\mu_{i|R}$ of part i relative to the reference part and sum all of these shifted quality maps together with the quality map for the reference part $C_M(I|l_R)$. The resulting cost map for the sample image is shown in Figure 3.2(c). An optimal location l_R* for the reference part (the back wheel) is determined by picking a minimum-cost location in this map. Then, the locations of the other parts can be found by choosing l_i^* for each individual non-reference part so as to minimize $C_M(l_i|l_R^*) + C_M(I|l_i)$. The final localization results for the sample image are shown in Figure 3.2(d).

For a 1-fan $F(l_R)$ is a two-dimensional cost map but for general k it is a $2k$ dimensional map. To compute $F(l_R)$ we iterate over all possible reference locations and evaluate the sum above. In the next section, we introduce a branch and bound technique that significantly reduces computation cost by eliminating the need to explicitly consider all possible reference set configurations.

3.2.3 Faster inference using branch-and-bound search

The localization procedure described in Section 3.2.2 can be thought of as a search over all possible configurations of the reference parts in order to maximize the likelihood in equation (3.7). Algorithmic techniques like min-convolution and dynamic programming help us to compute the marginal probability of any given reference part configuration efficiently, but performing localization still requires a search over a space of configurations that grows exponentially with k. Thus localization using the procedure from the last section is tractable only when k is very small (typically less than or equal to 2).

For most models and images, however, the majority of the reference part configuration space has very low probability. Branch-and-bound search [68] is a well-known general optimization technique that works well in such situations where the optimal solution is much better than the majority of the solution space. The idea is to discard large portions of the solution space *en masse* that are provably not optimal, assuming that bounds on regions of the configuration space can be computed efficiently.

In this section we present a localization technique based on branch-and-bound search. Although the worst-case running time is the same as that of the algorithm presented in the last section, in practice the technique is several orders of magnitude faster for most models and images. We will first give an overview of the branch-and-bound-based algorithm in the next section. We then prove that the algorithm always converges to the optimal configuration of any model in any image. We discuss efficient bounding functions in Section 3.2.3 and turn to implementation issues in Section 3.2.3.

A branch-and-bound algorithm

Consider the localization problem on a $p \times q$ pixel image with a k-fan model. The space of possible reference part configurations is $\mathcal{I} = ([1,p] \times [1,q])^k$, where a given configuration l_R of the reference parts is represented by a single point in $\mathcal{I}$. The goal of the search is to find l_R^*, a configuration that maximizes $F(l_R)$ in equation (3.7). Once l_R^* is found, the optimal configuration of the other parts can be found in $O(nh)$ time using equation (3.4).

We define a *cell* to be a subset of the search space $\mathcal{I}$. Suppose that we have a bounding function $B(E)$ that for any cell $E \subseteq I$ satisfies two conditions,

$$\textit{condition (i):} \qquad B(E) \leq F(l_R) \text{ for all } l_R \in E,$$

$$\textit{condition (ii):} \qquad B(E) = F(l_R) \text{ if } E = l_R.$$

That is, $B(E)$ is a lower bound on the cost of any part configuration within the cell E, with the additional property that if E is a single point, $B(E)$ is exactly the cost of that reference configuration. A computationally-efficient bounding function that satisfies these properties is presented in the next section.

We now describe our branch and bound procedure for efficient k-fan localization. The procedure uses a priority queue that stores cells. We begin by adding $\mathcal{I}$ with priority $B(\mathcal{I})$ to the empty queue. Then, the lowest-cost cell E is removed from the queue and partitioned into several strictly smaller cells $E_1 ... E_q$. Each of these cells E_i is added to the queue, along with the corresponding priority $B(E_i)$. This process continues until the cell removed from the queue is a single point. We now prove that the algorithm always finds such a point, and that the point corresponds to an optimal reference part configuration.

Theorem 3.2.1 *For any image I and model M, the above algorithm always converges to an optimal reference part configuration.*

Proof We first show that the procedure always terminates. In each iteration, either a cell corresponding to a single point is removed from the queue, in which case the procedure completes, or a cell is replaced with a finite number of strictly smaller cells. The image I is finite, so eventually a single point must be the highest-priority item in the queue.

We now show that the the reference part configuration removed from the queue before termination is optimal. Denote this last point removed from the queue as l_R. The priority queue guarantees that for any cell E' still in the queue, $B(l_R) \leq B(E')$. Further, our bounding function guarantees that for any point $l'_R \in E'$, $B(E') \leq F(l'_R)$, so we have that $F(l_R) \leq F(l'_R)$. Now since any cell removed from the queue by the algorithm is always replaced by a set of cells whose union is the original cell, the union of all cells still in the priority queue at the end of the algorithm is $\mathcal{I} - \{l_R\}$. Thus there is no lower-cost configuration in $\mathcal{I}$ than l_R, and l_R is optimal. ∎

The algorithm just described finds a single maximum-likelihood configuration of the reference parts. For some applications it is useful to find multiple high-likelihood configurations. This can be accomplished using the above procedure by continuing the branch-and-bound algorithm after l_R^* has been found. Whenever the cell removed from the priority queue is a single point, the algorithm records it as the next most probable reference part configuration. It removes the next entry from the priority queue and continues the recursive descent. This process can be repeated until an arbitrary number of configurations has been produced.

A bounding function

Having described the general algorithm we now turn to producing a bounding function that satisfies conditions (i) and (ii) from the last section. The efficiency of the branch-and-bound optimization algorithm depends critically on the choice of bounding function. If the bound is too loose, the algorithm must consider many sub-optimal cells. On the other hand if the bounding function is slow to compute, the algorithm might be no faster than a brute-force search. In this section we describe a bounding function that we have found is tight enough to work well in practice.

Suppose that we had a bounding function B_i such that for any cell E,

$$\begin{aligned} &\textit{condition (iii):} \qquad B_i(E) \leq \mathcal{D}_i(\mu_{i|R}(l_R)) \text{ for all } l_R \in E, \text{ and} \\ &\textit{condition (iv):} \qquad B_i(E) = \mathcal{D}_i(\mu_{i|R}(l_R)) \text{ if } E = l_R. \end{aligned}$$

That is, $B_i(E)$ bounds the min-convolved likelihood map of part i over its possible mean locations corresponding to the possible reference configurations in E. Then using the fact that a lower bound on a summation is the sum of lower bounds on the individual terms, a bound on the cost $F(l_R)$ in equation (3.7) is,

$$B(E) = \min_{l_R \in E} C_M(l_R) + \sum_{v_i \in R} \min_{l_R \in E} C_M(I|l_i) + \sum_{v_i \in \overline{R}} B_i(E) - \log Z(I). \tag{3.9}$$

It is easy to verify that both of conditions (i) and (ii) from the last section are met by this function.

All that remains is a bound B_i on the min-convolved likelihood maps that satisfies conditions (iii) and (iv) above. A simple bound satisfying these conditions is,

$$B_i(E) = \min_{p \in t_i(E)} \mathcal{D}_i(p), \tag{3.10}$$

where $t_i(E)$ computes the set of possible mean locations for part i given $l_R \in E$,

$$t_i(E) = \{\mu_{i|R}(l_R) \,|\, l_R \in E\}. \tag{3.11}$$

It is useful to visualize $t_i(E)$ geometrically. From the form of $\mu_{i|R}(l_R)$ given in equation (2.4), one observes that $t_i(E)$ is an affine transformation of the set E followed by a projection onto the two-dimensional image coordinate space.

Implementation issues

In order for the branch and bound procedure to be computationally efficient, the bound function B must be easy to compute. Fortunately, the minimizations in equations (3.9) and (3.10) can be computed ahead of time, so that calculating the bound on a given cell involves only adding together a few pre-computed values. The key is to perform the cell partitioning step in such a way that only a few sizes of cells are used. The minimizations can then be pre-computed for all of the possible cell sizes using a *min filter* in time linear in the dimensions of the image [38].

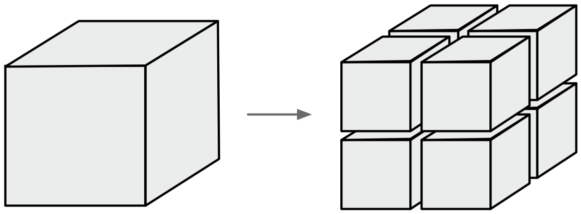


Figure 3.3: Illustration of cell partitioning. The cells are hypercubes of arbitrary dimensionality, but for visualization purposes three-dimensional cubes are shown.

For convenience we begin by padding the input image out to an $m \times m$ pixel square where m is a power of two. This allows us to restrict the cells to be hypercubes with side lengths that are also powers of two. During the branch and bound procedure, each cell is partitioned into $(2k)^2$ hypercubes, each having one-half the edge length of the original, as illustrated in Figure 3.3. The total number of different cell sizes used by this approach is only $O(\log m)$. The minimizations in equation (3.9) are then only over a small set of squares with dimensions that are powers of two.

When E is a hypercube, the affine projection function $t_i(E)$ in equation (3.11) produces a two-dimensional polytope (because as noted above, t_i is an affine transform followed by a projection into two-dimensional space). Thus each minimization in equation (3.10) is over one of a set of $O(\log m)$ polytopes, since the size and shape of the projected polytope depends only on the size of the hypercube E. To use a min filter to carry out these maximizations, it is convenient if $t_i(E)$ produces axis-oriented rectangles. We accomplish this by having $t_i(E)$ compute a rectangular superset of the true set of possible mean locations, at the expense of weakening the bound $B_i(E)$. To generate such a rectangle we compute the ideal location $\mu_{i|R}(l_R)$

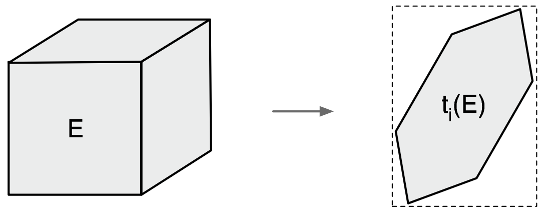


Figure 3.4: Geometric interpretation of the function t_i. On a cell E, $t_i(E)$ performs an affine transformation, a projection onto 2-D image space, and then computes a smallest enclosing rectangle.

for each vertex of E and then find the smallest axis-oriented rectangle enclosing these points. Any point in $t_i(E)$ is guaranteed to also be in this rectangle, because the affine projection of a polytope is the convex hull of the projection of its vertices, and an enclosing rectangle is a superset of this convex hull [90]. The process of computing this enclosing rectangle is illustrated geometrically in Figure 3.4. Maximizing or minimizing over rectangular subsets of distance-transformed likelihood maps has been called the *box distance transform* [66].

3.3 Sampling from the posterior

In Section 3.2 we showed how to find an exact maximum *a posteriori* configuration for a k-fan model in a given image. However many applications are interested in having multiple hypotheses for the location of the object — when multiple instances of the same object class appear in an image, for example. Even in images where there is only one object, the highest-likelihood model configuration may not correspond to the actual object because of modeling error. In these cases it is useful to retrieve multiple high-likelihood configurations of the model instead of only the maximum-likelihood configuration. The application can then use a separate criterion independent from the statistical model to choose among the multiple hypotheses. In our statistical framework, a natural way of generating these high-likelihood configurations is to sample from the posterior, $P_M(L|I)$.

With k-fan models it is possible to sample from the posterior efficiently. To do this, we first compute the marginal distribution over the reference set configuration,

$$P_M(l_R|I) = Z(I) P_M(l_R) \prod_{v_i \in R} P(I|l_i) \prod_{v_i \in \overline{R}} \alpha_i(l_R),$$

where $\alpha_i(l_R)$ is defined in equation (3.1). This takes time $O(nh^k)$ for Gaussian k-fans using convolutions to compute $\alpha_i(l_R)$, as discussed in Section 3.1. To sample from the posterior, we first draw a sample l'_R from $P_M(l_R|I)$ and then draw samples from $P_M(l_i|I, l'_R)$ for each non-reference part i. Note that the marginal need not be recomputed for each sample, so that drawing s samples from the posterior for a given model and image takes time $O(nh^k + sn)$.

CHAPTER 4

FEATURE OPERATORS

The last two chapters introduced a family of spatial models for object class recognition and presented efficient classification and localization algorithms for those models. In order to keep those chapters as general as possible, we did not assume the use of any particular part appearance model: we simply assumed that some mechanism existed to compute $P_M(I|l_i)$ for part v_i at any location l_i in the image. Thus the appearance model can be chosen independently of the spatial model and inference algorithm according to the requirements of the application. In fact, it is possible to use multiple types of appearance models in the same object model, as we do for scene modeling in Chapter 6.

A key difference between our approach and most others in the literature is that we do not use feature detection, as we discussed in Section 1.1.5. Instead of using a feature detector to produce a small set of possible locations for each part, we instead compute the likelihood for every part at *every possible location* in the image. In other words we can think of our appearance models as *feature operators* that when applied to an image produces a likelihood map over the entire configuration space. Our use of feature operators thus avoids the intermediate hard classification decisions that are made by feature detectors, by computing the entire factored posterior distribution in (1.1) using the full likelihood functions for all the parts. In Chapter 7 we demonstrate experimentally that this unified inference approach performs better than approaches that use feature detection and bottom-up inference.

We have experimented with a number of different part appearance models. We present here the details of two specific models that we have found to work well across a variety of object classes. The first appearance model uses small probabilistic rigid templates and can be applied to a range of image features including edges, texture, and color. The second approach models part appearance using histograms of image gradients, which have recently become popular for category recognition [9, 19, 31].

4.1 Template-based part models

We first introduce the simple template-based feature operators. A preprocessor is first run on the image to assign one of a small set of labels to each pixel. The preprocessor is chosen according to the type of image features of interest; this makes the approach general enough to handle a broad range of different image features. In our work we have used preprocessors tuned for color, edge, and surface orientation features. The parts are modeled as small rigid templates, with each position in the template having a probability distribution over the set of possible pixel labels.

More formally let I be the result of running the preprocessor on a given image. We assume that I has one of a small set of labels at each pixel, such that $I(p) \in \mathcal{E} = \{1, 2, ..., r\}$ for each pixel location $p \in I$. The foreground appearance model for a part i consists of a small template T_i that gives the probability of observing each possible label at each location within the template. That is, for part i the foreground appearance model is a function $f_i(p)[u]$ for all $p \in T_i$ and $u \in \mathcal{E}$ specifying the probability of observing label u at location p within the template when the patch is centered over a true part location. Another function $b[u]$ gives the probability of observing label u at a background pixel not corresponding to an object part. Assuming that background pixels are independent, the probability that an image is composed entirely of background pixels (which we call hypothesis

w_0) is,

$$P_M(I|w_0) = \prod_{p \in I} b[I(p)].$$

Given the location l_i of single part v_i, we can write a likelihood function,

$$P_M(I|l_i) = P_M(I|w_0) \prod_{p \in T_i} h_i(p + l_i, l_i), \tag{4.1}$$

where

$$h_i(p, l_i) = \frac{f_i(p - l_i)[I(p)]}{b[I(p)]}.$$

Then the likelihood for given a configuration L of several parts is,

$$\begin{aligned} P_M(I|L) &= P_M(I|w_0) \prod_{v_i \in V} \prod_{p \in T_i} h_i(p + l_i, l_i) & (4.2) \\ &= Z(I) \prod_{v_i \in V} P_M(I|l_i), & (4.3) \end{aligned}$$

where

$$Z(I) = P_M(I|w_0)^{-n+1}.$$

Note that this likelihood function is a true probability distribution (it sums to one over all possible images) as long as no patches overlap in the configuration L. For part configurations with overlap, equation (4.3) overcounts some pixels and thus gives only an approximation. In Section 4.1.3 we describe how this part overlap problem can be addressed.

In principle this template-based part model framework can be applied to any type of feature in which each pixel is assigned a discrete label. In practice it is also important that the set of possible pixel labels has small cardinality (e.g. less than a few dozen). Otherwise very large numbers of training exemplars are needed to estimate the non-parametric probability distributions accurately.

In our work we apply this framework to three different types of features: edges, color and surface orientation. We describe the edge features here; the color and surface orientation features are used for modeling scene context and are discussed in Chapter 6. Edges tend to capture useful local image information such as corners and boundaries of object regions. Instead of the simple presence or absence of an edge, we use coarsely-quantized edge orientation, dilated in both the spatial and orientation dimensions. Augmenting edge detection with dilation and edge orientation has been found to significantly improve results in other object recognition work [44]. We use the Canny algorithm [7] to detect edges in grayscale images. The gradient orientation at each edge pixel is also computed and quantized into one of four bins: north-south, east-west, northeast-southwest, and northwest-southeast. [1] We apply morphological dilation with a radius of 2.5 pixels to the binary edge map. Then we assign each nonzero pixel in the dilated map the orientation of the closest edge pixel. Note that this can be computed efficiently using a standard binary distance transform [39]. Thus for the edge features, the set $\mathcal{E}$ of possible edge labels consists of five elements: the four edge orientations and the non-edge label.

To implement this feature operator it is necessary to evaluate $P_M(I|l_i)$ at all pixel locations in image I. In practice the negative logarithms are computed instead, as we discussed in Section 3.2, to produce a cost map $C_M(I|l_i)$. The obvious way to compute this cost map is to slide the part template over the image, explicitly computing the likelihood at each position according to equation (4.1),

$$C_M(I|l_i) = C_M(I|w_0) - \sum_{p \in T_i} \log h_i(p + l_i, l_i). \tag{4.4}$$

This implementation takes time $O(|T_i|h)$ for each part v_i, where $|T_i|$ is the number of pixels in the part template and h is the number of pixels in the image. This

[1] The advantage to putting opposing directions into the same bin is for greater invariance to background color. For example, if opposing directions were placed in separate bins (e.g. by breaking up north-south into a north bin and a south bin), then the edge orientation labels along the boundary of a gray object would be different depending on whether the background is black or white.

straightforward algorithm is prohibitively slow for large templates.

4.1.1 Efficient implementation for sparse label maps

A faster implementation is possible when the labeled image maps are sparse — that is, when most pixels have the same label. This is the case for edge maps, where typically less than 5% of the pixels are labeled as edges. Let $z \in \mathcal{E}$ be the most common label (e.g. the non-edge label). Then equation (4.4) can be rewritten as,

$$C_M(I|l_i) = C_M(I|w_0) - \zeta - \sum_{p \in T_i} \log \frac{f_i(p)[I(l+p)]}{b[I(l+p)]} \frac{b[z]}{f_i(p)[z]}, \tag{4.5}$$

where ζ is a sum of the likelihood ratios in the part template having edge label z,

$$\zeta = \sum_{p \in T_i} \log \frac{f_i(p)[z]}{b(z)}.$$

The important property of equation (4.5) is that whenever the value of p is such that $I(l+p) = z$, the corresponding term in the summation is zero. Thus when computing the cost map for a sparse label map, most of the pixels in the image can be ignored because they have no effect on the sum.

An implementation of this approach proceeds as follows. Let B_i be the array that is to hold $C_M(I|l_i)$ computed at every location l_i. First B_i is initialized to be $C_M(I|w_0) - \zeta$ at every location. Then each pixel location l in the image is considered. If $I(l) = z$, then no action is taken and the pixel is skipped. Otherwise, for each location $p \in T_i$ in the part template, the cost map is updated,

$$B_i^{+1}(l-p) = B_i(l-p) - \log \frac{f_i(p)[I(l+p)]}{b[I(l+p)]} \frac{b[z]}{f_i(p)[z]}.$$

The worst-case running time of this implementation is still $O(|T_i|h)$ for each part, but in practice this method is typically 10-20 times faster than the naïve implementation described above for sparse label maps.

4.1.2 Efficient Fourier transform-based method

The optimization presented in the last section is fast for sparse label maps such as edge maps, but it provides little benefit when the frequency of the labels is roughly uniform, as with the color or surface orientation features discussed in Section 4.1. For these features it is faster to use the Fourier transform-based implementation presented in this section.

We can rewrite the likelihood function in terms of convolutions. To see this, note that equation (4.4) can be rewritten as a product over pixel labels,

$$C_M(I|l_i) = C_M(I|w_0) - \sum_{p \in T_i} \sum_{e \in \mathcal{E}} 1_e(I, l+p) \log \left(\frac{f_i(p)[e]}{b[e]} \right),$$

where $1_e(I, l)$ is an indicator function that is one if $I(l) = e$ and is zero otherwise. This is just a sum of cross correlations,

$$C_M(I|l_i) = C_M(I|w_0) - \sum_{e \in \mathcal{E}} \left(\log \frac{f_i[e]}{b[e]} \star 1_e(I) \right) (l_i),$$

and can be re-written in terms of convolutions,

$$C_M(I|l_i) = C_M(I|w_0) - \left(\sum_{e \in \mathcal{E}} \hat{f}_i[e] * 1_e(I) \right) (l_i), \tag{4.6}$$

where

$$\hat{f}_i(p)[e] = \log \frac{f_i(-p)[e]}{b[e]}.$$

Using the convolution theorem [39], equation (4.6) can be rewritten in terms of the Fourier transform,

$$C_M(I|l_i) = C_M(I|w_0) - \sum_{e \in \mathcal{E}} \mathcal{F}^{-1} \left\{ \mathcal{F}\{\hat{f}_i[e]\} \cdot \mathcal{F}\{1_e(I)\} \right\}, \tag{4.7}$$

where $\mathcal{F}$ denotes the forward Fourier transform, $\mathcal{F}^{-1}$ denotes the inverse transform, and $\cdot$ is the elementwise complex multiplication operator.

Thus convolution gives a faster method for computing the part likelihood maps. The Fourier transforms of the part template functions $\hat{f}_i$ can be pre-computed because they do not depend on I. Then during recognition, the Fourier transform of each of the binary indicator maps $1_e(I)$ is computed. Using the Fast Fourier Transform (FFT) algorithm [11] this takes time $O(|\mathcal{E}|h \log h)$ where h is the number of pixels in the image. Then for each part $v_i \in V$ and each edge label $e \in \mathcal{E}$, we compute the complex multiplication of the Fourier transform of the binary indicator function $1_e(I)$ and that of the part template $\hat{f}_i[e]$, and then take the inverse transform of each result. Finally an addition over the edge labels is performed for each part. Thus computing all of the part likelihood maps for an image requires $|\mathcal{E}|(2n+1)$ applications of the FFT and takes total asymptotic time $O(|\mathcal{E}|nh \log h)$. This approach is typically several times faster than the naïve $O(|T_i|hn)$ implementation described in Section 4.1 because $|T_i|$, the number of pixels per part template, is typically large (hundreds or thousands of pixels) while the number of possible edge labels is relatively small (fewer than a dozen).

4.1.3 Handling overlapping patches

The probabilistic framework presented in Section 1.1.2 assumes that the image likelihood factors into a product over parts. The local appearance of a part is thus assumed to be independent from that of any other part; in particular this implies that parts of an object never overlap one another, which is generally not true. For configurations with overlap, equation (1.3) overcounts some pixels and thus gives only an approximation to the true likelihood function. In [2] a model called Patchwork of Parts (POP) was introduced to address this problem. They propose averaging the probabilities of overlapping templates in order to eliminate the over-counting.

We can use a similar approach to address the patch overlap problem with our template-based models. For a given pixel p and object configuration L of patches, let $q(L,p)$ be the set of all patches that overlap p,

$$q(L,p) = \{v_i \,|\, p \in (T_i \oplus l_i)\},$$

where $\oplus$ denotes Minkowski addition, and let $Q(L)$ be the set of pixels covered by at least one patch,

$$Q(L) = \{p \,|\, q(L,p) \neq \emptyset\}.$$

Now the likelihood function in equation (4.2) can be rewritten in order to average the contributions of overlapping patches:

$$\hat{P}_M(I|L) = P_M(I|w_0) \prod_{p \in Q(L)} \frac{1}{|q(L,p)|} \sum_{v_i \in q(L,p)} h_i(p, l_i), \tag{4.8}$$

where $|\cdot|$ denotes set cardinality. For configurations in which no patches overlap, equation (4.8) simplifies to the factored likelihood function of equation (4.2).

Unfortunately an efficient inference algorithm for maximizing $\hat{P}_M(L|I,M)$ is not known, since the factorization in equation (3.3) is no longer possible. Given a particular part configuration L, however, it is easy to compute $\hat{P}_M(L|I,M)$. Thus we use a compromise approach that uses the simpler likelihood function to generate promising object configurations, but scores these configurations using the more accurate POP framework. In other words, we treat the simpler posterior $P_M(L|I)$ as a proposal distribution for the true posterior $\hat{P}_M(L|I)$. We then draw samples from the proposal distribution using the method described in Section 3.3 and evaluate the true posterior probability for each sampled configuration. The highest-likelihood configuration provides an approximation to the MAP estimate of the true posterior. In practice we have found that this process provides significantly better detection and localization results than just using the simple likelihood function.

4.2 Image gradient-based part models

The template-based models described in the last section are simple, fast, and work relatively well, but there are some situations in which these feature operators often fail. One problem is that they are sensitive to failures by the pixelwise classifier — a missed edge in a low-contrast region, for example. Another problem is that the models are very rigid, allowing for little deformation in part appearance. Thus these models work well for rigid objects like cars and motorbikes, but are unsuitable for highly deformable objects like people and animals.

Several alternative part appearance models were explored that allow for greater deformation and were less sensitive to local image contrast. We found the Histograms of Oriented Gradients (HOG) descriptor [19] to be the most successful approach. HOG descriptors capture the distribution of gradients across an image region, aggregated in both spatial and orientation dimensions. Thus they respond to edge features but operate directly on gradients instead of the output of a hard classifier like an edge detector. The quantization in spatial and orientation dimensions also allows for a greater degree of local part deformation.

In [19] the HOG descriptors were applied to object recognition using a rigid template model. Although the HOG descriptor allows for local deformation, the use of a single rigid template model for the entire object prevents more global deformations. In a sense this is the opposite of the problem with our part templates from the last section; there the deformable spatial model allows for large object-level deformations but the parts themselves are too rigid.

We combine the advantages of both approaches by showing how to use HOG features in our part-based statistical framework, thus allowing for both part-level and object-level deformations. A similar approach has recently been proposed in [31].

4.2.1 Histograms of Oriented Gradients

HOG descriptors are computed by dividing an image into small non-overlapping *cells* of about 8×8 pixels. Image gradients are computed, and then a histogram of gradient orientation weighted by gradient magnitude is built for each cell. The histograms are normalized within *blocks* of approximately 2×2 cells. More formally, for each pixel p in an image, let $G_x(p)$ and $G_y(p)$ be estimates of the image gradient in the x and y directions at pixel p,[2]

$$G_x(p) = \frac{1}{2}\left(I(p_x+1, p_y) - I(p_x-1, p_y)\right),$$

$$G_y(p) = \frac{1}{2}\left(I(p_x, p_y+1) - I(p_x, p_y-1)\right).$$

Then the gradient magnitude at p is given by,

$$M(p) = \sqrt{G_x(p)^2 + G_y(p)^2},$$

and the gradient direction is,

$$\theta(p) = Q\left(\arctan\frac{G_x(p)}{G_y(p)}\right),$$

[2] Other methods exist for estimating local image gradient [39]. We use the method presented here because it was found to work the best for object recognition in [19].

where Q is a quantization function that maps angles onto a small set $\mathcal{O}$ of orientations. A histogram is computed for each cell E_i,

$$H(E_i)[d] = \sum_{p \in E_i} M(p)\delta(\theta(p) - d),$$

for all $d \in \mathcal{O}$, where $\delta(n)$ is an indicator function that is 1 if $n = 0$ and 0 otherwise. These histograms are normalized on a per-block basis. Suppose that E_i belongs to a block B_j. Then the normalized histogram of E_i relative to B_j is,

$$\hat{H}(E_i, B_j)[d] = N(B_j)H(E_i)[d],$$

where $N(B_j)$ is a normalization function. Of the several normalization schemes studied in [19], normalizing based on L_2 distance was found to be the best,

$$N(B_j) = \left(\sum_{E_k \in B_j} \sum_{e \in \mathcal{O}} H(E_k)[e]^2 + \epsilon^2 \right)^{-\frac{1}{2}},$$

where ϵ is a small constant.

4.2.2 Part-based object detection using HOG features

In [19] these HOG descriptors are applied to object recognition using a simple rigid template approach. They build a feature vector for any given rectangular region of an image by concatenating the normalized histograms of all of the blocks underneath the region. An object model consists of a support vector machine (SVM) [23] trained on the HOG feature vectors corresponding to object regions marked in the ground truth. Object detection on a test image is performed using a sliding window classifier which exhaustively classifies subregions at many different scales and offsets.

In contrast, we use HOG descriptors to characterize the appearance of local image patches for use in our part-based object models. For simplicity we assume that the patch size is the same as the HOG block size, although this could be generalized. For a given image I and pixel location l, we define a vector consisting of the normalized HOG histograms in the block centered at l,

$$v(I, l) = \left(\hat{H}(E_1, B_l), \hat{H}(E_2, B_l), ..., \hat{H}(E_m, B_l) \right), \tag{4.9}$$

where $E_1, E_2, ...E_m$ are the cells belonging to B_l. Each vector $v(I, l)$ consists of m histograms each having $|\mathcal{O}|$ bins, so the total dimensionality of the vector is $m|\mathcal{O}|$. Using dynamic programming, $v(I, l)$ can be efficiently computed for all pixel locations in an image in time proportional to $h|\mathcal{O}|$, where h is the number of pixels in the image. In particular, note that the running time is independent of the cell size.

We use two different approaches for modeling part appearance using HOG features. The first approach uses a Gaussian distribution,

$$P_M(I|l_i) = \mathcal{N}\left(v(I, l_i)|\mu_i, \Sigma_i\right). \tag{4.10}$$

This model is used internally during the learning process, as we describe in Chapter 5. For performing recognition we use an SVM, interpreting the distance of a given HOG feature vector from the SVM's separating hyperplane as a log-likelihood ratio. That is, during recognition we use,

$$P_M(I|l_i) \propto \exp(w_i \cdot v(I, l_i) - b_i),$$

where w_i and b_i are parameters learned during the SVM training process.

CHAPTER 5

LEARNING THE MODELS

We have so far described a family of graphical models for object recognition and shown how to perform inference efficiently using these models. This chapter turns to the problem of choosing the model parameters for a given object category. This involves selecting a set of salient object parts, building an appearance model for each part, and building a spatial model that relates the geometric configuration of the parts.

In principal a human expert can develop an object model directly, using his or her intuition about what the object looks like. However setting the model parameters by hand is tedious and time-consuming. Moreover, research has repeatedly found that hand-tuned object models perform poorly compared to models produced by a learning algorithm [22, 64, 72, 87]. In this chapter we discuss algorithms for learning the object models automatically from labeled training images.

All learning algorithms require training exemplars, which in the case of object recognition are images that contain the object of interest. Approaches differ in the amount of information that they require about these exemplars, but most approaches can be categorized into one of four supervision paradigms:[1]

- *Fully-supervised* learning algorithms [14, 30] assume that the correct location of each object *part* is given as ground truth for every training image.
- *Partially-supervised* learning [19, 31] assumes that the training image data is annotated with bounding boxes enclosing each instance of the object. How-

[1] Note that the meaning of these terms is not standardized in the object recognition literature. For example, what we call weakly-supervised is referred to as “unsupervised” by [52]; what we call partially-supervised is called “supervised” in [1].

ever the individual part locations are not known.

- *Weakly-supervised* learning algorithms [2, 32, 33, 59, 86] expect an annotation for each training image indicating whether or not the object appears in it. No information on the position of the object within the image is assumed.
- *Unsupervised* techniques [67, 76] are given only a set of training images with no other ground truth data. These algorithms attempt to both discover the object classes in the training imagery and learn models for each one.

Less supervised learning methods require less human effort during training, but generally are more complex and require more computational resources. In practice, human time is usually considered more valuable than computer time and hence weaker supervision is preferred. However learning models in a fully unsupervised fashion is a very challenging problem with state-of-the-art approaches giving relatively poor results, which makes unsupervised learning impractical for now. Thus we consider the first three paradigms but do not consider unsupervised learning further in this thesis.

We frame learning the object models as a maximum-likelihood estimation problem. Given a set of positive training images $D = \{I_1, \ldots, I_T\}$, we seek a model that maximizes the likelihood of the training data,[2]

$$M^* = \arg\max_M P(D|M), \tag{5.1}$$

Exactly how the maximization in equation (5.1) is performed depends on the degree of supervision that is available. The remainder of this chapter describes algorithms for learning object models under the weakly-, partially-, and fully-supervised paradigms. First we show how to carry out the maximization in the

[2] In previous chapters we have used the notation $P_M(\cdot)$ to emphasize that the model M was a constant during inference. In this chapter we treat M as a random variable and thus switch to the alternative notation $P(\cdot|M)$.

weakly-supervised setting, where only per-image class membership labels are available. Then we show that the partially-supervised and fully-supervised learning problems can be viewed as special cases of our weakly-supervised approach. We discuss the partially-supervised and fully-supervised problems in Sections 5.2 and 5.3, respectively.

5.1 Weakly-supervised learning

In weakly-supervised learning, each training image has a binary annotation indicating whether the object of interest appears in the image. The label can be thought of as partitioning the training imagery into a set of positive images that contain the object and a set of negative images that do not. No information about the location of the object within the images is available.

Evaluating $P(D|M)$ for a particular model and set of T training images involves summing over the space of all possible model configurations in that dataset,

$$P(D|M) = \sum_{L \in \mathcal{I}^T} P(D, L|M). \tag{5.2}$$

where $\mathcal{I}$ is the set of possible object configurations for the model in an individual image. For example, if we consider only translation and assume that each training image is $p \times q$, then $\mathcal{I} = ([1,p] \times [1,q])^n$, where n is the number of parts in the model.

It is not tractable to perform the sum and maximization explicitly as the space of possible configurations is huge. Any given value of L records a specific configuration of each part of the model in each training image. Maximum likelihood estimation problems that involve latent or "nuisance" random variables like L can be solved approximately using an expectation maximization (EM) algorithm [20]. EM is an iterative hill-climbing optimization algorithm that is guaranteed to converge to a local maximum. Each iteration consists of two steps. In the first step, the posterior distribution of the latent variable is computed using an estimated model M^t. In the second step, the expectation of the likelihood function is maximized with respect to the model parameters, to produce an improved model M^{t+1}.

In our context, the general form of the EM update equation in [20] can be written,

$$M^{t+1} = \arg\max_M \mathbb{E}_L \left[\log P(D, L|M^t)|D\right] \tag{5.3}$$

where $\mathbb{E}_L\left[X|Y\right]$ denotes the expectation of random variable X conditioned on the value of another random variable Y. Assuming that the training images are independent, $P(D, L|M)$ factors into a product over images,

$$P(D, L|M) = \prod_{j=1}^{T} P(I_j, L|M),$$

so that the update equation becomes,

$$M^{t+1} = \arg\max_M \mathbb{E}_L \left[\sum_{j=1}^{T} \log P(I_j, L|M)|I_j\right].$$

Then using properties of expectations we have,

$$M^{t+1} \quad = \quad \arg\max_M \sum_{j=1}^{T} \sum_{L_j \in \mathcal{I}} P(L_j|I_j, M^t) \log P(I_j, L_j|M). \tag{5.4}$$

A k-fan object model M consists of an appearance model A that records the parameters of the image likelihood and a spatial model S that records the parameters of the prior. The logarithm in equation (5.4) factors further,

$$M^{t+1} = \arg\max_M \sum_{j=1}^{T} \sum_{L_j \in \mathcal{I}} P(L_j|I_j, M^t) \log P(I_j|L_j, A) P(L_j|S).$$

This means that the appearance model and spatial models can be maximized independently,

$$S^{t+1} = \arg\max_S \sum_{j=1}^{T} \sum_{L_j \in \mathcal{I}} P(L_j|I_j, M^t) \log P(L_j|S), \tag{5.5}$$

$$A^{t+1} = \arg\max_A \sum_{j=1}^{T} \sum_{L_j \in \mathcal{I}} P(L_j|I_j, M^t) \log P(I_j|L_j, A), \tag{5.6}$$

where $M^{t+1} = (A^{t+1}, S^{t+1})$.

We describe how to find A^{t+1} and S^{t+1}, in turn, in the following two sections.

5.1.1 Spatial model update equations

First we consider how to find the EM update equation for the spatial model. Recall from equation (2.1) that in a Gaussian k-fan the spatial prior factors as a product over cliques,

$$P(L|S) = P(l_R|S) \prod_{v_i \in \overline{R}} P(l_i|l_R, S), \tag{5.7}$$

where $R \subseteq V$ is the set of reference parts, $\bar{R} = V - R$ is the set of non-reference parts, and

$$P(l_R|S) = \mathcal{N}(l_R|\mu_R, \Sigma_R), \tag{5.8}$$

$$P(l_i|l_R, S) = \mathcal{N}(l_i|\mu_i(l_R), \Sigma_{i|R}) \text{ for each part } v_i. \tag{5.9}$$

The conditional parameters $\mu_{i|R}$ and $\Sigma_{i|R}$ for part v_i are computed by estimating $\mu_{i,R}$ and $\Sigma_{i,R}$, the mean and covariance for the clique $R \cup \{v_i\}$. In particular, $\mu_{i,R}$ and $\Sigma_{i,R}$ are treated as block matrices to define μ_i, μ_R, Σ_i, Σ_{iR}, and Σ_R using equation (2.3). The conditional parameters needed for recognition are then computed using equations (2.4) and (2.5).

We thus learn $\mu_{i,R}$ and $\Sigma_{i,R}$ for each maximal clique of the graph independently, using the joint form of the prior in equation (2.2). From equation (5.5) we have,

$$S_i^{t+1} = \arg \max_{\mu_{i,R}, \Sigma_{i,R}} \sum_{j=1}^{T} \sum_{l_R \in I_j} \sum_{l_i \in I_j} P(l_i, l_r | I_j, M^t) \log P(l_i, l_R | \mu_{i,R}, \Sigma_{i,R}).$$

The quantity being maximized here further simplifies to,

$$\sum_{j=1}^{T} \sum_{l_R \in I_j} \sum_{l_i \in I_j} -\frac{1}{2} P(l_{i,R} | I_j, M^t) \left((l_{i,R} - \mu_{i,R})^T \Sigma_{i,R}^{-1} (l_{i,R} - \mu_{i,R}) - \log Z |\Sigma_{i,R}| \right), \tag{5.10}$$

with a constant Z related only to the dimensionality of the Gaussians. We set the partial derivatives of (5.10) equal to zero and solve for $\mu_{i,R}^{t+1}$ and $\Sigma_{i,R}^{t+1}$. Taking the partial derivative with respect to $\mu_{i,R}$,

$$\sum_{j=1}^{T} \sum_{l_R \in I_j} \sum_{l_i \in I_j} P(l_i, l_R | I_j, M^t)(l_{i,R} - 2\mu_{i,R}) \Sigma_{i,R}^{-1} = 0$$

and multiplying both sides by $\Sigma_{i,R}$ and rearranging gives,

$$\mu_{i,R}^{t+1} = \frac{\sum_{j=1}^{T} \sum_{l_R \in I_j} \sum_{l_i \in I_j} P(l_i, l_R | I_j, M^t)(l_{i,R})}{\sum_{j=1}^{T} \sum_{l_R \in I_j} \sum_{l_i \in I_j} P(l_i, l_R | I_j, M^t)} \tag{5.11}$$

Taking the partial derivative of (5.10) with respect to $\Sigma_{i,R}$ and simplifying,

$$\sum_{j=1}^{T} \sum_{l_R \in I_j} \sum_{l_i \in I_j} \frac{1}{2} P(l_i, l_R | I_j, M^t) \left(\Sigma_{i,R}^{-1} (l_{i,R} - \mu_{i,R})(l_{i,R} - \mu_{i,R})^T \Sigma_{i,R}^{-1} - \Sigma_{i,R}^{-1} \right),$$

where we have used the facts that $\Sigma_{i,R} = \Sigma_{i,R}^T$, and the identities: [61]

$$\begin{aligned} \frac{\partial \mathbf{a}^T \mathbf{X}^{-1} \mathbf{b}}{\partial \mathbf{X}} &= -\mathbf{X}^{-T} \mathbf{a} \mathbf{b}^T \mathbf{X}^{-T}, \\ \frac{\partial |\mathbf{X}|}{\partial \mathbf{X}} &= |\mathbf{X}| \mathbf{X}^{-T}. \end{aligned}$$

Setting this equal to zero and solving for $\Sigma_{i,R}$ gives,

$$\Sigma_{i,R}^{t+1} = \frac{\sum_{j=1}^{T} \sum_{l_R \in I_j} \sum_{l_i \in I_j} P(l_i, l_r | I_j, M^t)(l_{i,R} - \mu_{i,R})(l_{i,R} - \mu_{i,R})^T}{\sum_{j=1}^{T} \sum_{l_R \in I_j} \sum_{l_i \in I_j} P(l_i, l_R | I_j, M^t)}, \tag{5.12}$$

which is the final update equation.

The update equations for a 1-fan model are particularly simple. In this thesis we are interested in translation-invariant models, in which $P(L_R|M)$ is a uniform distribution in a 1-fan. To make the models insensitive to absolute position, we learn the Gaussian spatial parameters relative to the reference part location. The dimensionality of Σ_R is thus zero, and from equations (2.3), (2.4), and (2.5) we have $\Sigma_i = \Sigma_{i,R} = \Sigma_{i|R}$ and $\mu_i = \mu_{i|R}$. Thus the update equations for a 1-fan are:

$$\mu_i^{t+1} = \frac{\sum_{j=1}^{T} \sum_{l_r \in I_j} \sum_{l_i \in I_j} P(l_i, l_r | I_j, M^t)(l_i - l_r)}{\sum_{j=1}^{T} \sum_{l_r \in I_j} \sum_{l_i \in I_j} P(l_i, l_r | I_j, M^t)}, \tag{5.13}$$

$$\Sigma_i^{t+1} = \frac{\sum_{j=1}^{T} \sum_{l_r \in I_j} \sum_{l_i \in I_j} P(l_i, l_r | I_j, M^t)(l_i - l_r - \mu_i)(l_i - l_r - \mu_i)^T}{\sum_{j=1}^{T} \sum_{l_r \in I_j} \sum_{l_i \in I_j} P(l_i, l_r | I_j, M^t)}. \tag{5.14}$$

5.1.2 Appearance model update equation

Now we turn to the equations for finding an appearance model A^{t+1} given an estimated model A^t. The update equations depend on the type of appearance model used. In this section we derive the equations for the two types of models proposed in Chapter 4.

Template models

We first consider the template-based models of Section 4.1. Recall that these appearance models consist of a background model and a foreground template T_i for each part,

$$P(I|L, A) = P(I|w_0, A) \prod_{v_i \in V} \prod_{p \in T_i} h_i(p + l_i, l_i), \tag{5.15}$$

where

$$h_i(p, l_i) = \frac{f_i(p - l_i)[I(p)]}{b[I(p)]},$$

and $f_i(p)[u]$ is the probability that a given pixel p in the template T_i corresponding to the i-th part has edge label u, and each background pixel has label u with probability $b[u]$. The background probabilities are trivial to estimate from the negative training images, so we will assume that b is fixed and known beforehand. Substituting equation (5.15) into (5.6) gives,

$$A^{t+1} = \arg\max_A \sum_{j=1}^{T} \sum_{L_j \in \mathcal{I}} P(L_j | I_j, M^t) \log \left(P(I_j | w_0) \prod_{v_i \in V} \prod_{p \in T_i} \frac{f_i(p)[I_j(p + L_{j,i})]}{b[I_j(P + L_{j,i})]} \right), \tag{5.16}$$

where $L_{j,i}$ denotes the location of the i-th part in configuration L_j.

From equation (5.16) we see that to estimate A^{t+1}, it is sufficient to maximize $f_i(p)[\cdot]$ independently for each part i and pixel $p \in T_j$,

$$f_i^{t+1}(p)[\cdot] = \arg\max_{f_i(p)[\cdot]} \sum_{j=1}^{T} \sum_{l_i \in I_j} P(l_i | I_j, M^t) \log f_i(p)[I_j(p + l_i)]. \tag{5.17}$$

where $P(l_i|I_j, M^t)$ is the marginal probability that part v_i occurs at location l_i in image I_j. Recall that an oriented edge detector has assigned a label from the set $\mathcal{E}$ to each pixel in the image. The quantity to maximize in equation (5.17) can be rewritten as a sum over the possible edge labels,

$$\arg\max_{f_i(p)[\cdot]} \sum_{j=1}^{T} \sum_{l_i \in I_j} \sum_{e \in \mathcal{E}} P(l_i | I_j, M^t) 1_e(I_j, p + l_i) \log f_i(p)[e],$$

subject to the constraint the sum of the probabilities over edge labels for a given pixel in the template must sum to one,

$$\sum_{e \in \mathcal{E}} f_j(p)[e] = 1,$$

and where $1_e(I, p)$ is an indicator function that is one if $I(p) = e$ and is 0 otherwise.

We use Lagrange multipliers to perform this maximization for each part v_i and pixel $p \in T_i$. The Lagrangian is,

$$\mathcal{L}(f_i(p), \lambda) = \lambda \left(\sum_{e \in \mathcal{E}} f_i(p)[e] - 1 \right) + \sum_{j=1}^{T} \sum_{l_i \in I_j} \sum_{e \in \mathcal{E}} P(l_i | I_j, M^t) 1_e(I_j, p + l_i) \log f_i(p)[e]$$

Taking partial derivatives with respect to $f_i(p)[e]$ for each edge label e, we obtain a system of $r+1$ equations for each pixel p in each part v_i. Each of these constraints has the form,

$$\frac{\partial \mathcal{L}(f_i(p), \lambda)}{\partial f_i(p)[e]} = \lambda f_i(p)[e] + \sum_{j=1}^{T} \sum_{l_i \in I_j} P(l_i | I_j, M^t) 1_e(I_j, p + l_i).$$

Solving the system of equations for $f_i(p)[e]$ for $e \in \mathcal{E}$ gives the EM appearance model update equation,

$$f_i^{t+1}(p)[e] = \frac{\sum_{j=1}^{T} \sum_{l_i \in I_j} P(l_i | I_j, M^t) 1_e(I_j, p + l_i)}{\sum_{j=1}^{T} \sum_{l_i \in I_j} \sum_{e' \in \mathcal{E}} P(l_i | I_j, M^t) 1_{e'}(I_j, p + l_i)}. \tag{5.18}$$

In summary, finding A^{t+1} involves computing $f_i^{t+1}(p)[e]$ using equation (5.18) for each combination of part v_i, template pixel $p \in T_i$, and edge label $e \in \mathcal{E}$.

Gradient-based models

For the gradient-based models of Section 4.2, we need to estimate the μ_i and Σ_i parameters of equation (4.10) for each part v_i. Using a derivation very similar to that shown for Gaussian k-fans in Section 5.1.1, it is straightforward to find the following update equations for the gradient-based appearance models:

$$\mu_i^{t+1} = \frac{\sum_{j=1}^{T} \sum_{l_i \in I_j} P(l_i | I_j, M^t) v(I, l_i)}{\sum_{j=1}^{T} \sum_{l_i \in I_j} P(l_i | I_j, M^t)}, \tag{5.19}$$

$$\Sigma_i^{t+1} = \frac{\sum_{j=1}^{T} \sum_{l_i \in I_j} P(l_i | I_j, M^t) v(I, l_i) v(I, l_i)^T}{\sum_{j=1}^{T} \sum_{l_i \in I_j} P(v(I, l_i) | I_j, M^t)}. \tag{5.20}$$

5.1.3 Learning an initial model

Since EM is a local search technique, it is important to start with a reasonable initial model M^0. Our approach is to generate a large set of candidate appearance templates that seem promising based only on how well they individually predict the training data. Then we examine the configurations of these templates in the training images both to choose which candidates to include in the initial appearance model A^0 and to define an initial spatial model S^0. We now discuss how to learn an initial model from weakly-supervised training data.

Candidate Patch Models

We first generate a large set of potential appearance models based on the training image data. We do this by randomly sampling image patches from the positive training images and generating a part operator for each sample. The way this is done depends on the type of feature operator being used.

For the template-based appearance models of Section 4.1, we uniformly sample patches of several different sizes at random from the edge maps of the positive training images. Then we generate a template model T_i from each sampled patch. We do this by dilating the patch in both the spatial and orientation dimensions, and then setting the template probabilities to reasonable values based on the dilated patch. We use a probability of 0.85 for the observed edge orientation and distribute the rest of the probability mass uniformly among the other orientations.

For the gradient-based models of Section 4.2, we sample image locations from the positive training images and compute the corresponding HOG feature vectors. We then build a feature operator for each vector by estimating the mean and covariance parameters of equation (4.10); in particular we set μ_i to the observed HOG feature vector and set $\Sigma_i = 0.1I$, where I is the identity matrix.

For the experiments reported in this paper we sampled approximately 10,000 initial patches at three different scales. Note that due to the redundancy in the sampling of the initial patches, many of the resulting feature operators may be quite similar. However we do not attempt to cluster or eliminate operators at this stage.

Building an initial model

The previous step generates a very large set of candidate patches. We want to build an initial k-fan model (for a given k) by choosing a small subset of these candidate patches so as to maximize the likelihood in equation (5.1). Combining equations (3.1), (3.2), and (5.2), we have

$$M^* = \arg\max_M \prod_{I_j \in D} Z(I_j) \sum_{l_R} P_M(l_R) \left(\prod_{v_i \in R} P_M(I_j|l_i) \prod_{v_i \in \overline{R}} \sum_{l_i} P_M(I_j|l_i) P_M(l_i|l_R) \right),$$

assuming as before that the parts do not overlap. Assuming that the set of reference patches R has already been chosen, the best non-reference part set $\overline{R}$ is

$$\arg\max_{\overline{R}} \prod_{v_i \in \overline{R}} q(R \cup \{v_i\}), \tag{5.21}$$

where

$$q(G) = \prod_{I_j \in D} \sum_{l_G} P(l_G|s_G) \prod_{v_i \in G} P(I_j|l_i). \tag{5.22}$$

and s_G is a maximum-likelihood full multivariate Gaussian on the relative locations of the patches in G. $q(G)$ is high for groups of patches that both individually predict the training data well and that appear at predictable relative spatial locations in the training images. This is a natural measure of how valuable a given set of patches would be as a clique in a k-fan object model.

These observations suggest a natural procedure for learning an initial spatial model. For every possible combination G of $k + 1$ candidate patches, the clique quality score $q(G)$ is computed. As we discuss below, this involves estimating a spatial model $s_G = (\Sigma_G, \mu_G)$ on the locations of the patches in the training data. Then all possible combinations of k candidate patches are considered for the reference set R. For each possible reference set, a k-fan model is built by choosing the cliques with the highest quality scores that include R. Because we assume that parts do not overlap, we use a greedy procedure that sorts the quality scores and chooses the first $n - k$ cliques in which the non-reference parts do not overlap one another. In practice a small degree of overlap is allowed. This greedy process continues until there are either no more cliques left to add, or until some pre-determined maximal number of parts is reached. After completing this process we have a large set of candidate k-fan models, one for every possible reference set. We then compute the likelihood $P_M(D)$ for each of the models and choose the one with highest likelihood. This model becomes our initial model $M^0 = (A^0, S^0)$ that is then improved using the EM procedure described previously.

For models using the HOG-based features, we train an SVM for each part's appearance after the last iteration of the EM procedure. To do the learning we use the `SVM-Light` toolkit of [47]. For the positive training exemplars we use the maximum *a priori* estimate for the part location in each training image according to the object model produced by the last iteration of EM. The exemplar corresponding to each of these estimates is the feature vector of equation (4.9). For the background exemplars we randomly sample many regions from the training images.

Implementation

The summations in the clique quality scores in (5.22) can be computed efficiently using convolutions, as described in Section 3.1. However computing the scores for all possible combinations of $k+1$ patches is still intractable when the number of candidate patches is large. To speed up the computation, we approximate the clique quality by sampling a small number of patch locations from $P_M(l_i|I_j)$ and perform the summation in (5.22) only over those sampled locations. Also, when $k \geq 2$, we can approximate the spatial prior $P(l_G|s_G)$ as the product of pairwise patch likelihoods,

$$P(l_G|s_G) \approx Z \prod_{\{p,q\} \subseteq G} P(l_p, l_q|s_{p,q}),$$

where $s_{p,q} = (\mu_G, \Sigma_G)$ is a Gaussian on the pairwise relative location of patches p and q, and Z is a normalization constant. In our implementation we use this approximation to select a small fraction of high-likelihood patch combinations, and then compute the true prior when computing the clique quality scores for these promising combinations.

Some objects have two or more distinct parts that are similar in appearance. Examples include the two wheels of a bicycle and the two eyes of a human face. For these objects, the relative configuration of a set of patches may be a multimodal distribution and thus is poorly modeled by a single Gaussian. In these cases we have found it is better to fit a model to the strongest mode and ignore the rest of the distribution. This approach is motivated by the high degree of redundancy in the patches, making it unnecessary at this stage to explicitly handle patches that match at multiple locations. In practice, we handle this case by fitting a mixture of Gaussians model with a small number of mixture components when a single Gaussian is not a good fit. We then choose the highest-likelihood mixture component and use the mean and covariance of that component as the parameters of s_G.

5.2 Partially-supervised learning

The generality of the weakly-supervised approach just described comes at a high computational cost: learning a single object model takes about 24 hours on a 40-processor cluster of Pentium 3 and Pentium 4 machines. However this level of generality is not required when additional ground truth information is available, such as object bounding boxes. For example, the training data of the PASCAL Visual Object Challenge (which we use in our experiments in Chapter 7) includes bounding boxes around object instances. These bounding boxes can be exploited to significantly reduce the training times.

Partially-supervised learning is implemented as a simple modification to the weakly-supervised technique just described. In particular we modify the procedure for learning the initial model estimate, M^0. Candidate parts are still sampled randomly from the training image set, but only from within regions marked with bounding boxes in the ground truth. Then the summation in equation (5.22) is performed only over configurations in which all of the parts lie inside the object's bounding box.

A further simplification is possible by learning the part locations with respect to the bounding boxes, by for example forcing one of the reference parts to lie at the center of each bounding box. Then we have to perform the maximization in equation (5.21) only $\binom{N}{k-1}$ times instead of $\binom{N}{k}$ times as with weakly-supervised learning, where N is the number of candidate parts. This is a significant speed-up because N is usually on the order of tens of thousands.

5.3 Fully-supervised learning

In the supervised learning case, the actual part configurations in the training exemplars are known. Assume for simplicity that each training image contains exactly one instance of the object of interest. Then the fully-supervised training exemplars are a set of image-object configuration pairs, $\{(I_1, L_1), \ldots, (I_T, L_T)\}$. As before we denote the location of the i-th part in the j-th training image as $L_{j,i}$.

This extra ground truth data simplifies the learning process considerably. We show how the fully-supervised learning equations can be derived directly from the weakly-supervised EM update equations (5.18), (5.11), and (5.12). In the supervised learning case $P(l_i|I_j, M)$ is equal to either one or zero, under the assumption that each part appears exactly once per training image; specifically, $P(l_i|I_j, M)$ is one if $l_i = L_{j,i}$ and zero otherwise. Substituting this into the appearance model EM update equations from Section 5.1, we have,

$$f_i(p)[e] = \frac{\sum_{j=1}^{T} 1_e(I_j, p + L_{j,i})}{\sum_{j=1}^{T} \sum_{e' \in \mathcal{E}} 1_{e'}(I_j, p + L_{j,i})}. \tag{5.23}$$

Similarly, $P(l_i, l_R|I_j, M)$ is either zero or one, so we can rewrite the spatial model equations as,

$$\mu_{i,R} = \frac{1}{T} \sum_{j=1}^{T} L_{j,iR} \tag{5.24}$$

$$\Sigma_{i,R} = \frac{1}{T} \sum_{j=1}^{T} (L_{j,iR} - \mu_{i,R})(L_{j,iR} - \mu_{i,R})^T. \tag{5.25}$$

Note that these are no longer update equations since M does not appear on the right hand side. Instead, these equations find a globally-maximal object model M^* in a single iteration, given a specific choice of the reference set R. Moreover, these equations are quite intuitive. Equation (5.23) simply computes the fraction of images in which a given location p in the template for part v_i has label e. Equations (5.24) and (5.25) are just the maximum-likelihood estimates of Gaussians on part configuration.

All that remains is to choose the reference part set R. To do this we learn a separate model for each choice of R and select the one that maximizes $P(D|M)$. Since the choice of reference set does not affect the appearance model A, it is sufficient to a learn a spatial model using equations (5.24) and (5.25) for each of the $\binom{n}{k}$ possible reference sets, and then choosing the one that maximizes $P(D|S)$,

$$P(D|S) = \prod_{j=1}^{T} P(L_j|S).$$

CHAPTER 6

HIERARCHICAL MODELS OF OBJECTS AND SCENES

In real images, objects do not usually appear in isolation but as one of many components of an overall scene. There are typically strong correlations between these components. For example, cars are usually pictured above some surface of support, which is usually pavement or gravel. Cars are rarely seen on grass and almost never on water; birds, on the other hand, can be observed on water, land, or in the air. Bicycles and people are very likely to co-occur; bicycles and fish are not.

In the last several chapters we have introduced the essential ingredients of an object category recognition system: a family of spatial models, part appearance operators, efficient inference algorithms, and learning techniques for various degrees of supervision. In this chapter we show how to combine these ingredients to build hierarchical models designed for object class recognition in consumer images. These hierarchical models collect evidence from the image at multiple scales, and are thus able to capture both fine object details as well as broad characteristics of a scene.

Recently there have been some impressive demonstrations of the power of scene context for classification and to a lesser extent for localization (e.g., [42, 57, 75, 82, 78]). The past few years have seen a resurgence of work on context-based recognition, which is an area where research dates back to the 1970's. Whereas much of the early work on scene context sought to parse the scene, accounting for all the objects and relations between those objects, more recent work has used contextual information to improve recognition. For instance, information about the locations of sidewalks, roads or the horizon has been shown to improve localization of automobiles and pedestrians (e.g., [42, 82]) and the co-occurrence of objects in an office scene can aid in recognition (e.g., [57]). There has also been relatively little work on the use of scene context for multi-part spatial models; in contrast, previous work has tended to focus on rigid template object models (e.g., [42, 57]) or on pixel-level classification (e.g., [75]).

In this chapter we show how the object recognition framework and techniques we have described in earlier chapters can be used to model both an object and the surrounding scene. We build a single statistical model that includes part operators tuned for different object scales, and then during inference we combine evidence from all of the scales in a single unified step. This is in contrast to other approaches like [78] that do bottom-up or top-down inference strategies where hard intermediate decisions are made. We learn the scene models and object models automatically, unlike [42] for instance where knowledge of priors on object height and scene geometry are assumed.

6.1 Combined scene and object models

We extend the part-based model framework described so far to not only represent an object in terms of its constituent parts but also to represent the surrounding scene context. The form of the object and scene components of our model are the same, differing only in the types of parts that are used for the object versus the scene. More formally, a model now consists of an object component and a scene component, each of which is composed of a collection of patches and spatial relations between pairs of those patches. Let $V^O = (v_1^O, \ldots, v_n^O)$ be the set of patches in the object model with pairwise spatial relations $E^O \subseteq V^O \times V^O$ among those parts. Similarly let the set of scene patches be $V^S = (v_1^S, \ldots, v_m^S)$ with pairwise spatial relations $E^S \subseteq V^S \times V^S$. In addition to the spatial relations between pairs of object patches and those between pairs of scene patches, there are also spatial relations between pairs that consist of one scene patch and one object patch, which we denote $E^{OS} \subseteq V^O \times V^S$. A model thus consists of the object and scene patches, the spatial relations within those sets of patches and the spatial relations between the two sets of patches, $M = (V^O, V^S, E^O, E^S, E^{OS})$.

Now a configuration L of the model is a particular placement of the object parts and the scene patches; that is, $L = (l_1^O, \ldots, l_n^O, l_1^S, \ldots l_m^S)$ where each l_i^S specifies the location of scene patch v_i^S and l_i^O the location of object patch v_i^O. While a configuration L specifies locations for each scene patch as well as each object patch, we consider only the locations of the object patches in localizing an instance of an object in an image (i.e., in placing a bounding box around the instance). The scene patches serve to constrain the configuration of object patches via the spatial relations of the model, but are not themselves part of the object location.

As before, we are interested in finding configurations that maximize the posterior, $P_M(L|I)$. As we saw in Chapter 3, the running time for performing inference on a spatial model is related to the maximal clique size of the underlying graphical model. Thus for computational reasons we seek a combined scene and object model that forms a tree. We do this by using a 1-fan model for the scene and 1-fan model for the object, where the root part of the object is one of the parts of the scene model. More formally, the underlying graphical model $G = (V, E)$ has vertices $V = V^O \cup V^S$ and edges $E = E^O \cup E^S \cup E^{OS}$, where E^O and E^S each form a tree of height 1 and E_{OS} is a single edge. The structure of the combined graphical model is illustrated in Figure 6.1. Then the independence assumption on the image likelihood from equation (1.3) implies that,

$$P_M(L|I) = P_M(L^O)P_M(L^S)P_M(L^{OS})P_M(I|L^O)P_M(I|L^S) \tag{6.1}$$

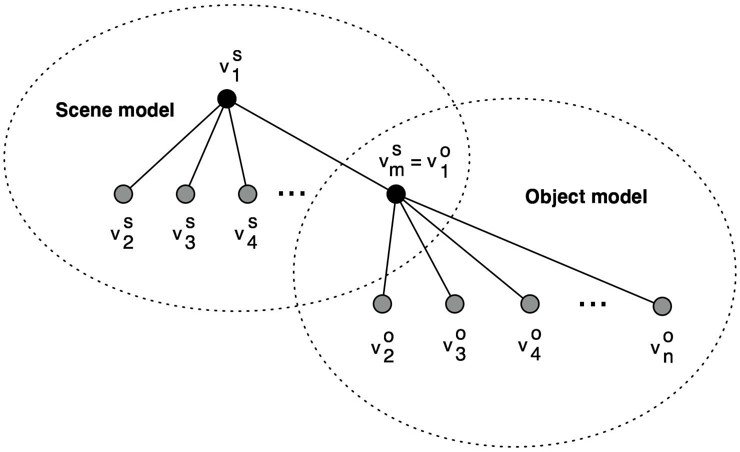


Figure 6.1: Graphical model of the combined scene and object models.

where

$$\begin{aligned}
P_M(L^O) &= \prod_{e_{ij} \in E^O} P_M(l_i^O, l_j^O) \\
P_M(L^S) &= \prod_{e_{ij} \in E^S} P_M(l_i^S, l_j^S) \\
P_M(L^{OS}) &= \prod_{e_{ij} \in E^{OS}} P_M(l_i^O, l_j^S) \\
P_M(I|L^O) &= Z^O(I) \prod_{v_i \in V^O} P_M(I|l_i^O) \\
P_M(I|L^S) &= Z^S(I) \prod_{v_i \in V^S} P_M(I|l_i^S).
\end{aligned}$$

A given object category may consist of more than one model of the form described in this section. For instance, there may be models corresponding to different viewpoints, to distinctive sub-categories of objects, or even to different scene contexts. In the experiments reported in this thesis, we used one scene context model per object category, with one object model per viewpoint as defined by the labels in the training data.

6.2 Scene appearance models

Any of the appearance operators presented in Chapter 4 can in principle be used to model the scene-level patches, however for the experiments in Chapter 7 we use the non-parametric template-based operators using several different types of image features. Regardless of the feature type, the scene appearance operators have a coarser spatial resolution than those of the object parts. We do this because the goal for the scene models is to represent general characteristics of the scene and not details of individual objects. Thus our combined scene and object models can be viewed as hierarchical: the scene-level component of the model captures coarse features over broad portions of the image, while the object-level component sees the finer detail.

We use three types of image features in particular:

- **Oriented edges:** These are the same oriented edge feature operators described in Chapter 4. For scene-level patches this type of operator tends to capture strong intensity variations corresponding to the horizon, roads, large buildings, etc.
- **Color:** Color is an important feature for establishing scene context. For examples, bicycles are often observed above an area of green (grass) or gray (pavement) but rarely appear above an area of blue (sky or water). The color quantization algorithm in [55] is applied to label each pixel with one of ten basic color clusters, yielding ten possible labels for each pixel.
- **Surface orientation:** The surface orientation of regions around an object often provides useful contextual cues. For example, many objects like cars and bicycles are usually observed resting on a horizontal support surface

such as a road. We use the surface orientation classification algorithm in [41] to classify each image pixel as one of three labels: ground, sky, or vertical surface, yielding three possible labels for each pixel.

6.3 Learning the combined models

For learning scene models we assume that bounding boxes around objects are available in the ground truth of the training imagery. This simplifies learning because we can use the partially-supervised technique of Chapter 5 and learn the object and scene models separately. In learning the object model (V^O, E^O), only the regions inside object bounding boxes (as given by the ground truth) are considered by the learning algorithm. In learning the scene context model (V^S, E^S), whole images are processed, but the algorithm is constrained to produce a model that includes the bounding box of the object as one of its patches. Finally the model for the edge connecting the object and scene models, E^{OS}, is learned by simply estimating a Gaussian on the relative location of the object model's reference part within the object bounding box.

Most state-of-the-art learning algorithms (including those presented in Chapter 5) assume that all of the training images show an object from the same viewpoint — frontal views of cars, for example. However for use in unconstrained consumer images, an object recognition system must be able to detect an object from many different viewpoints. We thus learn separate k-fan models for each of several canonical viewpoints for each object class. It is assumed that coarse viewpoint labels are provided in the ground truth for the training data (which is the case in most publicly available datasets), so that we can invoke the partially-supervised learning algorithm multiple times to learn each viewpoint-specific model independently.

CHAPTER 7

EXPERIMENTAL RESULTS

In this chapter we give experimental results for the models and algorithms proposed in this thesis. We present results on multiple datasets and for both the localization and classification tasks, unlike most recent papers which study only classification on a single dataset (e.g. [32, 33, 58]). Classification is a useful task for some applications — notably content-based image retrieval — but many applications also require some notion of the location of the object in the image. For example, a visual collision avoidance system installed in an automobile needs to decide whether or not a collision with another car is imminent — not just whether another car appears in the imaged scene. Thus we believe that results on the localization task are critical to a complete experimental evaluation of an object recognition approach.

The choice of test dataset is crucial to conduct a meaningful experimental evaluation. Most papers test against a single set of images, usually using one of the popular publicly-available datasets such as Caltech-4 [32], Caltech-101 [28], UIUC [70], MSR [17], and Graz [58]. While these datasets have fueled much of the dramatic progress on the category recognition problem over the last few years, each of them suffers from some biases related to the way they were collected [62]. The problem is that a set of images taken by a single photographer (and especially a single computer vision researcher) is not a representative sample of real-world images. We discuss one such bias in Caltech-4 in Section 7.2.7. A better approach is to use images taken by a large number of consumer photographers, and to use multiple datasets to guard against biases in any single collection.

In this chapter we describe extensive experimentation for both the classification and localization tasks. We first review the datasets that are used for our experiments, namely the popular Caltech-4 set [32] and the very challenging PASCAL VOC sets [25, 26]. We present classification experiments in Section 7.2. The main goal of these experiments is to understand how the parameter k affects recognition performance of k-fan spatial models. We also compare our results to other recent work using part-based spatial models, showing that our approach of using exact inference with weaker spatial models outperforms using approximate inference with rich models. We also compare classification performance of models learned using the weakly-supervised and fully-supervised paradigms, finding that the weakly-supervised models actually perform better despite having less ground truth information. We then turn to the localization experiments, first studying part-level localization in Section 7.3 and then object-level localization in Section 7.4. We show that the hierarchical models of Chapter 6 significantly outperform the single-scale models, and give state-of-the-art performance on our difficult test datasets.

7.1 Datasets

We use three publicly-available datasets for our evaluations: Caltech-4 [32], 2006 PASCAL VOC [25], and 2007 PASCAL VOC [26]. A summary of the datasets is presented in Table 7.1.

The Caltech-4 set [32] includes a total of about 4,400 images with ground truth for four airplanes, motorbikes, cars, and faces. There is also a negative image set consisting of 800 empty road scenes and 800 general scenes that do not contain any of these objects. The viewpoint of the objects is constant, with only side views of airplanes and motorbikes and only rear views of cars. The objects are

Table 7.1: Summary of test image sets.

	Caltech-4 [32]		**PASCAL 2006** [25]		**PASCAL 2007** [26]	
Object class	# of objects	# of images	# of objects	# of images	# of objects	# of images
Airplane	800	800	—	—	612	486
Bicycle	—	—	649	538	706	486
Bird	—	—	—	—	972	660
Boat	—	—	—	—	580	362
Bottle	—	—	—	—	1,010	488
Bus	—	—	468	354	458	372
Car	800	800	1,708	1,097	2,500	1,426
Cat	—	—	774	858	752	674
Chair	—	—	—	—	1,596	890
Cow	—	—	628	403	518	282
Dining table	—	—	—	—	430	400
Dog	—	—	845	735	1,020	842
Faces	435	435	—	—	—	—
Horse	—	—	650	501	724	574
Motorbike	800	800	549	469	678	490
People	—	—	2,309	1,341	9,380	4,016
Potted plant	—	—	—	—	1,028	490
Sheep	—	—	843	489	192	514
Train	—	—	—	—	594	522
Television	—	—	—	—	648	512

large and centered in many images, although object scale does vary considerably. Most images contain either zero or one instance of exactly one of the four object categories. A random subset of images from Caltech-4 is shown in Figure 7.1.

The PASCAL 2006 Visual Object Classes (VOC) Challenge data [25] includes a total of 5,304 images with ground truth for 9,507 instances of ten object categories, as shown in Table 7.1. The set is a combination of consumer images collected from the online photo sharing site Flickr.com and images from the Microsoft Research

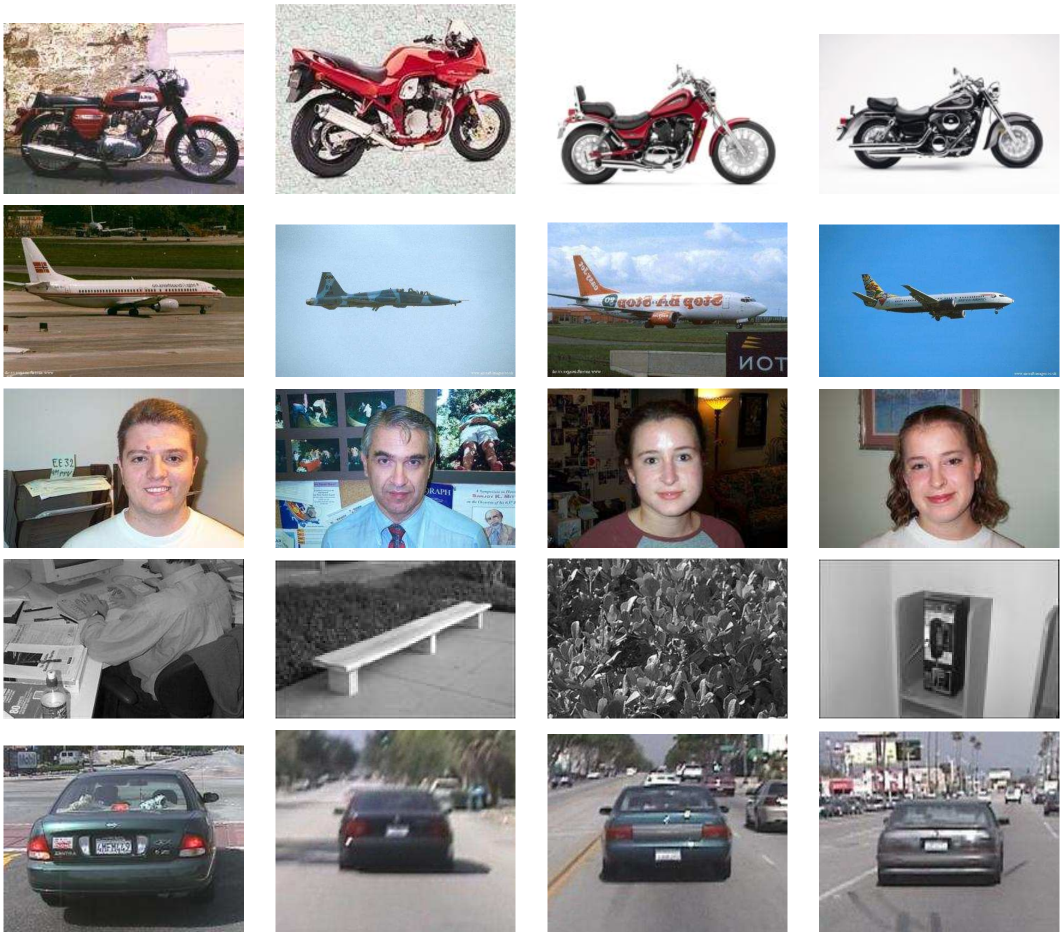


Figure 7.1: A random subset of images from the Caltech-4 dataset.

Cambridge Object Recognition Image Database [17]. The images contain a wide variety of scenes with unconstrained illumination and object scale and viewpoint, although the images from the MSR set are mostly objects in canonical viewpoints (e.g. cars viewed from the front, side, or rear). A sample set of these images is shown in Figure 7.2. The ground truth includes a bounding box for each object instance as well as a coarse viewpoint label (e.g. front, rear, left, right, or unspecified). The bounding boxes are aligned with the image axes (i.e. they do not have information about object orientation). Each bounding box optionally has a "truncated" flag indicating if only a portion of the object is visible due to occlusion or the frame boundary. Some bounding boxes are marked "difficult" if for example the viewpoint is particularly unusual or only a small portion of the object is visible. The ground truth includes bounding boxes for every visible object instance, even if the object is very small.

The PASCAL 2007 Visual Object Classes (VOC) Challenge data [26] is similar to the 2006 dataset but is larger and more challenging. It is composed of 9,963 images with ground truth for 24,640 instances of twenty object categories, and consists entirely of unconstrained consumer images from the photo sharing site Flickr.com. A sampling of images in the 2007 dataset is shown in Figure 7.3. To our knowledge this is the largest and most challenging publicly-available dataset for evaluating object category recognition algorithms available to date.

7.2 Classification experiments

Our first set of experiments concerns the *image classification* task: deciding whether an image contains an instance of a given object category (without having to determine its position). An important goal of these experiments is to determine the degree to which spatial structure affects object recognition performance. Since the running time for both learning and inference varies exponentially with k, in practice it is best to choose the lowest value of k that provides adequate classification and localization performance. We also compare our results to the constellation models of Fergus et al [32, 33] who use full multivariate Gaussians as the spatial priors (i.e. they use $n-1$-fans, where n is the number of parts). Their recognition algorithm uses feature detection and other search heuristics since exact inference

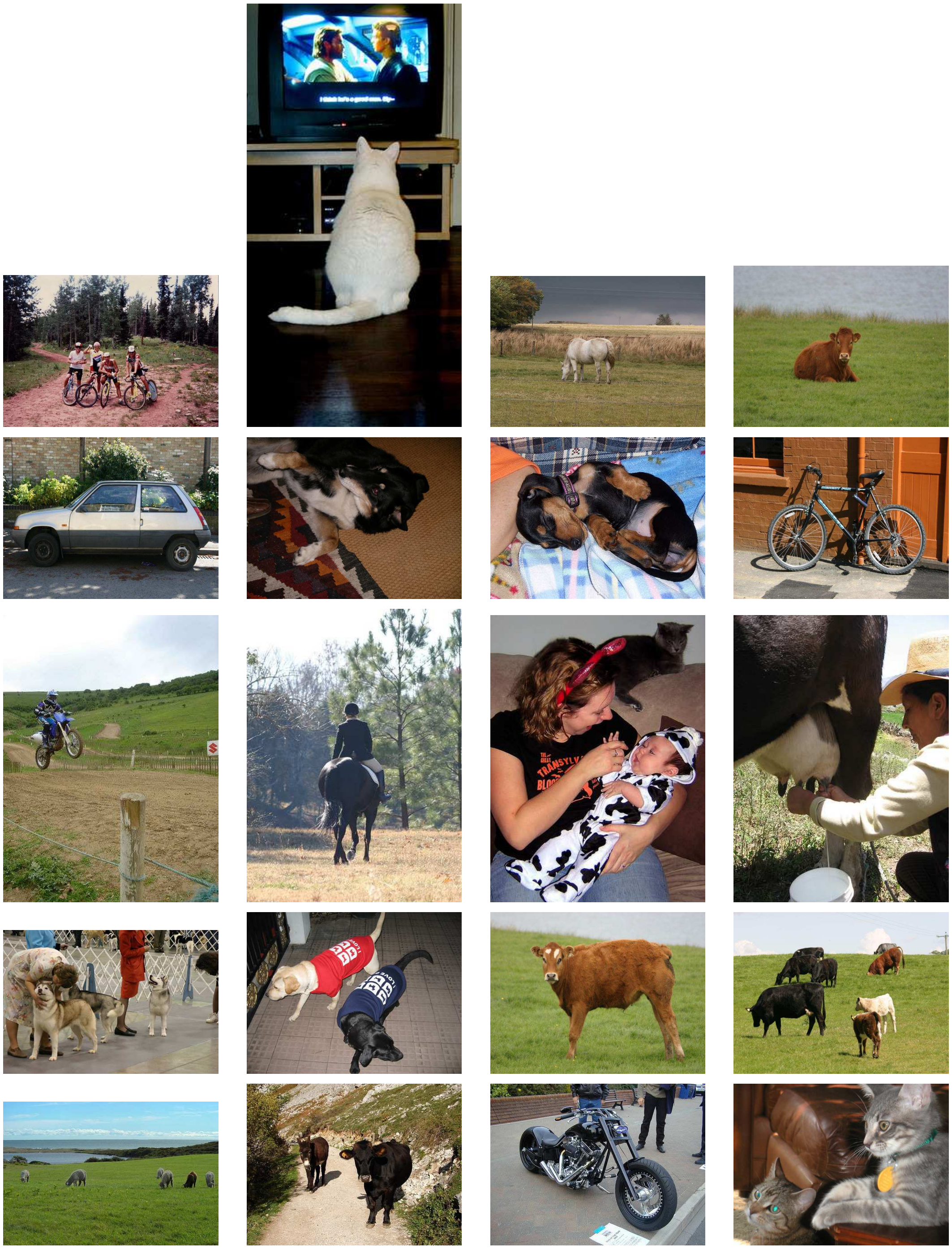


Figure 7.2: A random subset of images from the PASCAL 2006 VOC dataset.

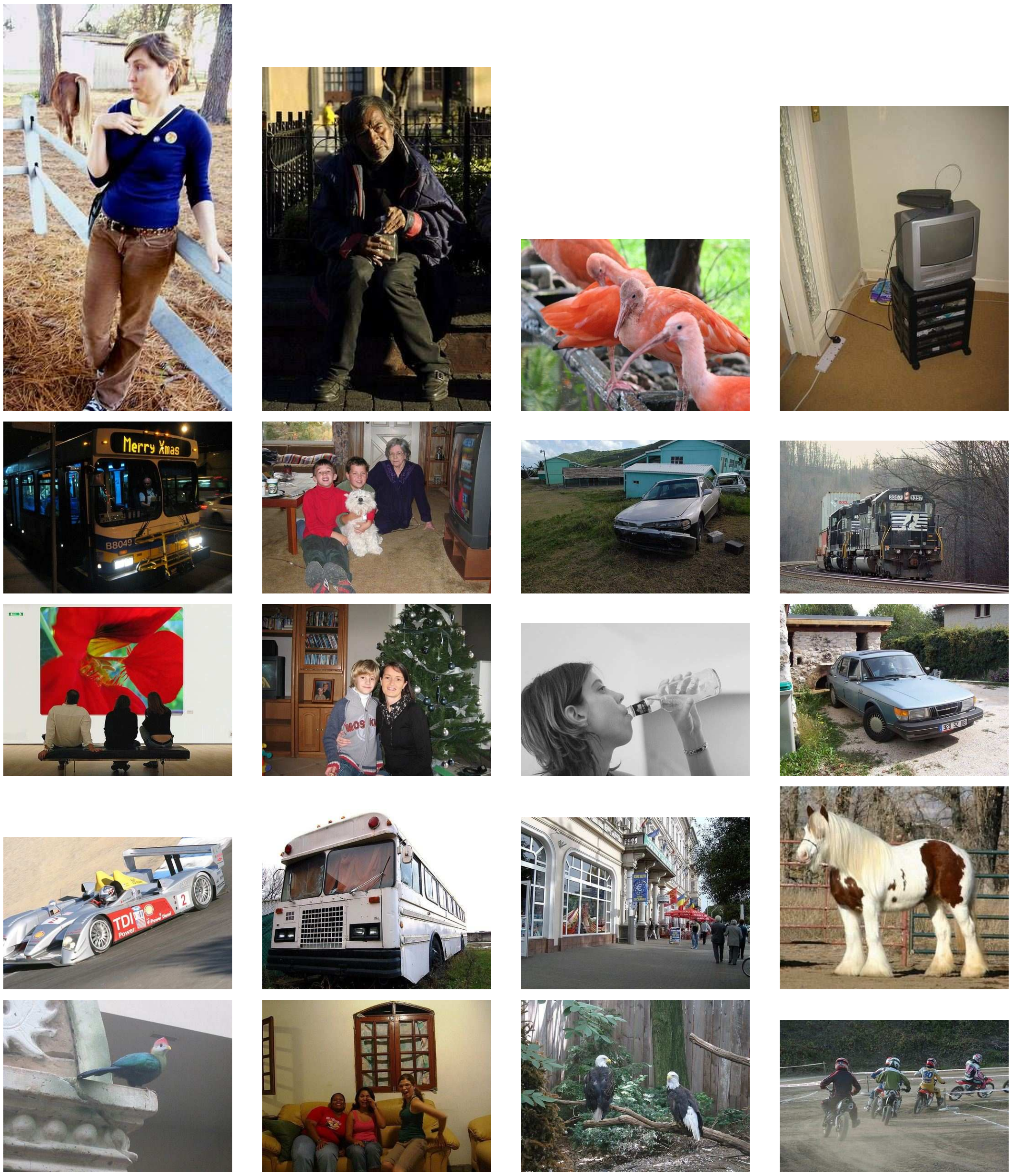


Figure 7.3: A random subset of images from the PASCAL 2007 VOC dataset.

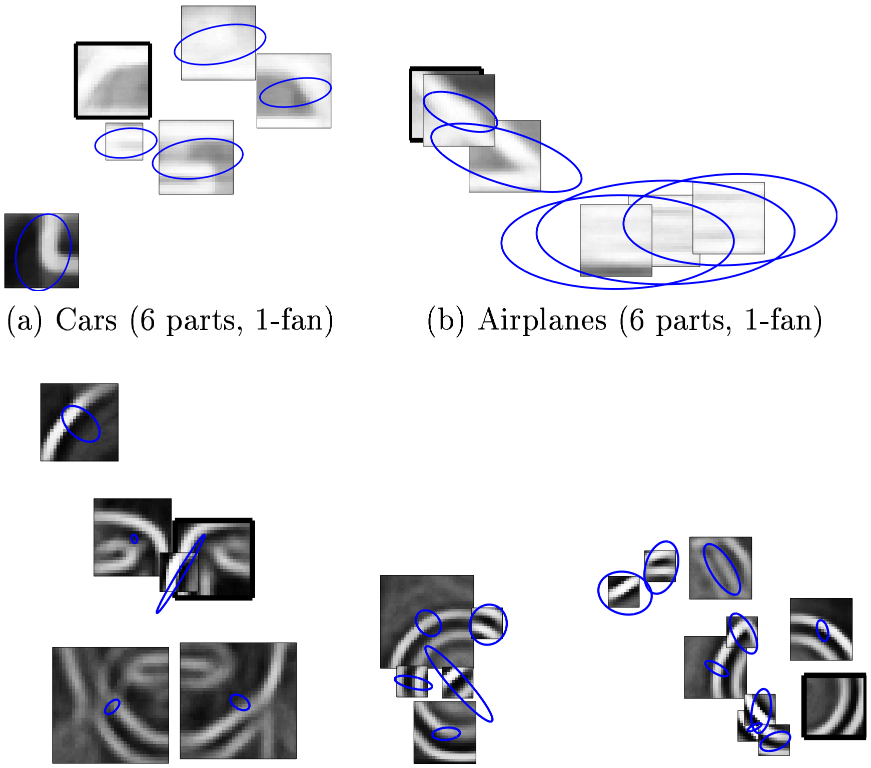
is not computationally tractable. Thus we also study whether it is best to use rich spatial models with approximate inference, as constellation models do, or weaker spatial models with exact inference, as we do.

7.2.1 Experimental protocol

To facilitate comparison we duplicated the experimental protocol and datasets of the constellation models work [32, 33]. In particular we used the same dataset (Caltech-4), used the same partitioning of test and training images, the same set of object categories (cars, faces, bicycles, motorbikes), and evaluated using the same ground truth.

Object models were trained using the weakly-supervised learning algorithm described in Chapter 5. For the classification experiments we used the edge templates of Section 4.1 as the appearance models. No positional information was given to the learning algorithm; the only ground truth information available was whether or not a given training image contained the object of interest. However we prescaled all training images so that object width was roughly uniform. As in [32] and [33], six parts were used to model each object. Figure 7.4 illustrates some of the models that were learned using the weakly-supervised procedure. Note that in each case the configuration of parts is readily recognizable as a prototype of the object. No hand-tuning of model parameters was carried out; all of the model parameters were either learned automatically by the weakly-supervised procedure or were constants that were fixed across all object classes.

To characterize classification performance we use the *equal-ROC rate* statistic, which is the point at which the true positive rate equals one minus the false positive



(a) Cars (6 parts, 1-fan)

(b) Airplanes (6 parts, 1-fan)

(c) Faces (15 parts, 1-fan)

(d) Motorbikes (6 parts, 2-fan)

Figure 7.4: Illustration of some models learned by the weakly-supervised algorithm. Each model is drawn with the part appearance templates positioned in their mean configuration and thick borders around the reference parts. Ellipses indicate the conditional covariances for the non-reference parts given the locations of the reference parts. High intensity pixels represent high edge probabilities. For clarity, just the probability of an edge is shown, even though the actual part models capture probabilities at multiple edge orientations.

rate. In other words, we plot a Receiver Operating Characteristic (ROC) curve over the range of possible values of the classification threshold, with the true positive rate on one axis and one minus the false positive rate on the other. The equal-ROC rate is the point at which this curve intersects the line $y = x$.

7.2.2 Scale-normalized classification

We first evaluate the performance of k-fan object models on scale-normalized versions of the Caltech and Graz image sets. Table 7.2 presents the results, showing the equal-ROC rate on each two-class classification task. Note that all the results shown in the table are directly comparable because the image data and experimental protocol were identical across all of these tests. The results show that classification accuracy increases as more structure is added to the spatial model (as k increases). There is a substantial improvement for all object classes between the 0-fan models and the 1-fan models. There is also some improvement as k increases from 1 to 2 for the airplanes, but little or no improvement for the other classes. This suggests that for some objects and image sets, increasing the degree of spatial constraint in the object model improves classification performance whereas for other objects and image sets additional spatial information provides little or no benefit. In part this may be due to the fact that the positive versus negative images in this database are highly different from one another, making it unnecessary to use spatial relationships to distinguish positive from negative.

The table also compares results to the constellation models of [32] and [33]. Our results are significantly better, suggesting that it is better to use weaker spatial models with exact inference, as we do, than rich spatial models with an approximate inference algorithm.

The table also compares recognition results achieved using models learned with the fully-supervised learning paradigm, in which the learning algorithm had ground truth specifying the actual location of each part in each training image. Surprisingly, the results from the models learned under weak supervision are significantly better. This is an encouraging result that speaks to the efficacy of our weakly-supervised learning algorithm, as one might expect that carefully hand-labeled data would yield better performance.

To explore this result further we repeated the supervised learning for airplanes on four additional sets of hand-labeled ground truth data produced by four different people. The people were not computer vision researchers and were not connected with our object recognition project in any way. They were instructed to select six parts of the airplane and then locate those six parts in each of the training images. We then used the supervised learning algorithm to produce an object model based on each set of ground truth and evaluated the classification accuracy of each of the resulting models. The equal ROC points for the four different models were 89.1%, 91.1%, 92.4%, and 93.4%. Thus there was a small amount of variation in performance across the supervised models, but none of them matched the performance of the model learned by weak supervision.

We compared the models learned by the weakly- and fully-supervised approaches to try to explain this surprising result. In labeling the training set, a human must make two sets of decisions, first choosing which n parts to label and then accurately localizing each part in every image. While humans are good at the latter task, their choice of parts appears to be sub-optimal from the perspective of object recognition. It appears that humans tend to choose parts that are semantically meaningful — such as those that have names — instead of the parts that are most visually distinctive. For example, three of the four humans in our experiment chose the wingtip as a part, but this is a poor feature: it appears as just a small horizontal line when the plane is viewed from the side. The weakly-supervised learning algorithm, unencumbered by prior knowledge about airplanes, chooses parts based only on their utility for object recognition and thus produces

Table 7.2: Classification performance on scale-normalized Caltech-4 images. Performance is reported by the equal ROC rate.

	Weakly-supervised			**Supervised** [14]			**Other results**
	0-fan	1-fan	2-fan	0-fan	1-fan	2-fan	[32]
Airplanes	90.5%	94.5%	95.8%	90.5%	91.3%	93.3%	90.2%
Cars (rear)	88.9%	94.6%	94.6%	—	—	—	—
Faces	86.0%	98.4%	98.4%	98.2%	98.2%	98.2%	96.4%
Motorbikes	96.7%	98.8%	98.8%	96.5%	97.0%	97.0%	92.5%

better models.

7.2.3 Scale invariant classification

We also tested the classification accuracy on the non-scale normalized version of the Caltech image set in which scale is not held constant. The results of this experiment are shown in Table 7.3. We carried out scale-invariant classification by applying the models at several different scales on each image and choosing the scale having the highest-likelihood classification. Note that the results match or outperform the scale-invariant classification performance reported with the constellation models of [32] and [33].

7.2.4 Varying the number of parts

Next we investigated how the number of parts in our k-fan models affects classification performance. This experiment was inspired by bag-of-parts models that use large numbers of features or "parts" ([18, 21, 86]). As the results in Table 7.4

Table 7.3: Classification performance on Caltech-4 with varying object scale. Performance is reported by the equal ROC rate.

	Weakly-supervised			**Other results**	
	0-fan	1-fan	2-fan	[32]	[33]
Airplanes	90.1%	93.6%	93.8%	93.0%	93.6%
Cars (rear)	88.4%	92.0%	92.1%	90.3%	84.2%
Faces	86.0%	98.4%	98.4%	96.4%	90.3%
Motorbikes	96.5%	97.6%	97.6%	93.3%	97.3%

show, both increasing the number of parts and increasing the degree of spatial constraint improve performance. Interestingly, even models consisting of a single part perform relatively well on these datasets.

7.2.5 Multi-category classification

We also conducted multi-category classification experiments to test the ability of the models to differentiate between the five different object classes and the background images. For each test image, the five object detectors were applied, and the object class with the highest likelihood was chosen. That likelihood was compared to the threshold at the equal ROC point to decide between that object class and the background class. Note that an advantage of our probabilistic approach is that the scores produced by each classifier are probabilities and hence can be compared directly without weighting parameters.

The results of the multi-class experiments are shown in Table 7.5. The performance of multi-class recognition was very similar to the single class case for 1-fans, dropping less than 1 percentage point in most cases. However, 0-fans performed

Table 7.4: Classification performance by number of model parts. Performance is reported using the equal-ROC rate.

	Airplanes			**Faces**		
	0-fan	1-fan	2-fan	0-fan	1-fan	2-fan
1 part	84.3%	—	—	76.3%	—	—
6 parts	90.1%	93.6%	93.8%	86.0%	95.6%	96.0%
15 parts	90.1%	94.8%	94.8%	86.1%	98.4%	98.4%
25 parts	90.3%	95.2%	95.2%	86.9%	98.0%	98.2%

dramatically worse on the multi-class task, suggesting that spatial constraints become more important as the difficulty of the classification task increases. This result underscores the need to study multi-class classification when evaluating the performance of an object recognition algorithm.

7.2.6 Running time

The running time of the entire weakly-supervised learning process is approximately 24 hours on a small cluster of 20 Pentium III nodes, or about one week on a single Pentium 4 node. Note that the majority of this processing time is spent performing the correlations between the training images and the tens of thousands of candidate part templates. The results of this part of the process can be cached and reused when learning models for different values of k or different numbers of parts. Once a model has been learned, the average time required to localize an object at a single scale in a 320×240 image is approximately 0.1 seconds for a 0-fan, 0.3 seconds for a 1-fan, and 2.5 seconds for a 2-fan.

Table 7.5: Confusion matrices for multi-category classification on Caltech-4. Rows correspond to actual classes, while columns correspond to predicted classes. The numbers in bold are the correct classification rates.

0-fan

		Detected class				
		Airplanes	Motorbikes	Cars	Faces	Background
Actual class	Airplanes	**70.1**%	0.5%	1.3%	12.8%	14.8%
	Motorbikes	0.0%	**65.5**%	0.0%	31.8%	2.8%
	Cars	6.8%	0.0%	**39.8**%	12.5%	41.0%
	Faces	0.5%	0.0%	0.5%	**82.5**%	16.6%
	Background	0.4%	0.0%	13.1%	17.5%	**69.0**%

1-fan

		Detected class				
		Airplanes	Motorbikes	Cars	Faces	Background
Actual class	Airplanes	**95.0**%	0.0%	0.0%	0.0%	5.0%
	Motorbikes	0.5%	**97.5**%	0.0%	0.3%	1.8%
	Cars	1.3%	0.0%	**93.3**%	0.0%	5.5%
	Faces	0.0%	0.0%	0.0%	**96.8**%	3.2%
	Background	0.6%	1.8%	1.5%	0.0%	**96.1**%

7.2.7 A caveat on the classification task

A problem with classification is that it is difficult to understand what image evidence is being used to make the classification decisions. For example, it is difficult to prevent a classification algorithm from exploiting biases in the test dataset. An alarming bias is present in the popular Caltech-4 dataset, for instance. Most of the positive images (those containing objects of interest) have larger physical dimensions than the background images. In fact, a trivial classifier whose only feature is image width achieves an equal-ROC rate of 99.4% on the airplane classification task — *without examining the content of the images at all!* It is thus possible that many classification methods tested on the Caltech-4 dataset inadvertently exploit this or other biases.[1]

Such biases become immediately apparent when the localization task is used for the experimental evaluation. The localizations themselves reveal the region of the image that the classifier used to make its decision: the localizations could not be accurate if the classifier uses evidence unrelated to the object, such as unrelated features of the background scene.

7.3 Part localization experiments

We now turn to the localization task, in which the goal is to accurately estimate the position of the objects in an image. This is a strictly more difficult problem than classification: the localization algorithm must first determine whether or not an object appears in an image before identifying the location of all of the instances. We examine two variants of the localization problem: part-level localization, in which the position of individual parts must be identified, and object-level localization, in which only a bounding box around the object class is expected. We give results for the part localization task in this section, and the object-level localization results in Section 7.4.

To conduct the part-level localization experiments we need ground truth specifying the true location of each part in every test image. This means the parts of the object model must correspond to meaningful parts of the object so that a human can annotate the true part locations. We could not use the weakly-supervised learning algorithm to produce the object models because the parts learned by that procedure are not guaranteed to correspond with meaningful object parts. Thus we

[1] We guard against this bias in our experiments by padding all images out to the same size.

learned the object models using the supervised algorithm described in Section 5.3.

We used the Caltech-4 dataset for this evaluation. The training images were annotated with the true locations of six parts for each object:

- for airplanes, the front and back landing gear, nose, wing tip, rear-most point of plane, and tail
- for faces, the left eye, right eye, nose, two corners of the mouth, and chin
- for motorbikes, the centers of the front and back wheels, headlight, tail light, and the front and back of the seat

For scoring, we examined the test images that were correctly classified as positives at the equal-ROC point in the experiments reported in Section 7.2.2. The part localizations on this subset of images were then compared to the hand-labeled ground truth. We computed the trimmed mean (at 75% and 90%) of the Euclidean distance between estimated locations and the ground truth for each part.

We observed a dramatic drop in localization error between 0-fans and 1-fans for most parts, with the average 75% trimmed mean error decreasing from 20.13 to 7.16 pixels. The 2-fan models offer a minor improvement with the localization error dropping to 7.02 pixels. Table 7.6 presents more detailed results. We see that the localization errors for 0-fan models are quite high for most parts, with the exception of visually distinctive parts such as the features of the face and front wheel of the motorbike. In fact, the facial features are so distinctive that the 0-fan part localizations are excellent, with errors less than two pixels. On the other hand, accurate localization of the less distinctive parts like the motorbike seat and airplane landing gear requires a spatial model. These results illustrate the advantage of our unified estimation approach in which weak evidence from

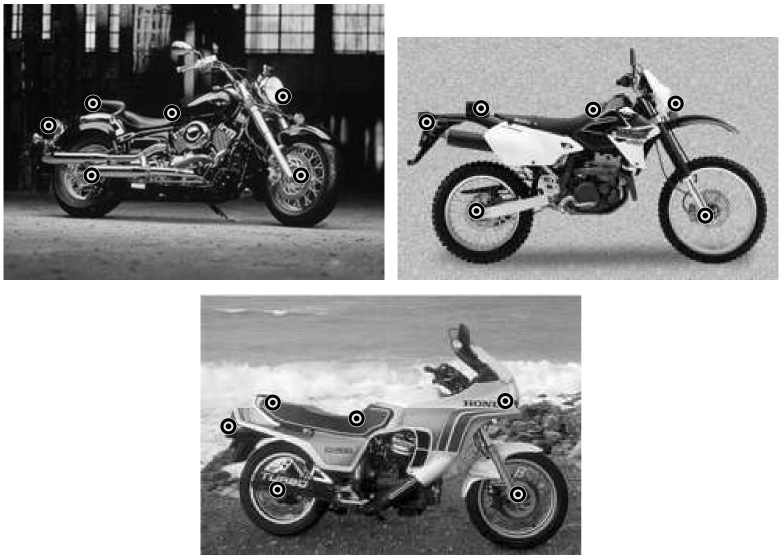


Figure 7.5: Sample part-level localization results.

multiple sources is combined to produce excellent overall results, even for parts that are not easily localized individually.

Figure 7.5 presents some sample localization results produced by our system on the motorbike dataset, showing precise localization of the parts despite substantial variability in their appearances and configurations.

7.4 Object localization experiments

In the previous section we present experiments on localization of individual object parts. Unfortunately collecting the ground truth for those experiments is expensive so it is difficult to run large-scale experiments on many object classes. Furthermore evaluating part localization accuracy is problematic when the model is learned in a

Table 7.6: Part localization results on Caltech-4. Shown are 75% and 90% trimmed means of the part localization errors, in pixels, for the correctly-classified Caltech-4 images.

Airplanes

	Nose		Front gear		Back gear	
	75%	90%	75%	90%	75%	90%
0-fan	33.3	63.3	73.3	91.8	57.5	67.9
1-fan	6.7	12.1	9.5	14.7	13.1	20.0
2-fan	6.3	12.1	8.7	14.4	12.2	20.0

	Rear		Tail		Wingtip	
	75%	90%	75%	90%	75%	90%
0-fan	34.9	55.7	19.3	42.2	63.8	79.0
1-fan	9.4	14.0	9.4	14.6	41.2	49.1
2-fan	9.4	14.0	10.2	15.2	41.6	49.9

Faces

	Left eye		Right eye		Nose	
	75%	90%	75%	90%	75%	90%
0-fan	0.38	0.52	0.54	0.70	1.11	1.31
1-fan	0.39	0.51	0.53	0.67	1.09	1.28
2-fan	0.73	0.87	0.53	0.68	1.25	1.45

	L. mouth		R. mouth		Chin	
	75%	90%	75%	90%	75%	90%
0-fan	0.93	1.24	0.98	1.21	1.52	3.12
1-fan	0.92	1.13	1.04	1.25	1.34	1.61
2-fan	0.92	1.15	1.12	1.33	1.55	1.88

Motorbikes

	Rear wheel		Front wheel		Headlight	
	75%	90%	75%	90%	75%	90%
0-fan	17.3	35.7	1.5	1.9	12.3	21.0
1-fan	1.7	2.1	1.5	1.9	8.4	13.0
2-fan	1.7	2.1	1.6	1.9	7.6	11.7

	Tail light		Seat back		Seat front	
	75%	90%	75%	90%	75%	90%
0-fan	12.7	19.8	23.5	36.0	7.4	13.8
1-fan	5.6	10.7	11.8	16.1	5.2	9.1
2-fan	5.0	9.0	11.0	14.5	4.9	8.4

weakly-supervised setting. Many applications of object recognition do not require localizations of individual parts anyway.

We therefore turn our attention to object-level localization, where the goal is to produce an accurate bounding box around each object instance. For these experiments we use the PASCAL Visual Object Classes (VOC) challenge datasets [25, 26], which include thousands of unconstrained consumer images with a wide variety of objects and scenes.

7.4.1 Experimental protocol

To facilitate comparison with other published results, we conducted our object localization experiments according to the rules of the VOC competitions. The object models were trained using the training and validation subset of the VOC data. All model parameters were either learned automatically or kept constant for all of the experiments. The partially-supervised algorithm of Section 5.2 was used to learn the models, using the object bounding boxes present in the ground truth. The ground truth also includes coarse viewpoint labels (namely right, front, left and rear) for some of the object instances. We used these labels by learning a separate model for each of these viewpoints. Thus between one and four viewpoint models were trained per object category, depending on the available ground truth. Object instances marked as “difficult” or “truncated” in the ground truth were excluded from the training procedure. Some of the models we learned are shown in Figure 7.6.

Localization was performed on all of the test images for each object class. Note that for any given object class, most of the images do not contain an instance of

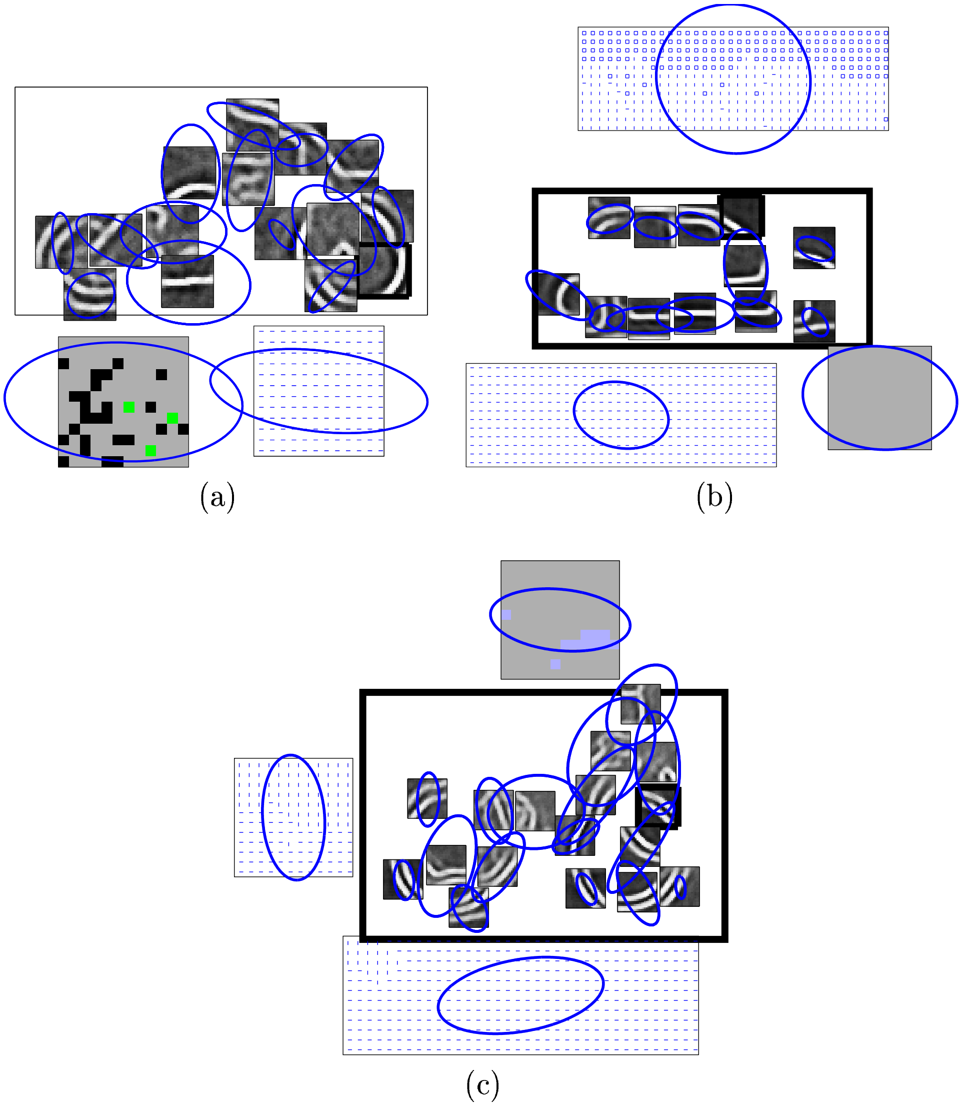


Figure 7.6: Sample scene and object models learned under partial supervision: (a) motorbike side view, (b) car side view, (c) bicycle side view. Patches are drawn at the mean configuration with ellipses showing spatial covariance (at a 2σ level set). Thick outlines designate the root patches of the scene and object models. Simple illustrations of the appearance models are also shown. For the part appearance models the probability of an edge is shown, with brighter pixels indicating higher probabilities, while for the color and surface orientation patches the mode at each pixel is shown. For surface orientation patches, horizontal dashes represent ground, vertical lines represent vertical surfaces, and boxes represent sky.

the object and therefore serve as distractor images. To count as a true positive, the size and position of a localized bounding box must be accurate according to the ground truth and must have the correct object class label. More specifically a localized bounding box B_p is considered correct if and only if

$$\exists B_{gt} \text{ such that } \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})} > 0.5,$$

where B_{gt} is a bounding box in the ground truth such that the object category labels for B_{gt} and B_p are the same. However if multiple detected bounding boxes satisfy this criterion using the same ground truth bounding box, only one of them is counted as a true detect. Thus duplicate localizations of the same object instance are counted as false positives. In accordance with the VOC rules, bounding boxes marked "difficult" were ignored during the evaluation: they did not count as true positives, false negatives, or false positives.

Localization performance is characterized according to average precision (AP), a standard metric in information retrieval [69]. Average precision is computed by taking the mean precision over a range of recall values, with precision and recall defined as,

$$\begin{aligned} \text{Precision} &= \frac{\text{number of correctly detected object instances}}{\text{number of detected object instances}} \\ \text{Recall} &= \frac{\text{number of correctly detected object instances}}{\text{number of object instances}} \end{aligned}$$

7.4.2 Results

We ran experiments for both the 2006 and 2007 PASCAL VOC challenges. The experimental protocol for the two challenges are exactly the same, so the only difference is in the datasets. The 2007 dataset is twice as large (about 10,000 versus 5,000 images) and has ground truth annotations for twice as many object categories (20 versus 10). The 2007 set is also considerably more difficult: it consists only of unconstrained consumer images, whereas the 2006 set also includes some images from the MSR dataset [17]. We present results on both the 2006 and 2007 image sets to allow comparisons with other published work that evaluate on only one or the other.

2006 Results

Table 7.7 presents the results for the 2006 set on four object categories: bicycles, buses, cars, and motorbikes. We used 1-fans as the spatial models, and the edge-based template models of Section 4.1 as the part appearance models. We ran the experiments both with and without the scene models proposed in Chapter 6. As the table shows, the scene context models improved localization performance for all four classes compared to using object models alone. These improvements are all statistically significant at a 99% confidence level according to the test of [24].

Table 7.7 also shows the best average precision obtained by any of the entries in the 2006 PASCAL VOC challenge for each object class as reported in [25]. and object models outperformed the best VOC results by a substantial margin for the bus, car, and bicycle classes. For the motorbike class our average precision was slightly lower, but the difference is probably not statistically significant. Moreover, unlike our method which performed uniformly well, none of the other methods entered in the competition performed well on all categories. For example, the algorithm that performed best on buses gave relatively poor results on bicycles and motorbikes.

Table 7.7: Object-level localization results on the 2006 VOC data, in terms of average precision.

Object class	Object model only	Scene + object model	Best VOC result [25]
Bicycle	0.421	0.498	0.440
Bus	0.172	0.185	0.169
Car	0.429	0.458	0.444
Motorbike	0.342	0.388	0.390

2007 Results

Experimental results for many of the object categories in the PASCAL VOC 2007 dataset are shown in Table 7.8. We compared the performance of three variants of our approach: 1-fan spatial models with the template-based appearance models of Section 4.1, 1-fans with the gradient-based appearance models of Section 4.2, and the multiscale models of Chapter 6 with the gradient appearance operators. For the experiments using the gradient appearance operators, the HOG parameters were set as follows: 2×2 cells per block, 8×8 pixels per cell, and 20 quantized edge orientations. For each object class we learned between one and four viewpoint-specific models.

Also shown in Figure 7.8 is a summary of the scores achieved by entrants in the 2007 VOC competition [26]. Nine research groups submitted entries to the competition; the maximum and median scores for each category are shown in the table. Also shown is the place that our method would have earned in the competition, had we entered. Note that most groups did not submit entries for all of the object categories (presumably because the results for some objects were too poor to report), so the median statistics in the table are likely overestimates. None

Table 7.8: Object-level localization results on the 2007 PASCAL VOC data, in terms of average precision. Results for three model variants are shown: "Edges" is with the part appearance operators of Section 4.1, "HOG" is with the gradient-based appearance operators of Section 4.2, "MS HOG" is with the gradient operators and the multi-scale spatial models proposed in Chapter 6. "Place" shows where the multiscale HOG-based method would have placed in the competition among the ten competitors.

Object	**Our results**			**VOC results [26]**		
category	Edges	HOG	MS HOG	Place	Median	Max
Airplane	18.1	18.5	21.1	2	15.2	26.2
Bicycle	30.3	40.0	40.3	2	26.4	40.9
Boat	—	1.2	3.0	4	2.8	9.4
Bus	—	14.6	18.9	4	19.7	39.3
Car	23.1	24.2	30.4	4	29.4	43.2
Cow	—	10.9	12.1	3	10.0	14.0
Dog	0.9	2.6	5.3	4	10.6	16.2
Horse	—	10.0	19.5	5	19.8	33.5
Motorbike	28.4	31.0	33.6	2	21.7	37.5
Television	18.6	21.8	25.5	3	24.2	28.9
Potted plant	0.3	1.9	2.5	4	4.6	12.0

of the nine research groups consistently dominated the competition; all approaches had weak or missing results for some object categories. The declared winner of the competition, for example, only entered six of the twenty object categories. For the bicycle, airplane, and motorbike classes, our results are a close second to the highest result reported in the competition. For the other classes we placed among the top four results from the competition, with the exception of "horse" in which we scored fifth.

To give some context for these average precision statistics, we present sample correct localizations in Figures 7.7, 7.8, and 7.9 for bicycles, cars, and motorbikes, respectively. We also show some incorrect localizations in the form of false positives in Figure 7.10 and false negatives in Figure 7.11.

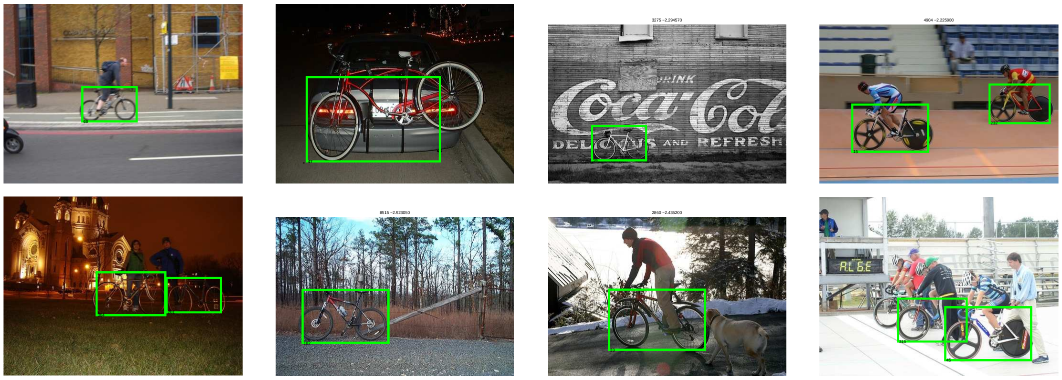


Figure 7.7: Some correct bicycle localizations. Note the false negative in the bottom right image.

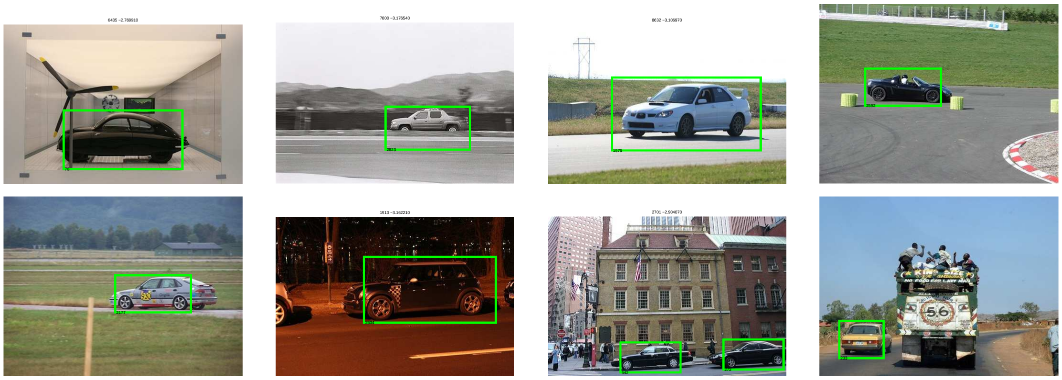


Figure 7.8: Some correct car localizations. Note the missed detections in the right three images of the bottom row caused by truncation and distant objects.

Another way of qualitatively visualizing the object detection performance is to examine the highest-likelihood object localizations for each class. Figures 7.12 through 7.17 present such visualizations for six categories: airplanes, bicycles, cars, cows, motorbikes, and televisions. Each of the figures shows the localization algorithm's top 32 most-probable object localizations for a given object category. In each of the figures, the localizations are arranged in order of decreasing likelihood,

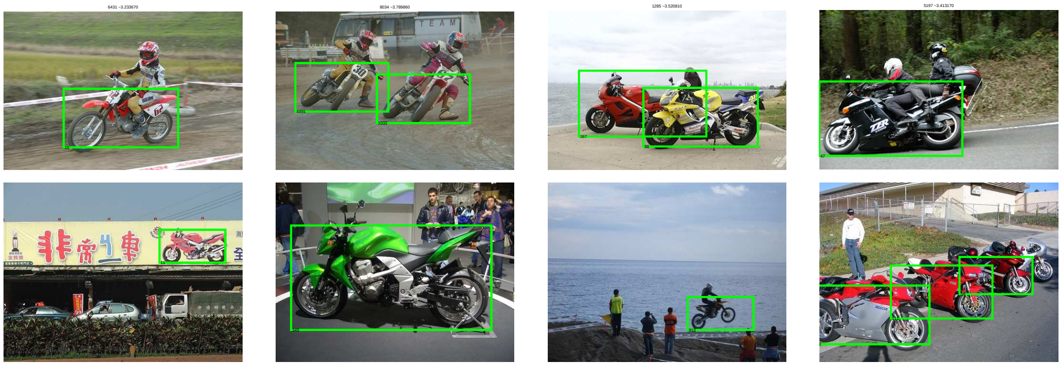
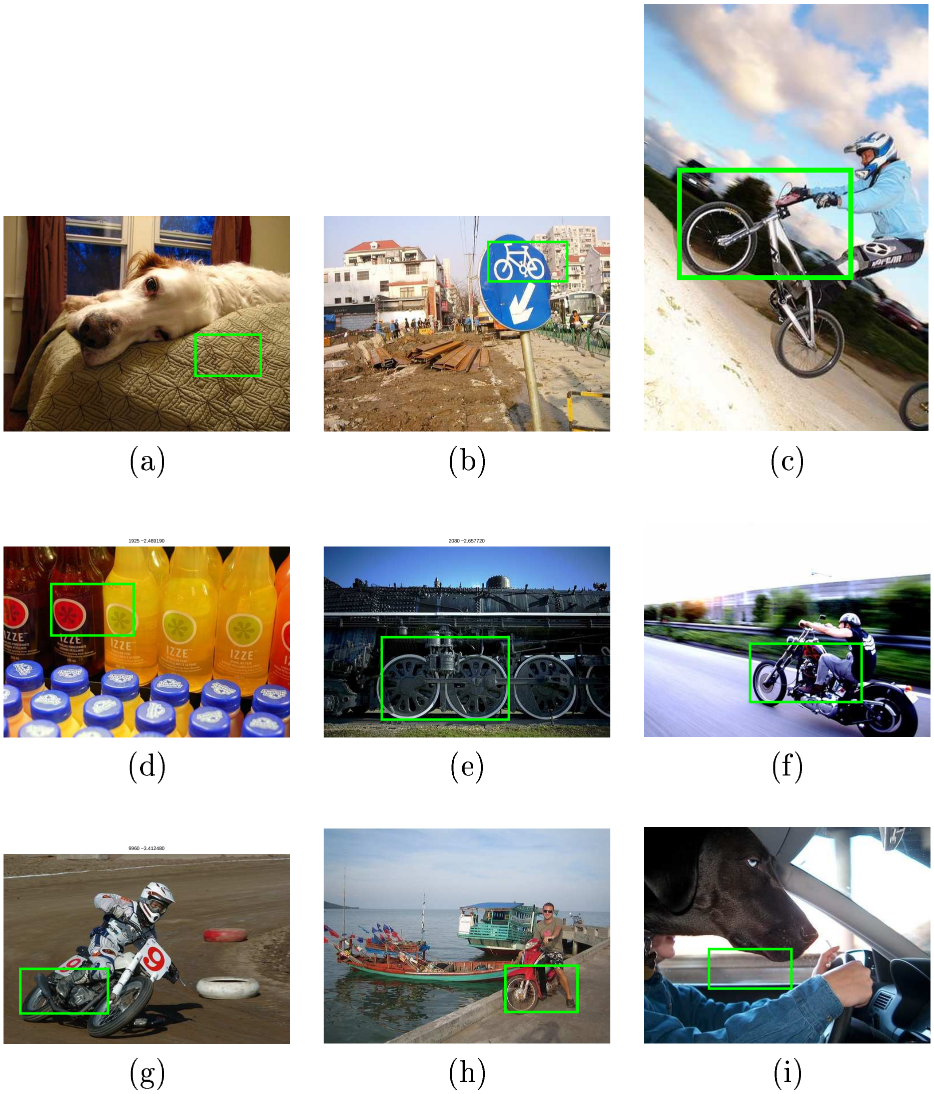


Figure 7.9: Some correct motorbike localizations. Note the false negative in the bottom right image.

with the highest-likelihood localization in the top-left, the second-highest localization in the second column of the first row, and so on. Images with localizations that are incorrect according to the VOC ground truth and scoring criterion are outlined with a red border. The images without borders are considered correct. These figures give a qualitative sense for both the precision and recall of the localization algorithm, as well as some insight into failure modes. For example, in the airplane results in Figure 7.12, six of the top ten false positives are actually poorly-localized true positives, where the bounding box produced by the localization algorithm was either too small or too large. Meanwhile many of the top bicycle false positives are actually motorbikes, and many of the false positives during motorbike localization are bicycles.

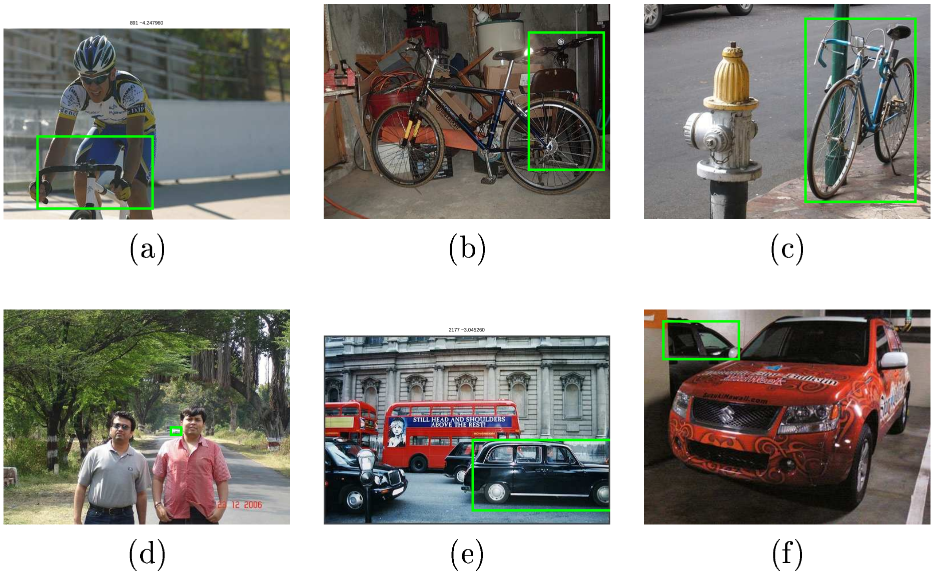


(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 7.10: Some sample false positives: bicycle false positives due to (a) textured regions, (b) drawing of a bicycle, (c) rotation, (d)-(e) confusing image features; motorbike false positives due to (f)-(g) incorrect scale estimation, (h) confusion with bicycle class; car false positive due to (i) image region coincidentally having edges similar to the contour of a car.



(a) (b) (c)

(d) (e) (f)

Figure 7.11: Some sample false negatives shown in green: missed bicycles due to (a) severe truncation, (b) frontal bike view, (c) perspective distortion; car false negatives caused by (d) very small scale, (e) unusual antique car, (f) severe occlusion.

7.4.3 Failure modes

In addition to the quantitative results reported in the last section, we have examined the results qualitatively to understand the situations that tend to cause localization errors. The error analysis was conducted for three object classes: cars, bicycles and motorbikes. For each class we chose an operating point near the midpoint of the precision-recall curve and examined the false positives and false negatives at that point by hand. For each error we identified its likely cause. Speculating on the causes is a subjective process, although for most of the failures the cause was obvious.

We found that most false positives could be explained by three broad failure

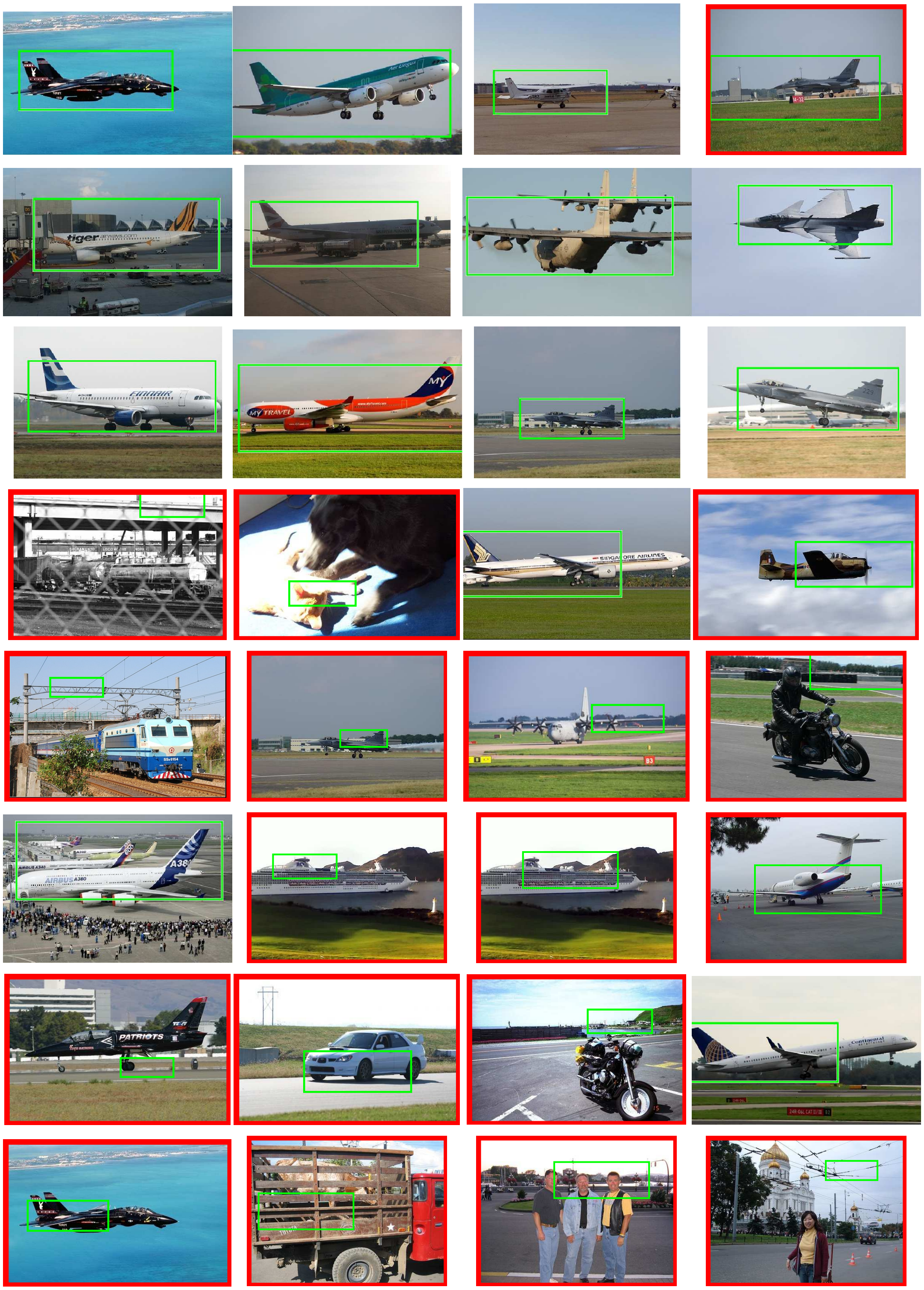


Figure 7.12: The 32 highest-probability airplane localizations.

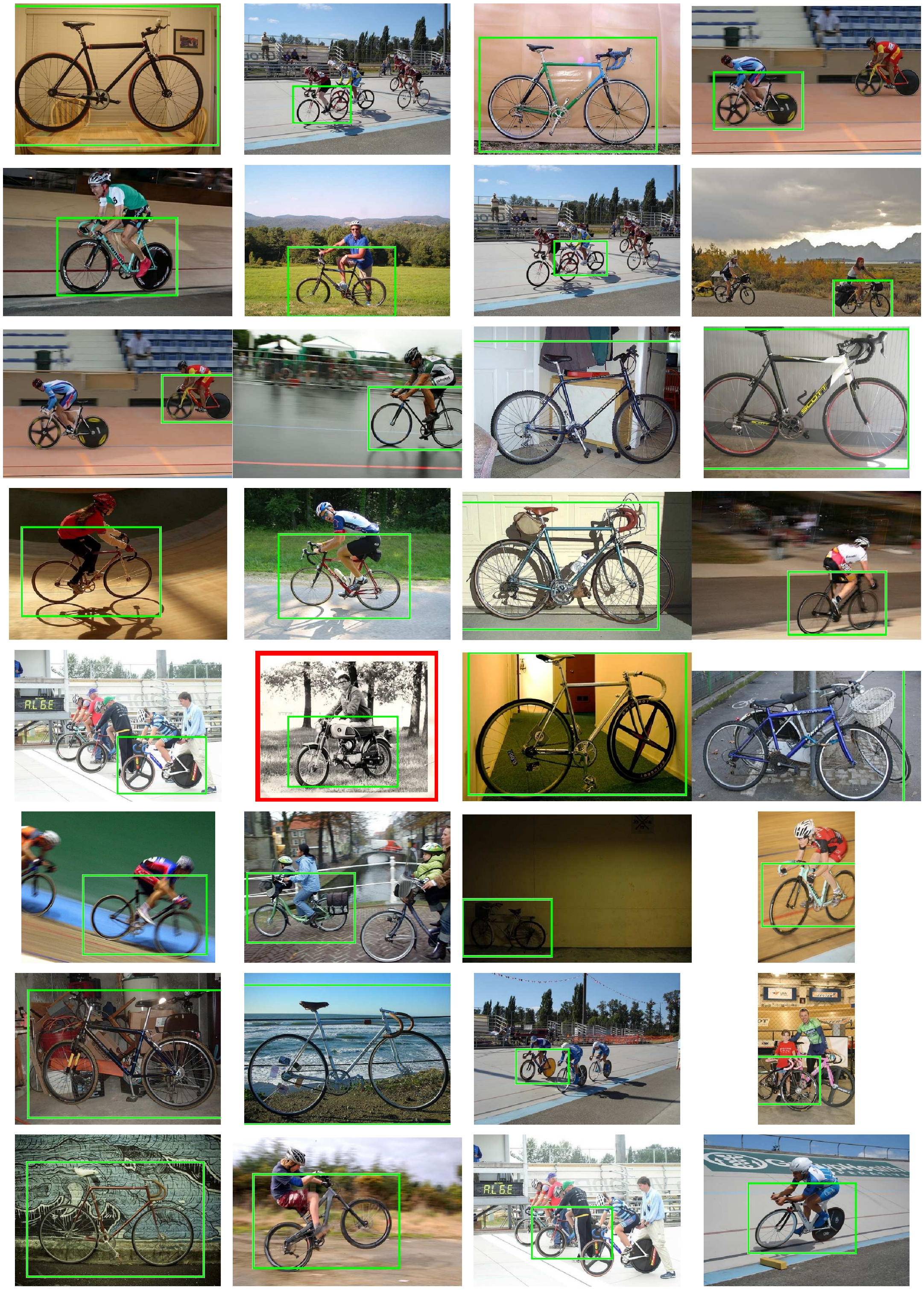


Figure 7.13: The 32 highest-probability bicycle localizations.

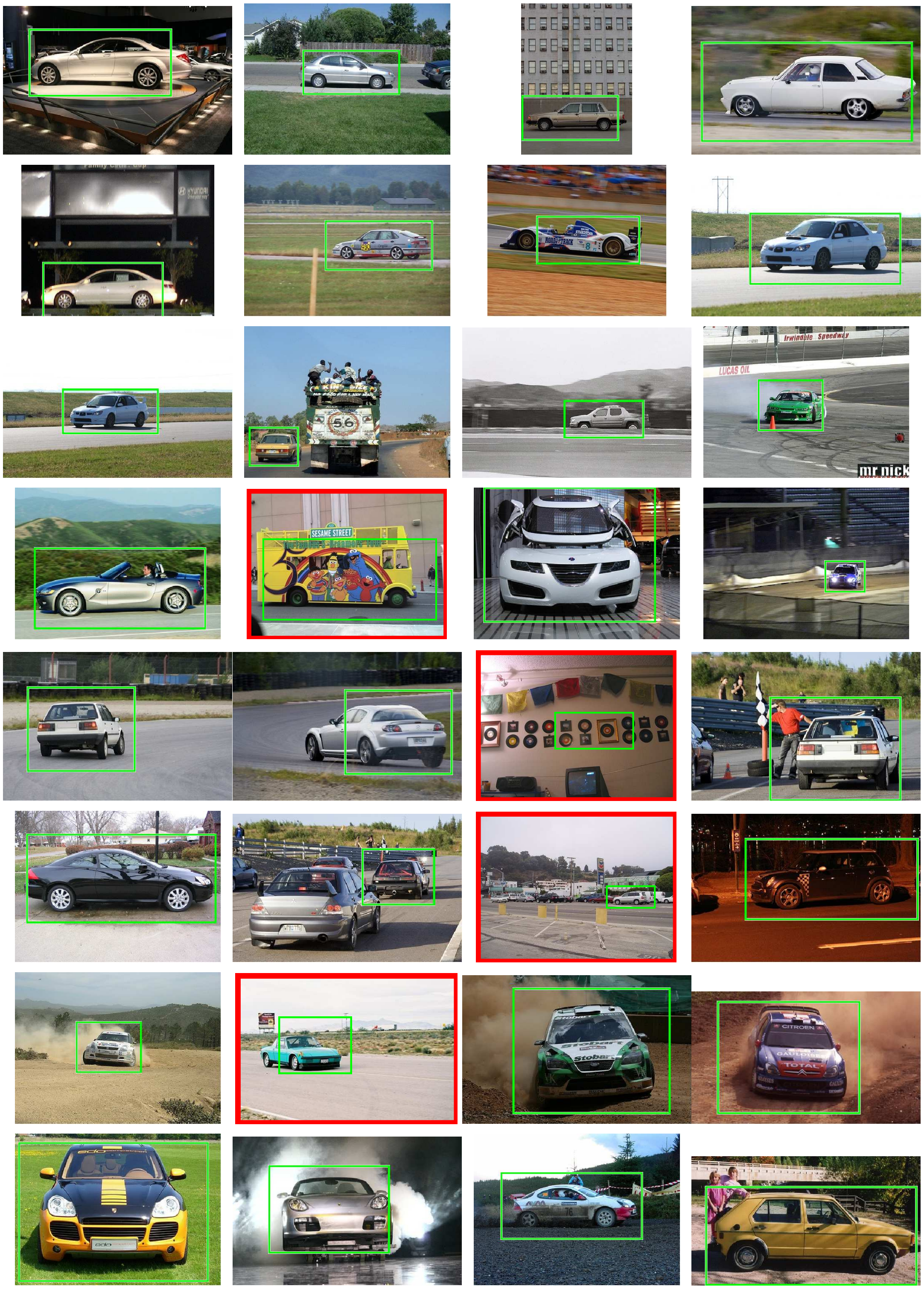


Figure 7.14: The 32 highest-probability car localizations.

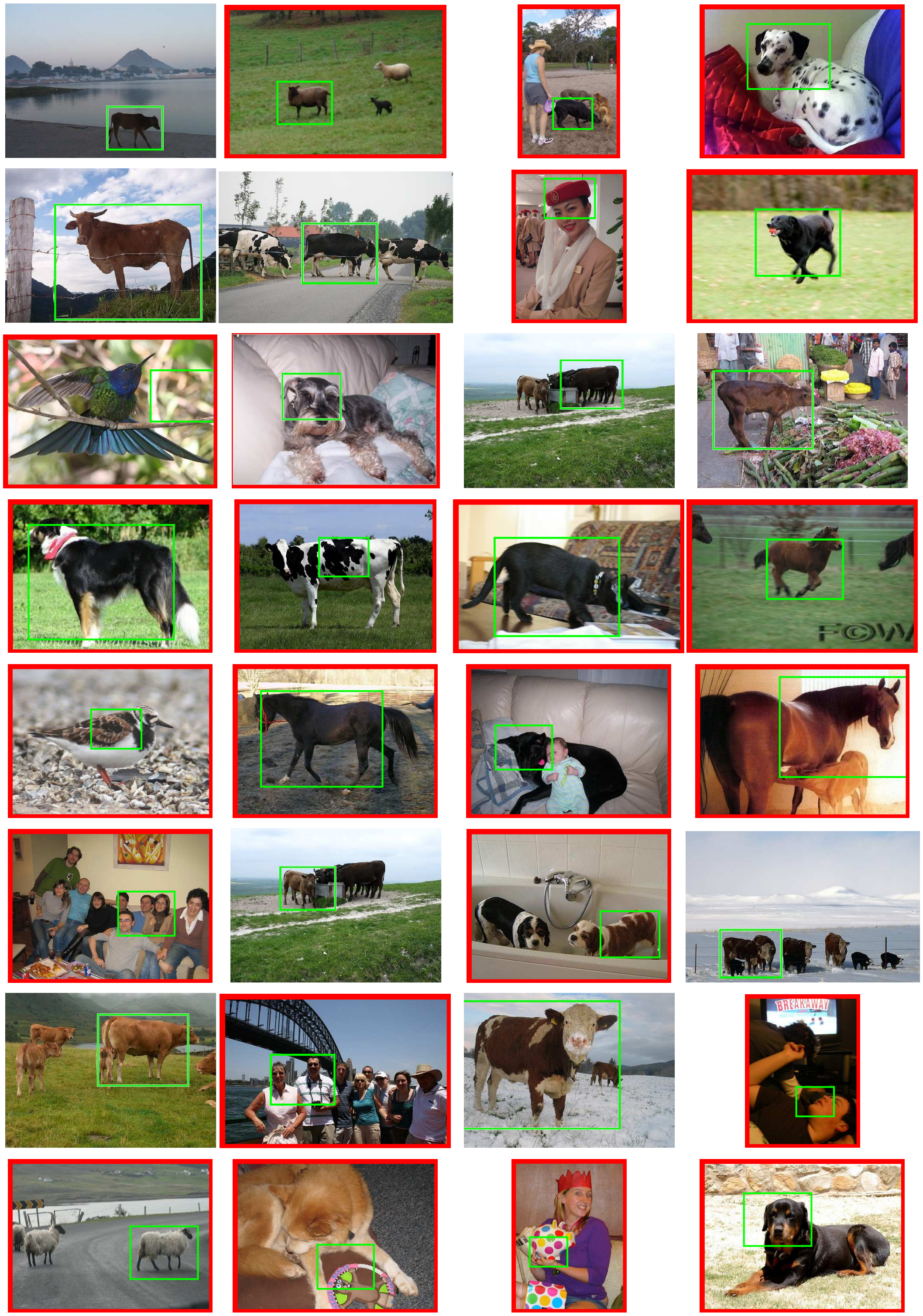


Figure 7.15: The 32 highest-probability cow localizations.

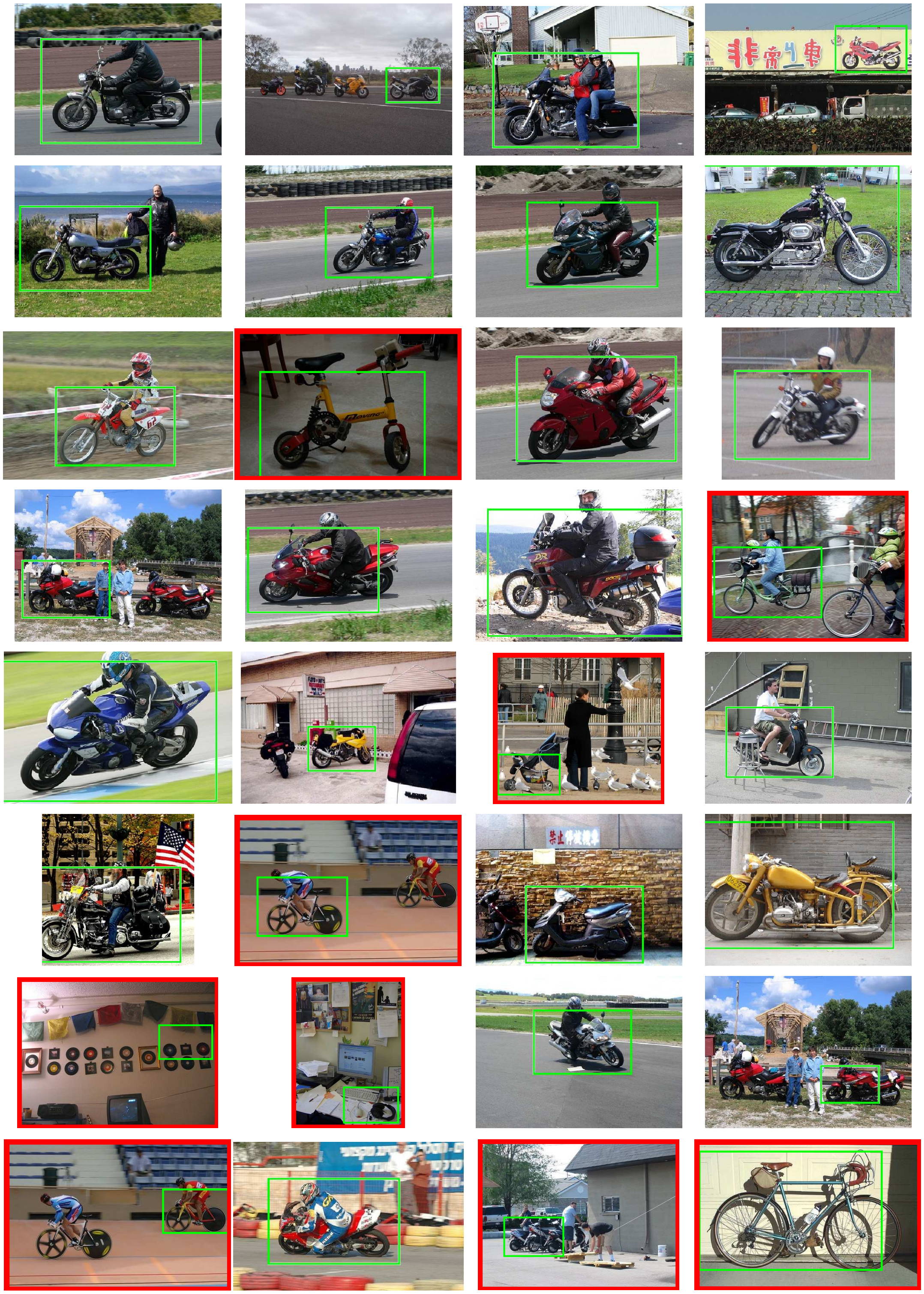


Figure 7.16: The 32 highest-probability motorbike localizations.

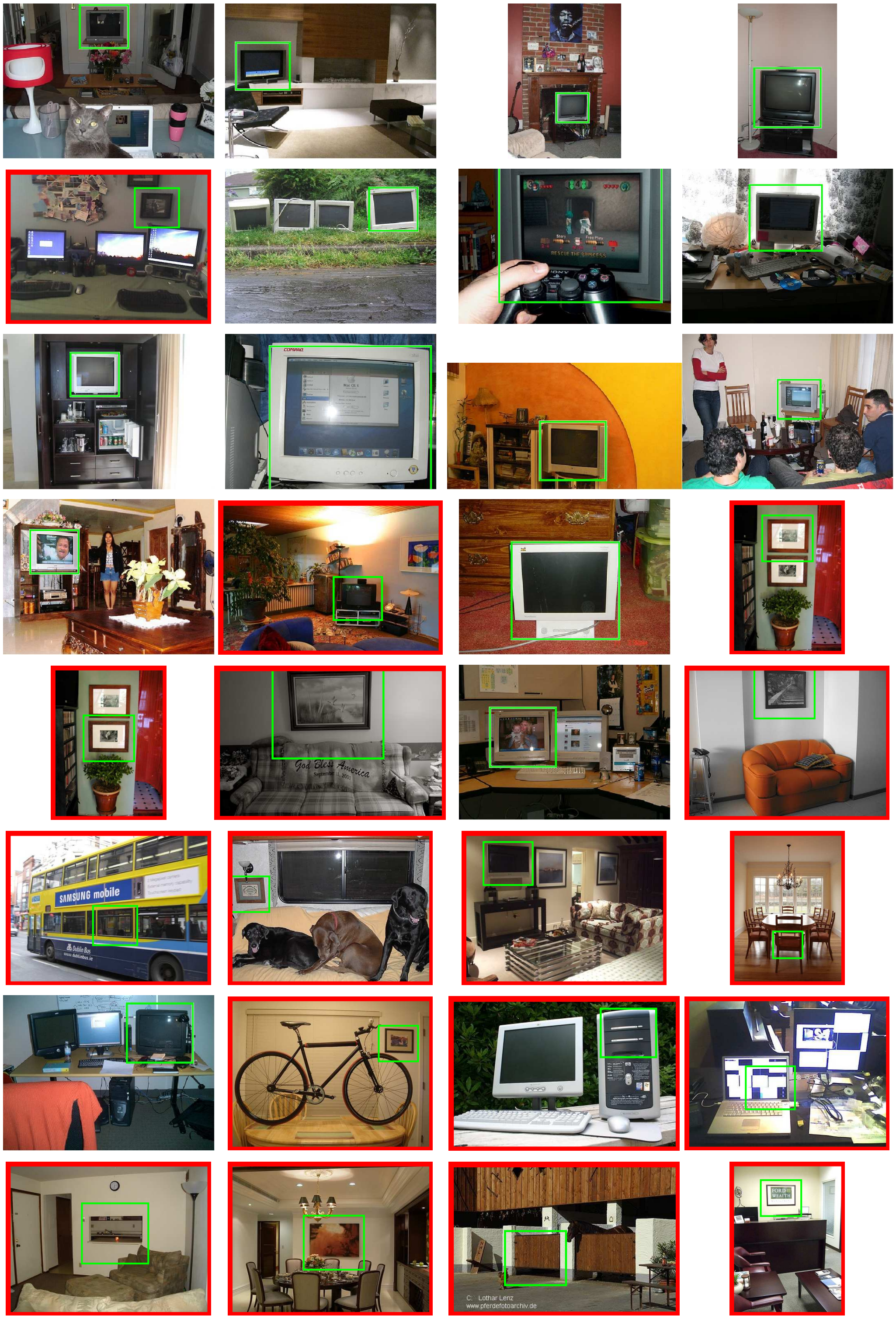


Figure 7.17: The 32 highest-probability television localizations.

modes:

1. **Confusion with other classes:** There was a significant amount of cross-confusion between the object classes. The most common errors involved the vehicular classes like bicycles, cars, buses, and motorbikes, presumably because they all share a For motorbikes, confusion with bicycles and cars accounted for over 60% of the false positives, and for bicycles, confusion with cars and motorbikes accounted for almost 50%. Most false positives from the television detector were actually framed pictures, which look nearly identical to flat-panel displays.

2. **Other confounding regions in the scene:** About 15% of false alarms occurred in densely textured regions, which can cause our appearance models to hallucinate small object parts and cause false positives. Figure 7.10(a) shows an example. Pairs of circles appearing in a scene also were a source of false positives, since they are visually similar to the wheels of a bicycle or motorbike. Figures 7.10(d) and (e) show examples of false localizations produced by pairs of circles.

3. **Correct detection but poor localization:** The detectors sometimes correctly identify an object but give a poor-quality bounding box. These poor localizations are consequently scored as false positives by the evaluation criterion. This typically occurs when the detector incorrectly estimates the object scale (as in Figure 7.10(f) and (g)) or when the object is rotated (e.g. Figure 7.10(c)).

Most false negatives occurred in one of the following situations:

1. **Occlusion and truncation**: A large fraction of the object instances in the

VOC dataset are either occluded by other objects or truncated by the image boundary. In some cases the objects are barely visible. Examples are shown in Figure 7.11(a) and (f).

2. **Small objects**: A large percentage of car and motorbike false negatives are due to small object scale. The VOC ground truth data includes bounding boxes for cars of any visible size, even those that are so small that individual parts cannot be made out. Our part-based object localization approach is unlikely to be able to detect such small objects. For example, the barely perceptible car in Figure 7.11(d) counts as a false positive.

3. **Unusual viewpoints**: Objects viewed from certain unusual viewpoints were difficult for our object detector. For example, object close-ups were detected poorly because of the geometric distortion that occurs from such a viewpoint. Also difficult were views for which models had not been trained, such as top views of cars and head-on views of bicycles. For bicycles and motorbikes, the greatest source of viewpoint-related false negatives occurred for views from the side-front or side-rear (e.g. see Figure 7.11(c)).

4. **Unusual instances of the object class**: All three detectors failed to recognize some of the more unusual instances of the object class. For example, pick-up trucks and vans were often missed by the car detector because they represent such a small fraction of the training data. Another example is the British taxicab in Figure 7.11(e).

CHAPTER 8

SUMMARY AND CONCLUSIONS

In this thesis we have studied techniques for visual object class recognition. Our approach is based on deformable part-based models, which represent an object class with a local appearance model for each individual part as well as a geometric model that captures the relative spatial constraints between parts. We used a statistical framework to represent uncertainty in a principled way, and posed recognition as a statistical inference problem.

A major contribution of the thesis is a family of probabilistic spatial models called k-fans, introduced in Chapter 2. In these models, the degree of spatial constraint is explicitly controlled by a parameter k. We showed that this family includes several approaches in the literature as special cases, including bags-of-parts, constellation models, and pictorial structures. By bringing these approaches into a common framework, we were able to reason about the assumptions that different approaches make and to directly compare the advantages and disadvantages of each approach.

In Chapter 3 we presented efficient algorithms for performing classification and localization using the k-fan models. These inference algorithms are *exact*, in that the answer they give is an actual global maximum of the posterior (in the case of localization) or the actual sum over all configurations of the model (in the case of classification). This is in contrast to almost all other part-based approaches in the literature which use approximations like bottom-up inference with feature detection. The computational cost of our inference algorithms is linear in the number of parts in the model and number of pixels in the image, but exponential in k. Thus fast, exact inference is possible for small k. In fact, we showed that the asymptotic running time for a Gaussian 1-fan is identical to that of a 0-fan, meaning that simple bag-of-parts models could include some spatial constraints at no additional computational cost.

The k-fan model and exact inference algorithms allowed us to explicitly study the trade-off between the degree of spatial constraint in the model and the computation complexity of inference. In experimental results we found a large improvement in recognition performance between 0-fans and 1-fans, but little improvement with values of k greater than 2. We also found that exact inference with 1-fan models outperformed constellation models, which use a large value of k but approximate inference. Thus our results suggest that for many object classes, using a relatively weak spatial model but an exact inference algorithm is the best trade-off position.

Most other part-based approaches begin by running a feature detector that identifies a small set of possible locations for each part. In contrast, our exact inference algorithm employs feature *operators* that give a likelihood at every location in the image. Our statistical framework is general enough to allow almost any kind of feature operator. We described two specific feature operators in Chapter 4, one based on small templates of edge features and another based on gradient histograms.

In Chapter 5 we presented algorithms for automatically learning object models from training images. Unlike most other learning approaches, ours does not use feature detection or assume a pre-defined part appearance model. In contrast, we learn the appearance and spatial components of the model simultaneously. We presented three different algorithms that differ in the amount of supervision they require: a weakly-supervised method that needs only a set of images known to contain the object of interest, a partially-supervised method that requires bounding boxes around the object instances, and a fully-supervised method that requires the location of each part in each training image. The less supervised methods require less human effort but are also more complex and take more computation to complete the learning process. Thus the family of algorithms offers a trade-off between computation cost and human effort. Surprisingly, more supervision does not necessarily lead to better models; in fact, our experimental results show that models trained by the weakly-supervised algorithm consistently outperforms those learned under full supervision.

In Chapter 6 we showed how the models and inference algorithms can be generalized to capture evidence across multiple image scales. This is useful for including contextual cues from the surrounding scene in making classification and localization decisions. We showed how to build a single model that includes evidence from both the scene and the object and how to perform exact inference on this unified model. The scene and object models were trained automatically using our partially-supervised learning algorithm.

Finally we presented extensive experimental results in Chapter 7, testing the proposed methods on both the classification and localization tasks. We used several test datasets including two very challenging collections of consumer images. We found that our approach outperformed other part-based spatial models such as constellation models on the classification task. On the localization task, we found that the hierarchical and scene models proposed in Chapter 6 improved the performance significantly. Our approach works very well for relatively rigid objects like airplanes, motorbikes, and bicycles, meeting or exceeding the performance of the best known algorithms. For less rigid objects like animals, our approach performs less well but is still among the best algorithms in the literature.

APPENDIX A

COMPUTING CONVOLUTION AND MIN-CONVOLUTION

The efficient inference algorithms presented in Chapter 3 for Gaussian k-fan models use two variants of the convolution operator. This appendix defines these operations and presents several techniques for computing them; the best technique for a given application depends on the form of the input functions and the accuracy that is desired. We first describe the regular convolution operation and explain how to compute it exactly using the Fourier transform. If one of the signals is Gaussian we show how to compute a good approximation of the convolution in linear time, first in one dimension and then generalizing to multiple dimensions. These techniques are well-known as the convolution operator is quite pervasive in computer vision and signal processing in general. We describe them here as a way of introducing analogous techniques for the lesser-known minimum-convolution operator, which we describe in Section A.2. We show that min-convolution can be viewed as a generalization of the distance transform. We describe how to compute the min-convolution efficiently when one of the functions is convex, as is the log-normal used by the localization algorithm of Section 3.2.1.

A.1 Convolution

Discrete convolution is a well-known technique that is used extensively in signal and image processing [39]. The convolution of two discrete functions f and g is,

$$(f * g)(p) = \sum_q f(q) \cdot g(p-q). \tag{A.1}$$

In practice the functions are finite in that they are non-zero over some finite portion of their domain and are zero elsewhere. Also, in many applications one of the functions has a much smaller non-zero domain than the other. This is the case for the classification algorithm of Section 3.1, in which we convolve a probability map with a relatively small Gaussian kernel. We will generally be interested in functions of two dimensions, but all of the discussion and techniques presented here generalize to arbitrary dimensionality.

The naïve implementation of convolution involves computing the sum in equation (A.1) explicitly. This takes time $O(|f| \cdot |g|)$, where $|f|$ and $|g|$ represent the number of discrete elements in the non-zero domain of f and g, respectively. This approach is reasonable when one of the functions is small but is otherwise prohibitively expensive. We now review some of the faster implementations of convolution. All of these techniques are well-known, but we review them here to serve as a basis for the discussion of min-convolution in Section A.2.

A.1.1 Frequency-based method

A faster method is to compute the convolution in the frequency domain. The convolution theorem states that the Fourier transform of the convolution of two signals is equal to the product of their Fourier transforms [39]. Thus convolution can be computed with two Fourier transforms, a multiplication, and an inverse Fourier transform,

$$f * g = \mathcal{F}^{-1}\left\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\right\},$$

where $\cdot$ denotes pointwise multiplication of complex numbers. Using the Fast Fourier Transform (FFT) algorithm [11], this computation takes asymptotic time $O(|f| \log |f| + |g| \log |g|)$. In practice this method is the best choice for large convolution kernels. For smaller kernels the naïve algorithm above may be faster because of the large hidden constants in the running times for the FFT and complex multiplications.

A.1.2 Separable kernels

Faster convolution is possible when one of the signals is *separable*, in that it can be written as a product over dimensions,

$$g(p) = \prod_{i=1}^{d} g_i(p_i),$$

where p_i denotes the i-th component of the vector p and d is the dimensionality of p. The convolution can then be computed as successive one-dimensional convolutions in each of the dimensions,

$$f * g = f * g_1 * g_2 * \ldots * g_d.$$

This takes time $O(|f| \cdot |g_i| \cdot d)$, where $|g_i|$ denotes the size of the largest dimension in the domain of g.

A.1.3 Gaussian kernel approximations

Further speed-ups in convolution are possible if at least one of the signals is a Gaussian. Without loss of generality, assume that f is an arbitrary sampled function and g is the Gaussian,

$$g(p) = \mathcal{N}(p|\mu, \Sigma).$$

This situation is of interest in our application to Gaussian k-fan inference in Chapter 3, in which one function is an arbitrary probability distribution produced by a part operator and the other is a small Gaussian kernel. If the covariance matrix Σ is diagonal, then the convolution kernel is separable,

$$g(p) = \prod_{i=1}^{d} \mathcal{N}(p_i|\mu_i, \Sigma_{i,i}), \tag{A.2}$$

and the convolution can be computed using the method presented in the last section. In fact, very good approximations of one-dimensional Gaussian convolutions can be obtained in linear time using multiple convolutions with a box filter [85]. Thus for diagonal Σ, a good approximation to convolution can be computed in total time $O(|f| \cdot d)$.

If the covariance matrix is not diagonal, the factorization over axis-oriented dimensions in equation (A.2) does not hold. One approach to overcoming this problem is to rotate the signals f and g such that the Gaussian in g is axis-oriented. In other words, we can rewrite the convolution as,

$$(f * g)(p) = f(R^T p) * \mathcal{N}(R^T p|R^T \mu, \Sigma'),$$

where

$$\Sigma' = \left(R^T \Sigma R\right),$$

and R is a rotation matrix (i.e. a real square matrix with $R^T = R^{-1}$ and $|R| = 1$). We can find R such that Σ' is a diagonal matrix using the eigendecomposition of Σ [53],

$$\Sigma = R \Sigma' R^T.$$

Such a decomposition is guaranteed to exist because as a covariance matrix, Σ is symmetric and positive definite [77].

This suggests an approach for "converting" a convolution with an arbitrary multivariate Gaussian kernel into a separable problem. We first find a rotation matrix R and diagonal matrix Σ' using the eigendecomposition of Σ. Then we rotate the coordinates of the signal f using R, run separable convolution with Σ', and then rotate the signal back. Each rotation requires $O(|f|)$ time. The total time required to compute the convolution thus takes $O(|f| \cdot d)$ time, using the box filter approximation for the 1-D Gaussian convolutions.

We have found that this method is the fastest in practice. Very efficient algorithms exist for image rotation based on several passes of simple shearing operations [60]. On most modern computers the rotations can be performed by the Graphics Processing Unit (GPU), thus taking very little CPU time [34]. Note however that some error is introduced because interpolation is necessary when rotating the discrete function f and re-sampling it on a rectangular grid. Thus this method produces an approximation to convolution with a Gaussian kernel. We have found the approximation to be very good in practice and adequate for our application. If an exact answer is required, the method based on Fourier transforms may be used instead.

A.2 Min-convolution

A variant of standard convolution is minimum-convolution, in which the sum is replaced by a minimization,

$$(f \otimes g)(p) = \min_q f(q) + g(p - q).$$

As with our discussion of convolution, we will assume that f and g are functions of arbitrary dimensionality sampled along a discrete grid, and that they are non-zero over some finite domain.

Min-convolution can be computed by brute force in $O(|f| \cdot |g|)$ time for arbitrary sampled functions f and g. Curiously, the fastest known algorithm for computing the min-convolution of two arbitrary functions is $O(\frac{|f| \cdot |g|}{\log |f| \cdot |g|})$ — much worse than the $O(|f| \log |f| + |g| \log |g|)$ bound for standard convolution [3]. The difficulty stems from the fact that min-convolution is not invertible, whereas the standard convolution is. Thus there is no known analogue to the Fourier transform-based approach from standard convolution.

However for object localization with Gaussian k-fans we are especially interested in min-convolutions in which one of the functions is the logarithm of a Gaussian, as discussed in Section 3.2. The remainder of this section describes several methods for computing the min-convolution efficiently with this restriction placed on one of the functions. We first discuss the connection between min-convolution and a related operation, the generalized distance transform. We then give an $O(n \log n)$ algorithm for computing the 1-D min-convolution between two functions where one of them is convex. Finally we consider how to extend this result to functions of arbitrary dimensionality, assuming that one of the functions is a log-Gaussian.

A.2.1 Connection with distance transform

The *distance transform* of a binary image is a well-known technique in image processing and computer vision [39]. It is used to find skeletons of digital shapes and to implement morphological operations such as dilation, for example. It is also key to efficient implementations of object recognition using Hausdorff distance [44]. For each pixel, the transform computes the distance to the nearest pixel whose binary value is one. More formally, the distance transform of a binary image I is,

$$\mathcal{D}_d\{I\}(p) = \min_{q \in I} \left(d(p,q) + 1(q) \right),$$

where $1(q)$ is an indicator function that is one if $I(q) = 1$ and is ∞ otherwise. The function $d(p,q)$ is an arbitrary distance metric, with the most common choice being Euclidean distance (L_2 norm). Fast, linear-time algorithms exist for this and other common choices of distance metric [27].

A generalization of the distance transform is possible by removing the restriction that the input is binary,

$$\mathcal{D}_d\{I\}(p) = \min_{q \in I} \left(d(p,q) + I(q) \right). \tag{A.3}$$

This has been called the *generalized distance transform* [30]. Note that the input need not be an image, but may be any function sampled on a regular grid.

There is an interesting connection between the generalized distance transform and min-convolution. In particular, for any functions f and g,

$$f \otimes g = \mathcal{D}_{g'}\{f\},$$

where

$$g'(p,q) = g(p-q).$$

A.2.2 Min-convolution in one dimension

An efficient algorithm for the one-dimensional generalized distance transform is presented in [29] for the special case that the distance metric is a convex function. Many useful distance functions are convex, including the Euclidean distance and the Mahalanobis distance. That algorithm uses the fact that the distance transform is the lower envelope of a set of convex functions. The algorithm runs in amortized $O(|f|+|g|)$ time, but the asymptotic running time analysis hides a relatively large constant.

We present a much simpler algorithm for computing the one-dimensional distance transform with a convex function. The asymptotic running time is worse than that of [29] by a logarithmic factor, but it is faster in practice for typical images because it has a smaller hidden constant. The algorithm uses a divide-and-conquer strategy that depends upon the following two theorems.

Theorem A.2.1 *For any functions $f, g : \mathbb{R} \mapsto \mathbb{R}$ with g convex, and any $p, d \in \mathbb{R}$ with $p < d$, there exists $r \in \mathbb{R}$ such that*

$$f(r) + g(p - r) = (f \otimes g)(p),$$

where

$$r \leq \arg\min_{q} f(q) + g(d - q).$$

Proof We prove the theorem by contradiction. Suppose that for some f, g, p, and d there is no such r. Then let $q^* > s$ be the smallest value such that,

$$(f \otimes g)(p) = f(q^*) + g(p - q^*),$$

where s is a minimizing parameter of $(f \otimes g)(d)$,

$$s = \arg\min_{q} f(q) + g(d - q).$$

Because s is a minimizing parameter it must be that,

$$f(s) + g(d - s) \leq f(q^*) + g(d - q^*),$$

and since q^* is the least minimizing parameter of $(f \otimes g)(p)$,

$$f(q^*) + g(p - q^*) < f(s) + g(p - s).$$

By combining these two inequalities we obtain,

$$g(p - s) + g(d - q^*) > g(d - s) + g(p - q^*). \tag{A.4}$$

From the facts that $s < q^*$ and $p < d$, we have $p - q^* \leq d - q^* \leq d - s$ and $p - q^* \leq p - s \leq d - s$. Since g is convex, by the definition of convexity we have that for any real numbers x and y and any $t \in [0, 1]$,

$$g(tx + (1 - t)y) \leq tg(x) + (1 - t)g(y),$$

and in particular,

$$\begin{aligned} g(d - q^*) &\leq \frac{q^* - s}{d - s - p + q^*} g(p - q^*) + \frac{p - d}{d - s - p + q^*} g(d - s), \\ g(p - s) &\leq \frac{d - p}{d - s - p + q^*} g(p - q^*) + \frac{q - s}{d - s - p + q^*} g(d - s). \end{aligned}$$

By combining these inequalities we can write,

$$g(p - s) + g(d - q^*) \leq g(d - s) + g(p - q^*).$$

But this inequality contradicts equation (A.4). Thus it must be that $q^* \leq s$, and the theorem is proven. ▮

Theorem A.2.2 *For any functions $f, g : \mathbb{R} \mapsto \mathbb{R}$ with g convex, and any $p, d \in \mathbb{R}$ with $p > d$, there exists $r \in \mathbb{R}$ such that*

$$f(r) + g(p - r) = (f \otimes g)(p),$$

where

$$r \geq \arg\min_q f(q) + g(d - q).$$

Proof By analogy with the above proof. ■

Taken together, these theorems mean that once the min-convolution at some point d is evaluated, the domain of f can be partitioned such that evaluating the min-convolution at points to the left of d involves only the left partition, and evaluating it at points to the right of d involves only the right partition. This suggests a recursive algorithm for computing the min-convolution at all points in the domain of f, which we now present.

```
MinConvolveSinglePoint($F, G, f_1, f_2, d$)
s ⇐ $f_1$
for $i = f_1$ to $f_2$ do
    if $F[i] + G[d - i] < F[s] + G[d - s]$ then
        $s \Leftarrow i$;
    end if
end for
return $s$
```

```
MinConvolve($F, G, D, f_1, f_2, d_1, d_2$)
if $d_2 - d_1 \geq 0$ then
    $d \Leftarrow \lfloor (d_1 + d_2)/2 \rfloor$
    $s \Leftarrow$ MinConvolveSinglePoint($F, G, f_1, f_2, d$)
    $D[d] \Leftarrow F[s] + G[d - s]$
    MinConvolve($F, G, D, f_1, s, d_1, d - 1$)
    MinConvolve($F, G, D, s, f_2, d + 1, d_2$)
end if
```

Calling MinConvolve$(F, G, D, f_1, f_2, f_1, f_2)$ computes the min-convolution of array F having domain $[f_1, f_2]$ with array G and places the result into an output array D. The worst-case running time of the algorithm is $O(|f| \log |f|)$ (where $|f| = f_2 - f_1 + 1$). To see this, note that each recursive call to MinConvolve divides the output array in half, so the maximum depth of recursive calls is $O(\log |f|)$. Moreover at any given depth in the call chain of MinConvolve, all of the calls to MinConvolveSinglePoint take total time $O(|f|)$. Thus the overall running time is $O(|f| \log |f|)$.

In practice, this algorithm is typically faster than the linear time algorithm of [29] for signals with $|f|$ on the order of a thousand or less. This is the typical scenario for the object detection work we present here, where f is a row or column of an image. To measure this more concretely, we performed an experiment in which we min-convolved random signals of different sizes with the function $g(x) = x^2$ and measured the running times on a 3.0 GHz Xeon system. We found that our recursive algorithm was 32.4% faster when $|f| = 10$, 9.9% faster when $|f| = 100$, 8.6% faster when $|f| = 1,000$, and 21.9% slower when $|f| = 10,000$.

A.2.3 Multidimensional min-convolution

As with regular convolution, min-convolution with multi-dimensional functions can be computed using successive applications of the one-dimensional operation if a separability condition is satisfied. In particular, if one of the functions can be written as a sum over dimensions,

$$g(p) = \sum_{i=1}^{d} g_i(p_i),$$

then the min-convolution can be written in terms of one-dimensional operations,

$$f \otimes g = f \otimes g_1 \otimes g_2 \otimes ... \otimes g_d.$$

This is analogous to convolution with separable filters discussed in Section A.1.2 except that the factorization is a sum over dimensions instead of a product.

Thus we have a procedure for performing the min-convolution of an arbitrary signal with another function that is separable and convex. One such function that satisfies both requirements is the squared Euclidean distance,

$$d_E(p, q) = \sum_{i=1}^{d} (p_i - q_i)^2,$$

where d is the number of dimensions in the signal. Min-convolution with the Euclidean distance is useful for applications like Hausdorff matching and performing the medial axis transform [39].

As we discuss in Section 3.2, for localization with Gaussian k-fans we are interested in the squared Mahalanobis distance,

$$d_M(p, q) = (p - q)^T \Sigma^{-1} (p - q). \tag{A.5}$$

This distance function can be written as a sum over dimensions if and only if Σ is diagonal.[1] Moreover the Mahalanobis distance is convex if Σ is positive definite.[2] The restriction on positive definiteness is not a problem in our application because Gaussian covariance matrices are positive definite by definition. The restriction that Σ is diagonal is more problematic because in general we will be interested in arbitrary covariance matrices. One approach to overcoming this problem is to rotate Σ such that is diagonal, as we did in Section A.1.3 for Gaussian convolution. That is, equation (A.5) can be rewritten,

$$d_M(p, q) = \left(R^T (p - q)\right)^T \Sigma'^{-1} \left(R^T (p - q)\right),$$

where

$$\Sigma' = \left(R^T \Sigma R\right),$$

and R is a rotation matrix such that Σ' is diagonal.

[1] Note that the Euclidean distance is a special case of the Mahalanobis distance in which Σ is the identity matrix.

[2] Here is a sketch of the proof of this fact. A continuous, twice-differentiable function is convex if and only if its Hessian is positive definite. It is easy to show that the Hessian H of the Mahalanobis distance between any two functions is equal to $2\Sigma^{-1}$. Thus if Σ is positive definite, then so is $2\Sigma^{-1}$, as the inverse of a positive definite matrix is also positive definite.

Thus to perform min-convolution of a signal f with an arbitrary Mahalanobis distance, we rotate the coordinate system of the signal according to R, run the one-dimensional distance transform algorithm of Section A.2.2 along each dimension using Σ', and then perform the inverse rotation on the result. The total time required to compute the d-dimensional min-convolution with the Mahalanobis distance is $O(d|f|\log|f|)$ using the algorithm in A.2.2 and $O(d|f|)$ using the algorithm in [29]. The result is exact if Σ is diagonal; otherwise it is an approximation as some error is introduced due to interpolation during rotation.

BIBLIOGRAPHY

[1] Saad Ali and Mubarak Shah. A supervised learning framework for generic object detection in images. In *IEEE International Conference on Computer Vision*, pages 1347–1354, Washington, DC, USA, 2005.

[2] Yali Amit and Alain Trouvé. Pop: Patchwork of parts models for object recognition. *International Journal of Computer Vision*, 75(2):267–282, November 2007.

[3] László Babai and Pedro Felzenszwalb. Computing rank convolutions with a mask. Technical report, University of Chicago, 2006.

[4] Irving Biederman. Perceiving real-world scenes. *Science*, 177(4043):77–80, 1972.

[5] Michael C. Burl and Pietro Perona. Recognition of planar object classes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1996.

[6] Michael C. Burl, Markus Weber, and Pietro Perona. A probabilistic approach to object recognition using local photometry and global geometry. In *European Conference on Computer Vision*, 1998.

[7] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), 1986.

[8] Stefan Carlsson. Geometric structure and view invariant recognition. *Philosphical Transactions: Mathematical, Physical and Engineering Sciences*, 356(1740), 1998.

[9] Ondřej Chum and Andrew Zisserman. An exemplar model for learning object classes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[10] Peter Clifford. Markov Random Fields in statistics. In Geoffrey R. Grimmett and Dominic J. A. Welsh, editors, *Disorder in physical systems*, pages 19–32, Oxford, 1990. Clarendon Press.

[11] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms (second edition)*. MIT Press and McGraw-Hill, 2001.

[12] Robert G. Cowell, A. Philip Dawid, Steffen L. Lauritzen, and David J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 1999.

[13] David J. Crandall, Pedro Felzenszwalb, and Daniel P. Huttenlocher. Object recognition by combining appearance and geometry. In *Toward Category-Level Object Recognition*. Springer, 2007.

[14] David J. Crandall, Pedro F. Felzenszwalb, and Daniel P. Huttenlocher. Spatial priors for part-based recognition using statistical models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 10–17, 2005.

[15] David J. Crandall and Daniel P. Huttenlocher. Weakly supervised learning of part-based spatial models for visual object recognition. In *European Conference on Computer Vision*, 2006.

[16] David J. Crandall and Daniel P. Huttenlocher. Composite models of objects and scenes for category recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[17] Antonio Criminisi. Microsoft Research Cambridge object recognition image database version 1.0. 2004.

[18] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *European Conference on Computer Vision Workshop on Statistical Learning in Computer Vision*, 2004.

[19] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893, Washington, DC, USA, 2005. IEEE Computer Society.

[20] Arthur Dempster, Nan Laird, and Donald Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

[21] Gyuri Dorko and Cordelia Schmid. Object class recognition using discriminative local features. Technical report, INRIA Grenoble, September 2005.

[22] Bruce A. Draper, Jose Bins, and Kyungim Baek. ADORE: adaptive object recognition. In *Computer Vision Systems*, pages 522 – 537, 1999.

[23] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, 2000.

[24] DeLong E.R., DeLong D.M., and Clarke-Pearson D.L. Comparing the areas under two or more correlated roc curves: a non-parametric approach. *Biometrics*, 44(3), 1998.

[25] Mark Everingham, Andrew Zisserman, Christopher K. I. Williams, and Luc Van Gool. The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results. http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf.

[26] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[27] Ricardo Fabbri, Luciano da F. Costa, Julio C. Torelli, and Odemir M. Bruno. 2D Euclidean distance transform algorithms: A comparative survey. *ACM Computing Surveys*, 40(1):2:1–2:44, 2007.

[28] 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 Conference on Computer Vision and Pattern Recognition Workshop on Generative Model-based Vision*, 2004.

[29] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Distance transforms of sampled functions. Technical Report TR2004-1963, Cornell Computing and Information Science, September 2004.

[30] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1), 2005.

[31] Pedro F. Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[32] Rob Fergus, Pietro Perona, and Andrew Zisserman. Object class recognition by unsupervised scale-invariant learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[33] Rob Fergus, Pietro Perona, and Andrew Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 380–387, 2005.

[34] Randima Fernando. *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[35] Vittorio Ferrari, Tinne Tuytelaars, and Luc Van Gool. Integrating multiple model views for object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[36] Martin A. Fischler and Robert A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computer*, 22(1), 1973.

[37] David Gerónimo, Antonio López, and Angel D. Sappa. Computer vision approaches to pedestrian detection: Visible spectrum survey. In *Recognition and Image Analysis*, pages 547–554. Springer, 2007.

[38] Joseph Gil and Michael Werman. Computing 2-d min, median, and max filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):504–507, May 1993.

[39] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[40] John Gribbin. *In search of the Big Bang: The life and death of the universe.* Penguin, 1999.

[41] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Geometric context from a single image. In *IEEE International Conference on Computer Vision*, pages 654–661, 2005.

[42] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Putting objects in perspective. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2137–2144, 2006.

[43] Rui Huang, Vladimir Pavlovic, and Dimitris N. Metaxas. A graphical model framework for coupling MRFs and deformable models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages II: 739–746, 2004.

[44] Daniel P. Huttenlocher, Gregory A. Klanderman, and William J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, September 1993.

[45] Daniel P. Huttenlocher and Shimon Ullman. Recognizing solid objects by alignment with an image. *International Journal of Computer Vision*, 5(2):195–212, November 1990.

[46] Harold Jeffreys. *The Theory of Probability.* Oxford University Press, third edition, 1961.

[47] Thorstem Joachims. Making large-scale SVM learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in kernel methods – support vector learning.* MIT Press, 1999.

[48] Timor Kadir and Michael Brady. Scale, saliency and image description. *International Journal of Computer Vision*, 45(2):83–105, 2001.

[49] Xiangyang Lan and Daniel P. Huttenlocher. A unified spatio-temporal articulated model for tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[50] Xiangyang Lan, Stefan Roth, Daniel P. Huttenlocher, and Michael Black. Efficient belief propagation with learned higher-order Markov Random Fields. In *European Conference on Computer Vision*, 2006.

[51] Steffen Lauritzen. *Graphical Models.* Oxford, 1996.

[52] Marius Leordeanu and Robert Collins. Unsupervised learning of object models from video sequences. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1142 – 1149, June 2005.

[53] Charles F. Van Loan. *Introduction to Scientific Computing.* Prentice-Hall, 2000.

[54] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[55] Jiebo Luo and David J. Crandall. Robust color object detection using spatial-color joint probability functions. *IEEE Transactions on Image Processing*, 15(6):1443–1453, 2006.

[56] Pierre Moreels and Pietro Perona. Evaluation of features detectors and descriptors based on 3D objects. In *IEEE International Conference on Computer Vision*, volume 1, pages 800–807, 2005.

[57] Kevin P. Murphy, Antonio B. Torralba, and William T. Freeman. Graphical model for recognizing scenes and objects. In *Proceedings of Neural Information Processing Systems*, 2003.

[58] Andreas Opelt, Michael Fussenegger, Axel Pinz, and Peter Auer. Weak hypotheses and boosting for generic object detection and recognition. In *European Conference on Computer Vision*, pages 71–84, 2004.

[59] Andreas Opelt, Axel Pinz, Michael Fussenegger, and Peter Auer. Generic object recognition with boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):416–431, 2006.

[60] Alan Paeth. A fast algorithm for general raster rotation. In *Graphics Interface*, pages 77–81, 1986.

[61] Karre B. Peterson and Michael S. Peterson. The matrix cookbook. September 2007.

[62] Jean Ponce, Tamara L. Berg, Michael Everingham, David Forsyth, Martial Hebert, Svetlana Lazebnik, Marcin Marszałek, Cordelia Schmid, Bryan C. Russell, Antonio Torralba, Christopher Williams, Jianguo Zhang, and Andrew Zisserman. Dataset issues in object recognition. In *Towards Category-Level Object Recognition*, pages 29–48. Springer, 2006.

[63] Jean Ponce, Svetlana Lazebnik, Fred Rothganger, and Cordelia Schmid. Towards true 3D object recognition. In *Reconnaissance des Formes et Intelligence Artificielle*, 2004.

[64] Marc'Aurelio Ranzato, Fu Jie Huang, Y-Lan Boureau, and Yann LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 411–426, 2007.

[65] Donald J. Rose. On simple characterizations of k-trees. *Discrete Mathematics*, 7(3-4):317–322, 1974.

[66] William Rucklidge. *Efficient Visual Recognition Using the Hausdorff Distance.* Springer-Verlag, 1996. Lecture Notes in Computer Science volume 1173.

[67] Bryan C. Russell, Alexei A. Efros, Josef Sivic, William T. Freeman, and Andrew Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[68] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, NJ, 1995.

[69] Gerard Salton. *Introduction to Modern Information Retrieval (McGraw-Hill Computer Science Series).* McGraw-Hill Companies, September 1983.

[70] Silvio Savarese and Li Fei-Fei. 3D generic object categorization, localization and pose estimation. In *IEEE International Conference on Computer Vision*, 2007.

[71] Andrea Selinger and Randal C. Nelson. A perceptual grouping hierarchy for appearance-based 3D object recognition. *Computer Vision and Image*

Understanding, 76(1):83–92, October 1999.

[72] Thomas Serre, Lior Wolf, Stanley Bileschi, and Maximilian Riesenhuber. Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):411–426, 2007.

[73] Thomas Serre, Lior Wolf, and Tomaso Poggio. A new biologically motivated framework for robust object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[74] Raymond A. Serway. *Physics for Scientists and Engineers.* Harcourt, 5th edition, 1999.

[75] Jamie Shotton, John M. Winn, Carsten Rother, and Antonio Criminisi. *TextonBoost*: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conference on Computer Vision*, pages 1–15, 2006.

[76] Josef Sivic, Bryan Russell, Alexei A. Efros, Andrew Zisserman, and William T. Freeman. Discovering objects and their location in images. In *IEEE International Conference on Computer Vision*, 2005.

[77] Gilbert Strang. *Introduction to Linear Algebra.* Wellesley Cambridge, 3rd edition, 2003.

[78] Erik B. Sudderth, Antonio Torralba, William T. Freeman, and Alan S. Willsky. Learning hierarchial models of scenes, objects, and parts. In *IEEE International Conference on Computer Vision*, 2005.

[79] Jian Sun, Nan-Ning Zheng, and Heung-Yeung Shum. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7), 2003.

[80] Zehang Sun, George Bebis, and Ronald Miller. On-road vehicle detection using optical sensors: a review. In *IEEE Conference on Intelligent Transportation Systems*, pages 585–590, 2004.

[81] Carlo Tomasi and Takeo Kanade. Detection and tracking of feature points. Technical Report CMU-CS-91-132, Carnegie Mellon University, 1992.

[82] Antonio B. Torralba, Kevin P. Murphy, William T. Freeman, and Mark A. Rubin. Context-based vision system for place and object recognition. In *IEEE International Conference on Computer Vision*, pages 273–280, 2003.

[83] Douglas L. Vail, Manuela M. Veloso, and John D. Lafferty. Conditional random fields for activity recognition. In *Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems*, 2007.

[84] Yair Weiss and William T. Freeman. What makes a good model of natural images? In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[85] Williams M. Wells, III. Efficient synthesis of Gaussian filters by cascaded uniform filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2), 1986.

[86] John Winn, Antonio Criminisi, and Thomas Minka. Object categorization by learned universal visual dictionary. In *IEEE International Conference on Computer Vision*, 2005.

[87] Kevin Woods, Diane Cook, Lawrence Hall, Kevin W. Bowyer, and Louise Stark. Learning membership functions in a function-based object recognition system. *Journal of Artificial Intelligence Research*, 3:187–222, 1995.

[88] Hanhong Xue and Venu Govindaraju. Hidden Markov Models combining discrete symbols and continuous attributes in handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):458–462, 2006.

[89] Wenyi Zhao, Rama Chellappa, P. Jonathon Phillips, and Azriel Rosenfeld. Face recognition: A literature survey. *ACM Compututing Surveys*, 35(4):399–458, 2003.

[90] Günter M. Ziegler. *Lectures on polytopes.* Springer-Verlag, New York, 1995.