Object Recognition with Many Local Features

by

Scott Helmer

B.Sc. University of Toronto, 2002

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

We accept this thesis as conforming
to the required standard

The University of British Columbia

September 2004

Library Authorization

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Scott Helmer
Name of Author *(please print)*

08/10/2004
Date (dd/mm/yyyy)

Title of Thesis: Object Recognition with Many Local Features

Degree: Master of Science Year: 2004

Department of Computer Science
The University of British Columbia
Vancouver, BC Canada

Abstract

There has been a great deal of attention focused on part-based approaches to object classification in recent research in computer vision, and some approaches have achieved a surprising amount of success. However, learning models with a large number of parts has been a particular challenge. One of the most successful approaches is that of Fergus *et al.* [5] who have developed a generative model for recognition that achieves excellent results on a variety of datasets. The learning method that they present to learn the parameters for the model, however, requires an exponential amount of time to train as the number of parts increase. The primary contribution of this thesis is the extension of their generative model, and the development of a learning algorithm that can learn a large number of parts in a reasonable amount of time. In particular, we have developed an incremental learning algorithm where the model is initialized intelligently with a small number of parts, and parts are added to the model one at a time. By taking such an approach we are able to learn models with a large number of parts in nearly a linear amount of time in the number of parts. The approach is validated on a number of datasets, including cars, motorbikes, and faces, and demonstrates excellent recognition results along with large models learned in a reasonable amount of time.

Contents

List of Figures

List of Tables

Acknowledgements

First and foremost I'd like to thank my supervisor Prof. David Lowe for his patience, kindness, and for his many insights that allowed me to complete this thesis. Thanks also goes out to my co-supervisor Prof. Nando de Freitas for guidance and for teaching me that the math and the ideas behind the math go hand in hand.

Many thanks also goes out to the Orphanage, where my generous house mates fed me when I was working frantically, and also provided me with much needed cheer. Long live the Lord of Catan, who ever he or she may be.

I'd also like to thank my family, who have provided me with support through this whole process.

SCOTT HELMER

The University of British Columbia
October 2004

Chapter 1

Introduction

Of all the tasks that humans perform with intelligence, visual object recognition seems to be one of the most effortless, done mostly without conscious thought. Yet to solve this problem consciously is far from effortless. The primary source of the difficulty that underlies solving the problem of object recognition is the notion of similarity. To see this, note that an object class can be defined by its functionality, such as chairs, or by its shape, such as cylinders, or by its intrinsic nature, such as mammals. As a result, the possible features that are relevant in object recognition are many, and vary in importance for each class. In many cases, even defining an object class explicitly in terms of necessary and sufficient characteristics is nearly impossible. Even more important is that many of the primary characteristics of an object class are not manifested visually, at least not directly. For example, an image of a dog is also an image of a living thing, yet the class of living things displays no regular visual characteristics. However, if we first judged the image as containing a dog, we could also infer that it is also contains a living thing. As a result, solving the object recognition problem requires significant machinery that is unrelated to visual phenomena.

A great deal of success has been achieved in recognizing specific objects [8], such as a particular teddy bear, rather than an object class, such as teddy bears. The reason for this is that recognizing an object class is arguably a more difficult task in computer vision, and this is because there is variation in the appearance and shape across an object class, whereas this is not the case for single objects. Moreover, some success has been achieved in recognizing an object class visually with specific object classes. Examples of such success are face recognition [18] and digit recognition [10]. However, the approaches developed for these object classes generally don't extend to general object classes because they make restrictive assumptions, require vast amounts of training time, or utilize visual features that are only useful in classification for a few object classes. Applications such as image labelling on the Internet and robot navigation require a vision system that can recognize a wide variety of object classes, while remaining computationally tractable. The concern of this thesis is the

development of an approach that can recognize general object classes, such as cars, faces, and motorbikes, and generally from canonical views.

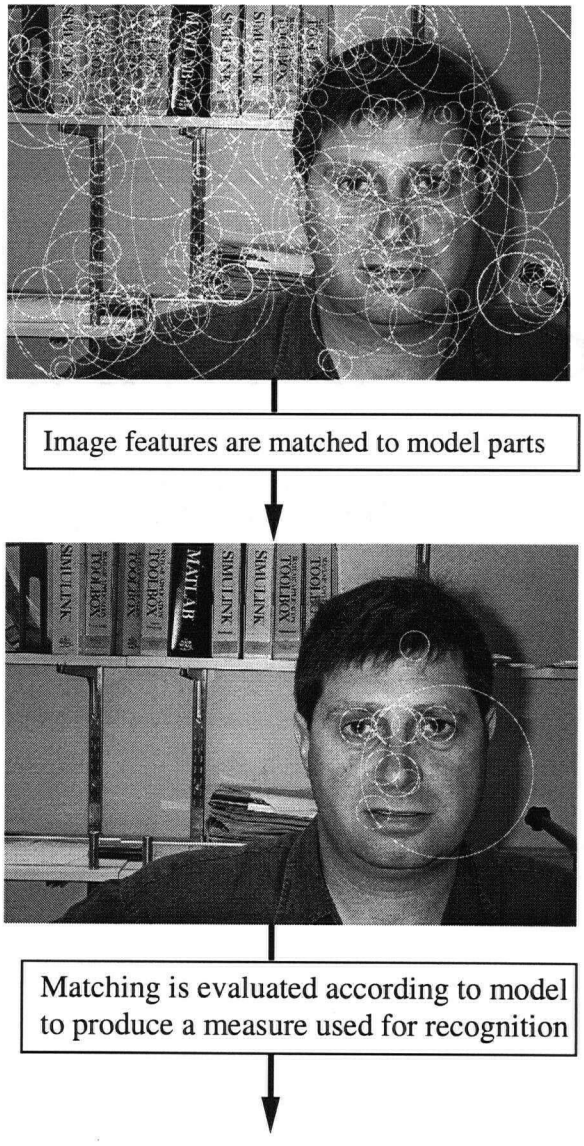


Figure 1.1: An example of finding an optimal matching from features in the image to parts in the model. There can be more that one matching that contributes to the final decision as to whether the image contains an instance of the object class.

The primary job of a vision system designed to recognize an object class from a 2D image is to determine what visual characteristics distinguish the object class from others. Such visual characteristics could be the colour, shape, or texture of the object

or parts of the object. However, the appearance of an image containing the object is dependent upon the what angle the object is viewed at, the distance the object is from the camera, its location and illumination, and other occluding objects. Yet whether an object is in a particular class is independent of all of these variables. Any successful vision system designed to recognize objects from 2D images must also be at least partially invariant to these variables. As a result we have taken a part-based approach, where we model the object as a collection of parts. By doing this, if the object is occluded, the occlusion only affects the occluded parts, and the object can still be recognized based on the parts that are not occluded. Moreover, translation and scale invariance become computationally tractable, as outlined later.

A general part-based approach to object recognition extracts a set of features from an image, determines a matching of these features to parts in the model, sometimes multiple matchings, and produces a score that indicates how likely it is that the image contains an instance of the object class. See Figure 1.1 for an example. Our approach in particular evaluates a matching based both upon the spatial arrangement of the parts, the scale of these parts, and the appearance. A motivation for taking a part-based approach is that many object classes have characteristics that manifest themselves as distinct visual parts. A face for example is composed of eyes, a nose, and a mouth, and each of these are local in the sense that their appearance is captured in a region that is smaller than the whole object. Thus, modelling an object class by its parts can decompose the problem into recognizing pieces of the object, each of which will contain less variability than the variability of object as a whole, which makes it significantly easier to model.

The approach presented in the thesis is an extension of the constellation model presented in Fergus *et al.*[5], with a number of novel extensions and explorations. The object class is modelled using a generative model, with distributions over the appearance, location and scale of parts. The measure of how likely an image is to contain an object is the sum of the likelihood of seeing the features of the image over all possible matchings of features to parts of the model. The difficulty with this model is that the number of matchings is exponential in the number of parts of the model, which makes evaluation and learning difficult. The Fergus *et al.*approach learns the parameters for the probability model simply by using Expectation Maximization (EM). In EM it is required that the probability is evaluated for all non-zero matchings, and with a random initialization almost all possible matchings must therefore be evaluated. By doing this they are restricted to models with less than 7 parts for reasons of efficiency.

The primary contribution of this thesis is the development of a tractable approach

to learning models with more than a few parts. This is done by intelligently initializing a model with a small number of parts, and then adding parts incrementally to the model and updating parameters using EM after each addition. One of the primary advantages of such an approach is that not all possible matchings need to be reevaluated when adding a new part. From previous iterations we know the non-zero matchings for the previous parts, so the search space is greatly reduced.

Another contribution is a new approach to scale and translation invariance that results in more a accurate geometric model on parts, which accelerates learning and improves recognition results. To achieve the scale and translation invariance, the location and scale of image features are first transformed into a model coordinate space before being evaluated by the model. The particular transformation chosen results in the features being transformed to the most likely model coordinates given the model parameters.

In the remainder of this thesis we will first describe a variety of related work and background related to object recognition. This will be followed by the description of the probabilistic model, in Chapter 3, and how the parameters are learned, in Chapter 4. Chapter 5 will describe a variety of techniques that are used to make the approach tractable. The approach is then validated with a set of experiments on a variety of object classes, including motorbikes, faces, and cars. The final chapter will then discuss some short falls of the approach and future work.

Note

A portion of material presented in the introduction and succeeding chapters appeared in the Generative-Model Based Vision Workshop 2004, Helmer *et al.* [7]

Chapter 2

Part-Based Approaches to Object Recognition

The computer vision literature is rich with approaches to solving the object recognition problem, and approaches can be divided along many different lines. A complete overview of the field however is beyond the scope of this thesis. As a result, we outline the approaches that deal with the issues that were relevant in the design of the system outlined later in the thesis. The bulk of this chapter discusses part-based approaches to object recognition, and in Section 2.2.1 onwards some notation is introduced that is used throughout the thesis.

2.1 Local versus global approaches

Two distinctive methodologies exist to solving the object recognition problem: global and local approaches. Global approaches extract features of the object that are generally properties of the object as a whole, such as colour histograms [15] or texture and shape [12][16] . Local approaches, on the other hand, divide the object into smaller regions, and these are then used to recognize the object [13][17][14][8][5]. It should be noted however that it is not necessary that a vision system for object recognition take a strictly local or strictly global approach. In fact, certain object classes are devoid of any clear notion of parts, such as lakes or walls, whereas some object classes are little more than the sum of their parts, at least visually.

Much of the current research in object recognition is devoted to part based approaches because of several of its natural advantages. One of the primary advantages is that the background and occluding objects can be more easily discarded by simply looking only at the features on the object. In global approaches, however, difficult segmentation, or other restrictive assumptions, such as a uniform background, must be made so that the background and occluding objects do not influence the classification of the foreground. Another advantage is modelling the appearance of an object

class as a whole is a more complex task than modelling a subset of its parts.

2.2 Parts

Using a part based approach first requires asking the question, how should a part be represented?

Early research using a part-based approach by Biederman *et al.* focused on modelling the 3D structure of an object class, and inferring the 3D structure of an object from an image in order to do classification [2]. In Biederman's work, he theorizes that objects can be modelled using geons, or geometric ions, which are small 3D structures such as cylinders, cubes, etc. An object class in this case is a collection of geons with some sort of geometric relations and determining whether an object is present reduces to inferring the geons present, their geometric relations, and determining if this coincides with the object class model.

A prior definition of parts such as geons may seem attractive, but determining their presence is difficult since it involves inferring 3D structure from a 2D signal. An alternative approach uses an appearance based approach where a part is modelled only as the 2D signal it presents in an image. In this way inferring the 3D structure can be avoided entirely. In an appearance based representation a part is modelled upon the pixel values of image patch directly, or on some transformation performed on the pixel values, such as PCA.

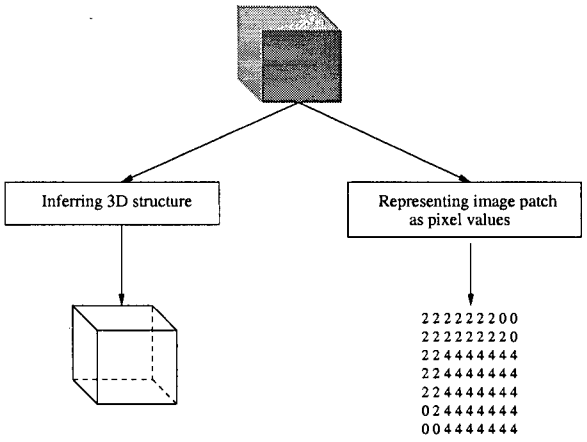


Figure 2.1: Representing and modelling parts based directly on its 2D appearance is an easier task then having to infer 3D structure.

Many current approaches to object recognition take an appearance based ap-

proach, but there is still the question of how to obtain parts that can be used for recognition. There are a variety of supervised approaches, Burl *et al.* [3] for example, where humans hand label parts on an image dataset. By taking such an approach, the question can be avoided, but labelling is time consuming, and it is desirable to be able to learn a model of an object class with as little supervision as possible.

Another approach is to learn the parts from a dataset of images containing the object class in an unsupervised manner. However, any region of an image can serve as a possible part, and searching through regions at every location and at multiples sizes in all images in the dataset is prohibitively expensive. Shimon Ullman *et al.* [17] take this approach to part selection, incrementally selecting features that reduce the entropy of classification. However, as mentioned, searching over all locations and multiple scales is computationally expensive.

2.2.1 Representation of an image

One way to overcome this problem is to select interesting regions in the image, encode these local regions, and then to use only these to learn parts and for recognition. The motivation behind such an approach is that an image of an object class is generally distinguished by regions that contain texture rather than those that do not. Thus, by representing an image as a selection of features, we reduce the search space to interesting regions. Moreover, many of these local feature approaches can help with dealing with invariance to scale, translation, and rotation. Additional discussion concerning feature detectors and descriptors can be found in Section 6.1 of the Experiments chapter, where we outline the particular feature detector and descriptor we used in our experiments.

An input image i is represented as a set of N features $\mathcal{F}^i$, which are extracted as a preprocessing step. For notational convenience we drop the reference to the image i in some sections of the thesis.

Each feature $f \in \mathcal{F}$ is composed of a d dimensional appearance vector $\mathbf{A}_f$, which describes a local region of scale $\mathbf{S}_f$, centred at location vector $\mathbf{X}_f$ which is the 2 dimensional tuple $(X_{f,1}, X_{f,2})$. The scale and location are described relative to image pixel coordinates. See Figure 2.2 for an illustration.

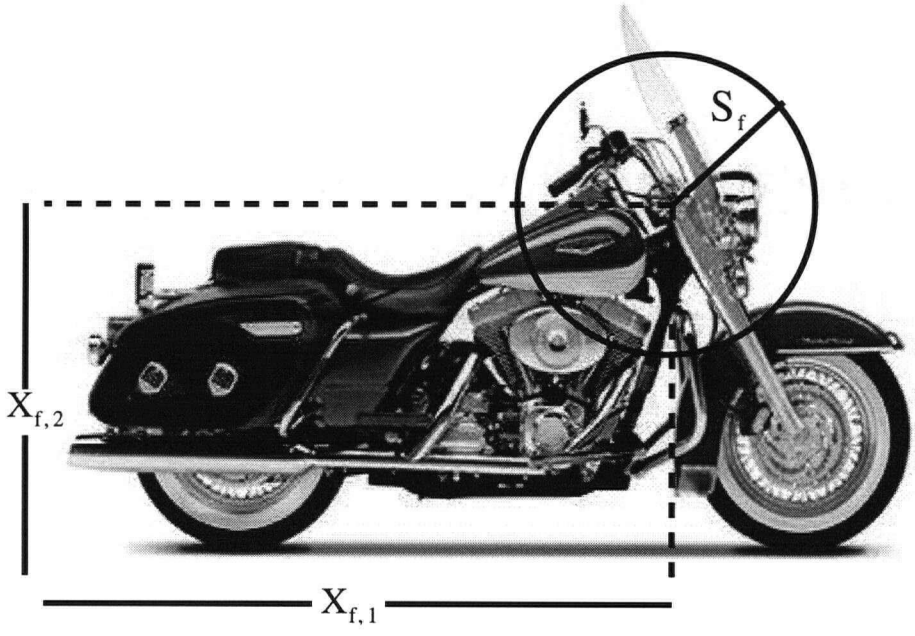


Figure 2.2: A feature f is describe by an image location $\mathbf{X}_f = (\mathbf{X}_{f,1}, \mathbf{X}_{f,2})$, a scale $\mathbf{S}$ and an appearance vector $\mathbf{A}_f$, which is some description based upon the pixel values within the circular region.

2.3 Recognizing an object class with parts

When an image is given to an object recognition system, the system outputs a score that is then used to determine whether the object is present. Exactly how this score is produced and what it means varies widely across part-based approaches. In general, part based approaches are concerned with finding a matching, or matchings, that matches features to parts in the model, and the score is based upon this matching. We define a matching as $\mathbf{h}$, where $\mathbf{h}(p) = f$ means that a feature f of the image matches part p. If a part is occluded or not present then $\mathbf{h}(p) = 0$. An example can be found in Figure 2.3.

Most part-based approaches to recognition have a model over the appearance of an object class' parts and a model over the geometric relationship between the parts. One approach to recognition by Schmid *et al.* [14], uses the the model over the appearance to determine a possible matching $\mathbf{h}$, then uses the geometric model to refine and score this matching. To find an initial matching they use a nearest neighbour approach, based upon the Mahalanobis distance, to extract matches from features to parts. Moreover, their geometric model encodes for each part the angles between neighbouring parts, and in this manner they can achieve scale invariance in their geometric model. After finding the initial matching, the matches that don't agree with the geometric model are removed from the matching. The final score is then the number of matches remaining in the matching. Although this is conceptually an effective approach, the uncertainty of the matches is not represented, and it does not represent the relationship between false matches and model size.

2.3.1 Probabilistic approaches to recognition

Most recent approaches take a probabilistic approach to object recognition instead of basing the measure on the number of matches. Given the object class C that we seek to recognize, the most obvious measure as to whether an object class is present in an image is the probability that the object class is in the image. That is,

$$p(C|\mathcal{F}) \tag{2.1}$$

Pope *et al.* [13] propose an approach to recognize objects using $p(C|\mathcal{F})$ by first finding a good matching $\mathbf{h}$, and a transformation, $\mathbf{t}$, that transforms image feature coordinates into model coordinate space. They then estimate $p(C|\mathcal{F})$ based up this single matching and transformation. Object classes are represented as containing

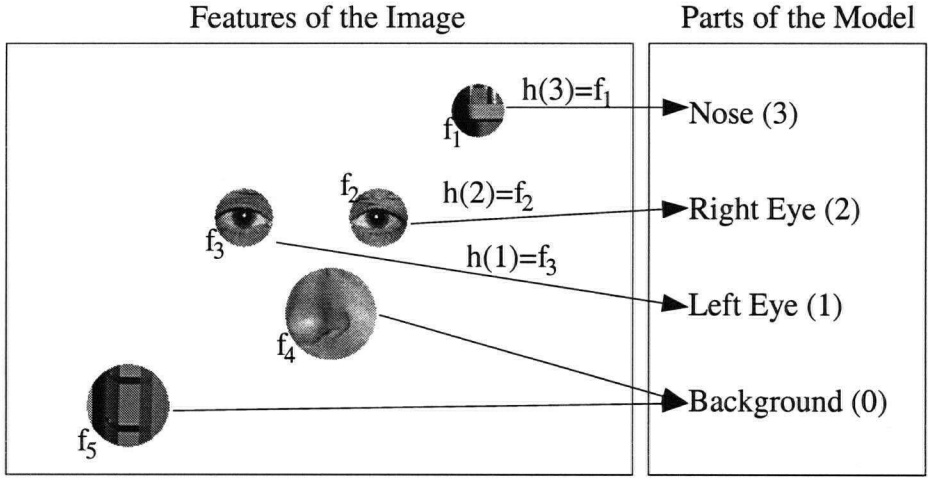


Figure 2.3: An example of a matching $\mathbf{h}$. Note that $\mathbf{h}$ does not have to be the true matching. For each of the features that are matched to a point it is implied that the remaining features are matched to the background

parts of various types with with a Gaussian distribution over the part locations in model coordinate space. In their approach, $p(C|\mathcal{F})$ is approximated as

$$\begin{aligned} p(C|\mathcal{F}) &\approx p(C|\mathcal{F}, \mathbf{h}, \mathbf{t}) && (2.2) \\ &= \frac{p(\mathcal{F}, \mathbf{h}, \mathbf{t}|C)}{p(\mathcal{F}, \mathbf{h}, \mathbf{t})} p(C) && (2.3) \\ &= \frac{p(\mathbf{A}|\mathbf{h}, \mathbf{t}, C) p(\mathbf{X}|\mathbf{h}, \mathbf{t}) p(\mathbf{h}, \mathbf{t}|C)}{p(\mathcal{F}|\mathbf{h}, \mathbf{t}) p(\mathbf{h}, \mathbf{t})} p(C) && (2.4) \end{aligned}$$

where they assume that part locations and appearance are independent given the transformations $\mathbf{t}$ and matching $\mathbf{h}$, and that $p(\mathbf{h}, \mathbf{t}|C)$ and $p(\mathbf{h}, \mathbf{t})$ are uniform. In their matching procedure they iteratively add a match of a feature to a part such that the match, along with the updated transformation $\mathbf{t}$, improves the probability of the match the most. The difficulty with this approach is that it is designed for recognition of a single object, and it is unclear how to learn parts effectively for a general object class.

An alternate approach is to learn the density $p(\mathcal{F}|C)$, the probability of seeing these features given that the object is present. In an approach by Weber *et al.* [19] they model $p(\mathcal{F}|C)$, and use the likelihood ratio

$$R = \frac{p(\mathcal{F}|C)}{p(\mathcal{F}|\neg C)} \tag{2.5}$$

to compute a measure for whether the object is in the image. This ratio compares the likelihood of seeing this set of features given that the object class is present against the likelihood of seeing the features if the object class was not present. The main advantage of using the ratio R for recognition is that the parameters θ for $p(\mathcal{F}|C)$ can be learned using EM, which requires little supervision.

2.4 The constellation model

One of the most successful approaches to general object recognition is that of Fergus *et al.*[5]. The approach they present is an extension of the constellation model that was developed by Burl *et al.* [3] and extended by Weber *et al.* [19]. Although they attain good results on a variety of datasets, it takes an enormous amount of time to learn and is limited to just 6 or 7 parts to keep learning tractable. Since the approach presented in this thesis builds on the constellation model, it is useful to introduce some of the details.

2.4.1 Matching features to parts

Suppose an image contains features $\mathcal{F}$, and the model contains P parts with parameters θ, there still is the problem of matching features to parts. Most previous approaches first match features to parts based solely on appearance and then later remove false matches by utilizing the learned geometric relations between parts. The approach taken by Fergus *et al.*, on the other hand, is to evaluate the value of a matching $\mathbf{h}$ jointly based upon the appearance and location. Now, they propose that if they had the matching $\mathbf{h}$ then

$$\begin{aligned} p(\mathcal{F}|\mathbf{h}) &= p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h}) && (2.6) \\ &= p(\mathbf{A}|\mathbf{h})p(\mathbf{X}|\mathbf{h})p(\mathbf{S}|\mathbf{h}) && (2.7) \end{aligned}$$

They assume that the appearance, scale, and the location of a part are independent of each other when the matching $\mathbf{h}$ is given.

In general, $\mathbf{h}$ is unknown, so they integrate it out of the joint $p(\mathbf{A}, \mathbf{X}, \mathbf{S}, \mathbf{h})$ to obtain $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$. That is, they define

$$p(\mathbf{A}, \mathbf{X}, \mathbf{S}) = \sum_{\mathbf{h} \in \mathcal{H}} p(\mathbf{A}|\mathbf{h}) p(\mathbf{X}|\mathbf{h}) p(\mathbf{S}|\mathbf{h}) p(\mathbf{h}) \tag{2.8}$$

where the hypothesis space $\mathcal{H}$ contains all possible matchings of parts to features.

The distributions $p(\mathbf{A}|\mathbf{h})$, $p(\mathbf{X}|\mathbf{h})$, and $p(\mathbf{S}|\mathbf{h})$ are all Gaussian distributions with distinct parameters for each part. However, they evaluate $\mathbf{X}$ and $\mathbf{S}$ in model coordinate space rather than image coordinate space. To achieve this scale and translation invariance, they use parts p that are matched to a feature f as a landmark point, and translate and scale the other features locations to model coordinate space by using the parameters of part p. This poses a difficulty because which feature is chosen dictates where the other features are transformed to in model coordinate space and their is no clear way to chose the landmark point.

One thing to note in Equation 2.7 is that $|\mathcal{H}| \in O(|F|^P)$, so evaluating $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$ exactly is intractable for models with more than 6 or 7 parts. If the model is accurate than A* can be used to approximate $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$. However, the manner in which they learn the parameters for the model is intractable for models with more than 7 parts. The reason for this will be discussed more fully in Section 4.1.1. One of the primary contributions of this thesis is the introduction of a learning scheme that remains tractable even for models with a large number of parts.

Chapter 3

Model

The approach we take to the object recognition problem is to learn a distribution $p(\mathcal{F}|C)$, and use the likelihood ratio $R = p(\mathcal{F}|C)/p(\mathcal{F}|\neg C)$ to recognize an object. Given a scene with an instance of the object class, a 2D image is generated by the foreground object, and by the background scene. This naturally leads to the choice of modelling $p(\mathcal{F}|C)$ as a generative model, where the features $\mathcal{F}$ are generated by the parts of the object and the background. The model presented in this chapter builds upon the constellation model of Fergus *et al.*; the similarities and differences will be noted as they are presented.

If a part p generates a particular feature f, then $\mathbf{h}(p) = f$, so $\mathbf{h}$ is the matching from parts to features. If a feature f is not assigned to a part then it is assigned to the background, the set of background features for a particular matching h is denoted as $\mathcal{B}_{\mathbf{h}}$. In addition, if the object is in the image, then it will occur at some location and scale, denoted $\mathbf{c} = \{\mathbf{X_c}, \mathbf{S_c}\}$. Naturally $\mathbf{X_c} \in \mathcal{R}^2$ such that $\mathbf{X_c}$ does not exceed the dimensions of the image, and $\mathbf{S_c} \in \mathcal{R}$ such that it does not exceed the maximum dimension of the image. The joint space to which $\mathbf{c}$ belongs is referred to as $\mathcal{C}$. So, for each of the parts that generated a feature in the image, the feature will occur at some location and scale relative to $\mathbf{c}$, and with some appearance. An example of how the model is generative is illustrated in Figure 3.1.

For notational convenience, we refer to $p(\mathcal{F}|C)$ as the density $p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\theta)$ where in most instances θ is dropped. The background density $p(\mathcal{F}|\neg C)$ is sometimes referred to as $p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h}_\emptyset)$ where $\mathbf{h}_\emptyset$ is the matching that matches all features in the image to the background.

Let the probability of seeing the features of an image, given that we have a matching $\mathbf{h}$, and the object's centre and scale, $\mathbf{c}$, and model parameters θ, be

$$p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h}, \mathbf{c}) = p(\mathbf{A}|\mathbf{h})p(\mathbf{X}|\mathbf{h}, \mathbf{c})p(\mathbf{S}|\mathbf{h}, \mathbf{c}) \tag{3.1}$$

That is, if we know $\mathbf{c}$ and $\mathbf{h}$, then the appearance, location, and scale of the features are all independent. This assumption is similar to that of Fergus *et al.*, with the exception that Fergus *et al.* don't introduce the parameter $\mathbf{c}$. Although the indepen-

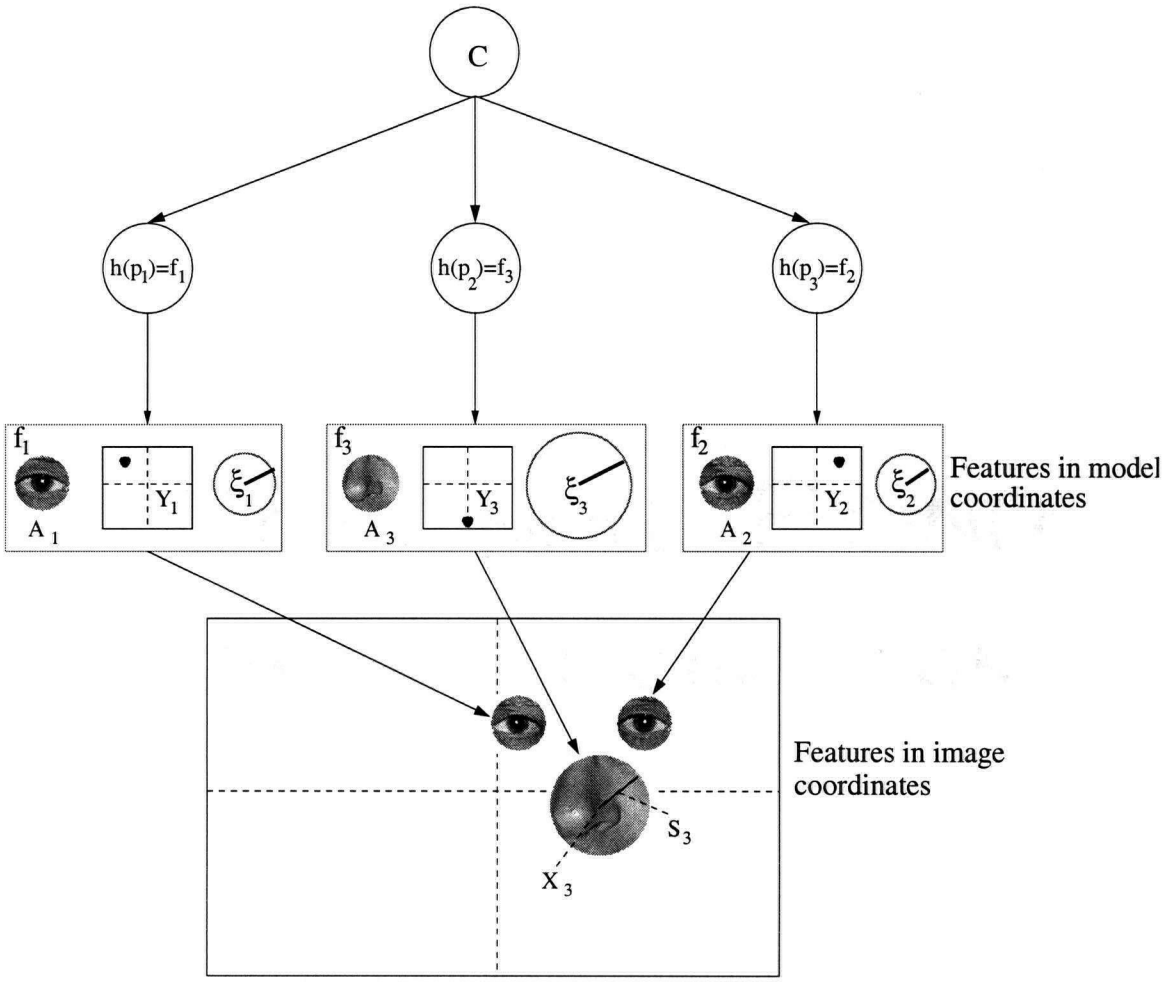


Figure 3.1: When an object class is in an image, its constituent parts that are present generate features, where each feature f has an appearance A_f, a location and scale in model coordinate space Y_f and $\boldsymbol{\xi}_f$. The object then generates a center and scale of the object in image coordinate space $\mathbf{c} = \{\mathbf{X_c}, \mathbf{S_c}\}$. As a result, the image location of feature f is $\mathbf{X}_f = \frac{(\mathbf{Y}_f - \mathbf{X_c})}{\mathbf{S_c}}$ and the scale is $\mathbf{S}_f = \frac{\boldsymbol{\xi}_f}{\mathbf{S_c}}$

dence assumption makes modelling easier, it is easy to conjure up examples where the appearance, scale or location of a part are not independent. One clear example of this is that of a truck, where the location of a truck door is dependent upon the size of the wheel. Modelling these dependencies, however, is more difficult, and it is not clear that these dependecies are crucial in object classification.

Although we have assumed independence between $\mathbf{A}$, $\mathbf{X}$, and $\mathbf{S}$ when $\mathbf{h}$ and $\mathbf{c}$ are given, it is not generally the case that these variables are known in object classification. In order to overcome this we can integrate $\mathbf{h}$ and $\mathbf{c}$ out of the joint distribution $p(\mathbf{A}, \mathbf{X}, \mathbf{S}, \mathbf{h}, \mathbf{c}|C)$ to determine $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$. So,

$$\begin{aligned} p(\mathbf{A}, \mathbf{X}, \mathbf{S}) &= \sum_{\mathbf{h}\in\mathcal{H}} \int_{\mathbf{c}\in\mathcal{C}} p(\mathbf{A}, \mathbf{X}, \mathbf{S}, \mathbf{h}, \mathbf{c}) d\mathbf{c} \\ &= \sum_{\mathbf{h}\in\mathcal{H}} \int_{\mathbf{c}\in\mathcal{C}} p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h}, \mathbf{c}) p(\mathbf{h}) p(\mathbf{c}) d\mathbf{c} \\ &= \sum_{\mathbf{h}\in\mathcal{H}} p(\mathbf{A}|\mathbf{h}) p(\mathbf{h}) \int_{\mathbf{c}\in\mathcal{C}} p(\mathbf{X}|\mathbf{h}, \mathbf{c}) p(\mathbf{S}|\mathbf{h}, \mathbf{c}) p(\mathbf{c}) d\mathbf{c} \end{aligned} \tag{3.2}$$

is the probability of seeing this set of features given that the object is in the image. However, the integral over $\mathcal{C}$ is not analytically tractable with most distributions, so we instead use a point mass estimate to approximate the integral.

$$p(\mathbf{A}, \mathbf{X}, \mathbf{S}) \approx \sum_{\mathbf{h}\in\mathcal{H}} p(\mathbf{A}|\mathbf{h}) p(\mathbf{h}) \max_{\mathbf{c}\in\mathcal{C}} p(\mathbf{X}, \mathbf{S}, \mathbf{c}|\mathbf{h}) \tag{3.3}$$

The intuition behind this approximation is that we find the best transformation of feature locations and scales into model coordinate space, and approximate $p(\mathbf{X}, \mathbf{S}|\mathbf{h})$ based solely on this best transformation, rather than all possible transformations. This is accurate in the normal case when there is one transformation that dominates all others. This approximation will be discussed in detail in Section 3.3.

Moreover, as with the Fergus *et al.* approach $|\mathcal{H}| = O(|\mathcal{F}|^P)$, so evaluating the sum in Equation 3.3 is expensive. Tractable approximations to compute $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$ are presented in Chapter 5. The remainder of this chapter descibes the foreground and background model.

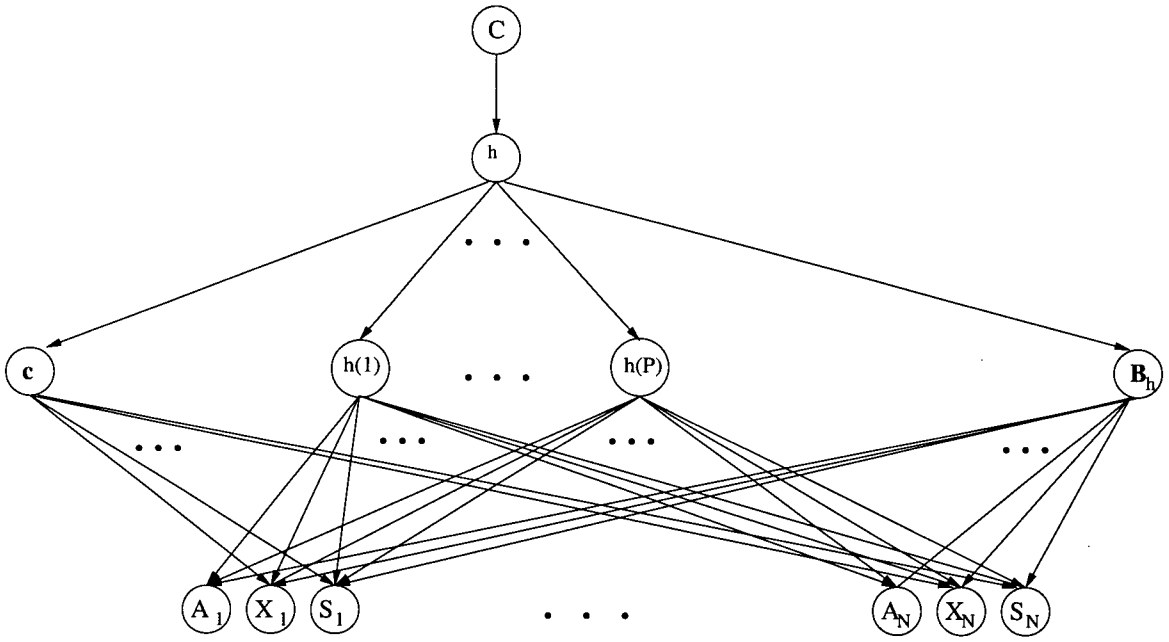


Figure 3.2: The graphical model for the random variables $\mathbf{h}$, $\mathbf{A}$, $\mathbf{X}$, $\mathbf{S}$, $\mathbf{c}$, and $\mathbf{B_h}$

3.1 Background model

Any given scene is composed of a number of objects, which belong to different object classes, and it is these scene elements, under illumination, that generate a 2D image. In a generative model approach, the primary concern is modelling of the foreground. However, in order to come up with the likelihood ratio, R, it is necessary to have some model of $p(\mathcal{F}|\neg C) = p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h}_\emptyset)$, so that there is some measure as to how likely it is to see the features $\mathcal{F}$ in general. We assume that if a feature is generated by the background, then it is independent of other features, and that the appearance, location, and scale are also independent of one another. So, $p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h}_\emptyset)$ becomes

$$p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h}_\emptyset) = \prod_{f \in \mathcal{F}} p(\mathbf{A}_f|\mathbf{h}_\emptyset) p(\mathbf{X}_f|\mathbf{h}_\emptyset) p(\mathbf{S}_f|\mathbf{h}_\emptyset) \tag{3.4}$$

where $\mathbf{h}_\emptyset$ is the matching where all features are matched to the background.

Moreover, when the object is in the image it is usually the case that not all of the features in $\mathcal{F}$ are generated by that object. As a result, it is necessary that the distribution $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$ also models those features that are part of the background.

3.1.1 Background scale

In general, a feature generated by the background could be any size. However, for most feature detectors that operate at multiple scales, the probability of a random feature f having scale of $\mathbf{S}_f$ is proportional to $\frac{1}{\mathbf{S}_f^2}$. The reason for this is that feature detectors generally construct an image pyramid, where the image at each level $\mathbf{S}_f$ is a Gaussian blur with $\sigma = \mathbf{S}_f$ of the original image. This is then resampled according to $\mathbf{S}_f$, resulting in an image $\frac{1}{\mathbf{S}_f^2}$ the size of the original image. As a result, the number of features detected at a particular scale is $\frac{1}{\mathbf{S}_f}$ the original image size. If the largest scale at which a feature can be detected for an image is α, then

$$p(\mathbf{S}_f | f \in \mathcal{B}) = \frac{2 \log \alpha}{\mathbf{S}_f^2} \tag{3.5}$$

This background distribution on scale differs from that of Fergus *et al.*who represents the background distribution simply as a uniform distribution. The motivation for the change is that feature detections are not usually generated uniformly across scales.

3.1.2 Background location

In general it can be expected that a feature from the background can be found uniformly throughout the image, so in this case the location is modelled as

$$p(\mathbf{X}_f | f \in \mathcal{B}) = \frac{1}{\gamma}. \tag{3.6}$$

where γ is the area of the image. This distribution is in accordance with the that chosen by Fergus *et al.*

3.1.3 Background appearance

The appearance of the average feature returned by the feature detector is highly dependent upon both the detector and descriptor. As a result the appearance is modelled with a kernel density estimator with a Gaussian kernel.

In the thesis, we primarily work with Gaussian distributions where the coovariance matrix is restricted to be diagonal. If a d dimensional random variable X is distributed according to a Gaussian distribution with mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$, ie. that $X \sim \mathcal{N}(\boldsymbol{\mu}, (\boldsymbol{\sigma})^2)$, the probability of X is described as

$$G(X|\boldsymbol{\mu}, \boldsymbol{\sigma}^2) = \frac{1}{(2\pi)^{0.5d} \prod_i |\boldsymbol{\sigma}_i|} exp\left(\sum_i^d \frac{(X_i - \boldsymbol{\mu}_i)^2}{2\boldsymbol{\sigma}_i^2}\right) \tag{3.7}$$

The probability of the appearance given that it was generated by the background is

$$p(\mathbf{A}_f | f \in \mathcal{B}) = \frac{\sum_{f' \in \mathcal{B}} G(\mathbf{A}_{f'} - \mathbf{A}_f | 0, \boldsymbol{\sigma}_{bf})}{|BackGround|} \tag{3.8}$$

where *BackGround* are a set of features that are extracted from a set of background images, and where $\boldsymbol{\sigma}_{bf}$ is chosen experimentally by cross validation. This distribution differs from the model presented by Fergus *et al.*, which uses a single Gaussian for all features. The motive for this change is that for high dimensional features, a single Gaussian does not provide an adequate model for the appearance of features generated by the background. This issue is a complicated one, and constructing a better model for the background distribution of appearances is left for future research.

3.2 Foreground model

As mentioned previously, given a matching $\mathbf{h}$ and $\mathbf{c}$ we assume that appearance, location, and scale of the features are independent. We also assume that given $\mathbf{h}$ and $\mathbf{c}$ that all the attributes for each part are independent. In this sense, given that an object is part of a particular class, and its center $\mathbf{c}$ is known, then which parts are generated, where these parts are generated, and their appearance are all independent of each other.

Clearly a lot of information is lost by this independence assumption. A face model that assumes such independence would probably consider a woman's face with a beard as likely as a man's face with a beard. However, in terms of object classification, these examples are pathological in the sense that real data rarely presents cases where the dependence needs to be modelled in order to do correct classification. In cases where modelling this dependence is necessary, the object class can be split into subclasses. One possible way to do this is during learning, detect when there are large segments of the training data that are not being recognized by the current model. At this point the algorithm could then learn a seperate model for this segment of the training data, thereby automatically creating a model for different subclasses. Development of a technique to do this is left for future research.

3.2.1 Part appearance

If a part p is present in an image as feature f, then we assume that the appearance of the part $\mathbf{A}_f \sim \mathcal{N}(\boldsymbol{\mu}_p^{app}, (\boldsymbol{\sigma}_p^{app})^2)$. That is that,

$$p(\mathbf{A}_f|\mathbf{h}(p) = f) = G(\mathbf{A}_f|\boldsymbol{\mu}_p^{app}, (\boldsymbol{\sigma}_p^{app})^2) \tag{3.9}$$

We assume that each dimension of the appearance is independent. There is no reason to believe that the dimension of each part is independent, but estimating a full covariance is prohibitely expensive.

Given a matching $\mathbf{h}$, we evaluate the appearance of a feature f according to the density for part p only if $\mathbf{h}(p) = f$, and if f is assigned to no part then it is evaluated according the background distribution, $p(\mathbf{A}_f|f \in \mathcal{B}_\mathbf{h})$. So, for the entire set of appearances $\mathbf{A}$,

$$p(\mathbf{A}|\mathbf{h}) = \prod_{f|\mathbf{h}(p)=f} G(\mathbf{A}_f|\boldsymbol{\mu}_p^{app}, (\boldsymbol{\sigma}_p^{app})^2) \prod_{f|f\in\mathcal{B}_\mathbf{h}} p(\mathbf{A}_f|f \in \mathcal{B}) \tag{3.10}$$

3.2.2 Part location

If a feature f is generated by a part p, then we assume that in model coordinate space that the location of the part $\mathbf{Y}_f \sim \mathcal{N}(\boldsymbol{\mu}_p^{loc}, (\boldsymbol{\sigma}_p^{loc})^2)$. The ith element of $\boldsymbol{\sigma}_p^{loc}$ is referred to as $\boldsymbol{\sigma}_{p,i}^{loc}$.

The problem is that since $\mathbf{X}_f$ is in image coordinate space, it must be transformed into model coordinate space. If we are given the object center and scale $\mathbf{c}$, then

$$\mathbf{Y}_f = \mathbf{S_c}\mathbf{X}_f + \mathbf{X_c} \tag{3.11}$$

$$\Leftrightarrow \mathbf{X}_f = \frac{(\mathbf{Y}_f - \mathbf{X_c})}{\mathbf{S_c}} \tag{3.12}$$

which implies that $\mathbf{X}_f \sim \mathcal{N}\left(\frac{(\boldsymbol{\mu}_p^{loc} - \mathbf{X}_c)}{\mathbf{S_c}}, (\frac{\boldsymbol{\sigma}_p^{loc}}{\mathbf{S_c}})^2\right)$. As a result

$$p(\mathbf{X}_f|\mathbf{h}(p) = f, \mathbf{c}) = G\left(\mathbf{X}_f \middle| \frac{(\boldsymbol{\mu}_p^{loc} - \mathbf{X_c})}{\mathbf{S_c}}, \left(\frac{\boldsymbol{\sigma}_p^{loc}}{\mathbf{S_c}}\right)^2\right) \tag{3.13}$$

However, if f is part of the background, $f \in \mathcal{B}_\mathbf{h}$, then we evaluate its location according to a background distribution. The resulting probability of seeing the locations $\mathbf{X}$ is therefore

$$p(\mathbf{X}|\mathbf{h},\mathbf{c}) = \prod_{p|\mathbf{h}(p)=f} G\left(\mathbf{X}_f \middle| \frac{(\boldsymbol{\mu}_p^{loc} - \mathbf{X}_\mathbf{c})}{\mathbf{S}_\mathbf{c}}, \left(\frac{\boldsymbol{\sigma}_p^{loc}}{\mathbf{S}_\mathbf{c}}\right)^2\right) \prod_{f|f\in\mathcal{B}_\mathbf{h}} p(\mathbf{X}_f|f \in \mathcal{B}) \quad (3.14)$$

3.2.3 Part scale

If $\mathbf{h}(p) = f$, then the scale of part p in model coordinate space, $\boldsymbol{\xi}_f$, is generated according to a Gaussian with, with mean μ_p^{scale} and standard deviation σ_p^{scale}. Since the scales $\mathbf{S}$ are given in image coordinates, they are transformed into model coordinates,

$$\boldsymbol{\xi}_f = \mathbf{S}_\mathbf{c}\mathbf{S}_f \quad (3.15)$$

This implies that $\mathbf{S}_f \sim \mathcal{N}(\frac{\mu_p^{scale}}{\mathbf{S}_\mathbf{c}}, (\frac{\sigma_p^{scale}}{\mathbf{S}_\mathbf{c}})^2)$.

If f is part of the background, $f \in \mathcal{B}_\mathbf{h}$, then we evaluate its location according to a background distribution. As a result,

$$p(\mathbf{S}|\mathbf{h},\mathbf{c}) = \prod_{p|\mathbf{h}(p)=f} G\left(\mathbf{S}_f \middle| \frac{\mu_p^{scale}}{\mathbf{S}_\mathbf{c}}, \left(\frac{\sigma_{\mathbf{h}(p)}^{scale}}{\mathbf{S}_\mathbf{c}}\right)^2\right) \prod_{f|f\in\mathcal{B}_\mathbf{h}} p(\mathbf{S}_f|f \in \mathcal{B}) \quad (3.16)$$

3.2.4 Part statistics

We model the distribution $p(\mathbf{h})$ as

$$p(\mathbf{h}) = \frac{1}{|\mathcal{H}|} \prod_{p|\mathbf{h}(p)\neq 0} w_p \prod_{p|\mathbf{h}(p)=0} (1 - w_p) \quad (3.17)$$

where w_p is the probability that the part is present. A more complicated model is utilized in Fergus *et al.*, but as the number of parts increase, the number of parameters increases quadratically, which poses a problem for our approach. Implementing a more complicated model is left for further research.

3.3 Approximating object center and scale

Integrating $\mathbf{c}$ out of the joint $p(\mathbf{A}, \mathbf{X}, \mathbf{S}, \mathbf{h}, \mathbf{c})$, as mentioned previously, is not analytically tractable. Instead, we make the approximation

$$p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h}) \approx \max_{\mathbf{c}\in\mathcal{C}} p(\mathbf{X}, \mathbf{S}, \mathbf{c}|\mathbf{h}) \tag{3.18}$$

That is, we replace the integral with a point mass estimate at the maximum, since when $\mathbf{X}$ and $\mathbf{S}$ are fixed then $p(\mathbf{X}, \mathbf{S}, \mathbf{c}|\mathbf{h})$ is a peaked distribution. In essence, this means that we are primarily concerned with the measurement of $p(\mathbf{X}, \mathbf{S}, \mathbf{c}|\mathbf{h})$ at the location where the parts are most likely to be centered, given the matching $\mathbf{h}$. By making such an assumption we can find the $\mathbf{c}_{opt}$ that maximizes $p(\mathbf{X}, \mathbf{S}, \mathbf{c}|\mathbf{h})$, and use this to transform $\mathbf{X}$ and $\mathbf{S}$ in to model coordinate space. To find the maximum, notice that

$$p(\mathbf{X}, \mathbf{S}, \mathbf{c}|\mathbf{h}) = p(\mathbf{X}|\mathbf{c}, \mathbf{h})p(\mathbf{S}|\mathbf{c}, \mathbf{h})p(\mathbf{c}) \tag{3.19}$$

$$= \Big(\prod_{p|\mathbf{h}(p)=f} G\left(\mathbf{X}_f \middle| \frac{(\boldsymbol{\mu}_p^{loc} - \mathbf{X}_\mathbf{c})}{\mathbf{S}_\mathbf{c}}, \left(\frac{\boldsymbol{\sigma}_p^{loc}}{\mathbf{S}_\mathbf{c}}\right)^2\right) \tag{3.20}$$

$$G\left(\mathbf{S}_f \middle| \frac{\mu_p^{scale}}{\mathbf{S}_\mathbf{c}}, \left(\frac{\sigma_{\mathbf{h}(p)}^{scale}}{\mathbf{S}_\mathbf{c}}\right)^2\right) \Big)$$

$$\Big(\prod_{f|f\in\mathcal{B}} p(\mathbf{X}_f|f \in \mathcal{B}_\mathbf{h}) p(\mathbf{S}_f|f \in \mathcal{B}) \Big) \tag{3.21}$$

The second term in 3.20 is independent of $\mathbf{c}$, and as a result is a constant when $\mathbf{X}$ and $\mathbf{S}$ are fixed. The first term, which we will denote with the function $\mathcal{V}(\mathbf{c}|\mathbf{h}, \mathbf{X}, \mathbf{S})$ is the term that must be maximized. Taking a closer look at this function reveals that the value $\mathbf{c}_{opt}$ which maximizes $\log(\mathcal{V})$ also maximize $\mathcal{V}$ and thus $p(\mathbf{X}, \mathbf{S}, \mathbf{c}|\mathbf{h})$. To determine $\mathbf{c}_{opt}$ we maximize

$$\log(\mathcal{V}) = \sum_{p|\mathbf{h}(p)=f} \left[\log G\left(\mathbf{X}_f \left| \frac{(\boldsymbol{\mu}_p^{loc} - \mathbf{X_c})}{\mathbf{S_c}}, \left(\frac{\boldsymbol{\sigma}_{p\cdot}^{loc}}{\mathbf{S_c}} \right)^2 \right. \right) \right. \tag{3.22}$$

$$\left. + \log G\left(\mathbf{S}_f \left| \frac{\mu_p^{scale}}{\mathbf{S_c}}, \left(\frac{\sigma_{\mathbf{h}(p)}^{scale}}{\mathbf{S_c}} \right)^2 \right. \right) \right] \tag{3.23}$$

$$= -\sum_{p|\mathbf{h}(p)=f} \frac{(\mathbf{X}_{f,1}\mathbf{S_c} + \mathbf{X}_{\mathbf{c},1} - \boldsymbol{\mu}_{p,1}^{loc})^2}{2(\sigma_{p,1}^{loc})^2} + \log(\mathbf{S_c}) \tag{3.24}$$

$$-\sum_{p|\mathbf{h}(p)=f} \frac{(\mathbf{X}_{f,2}\mathbf{S_c} + \mathbf{X}_{\mathbf{c},2} - \boldsymbol{\mu}_{p,2}^{loc})^2}{2(\sigma_{\mathbf{h}(f),2}^{loc})^2} + \log(\mathbf{S_c}) \tag{3.25}$$

$$-\sum_{p|\mathbf{h}(p)=f} \frac{(\mathbf{S}_f\mathbf{S_c} - \boldsymbol{\mu}_p^{scale})^2}{2(\sigma_p^{scale})^2} + \log(\mathbf{S_c}) + \text{constant} \tag{3.26}$$

The most straightforward way to find the set of parameters $\mathbf{c}_{opt}$ that minimizes $\log(\mathcal{V})$ is to take the derivative and set it to 0 and solve for the parameters. There is a closed form solution, where the solution requires solving a quadratic equation in $\mathbf{S_c}$, but the actual solution is omitted for brevity.

3.4 Probability on sets

The technical details addressed in this section deal with defining probabilities on sets with a variable size, and can be skipped without loss of continuity.

In the model proposed by Fergus *et al.*, the number of features extracted from an image, k, are fixed apriori to some number, usually less than 20. That is, despite the fact that a highly textured image could contain 500 features, the image is represented by only a small subset of these features. The difficulty that this poses is that if a face appears in a very cluttered scene, then it's possible that very few of the features chosen to represent the image actually fall on the face. To overcome such difficulty, we do not fix the number of features k that represent an image. The number of features is entirely dependent upon the feature detector. However, if k varies per image, there are a number of technical details that need to be addressed. First, $p(\mathcal{F}|C)$, as defined above is not a probability since

$$\sum_{1}^{\infty} \int_{\mathbf{X} \in \mathcal{R}^{kx2}} \int_{\mathbf{S} \in \mathcal{R}^{k}} \int_{\mathbf{A} \in \mathcal{R}^{kxd}} p(\mathbf{A}, \mathbf{X}, \mathbf{S}) d\mathbf{A} d\mathbf{S} d\mathbf{X} = \infty \tag{3.27}$$

That is, if we sum over all feature sets of all possible sizes, the distribution does not sum to 1, but to ∞. However, this can be easily over come if we define

$$p(\mathcal{F}, k|C) = p(\mathcal{F}|C)p(k) \tag{3.28}$$

and if we define $p(k)$ to be uniform over $[0, M]$, where M in this case is some large number. If we implicitly redefine $p(\mathcal{F}|C)$ to be $p(\mathcal{F}, k|C)$, then $p(\mathcal{F}|C)$ is once again a proper probability. In practice $p(k)$ can be ignored in both learning and recognition. The reason is that $p(k)$ is a constant and thus cancels out in the likelihood ratio R and does not affect parameter estimation in learning.

Chapter 4

Learning

If we are given a training set D for which every image in the dataset contains an instance of the object class, we would like to learn a set of θ for the posterior distribution $p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\theta)$. This chapter presents a method to learn these parameters quickly.

The general learning algorithm is as follows

1. Extract a set of candidate parts $\mathcal{J}$ from the dataset
2. Initialize a small model from these candidate parts
3. For parts $p = k + 1$ to P
   (a) Maximize the parameters for the parts 1 to p using EM
   (b) For each potential part $j \in \mathcal{J}$ that has not been used
      i. Determine the increase in the likelihood, K_j, of the data if part j is part of the model
      ii. Select the part j with maximum K_j and add it to the model

The rest of the chapter is devoted to explaining each of these steps. The first section describes the maximum likelihood framework and the EM algorithm, and discusses why learning all P parts jointly from a random initialization is intractable. Section 4.2 describes incremental learning, its motivation and its pitfalls. The final section describes how to initialize a small model and how to select a set of candidate parts from the dataset. Readers familar with EM can skip section 4.1.

4.1 Parameter estimation by Expectation Maximization

Given a set of training data D, we'd like to determine the posterior distribution $p(\mathbf{A}, \mathbf{X}, \mathbf{S}|D)$, which we usually denote $p(\mathbf{A}, \mathbf{X}, \mathbf{S}|D)$, using the information in D. Ideally, we'd like to determine

$$p(\mathbf{A}, \mathbf{X}, \mathbf{S}|D) = \int_\theta p(\mathbf{A}, \mathbf{X}, \mathbf{S}, \theta|D) \tag{4.1}$$

$$= \int_\theta p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\theta)p(\theta|D) \tag{4.2}$$

However, this is computationally intractable for the model outlined, so instead we approximate the posterior where the integral is estimated by a point mass,

$$p(\mathbf{A}, \mathbf{X}, \mathbf{S}) \approx p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\theta) \tag{4.3}$$

that is, we use a single set of parameters to evaluate $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$. Within the machine learning community, there are a variety of approaches to chose this set of parameters, and one of the most common methodologies is to select the set of parameters θ that maximizes the likelihood of the data $p(D|\theta)$. In this case,

$$\theta_{ML} = \underset{\theta}{\text{argmax}}\, p(D|\theta) \tag{4.4}$$

One of the pitfalls of using θ_{ML} is that the parameters may overfit to the data. That is, the parameters may also fit not only to object class present in the images, but also to the noise or background in the data. This entails that θ_{ML} may not generalize well for the object classification problem. Another methodology for selecting a set of model parameters is to select the maximum a posteriori (MAP) parameters θ_{MAP}:

$$\theta_{MAP} = \underset{\theta}{\text{argmax}}\, p(\theta|D) \tag{4.5}$$

$$= \underset{\theta}{\text{argmax}} \frac{p(D|\theta)p(\theta)}{p(D)} \tag{4.6}$$

$$= \underset{\theta}{\text{argmax}}\, p(D|\theta)p(\theta) \tag{4.7}$$

This set of parameters are the most likely given the data and the prior, where $p(\theta)$ is a prior over the parameters. It is this prior that can be used to prevent the overfitting

that can plague learning the maximum likelihood parameters θ_{ML}. The prior basically acts as a constraint over the set of parameters. For the model we are learning we found it was only necessary to impose a prior on the parameters $\boldsymbol{\sigma}^{loc}$. The specification of the prior is stated in Section 4.1.4.

So we seek to find the set of parameters θ that maximizes

$$\mathcal{L}(\theta|D) = p(D|\theta)p(\theta) \tag{4.8}$$

which is referred to as the penalized likelihood. The first thing to note is that $\mathcal{L}(\theta|D)$ is a function with many local maxima for the particular model we are interested in. As a result, any numerical technique that seeks to find the set parameters to maximize $\mathcal{L}(\theta|D)$ will settle on a set of parameters that is but a local maxima of $\mathcal{L}(\theta|D)$.

One of the standard techniques to learn the parameters θ_{MAP} is the EM algorithm [4], which iteratively improves the parameters until a local maximum is achieved. To explain EM a few facts must first be noted. One is that maximizing $\mathcal{L}(\theta|D)$ is equivalent to maximizing $l(\theta|D) = \log \mathcal{L}(\theta|D)$ because of the monotonicity of the logarithmic function.

In addition,

$$\begin{aligned}
l(\theta|D) &= \sum_i \log \sum_{\mathbf{h}\in\mathcal{H}} p(\mathcal{F}^i, \mathbf{h}|\theta) + \log p(\theta) && (4.9)\\
&= \sum_i \log \sum_{\mathbf{h}\in\mathcal{H}} q(\mathbf{h}|\mathcal{F}^i, \theta)\frac{p(\mathcal{F}^i, \mathbf{h}|\theta)}{q(\mathbf{h}|\mathcal{F}^i, \theta)} + \log p(\theta) && (4.10)\\
&\geq \sum_i \sum_{\mathbf{h}\in\mathcal{H}} q(\mathbf{h}|\mathcal{F}^i) \log \frac{p(\mathcal{F}^i, \mathbf{h}|\theta)}{q(\mathbf{h}|\mathcal{F}^i)} + \log p(\theta) && (4.11)\\
&= l_b(\theta|D) && (4.12)
\end{aligned}$$

where Equation 4.11 holds true by the Jensen inequality only if q sums to 1.

So, if we fix the values of the function q, then for all θ, $l(\theta|D) \geq l_b(\theta|D)$. Also note that if we fix q to $p(\mathbf{h}|\mathcal{F}_i, \theta)$, then it is easy to see in Equation 4.11 that $l_b(\theta|D) = l(\theta|D)$.

Now, on the t^{th} iteration of EM we have parameters θ^t. EM first fixes q to $p(\mathbf{h}|\mathcal{F}_i, \theta^t)$, so that $l_b(\theta^t|D) = l(\theta^t|D)$, and this is referred to as the Expectation step. On the Maximization step the parameters θ' are found that maximizes $l_b(\theta|D)$ are found, so now it is the case that $l_b(\theta'|D) \geq l_b(\theta^t|D)$. However, since $l_b(\theta^t|D) = l(\theta^t|D)$, it is the case that $l(\theta'|D) \geq l(\theta^t|D)$. In other words, by increasing the lower bound l_b

we also increase l. At this point the parameters of the model are updated to $\theta^{t+1} = \theta'$, and the process is repeated. In this way the algorithm updates the parameters in such a way that it is guaranteed that $l(\theta^{t+1}|D) \geq l(\theta^t|D)$.

The algorithm is as follows

EM Algorithm
For t = 1 to T

1. Calculate $p(\mathbf{h}|F^i, \theta^t)$ (E step)
2. Calculate parameters θ^{t+1} that maximize $l(\theta|D)$ (M step)

4.1.1 Expectation

On the expectation step of the algorithm $p(\mathbf{h}|\mathcal{F}^i, \theta^t)$ needs to be calculated for all i and $\mathbf{h}$. By applying Bayes rule

$$p(\mathbf{h}|\mathcal{F}, \theta^t) = \frac{p(\mathcal{F}, \mathbf{h}|\theta^t)p(h)}{p(\mathcal{F}|\theta^t)} \tag{4.13}$$

$$= \frac{p(\mathcal{F}|\mathbf{h}, \theta^t)p(\mathbf{h})}{p(\mathcal{F}|\theta^t)} \tag{4.14}$$

$$= \frac{p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h})p(\mathbf{h})}{\sum_{h \in \mathcal{H}} p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h})p(\mathbf{h})} \tag{4.15}$$

The standard method to compute the normalization term $Z = \sum_{h \in \mathcal{H}} p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h})p(\mathbf{h})$ is to first compute $p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h})p(\mathbf{h})$ for all $h \in \mathcal{H}$. If the model parameters θ^t are at all accurate, then for most $\mathbf{h} \in \mathcal{H}$ the value $p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h}) \approx 0$, at least in terms of how much it contributes to Z. As a result Z can be approximated and $p(\mathbf{h}|\mathcal{F}, \theta^t)$ need only be calculated for a small subset of $\mathcal{H}$.

The problem is that if the parts are initialized randomly then techniques such as A* cannot reduce the magnitude of the problem greatly. The reason for this is that there needs to be a large number of matchings $\mathbf{h}$ for which $p(\mathbf{h}|\mathcal{F}, \theta^t)$ is non-zero, otherwise EM will never be able to converge onto a satisfactory solution. The primary contribution of this thesis is the development a method to overcome this very problem. If the parts of the model are initialized intelligently with a small number of parts, then learning an accurate distribution for a small number of parts is easy. By

adding parts incrementally we avoid the combinatorial explosion because the number of matchings where $p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h})$ actually needs to be evaluated for the Expectation step is minimized. Techniques that are used to calculate Z efficiently are discussed in Chapter 5.

4.1.2 Maximization

In the maximization step of EM we need to determine θ' that maximizes $l_b(\theta|D)$ when q is fixed. The lower bound l_b can be re-expressed as

$$l_b(\theta|D) = \overbrace{\log p(\theta) + \sum_i \sum_{\mathbf{h}\in\mathcal{H}^i} q(\mathbf{h}|\mathcal{F}_i) \log p(\mathcal{F}_i, \mathbf{h}|\theta)}^{Q(\theta|D)} \quad (4.16)$$

$$+ \underbrace{\sum_i \sum_{\mathbf{h}\in\mathcal{H}^i} q(\mathbf{h}|\mathcal{F}_i) \log q(\mathbf{h}|\mathcal{F}_i)}_{H(D)} \quad (4.17)$$

It is easy to see that maximizing l_b is equivalent to maximizing $Q(\theta|D)$ since the term H is only dependent upon the data D and not the model parameters. $Q(\theta|D)$ can be rewritten as

$$Q(\theta|D) = \log p(\theta) + \sum_i \sum_{\mathbf{h}\in\mathcal{H}} q(\mathbf{h}|\mathcal{F}^i)(\log p(\mathcal{F}^i|\mathbf{h}, \theta) + \log p(\mathbf{h}|\theta)) \quad (4.18)$$

$$= \log p(\theta) + \sum_i \sum_{\mathbf{h}\in\mathcal{H}} q(\mathbf{h}|\mathcal{F}^i)(\log p(\mathbf{A}^i|\mathbf{h}) + \log p(\mathbf{X}^i|\mathbf{h}) + \log p(\S^i|\mathbf{h}) + \log p(\mathbf{h}))$$

In order to find the set of parameters that maximizes Q, we take the derivative with respect to the parameters, set this to 0, and solve for the parameters. Since each of the distributions are Gaussian, except for $p(\mathbf{h})$, it turns out that the parameter updates for the means are basically weighted averages of the data, and that the variances are the weighted average of the variance of the data. The weights are the probability that each feature is a manifestation of a part.

In order to describe the parameter updates a few terms need to be introduced. Let $\mathcal{H}_p^i$ be the set of hypotheses where for every $\mathbf{h} \in \mathcal{H}_p^i$ there exists $f \in \mathcal{F}^i$ such that $\mathbf{h}(p) = f$. In other words $\mathcal{H}_p^i$ is the set of hypotheses that have some part that matches p.

4.1.3 Appearance updates

For part p, if we wish to obtain the parameters $\boldsymbol{\mu}_p^{app}$ then we first take the derivative of Q with respect to $\boldsymbol{\mu}_p^{app}$

$$\frac{dQ}{d\boldsymbol{\mu}_p^{app}} = \frac{\sum_i \sum_{h \in \mathcal{H}_p^i} q(\mathbf{h}|\mathcal{F}^i) \log p(\mathbf{A}^i|\mathbf{h})}{d\boldsymbol{\mu}_p^{app}} \tag{4.19}$$

$$= \sum_i \sum_{\mathbf{h} \in \mathcal{H}_p^i} q(\mathbf{h}|\mathcal{F}^i)(\boldsymbol{\mu}_p^{app} - \mathbf{A}_{\mathbf{h}(p)}) \tag{4.20}$$

Now if we set this to zero we can obtain the maximum since Q is a convex function.

$$0 = \sum_i \sum_{\mathbf{h} \in \mathcal{H}_p^i} q(\mathbf{h}|\mathcal{F}^i)(\boldsymbol{\mu}_p^{app} - \mathbf{A}_{\mathbf{h}(p)}) \Leftrightarrow \tag{4.21}$$

$$\boldsymbol{\mu}_p^{app} = \sum_i \sum_{h \in \mathcal{H}_p^i} \frac{q(h|\mathcal{F}^i)\mathbf{A}_{\mathbf{h}(p)}}{Z_p(D)} \tag{4.22}$$

where $Z_p(D) = \sum_i \sum_{h \in \mathcal{H}_p^i} q(\mathbf{h}|F^i)$. For the variances $(\boldsymbol{\sigma}_p^{app})^2$ we can perform a similar operation.

$$\frac{dQ}{d\boldsymbol{\sigma}_p^{app}} = \sum_i \sum_{\mathbf{h} \in \mathcal{H}_p^i} -\frac{q(\mathbf{h}|\mathcal{F}^i)}{\boldsymbol{\sigma}_p^{app}} + q(\mathbf{h}|\mathcal{F}^i)\frac{(\boldsymbol{\mu}_p^{app} - \mathbf{A}_{\mathbf{h}(p)})^2}{(\boldsymbol{\sigma}_p^{app})^3} \tag{4.23}$$

Setting this to 0 we get

$$(\boldsymbol{\sigma}_p^{app})^2 = \sum_i \sum_{\mathbf{h} \in \mathcal{H}_p^i} \frac{q(\mathbf{h}|\mathcal{F}^i)(\boldsymbol{\mu}_p^{app} - \mathbf{A}_{\mathbf{h}(p)})^2}{Z_p(D)} \tag{4.24}$$

4.1.4 Location updates

For a particular matching $\mathbf{h}$ we fix $\mathbf{c_h}$ before determining the parameters that maximize Q. That is, we are going to set $\mathbf{Y}_{f,\mathbf{h}} = \mathbf{S}_{\mathbf{c_h}}\mathbf{X}_f + \mathbf{X}_{\mathbf{c_h}}$, where $\mathbf{Y}_{f,\mathbf{h}}$ is the projection of $\mathbf{X}_f$ into model coordinate space with matching $\mathbf{h}$. Now if we note that $p(\mathbf{Y}_{f,\mathbf{h}}|\mathbf{h}(p) = f) = p(\mathbf{X}_f|\mathbf{h}(p) = f, \mathbf{c_h})$, then we can replace the probabilities over $\mathbf{X}$ in Q to probabilities over $\mathbf{Y}$.

Now if we want to maximize Q with respect to $\boldsymbol{\mu}_p^{loc}$, then we take the derivative

$$\frac{dQ}{d\boldsymbol{\mu}_p^{loc}} = \frac{d\sum_i \sum_{\mathbf{h}\in\mathcal{H}_p^i} q(\mathbf{h}|\mathcal{F}^i)\log p(\mathbf{X}^i|,\mathbf{c}_\mathbf{h})}{d\boldsymbol{\mu}_p^{loc}} \tag{4.25}$$

$$= \frac{d\sum_i \sum_{\mathbf{h}\in\mathcal{H}_p^i} q(\mathbf{h}|\mathcal{F}^i)\log p(\mathbf{Y}_{\mathbf{h}(p),\mathbf{h}}|\mathbf{h})}{d\boldsymbol{\mu}_p^{loc}} \tag{4.26}$$

$$= \sum_i \sum_{\mathbf{h}\in\mathcal{H}_p^i} q(\mathbf{h}|\mathcal{F}_i)(\boldsymbol{\mu}_p^{loc} - Y_{\mathbf{h}(p),\mathbf{h}}) \tag{4.27}$$

Which results in the parameter update if we set the above to 0,

$$\boldsymbol{\mu}_p^{app} = \sum_i \sum_{h\in\mathcal{H}_p^i} \frac{q(h|\mathcal{F}^i)\mathbf{Y}_{\mathbf{h}(p),\mathbf{h}}}{Z_p(D)} \tag{4.28}$$

Now, if we want to maximize Q with respect to $\boldsymbol{\sigma}_p^{loc}$ we first have to describe the prior that we have placed on $\boldsymbol{\sigma}_p^{loc}$. One pitfall to adding parts to the model incrementally is that in some cases there is a bias such that the parts learned earlier will have a smaller variance than parts learned later. If a new part is added to the model, the most likely centre, $\mathbf{c}_{opt}$, according to the model will be determined primarily by the parts that have a small variance. As a result, new parts added to the model will have less influence on where the $\mathbf{c}_{opt}$ should be, and so their location will be noisier.

In order to combat this bias we introduce an inverse Gamma prior on the $\boldsymbol{\sigma}_p^{loc}$ in order to prevent variances from becoming too small. This prior has the form that

$$p(\boldsymbol{\sigma}) \propto (\boldsymbol{\sigma})^{-\beta} e^{-\frac{\beta\gamma}{\sigma^2}} \tag{4.29}$$

We will first derive the parameter updates and then explain the intuition behind the choices for the parameters of γ, and β.

$$\frac{dQ}{d\boldsymbol{\sigma}_p^{loc}} = \frac{\sum_i \sum_{\mathbf{h}\in\mathcal{H}_p^i} q(\mathbf{h}|\mathcal{F}^i)\log p(\mathbf{Y}_{\mathbf{h}(p),\mathbf{h}}|\mathbf{h}) + \log p(\boldsymbol{\sigma}_p^{loc})}{d\boldsymbol{\sigma}_p^{loc}} \tag{4.30}$$

$$= \sum_i \sum_{\mathbf{h}\in\mathcal{H}_p^i} -\frac{q(\mathbf{h}|\mathcal{F}^i)}{\boldsymbol{\sigma}_p^{loc}} + q(\mathbf{h}|\mathcal{F}^i)\frac{(\boldsymbol{\mu}_p^{loc} - \mathbf{Y}_{\mathbf{h}(p),\mathbf{h}})^2}{(\boldsymbol{\sigma}_p^{loc})^3} - \frac{\beta}{\boldsymbol{\sigma}_p^{loc}} + \frac{\gamma}{(\boldsymbol{\sigma}_p^{loc})^3} \tag{4.31}$$

Setting this to 0 and solving for the parameter we get

$$(\boldsymbol{\sigma}_p^{loc})^2 = \sum_i \sum_{\mathbf{h}\in\mathcal{H}_p^i} \frac{q(\mathbf{h}|\mathcal{F}^i)(\boldsymbol{\mu}_p^{loc} - \mathbf{Y}_{\mathbf{h}(p),\mathbf{h}})^2 + \beta\gamma}{Z_p(D) + \beta} \tag{4.32}$$

In essence, the choice of γ is prior belief of what the variance on the location of a part should be, and β is the strength of that belief. These parameters essentially add a set of β pseudo-data that are γ in squared distance from the mean $\boldsymbol{\mu}_p^{loc}$. The particular choices of these parameters were set by experimentation.

4.1.5 Scale updates

Similar to the location updates, we fix $\mathbf{c_h}$ for $\mathbf{h}$ before parameter updates. So, we set $\boldsymbol{\xi}_{f,\mathbf{h}} = \mathbf{S}_{\mathbf{c_h}}\mathbf{S}_f$.

$$\frac{dQ}{d\sigma_p^{scale}} = \frac{d\sum_i \sum_{\mathbf{h}\in\mathcal{H}_p^i} q(\mathbf{h}|\mathcal{F}^i)\log p(\mathbf{S}^i|\mathbf{h})}{d\sigma_p^{scale}} \tag{4.33}$$

$$= \frac{d\sum_i \sum_{\mathbf{h}\in\mathcal{H}_p^i} \log p(\boldsymbol{\xi}_{\mathbf{h}(p),\mathbf{h}}|\mathbf{h})}{d\sigma_p^{scale}} \tag{4.34}$$

$$= \sum_i \sum_{\mathbf{h}\in\mathcal{H}_p^i} q(\mathbf{h}|\mathcal{F}^i)((\boldsymbol{\xi}_{\mathbf{h}(p),\mathbf{h}} - \mu_p^{scale}) \tag{4.35}$$

Setting this to 0 and solving we get

$$\mu_p^{scale} = \sum_i \sum_{\mathbf{h}\in\mathcal{H}_p^i} \frac{q(\mathbf{h}|\mathcal{F}^i)\boldsymbol{\xi}_{\mathbf{h}(p),\mathbf{h}}}{Z_p(D)} \tag{4.36}$$

For the variances $(\sigma_p^{scale})^2$ we can do the same, arriving at the parameter updates

$$\sigma_p^{scale} = \sum_i \sum_{\mathbf{h}\in\mathcal{H}_p^i} \frac{q(\mathbf{h}|\mathcal{F}^i)(\boldsymbol{\xi}_{\mathbf{h}(p),\mathbf{h}} - \mu_p^{scale})^2}{Z_p(D)} \tag{4.37}$$

4.1.6 Part weight updates

If we seek to determine the parameters w_p for $p(\mathbf{h})$ take the derivative of Q with respect to w_p.

$$\frac{dQ}{dw_p} = \frac{d\sum_i \sum_{\mathbf{h}\in\mathcal{H}} q(\mathbf{h}|\mathcal{F}^i)\log p(\mathbf{h})}{dw_p} \tag{4.38}$$

$$= \sum_i \sum_{\mathbf{h}\in\mathcal{H}|h(p)=0} \frac{q(\mathbf{h}|\mathcal{F}^i)}{1-w_p} + \sum_{\mathbf{h}\in\mathcal{H}|h(p)\neq 0} \frac{q(\mathbf{h}|\mathcal{F}^i)}{w_p} \tag{4.39}$$

Setting this to 0 results in the parameter updates

$$w_p = \sum_i \sum_{\mathbf{h}\in\mathcal{H}|h(p)\neq 0} \frac{q(\mathbf{h}|\mathcal{F}^i)}{Z(D)} \tag{4.40}$$

4.2 Incremental learning

Learning a model with random initialization with all P parts is generally not tractable for models with a large number of parts. The primary innovation in this thesis is the development of an incremental scheme to adding parts to the model to make learning models with a large number of parts tractable.

Now if we have an accurate model with $p-1$ parts, then if we add a new part, it is assumed that the parts we have learned so far will remain roughly the same. For example, if we have a model for motorbikes with parts that correspond to the front and back wheel, then adding a new part is unlikely to change the first 2 parts significantly enough that they no longer correspond to the front and back wheel. With this insight, learning a model with 3 parts, given the model with 2 parts, basically reduces to learning the parameters for the new part. The combinatorics of EM is reduced primarily because when we add a new part p we already have a good indication of which features match the first $p-1$ parts. As a result, we know which $h \in \mathcal{H}$ are irrelevant, which is generally a vast majority. Moreover, we also have a good estimate for the object's centre and scale, $\bar{\mathbf{c}}$, which is the average object centre over all matchings

$$\bar{\mathbf{c}} = \sum_{\mathbf{h}\in\mathcal{H}} p(\mathbf{h}|\mathbf{A},\mathbf{X},\mathbf{S})\mathbf{c}_\mathbf{h} \tag{4.41}$$

To add a new part to the model we have a set of potential parts $\mathcal{J}$, and for each $j \in \mathcal{J}$ we have an estimate for the appearance distribution, that is we have mean $\boldsymbol{\mu}_j^{app}$ and a variance $\boldsymbol{\sigma}_j^{app}$ for the potential part. The following algorithm describes how we add a new part to the model. To set $\boldsymbol{\mu}_p^{scale}$ and $\boldsymbol{\mu}_p^{loc}$ we find the image that j

originated from and use the image center $\bar{\mathbf{c}}$ from that image to find an approximate location and scale in model space for the part j. We set $\boldsymbol{\sigma}_p^{loc}$ to be the average variance of parts thus far, and we set $\boldsymbol{\sigma}_p^{scale}$ to be the scale of the part in model coordinate since we found that this generally varies with the scale of the part.

Adding a new part

1. For each $j \in \mathcal{J}$
   (a) Initialize a part p
   - Set appearance mean to $\boldsymbol{\mu}_p^{app} = \boldsymbol{\mu}_j^{app}$ and variance to $\boldsymbol{\sigma}_p^{app} = \boldsymbol{\sigma}_j^{app}$
   - Find the image that the sample came from, determine the expected object centre, $\bar{\mathbf{c}}$, and set $\boldsymbol{\mu}_p^{loc} = \mathbf{S}_{\bar{\mathbf{c}}}\mathbf{X}_f + \mathbf{X}_{\bar{\mathbf{c}}}$. Set the $\boldsymbol{\sigma}_p^{loc}$ to be the average variance of the parts learned thus far.
   - Using $\bar{\mathbf{c}}$, set $\boldsymbol{\mu}_p^{scale} = \mathbf{S}_{\mathbf{c}}\mathbf{S}_f$ and set $\boldsymbol{\sigma}_p^{scale} = \mathbf{S}_{\mathbf{c}}\mathbf{S}_f$.
2. Determine the increase in the likelihood of the data with this part. That is calculate $K_j = p(D|\theta)$ with this part included as part of the model.
3. Initialize the new part of the model to be the part that produced the maximum likelihood K_j

Initially we only have a small number of parts in the model, and as a result the model will only be able to recognize only a fraction of the dataset. For example, if we have a model with 3 parts, for many images in the dataset these parts may note be present, or the feature detector may not have produced a feature at these parts. However, as we add new parts to the model, more and more images in the dataset will be recognized by the model. By adding a new part that produces the maximum likelihood, we usually add that part that occurs in the most images in the data set. The bias that this creates is that if the dataset is divided into distinct subsets, then it is possible that this method may learn only one subset. Dealing with this issue will be left for future work.

4.3 Intelligent initialization for models and parts

In order to be able to learn the model incrementally we need a small model to start with, and we also need a set of potential parts that can be added to the model. The parts of the small model and the set of potential parts are all selected using image matching in the dataset. In image matching we match the features in one image to the features in another image based upon appearance, and remove those that don't agree geometrically. If there are enough features still left, we assume the match is correct, and we retain the features involved in that matching. The small model is chosen to be from the image that had the highest number of matches to other images in the dataset. The parts that are selected from this image are those features that were involved in the most matches to other images. To select the sample set of parts, we simply take those features that are involved in a high number of matches. An example of this can be found in Figure 4.1

4.3.1 Image matching

If two images contain the same object class, then it may be the case that they have features in common. In the image matching algorithm presented, the intent is to find those features in one image that match to the features in the other image. We obviously want to maximize the number of matches, but we also want to minimize the number of false matches since these will introduce additional noise into the initializations. In order to achieve this end we first match features in terms of appearance, and then find the subset of matches that agree geometrically.

To better explain the procedures some additional notation must be introduced. For images i and image j, we denote a $m(f^i) = f^j$ to be a match from $f^i \in \mathcal{F}^i$ to $f^j \in \mathcal{F}^j$, where $m(f^i) = f^j$ if $(\mathbf{A}_{f^i} - \mathbf{A}_{f^j})^2 \leq (\mathbf{A}_{f^i} - \mathbf{A}_g)^2$ for all $g \in \mathcal{F}^j$. We denote $\mathcal{G}$ as the set of features from image i in a match.

We say that the match agrees geometrically if the best least squares linear transformation has an error less than $\psi|\mathcal{G}|$. The best least square linear transformation is the set of parameters (a, s) that minimizes the error

$$E_{\mathcal{G}}(a, s) = \sum_{f^i \in \mathcal{G}} \left(\frac{X_{f^i}}{s} + a - X_{m(f^i)} \right)^2 \tag{4.42}$$

In essence it means that we find the best linear transformation of feature locations from image i to the image coordinate system of j, and then measure the error by

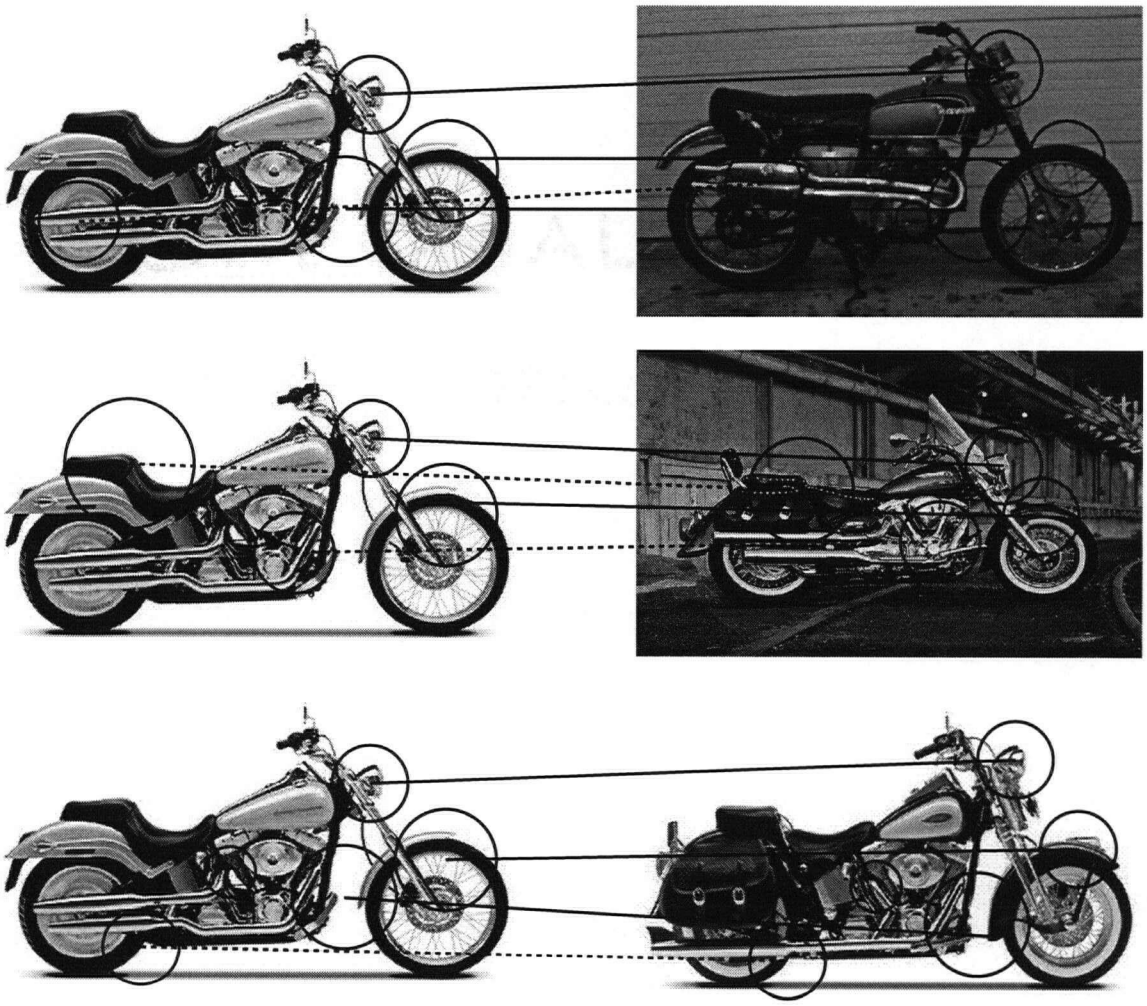


Figure 4.1: Each pair of images represents the final image matching, after geometric constraints have removed false matches. We want features that occur frequently in the dataset both for potential parts and for initializing a small model. In this case, those features connected by the solid lines would be good potential parts, where as those with the dotted lines occur infrequently.

squared distance from each feature to its match in the image coordinate system of j. For notational convenience we denote this error as $E_{\mathcal{G}} = minE_{\mathcal{G}}(a, s)$.

In addition we denote $M_{i,f}$ as the set of features from other images that are judged to be a match to $f \in \mathcal{F}^i$. This is not used directly in the image matching, but is used later for initializing the model and initializing the set of potential parts.

To determine an image matching between i and j

1. Find the closest matches in appearance for each feature in $f^i \in \mathcal{F}^i$ to a feature in $f^j \in \mathcal{F}^j$. That is find $m(f^i)$
2. Initialize $\mathcal{G}_{opt} = \emptyset$
3. For N iterations
   (a) Randomly select a set of 3 features in F^i, denote the set as $\mathcal{G}$
   (b) While $E_{\mathcal{G}} > \psi|\mathcal{G}|$
   (ie. add the best features to $\mathcal{G}$ until the geometric error is too great)
      i. For each feature f^i, let $\mathcal{G}_{f^i} = \mathcal{G} \cup f^i$ and determine $E_{\mathcal{G}_{f^i}}$
      ii. Select that feature f^i that has minimum $E_{\mathcal{G}_{f^i}}$ and add it to $\mathcal{G}$, that is $\mathcal{G} = \mathcal{G} \cup f^i$
   (c) If $|\mathcal{G}_{opt}| < |\mathcal{G}|$ then let $\mathcal{G}_{opt} = \mathcal{G}$
4. If $|\mathcal{G}_{opt}| > k$ then for each $f^i \in \mathcal{G}_{opt}$, $M_{i,f^i} = M_{i,f^i} \cup m(f^i)$
5. If $|G_{opt}| > k$ return $\mathcal{G}_{opt}$, otherwise there is no matching.

The parameter ψ is set by hand, and is meant to represent how far away on avera a feature $f^i \in \mathcal{G}$ can be from $m(f^i)$ in the best linear transformation to the image coordinate space of j. In practise we set it ψ to be about 0.1 times the minimum dimension of the image, assuming that the object takes up about 1/3 to 2/3 of the image.

Clearly it is also possible that the matching found is a false match, and the fewer number of features in $\mathcal{G}_{opt}$, the more likely it is that we have a false match. This is where the parameter k comes in, for it determines when there is a matching. In the experiments we set $k = 4$, which was determined experimentally to prevent most

false matchings between images from affecting the initialization of the potential parts or initializing the small model. For values of $k < 4$ we found there were many more image matchings where the matching was incorrect.

4.3.2 Initializing a small model and a sample set

To initialize the sample set $\mathcal{J}$ we match all images to each other and extract those features that were involved in the most matches to other images. By this we mean that we use those features that had a high number of both geometric and appearance matches to features in other images. Once we have these features we find the average appearance of these features across the other features they matched, and use these to initialize both $\boldsymbol{\mu}_p^{app}$ and $\boldsymbol{\sigma}_p^{app}$ for a potential part p. The motivation behind choosing these features is that these features are probably the most common and are likely to be part of a good model anyway.

To initialize a small model with δ, we first find that image i that matched the most other images in the dataset. At this point we select the top δ features from that image that were involved in the most matches to other images. These features are then used to initialize a small model. Similar to initializing potential parts for $\mathcal{J}$, we also find the average appearance of these features, across the other features they matched in other images, and use this to initialize both the appearance $\boldsymbol{\mu}_p^{app}$ and $\boldsymbol{\sigma}_p^{app}$. The reason that we choose one image and use the features only from that image is that it allows us to initialize both the appearance $\boldsymbol{\mu}_p^{app}$,$\boldsymbol{\sigma}_p^{app}$, and the location $\boldsymbol{\mu}_p^{loc}$, and scale μ_p^{scale} for a part p. Otherwise we would not have a coordinate system to initialize the parameters for location and scale.

Initializing a small model and a sample set $\mathcal{J}$

1. For each image i (or a subset)
   (a) Initialize $M_{i,f^i} = \emptyset$ for every feature $f^i \in \mathcal{F}^i$
   (b) For every other image j (or subset) determine $\mathcal{G}_{opt}$ by the matching procedure described above
2. Find the η features that are involved in the most matches and put in set $\mathcal{J}$
3. For each of $f^i \in \mathcal{J}$, where i is the image f comes from
   - Set $\boldsymbol{\mu}_{fi}^{app} = \frac{\sum_{f \in M_{i,f^i}} A_f}{|M_{i,f^i}|}$, the average appearance of features in M_{i,f^i}
   - Set $\boldsymbol{\sigma}_{fi}^{app} = \frac{\sum_{f \in M_{i,f^i}} (A_f - \boldsymbol{\mu}_i^{app})^2}{|M_{i,f^i}|}$
4. Select that image i that matched the most other images
5. For δ top features $f^i \in \mathcal{F}^i$ that are involved in the highest number of matches, $|M_{i,f^i}|$
   (a) Set $\boldsymbol{\mu}_{fi}^{app} = \frac{\sum_{f \in M_{i,f^i}} \mathbf{A}_f}{|M_{i,f^i}|}$, the average appearance of features in M_{I,f^i},
   (b) Set $\boldsymbol{\sigma}_{fi}^{app} = \frac{\sum_{f \in M_{i,f^i}} (\mathbf{A}_f - \boldsymbol{\mu}_{fi}^{app})^2}{|M_{i,f^i}|}$
   (c) Set $\boldsymbol{\mu}_{fi}^{loc} = \mathbf{X}_{f^i}$
   (d) Set $\boldsymbol{\sigma}_{fi}^{loc} = \psi$
   (e) Set $\mu_{fi}^{scale} = \mathbf{S}_{f^i}$
   (f) Set $\sigma_{fi}^{scale} = \mathbf{S}_{f^i}$
6. Return these top features as the parts for a small model

Chapter 5

Model Computation

In general, calculating $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$ exactly is intractable for a large number of parts as mentioned previously. There are, however, a variety of techniques that can be utilized to approximate $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$.

In general, if we have an accurate model of an object class then we expect that there is only one true matching $\mathbf{h}$, and one true center $\mathbf{c}$ for the object in an image. As a result, there will generally be very few matchings that contribute to $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$. In practice this is also the case, and usually there is only one matching $\mathbf{h}$ that contributes significantly to $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$. So, the task in object classification is to find these few matchings. In this chapter, we outline a few techniques to find the most likely matchings.

5.1 False matches and enforcing geometry

The density $p(\mathbf{X}, \mathbf{S}|\mathbf{h}) \approx \max_{\mathbf{c}} p(\mathbf{X}, \mathbf{S}, \mathbf{c}|\mathbf{h})$ as defined earlier is unconstrained. However, if we impose one constraint on this probability then computation is much more tractable. The constraint is that if

$$p(\mathbf{X}_f, \mathbf{S}_f|\mathbf{h}(p) = f) < p(\mathbf{X}_f, \mathbf{S}_f|\mathbf{h}_\emptyset) \tag{5.1}$$

for some f and p, then we set $p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h}) = 0$. What this means is that if f is not at all where part p should be according to the object model, then we discard the matching. The reason for making this restriction is two-fold. The first is that it can help reduce the amount of computation to approximate $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$ significantly. The second is that for high dimensional appearance vectors $\mathbf{A}_f$, the distribution for $p(\mathbf{A}_f|\mathbf{h})$ can range over a much larger range of numbers than $p(\mathbf{X}, \mathbf{S}|\mathbf{h})$, and as a result $P(\mathbf{A}|\mathbf{h})$ has a much greater influence on $p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h})$ than $p(\mathbf{X}, \mathbf{S}|\mathbf{h})$. One way to avoid false matches due to this is to enforce the constraint that for each feature f matched to a part, the feature's location should agree at least in some degree to where the part should be according to the object model. For example, consider a

theoretical object model for cars, which has two parts that correspond to the wheels of the car. If an image with two wheels in random locations is given to the system, this constraint would prevent a false match. The problem with imposing such a constraint is that $p(\mathbf{X}_f, \mathbf{S}_f|\mathbf{h})$ is no longer a probability distribution. In practice, however, this constraint rarely affects the end evaluation since $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$ relies primarily upon only a few matchings, and these generally satisfy this constraint.

It should be mentioned however, that this constraint would have a significant effect upon learning if the parts of the model were initialized randomly. The reason for this is that if the model is not at all accurate, then many of the matchings that do in fact contribute to $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$ have no influence in the parameter updates because of this additional constraint. However, in the learning scheme presented, it is never the case that a part is initialized randomly, so the above approximation is reasonable.

5.2 Search space techniques

Given a set of features $\mathcal{F}$, we can represent the hypothesis space $\mathcal{H}$ as a tree, where each level is a part, and nodes on level p represent a match from a feature to part p. Given this tree we can do a depth first search through the tree to determine $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$, trimming branches that are unlikely to contribute to $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$ and exploring those that are likely to contribute to this probability. Figure 5.1 illustrates the search space and the trimming techniques. One basic trimming technique is to not explore further down a branch in instances when $p(\mathbf{X}_f, \mathbf{S}_f|\mathbf{h}(p) = f) < p(\mathbf{X}_f, \mathbf{S}_f|\mathbf{h}_\emptyset)$, which is trimming based upon geometry. Another basic trimming technique, based upon appearance, is to not explore further down a branch in instances when

$$p(\mathbf{h}(p) = f)p(\mathbf{A}_f|\mathbf{h}(p) = f) < p(\mathbf{A}_f, \mathbf{X}_f, \mathbf{S}_f|f \in \mathcal{B})p(\mathbf{h}(p) = 0) \tag{5.2}$$

In both of these cases it is always better that the feature f be assigned to the background, that is $\mathbf{h}(p) = 0$. This will be explained in Section 5.2.1.

The technique presented in Figure 5.2 to approximate $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$, computes $p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h})$ for the top matchings $\mathbf{h}$, neglecting many matches that would not contribute to $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$. In practice, computing the above approximation $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$ grows roughly linearly with the number of parts in the model. For example, for a model with 5 parts it takes much less than a second, but with a model with 25 parts, it takes between 2 and 10 seconds. There are additional techniques that can be utilized to find only the optimal matching $\mathbf{h}_{opt}$ in much less than a second for even models with large parts.

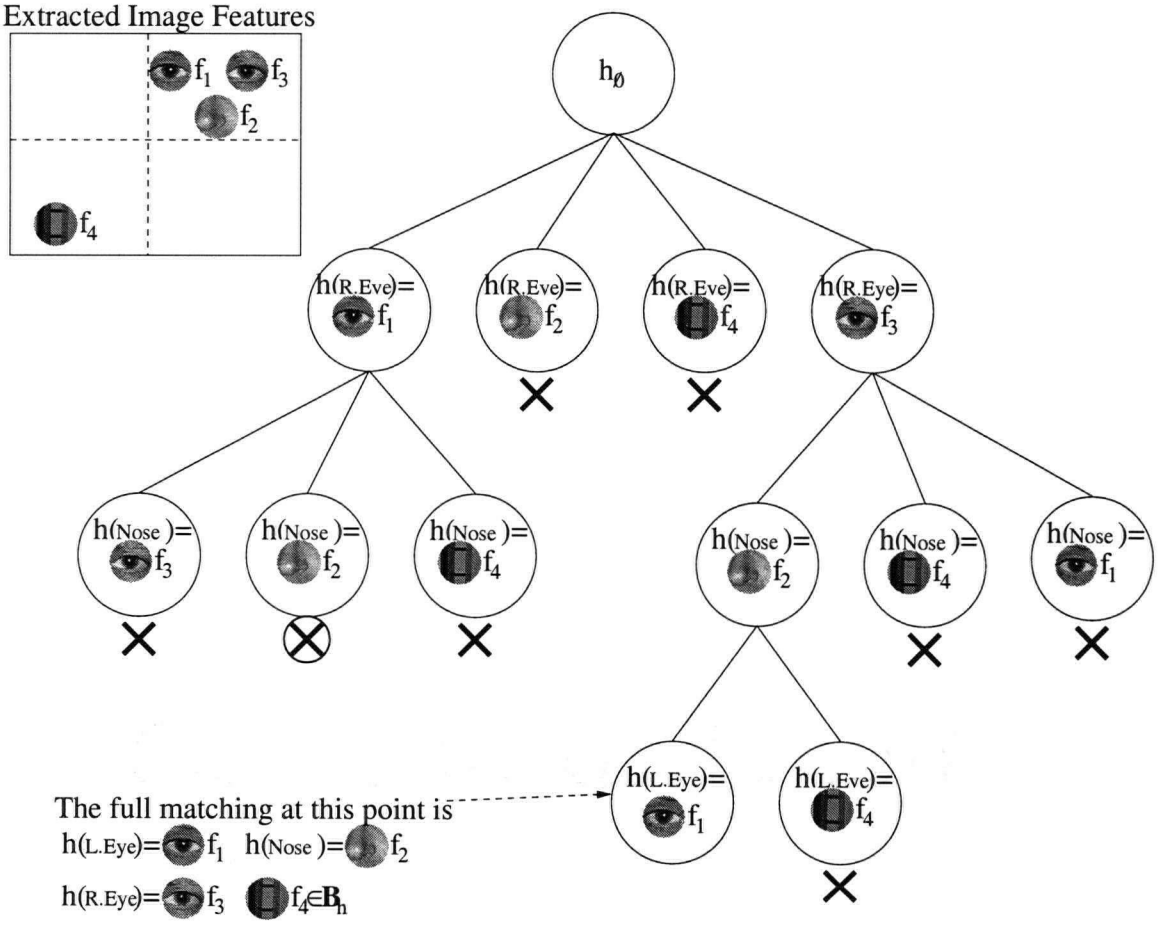


Figure 5.1: This diagram illustrates an example of the search space and the trimming techniques with a face model. A matching at a particular node is defined by the path from the root to the node, where the root assigns all parts to the background, and each node assigns a part to feature. ⊗ are cuts that have been made because of geometry, and X are cuts that have been made because of appearance.

Approximating $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$

1. Let $\mathbf{h}_r$ be the matching where $\mathbf{h}_r(p) = 0$ for all parts p
2. $SumAll = 0$
3. For every $f \in \mathcal{F}$
   (a) $SumAll = SumAll + computeChild(\mathbf{h}_r, 1, f)$
4. $p(\mathbf{A}, \mathbf{X}, \mathbf{S}) \approx SumAll$

Function $computeChild(\mathbf{h}, p, f)$

1. Let $Sum_{\mathbf{h}'} = 0$, $\mathbf{h}' = \mathbf{h}$, and set $\mathbf{h}'(p) = f$
2. If $p(\mathbf{h}'(p) \neq 0)p(\mathbf{A}_f|\mathbf{h}'(p) = f) < p(\mathbf{A}_f, \mathbf{X}_f, \mathbf{S}_f|f \in \mathcal{B})p(\mathbf{h}(p) = 0)$ return 0 (Trimming based upon appearance)
3. If $p(\mathbf{X}_f, \mathbf{S}_f|\mathbf{h}') < p(\mathbf{X}_f, \mathbf{S}_f|\mathbf{h}_\emptyset)$ return 0 (Trimming based upon geometry)
4. If $p < P$ then (Descending down child)
   - For every $f' \in \mathcal{B}_{\mathbf{h}'}$, the features matched to the background in $\mathbf{h}'$
     - $Sum_{\mathbf{h}'} = Sum_{\mathbf{h}'} + computeChild(\mathbf{h}', p + 1, f')$
5. return $p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h}')p(\mathbf{h}') + Sum_{\mathbf{h}'}$

Figure 5.2: Pseudo algorithm for approximating $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$

In practice, approximating $p(\mathbf{A}, \mathbf{X}, \mathbf{S}) \approx p(\mathbf{h}_{opt})p(\mathbf{A}, \mathbf{X}, \mathbf{S}|h_{opt})$ is accurate enough to obtain the same recognition results as using the full technique described above. In learning, however, we need $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$ to be as accurate as possible since EM requires $p(\mathbf{h}|\mathbf{A}, \mathbf{X}, \mathbf{S})$ for multiple $\mathbf{h}$ for parameter updates, so such additional techniques are of little use. In Section 5.3 we discuss how to reuse the search tree from previous iterations of EM to reduce computation.

5.2.1 Trimming the search space

If we have a node n at level p, we know the matching $\mathbf{h}_n$ specified at this node. If $p(\mathbf{X}, \mathbf{S}|\mathbf{h}_n) < p(\mathbf{X}, \mathbf{S}|\mathbf{h}_\emptyset)$, we know that for each matching $\mathbf{h}$ further down the tree that it is always the case that $p(\mathbf{X}_{\mathbf{h}(p)}, \mathbf{S}_{\mathbf{h}(p)}|\mathbf{h}_n) < p(\mathbf{X}_f, \mathbf{S}_f|\mathbf{h}_\emptyset)$ for some p. So, according to our constraint outlined in Section 5.1, all matchings that stem from this node do not contribute to $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$ and thus the subtree does not need to be explored.

To see this, note that when $p(\mathbf{X}, \mathbf{S}|\mathbf{h}_n) < p(\mathbf{X}, \mathbf{S}|\mathbf{h}_\emptyset)$, there must be at least one part p, where $\mathbf{h}_n(p) = f$, such that $p(\mathbf{X}_f, \mathbf{S}_f|\mathbf{h}_n) < p(\mathbf{X}_f, \mathbf{S}_f|\mathbf{h}_\emptyset)$. For the matching $\mathbf{h}_n$, where $\mathcal{K}$ is the set of features assigned to parts, the $\mathbf{c}_{\mathbf{h}_n}$ that determines where $\mathbf{X}_\mathcal{K}$ and $\mathbf{S}_\mathcal{K}$ are transformed into model coordinate space is the best transformation for these features, that is

$$p(\mathbf{X}_\mathcal{K}, \mathbf{S}_\mathcal{K}|\mathbf{h}_n) \approx p(\mathbf{X}_\mathcal{K}, \mathbf{S}_\mathcal{K}|\mathbf{h}_n, \mathbf{c}_{opt}) \tag{5.3}$$

$$\geq p(\mathbf{X}_\mathcal{K}, \mathbf{S}_\mathcal{K}|\mathbf{h}_i, \mathbf{c}) \forall \mathbf{c} \tag{5.4}$$

For each matching $\mathbf{h}$ below node n, it is the case that the matchings for the features in $\mathcal{K}$ are the same as in $\mathbf{h}_n$. As a result, $p(\mathbf{X}_\mathcal{K}, \mathbf{S}_\mathcal{K}|\mathbf{h}_n) \geq p(\mathbf{X}_\mathcal{K}, \mathbf{S}_\mathcal{K}|\mathbf{h})$, and so it is the case that there exists a feature $f \in \mathcal{K}$ such that $p(\mathbf{X}_f, \mathbf{S}_f|\mathbf{h}) < p(\mathbf{X}_f, \mathbf{S}_f|\mathbf{h}_\mathcal{B})$.

Another technique to trim the subtree at a node n is to evaluate the appearance of the feature f given that it is matched to a part p and compare it to the likelihood of the appearance given that the feature is generated by the background. If it is a lot more likely that the feature is generated by the background than by the part, then we do not need to explore the subtree below the node. In particular if

$$p(\mathbf{h}(p) = f)p(\mathbf{A}_f|\mathbf{h}(p) = f) < p(\mathbf{A}_f, \mathbf{X}_f, \mathbf{S}_f|f \in \mathcal{B})p(\mathbf{h}(p) = 0) \tag{5.5}$$

then the subtree can in practice be safely be removed. The reason for this is that for any $\mathbf{h}$ where $\mathbf{h}(p) = f$, $p(\mathbf{X}_f, \mathbf{S}_f|\mathbf{h}) < 1$, since we are using Gaussians and the

variances are never small enough to produce probabilities greater than 1, and as a result

$$\begin{aligned} p(\mathbf{h}(p) = f)p(\mathbf{A}_f|\mathbf{h}(p) = f) &< p(\mathbf{A}_f, \mathbf{X}_f, \mathbf{S}_f|f \in \mathcal{B})p(\mathbf{h}(p) = 0) \\ \Leftrightarrow p(\mathbf{h}(p) = f)p(\mathbf{A}_f, \mathbf{X}_f, \mathbf{S}_f|\mathbf{h}(p) = f) &< p(\mathbf{A}_f, \mathbf{X}_f, \mathbf{S}_f|f \in \mathcal{B})p(\mathbf{h}(p) = 0) \quad (5.6) \end{aligned}$$

This basically means that it is better that the feature be matched to the background in all matchings $\mathbf{h}$ where $\mathbf{h}(p) = f$.

5.3 Search space techniques for EM

In computing the parameters θ for the model, we repeatedly search through the hypothesis space $\mathcal{H}$ for matches that contribute to $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$. If we have $p - 1$ parts in the model, then if we run EM on the parameters then EM will converge to some local maximum after some number of iterations. At this point there is very little change in the parameters, so the matchings that are found to contribute to $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$ on each iteration will essentially be the same. Moreover, when we add a new part p to the model, we don't expect that the matchings for parts $p - 1$ to change a great deal. Given this, we can utilize information from previous iterations of EM in order to reduce the search space.

Let $\phi_{p,f}$ be a boolean variable, where $\phi_{p,f} = TRUE$ indicates that there still remains a matching $\mathbf{h}$ where $\mathbf{h}(p) = f$ and $p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h})$ contributes to $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$. $\phi_{p,f} = FALSE$ indicates that f definitely does not match p. We can use this in the approximating technique in Section 5.2 by checking that $\phi_{p,f} = TRUE$ in $computeChild(\mathbf{h}, p, f)$ and returning 0 immediately if it's false.

From previous iterations, we have determined $p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h})$ for a set of matches $\mathbf{h} \in \mathcal{W}$ that contributed to $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$. Let $\Psi_{p,f} = \sum_{h|h\in\mathcal{W}\ \&\ \mathbf{h}(p)=f} p(\mathbf{h})p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h})$, that is, the total contribution of matching f to p to $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$. Now, if the contribution of matching p to f is sufficiently small then it's unlikely that this match is ever going to contribute to $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$. In particular if

$$\frac{\Psi_{p,f}}{p(\mathbf{A}, \mathbf{X}, \mathbf{S})} < \lambda \quad (5.7)$$

then we set $\phi_{p,f} = FALSE$, that is that in future iterations we do not need to evaluate an matches $\mathbf{h}$ where $\mathbf{h}(p) = f$. This check can be performed at the end of every iteration.

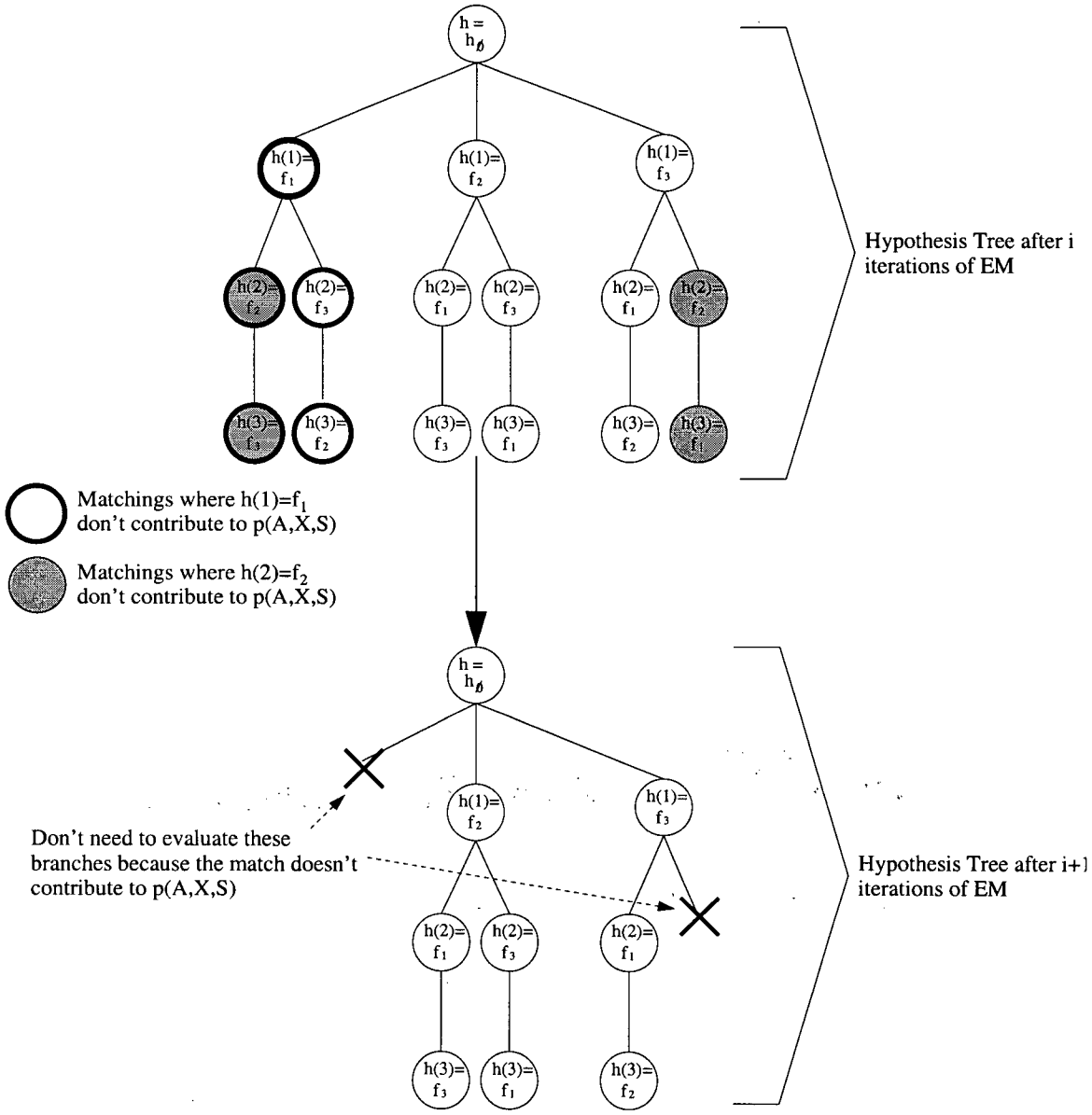


Figure 5.3: This diagram illustrates how we use information from a previous iteration of EM to trim the search space in order to approximate $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$

The question is, how do we set λ? With small models there really is no need to utilize previous iterations to reduce computation. However, for models with more than 10 parts, even the approximation scheme in the previous Section 5.2.1 is costly to compute for some model classes, like motorbikes. The key insight to setting λ is that $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$ is roughly proportional to the number of matches in the optimal match $\mathbf{h}_{opt}$ for an image. If there are a high number of matches in $\mathbf{h}_{opt}$, then it is assumed that the matches from parts to features are correct, and that for any additional parts added to the model, these parts will continue to be matched to the same features in the dominant matching. Clearly we want to reduce computation as much as possible, but we don't want to prematurely exclude possible matches of parts to features since the parameters for parts change on every iteration. So, we set λ to be the average error of optimal matches that contain 4 matches. The intuition is that if the optimal matching $\mathbf{h}_{opt}$ has 6 matches, then matchings $\mathbf{h}$ with only two matches are unlikely to ever contribute to $p(\mathbf{A}, \mathbf{X}, \mathbf{S})$ since we already have a very good matching of parts to features in $\mathbf{h}_{opt}$. In practice it was found that this setting produced the same results as learning without making any use of previous iterations to reduce the search space, except that we could learn large models of 25 parts in 25 minutes instead of 2 or 3 hours.

Chapter 6

Experiments

In order to validate the learning method and the models learned, a variety of experiments have been conducted on 4 different datasets. The first few sections of the chapter describe some parameter choices and the datasets, while later sections describe actual results.

6.1 Feature detectors and descriptors

Up until this point we have not restricted the model to any particular type of feature detector or descriptor. At this point we ask, what regions of the image are interesting? There are a huge variety of interest point detectors in the computer vision literature, some examples being [6][11][9]. However, there are a number of properties that are required for robust general object classification that narrows the number of choices. The first is that the feature detector should be scale and rotation invariant, that is, the same region will be chosen as a feature if the image is scaled or rotated. The particular feature detector we chose for our experiments is that of Lowe [9] , where interest points in this case are located at peaks in the difference of Gaussian function convolved with the image in scale space. In other words, they are located in regions and at scales where there are large gradients in all directions when compared to neighbouring scales and locations. By selecting such regions we are guaranteed to select highly textured regions of the image.

Aside from determining which regions of the image are interesting, there is also the question of how to represent this region. As a first step the region should be blurred and resampled according to the scale so that all feature descriptors are of the same size. The simplest approach from this point would be to simply use the resampled region, but this usually produces a very high dimensional vector. The particular approach that Fergus *et al.* take is to first apply PCA, and represent the region as the first 10-15 principle components.

There have also been a number of feature representations that have been developed by the computer vision community that are invariant to rotation, scale, and brightness

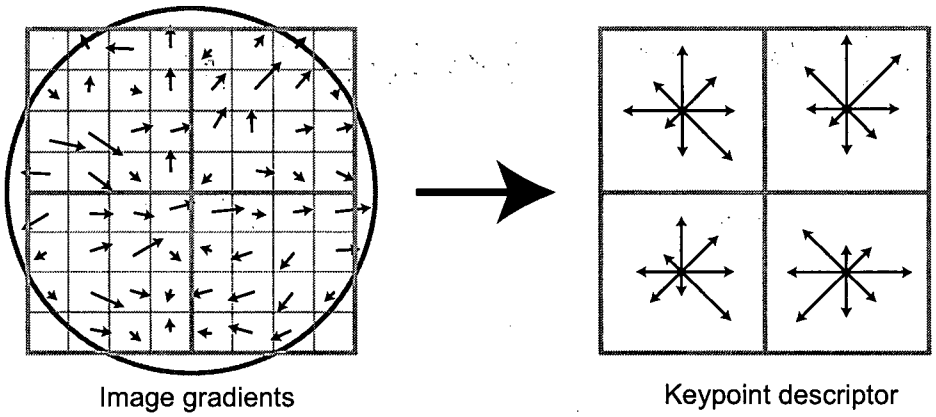


Figure 6.1: Example of a feature descriptor when the image region is divided into 2x2 patches, with the histogram of image gradients in each patch having $K = 8$ bins.

changes, and are less sensitive to changes in 3D viewpoint and noise. The particular approach adopted in this thesis is to use the approach developed by Lowe [9] in order to determine feature vectors $\mathbf{A}_f$. In this approach an image region is first blurred and resampled at the appropriate scale and then transformed into a patch of gradients. From here, the region is divided into *mxm* patches. Each patch is then described as a histogram of image gradients, with K bins. See Figure 6.1 for an example. The advantages of such a representation are that it is invariant to brightness changes, scale, and somewhat invariant to affine distortions. There is the option of making the feature rotationally invariant, but we have dropped this extension for simplicity. We have chosen to divide the image into a 3x3 patch, $m = 3$, and to have 4 gradient bins per patch, $K = 4$. This results in feature vectors $\mathbf{A}_f$ that are 36 dimensional. We have experimented with other parameter choices but these choices tended to produce good results in a reasonable amount of time.

6.2 Experimental setup

Each dataset was split into a test dataset and a training dataset. The training dataset was used in learning an object model and the test dataset was reserved for testing the model. Preprocessing for each of the datasets consisted primarily of extracting features, where an image usually had anywhere from 50 features to 700 features detected. This is very much in contrast to the method proposed by Fergus *et al.*, where the the maximum number of features was 20. Despite this difference, training

our models were an order of magnitude faster.

For each object class model, a set $\mathcal{J}$ of 100 potential parts, and a small initial model with 4 parts are extracted from the training data. Following this preprocessing step, the iterative learning procedure is then used to add parts to the model until the model contains P parts.

6.2.1 Datasets

There are 4 different datasets from Fergus *et al.* on which the method presented in the thesis is validated: frontal views of faces, side views of cars, rear views of cars, and side views of motorbikes.

The face dataset contains 440 views of faces, mostly of the same scale, similar lighting conditions, and varying backgrounds. The dataset only consists of 27 distinct individuals. In learning a model on this dataset, the dataset was randomly split into 300 training images, and 140 test images. This split was sometimes done numerous times to obtain averages.

The motorbike dataset contains 826 images of side views of motor bikes, with varying scale, similar lighting conditions, and varying backgrounds, although most were against a light-coloured background. The test sets consisted of 200 images while the training sets consisted of the remaining 626 images

The rear view car dataset contains 1155 images of rear views of cars, with varying scale, varying lighting conditions, and varying backgrounds, although all were taken on roads. The test sets consisted of 200 images and the training sets consisted of the remaining 955 images.

The main side view car dataset contains 540 images of various cars viewed from side. The scale of the cars were identical, but the background varied. The test sets consisted of 140 images and the training sets consisted of the remaining 400 images. However, for comparison with results presented in Fergus *et al.* there is also an additional test set that contains test images of cars, sometimes more than one. The task with this dataset is to locate the object class, not only classify the image as a car.

There are also two background datasets. The first contains 900 images of various sizes of various scenes. The second set contains 1155 images of various sizes of road scenes without the presences of a car. This is used for recognition results related to the dataset with cars viewed from the rear. In both cases, these datasets were used to compute the false positive rate, which is the percentage of the time that the object

recognition system classified a background image as being part of the object class.

6.3 Computation time required to learn object models

One of primary contributions of this thesis is a method to learn a part based model with a large number of parts quickly. In Fergus *et al.*, models of 6 or 7 parts take 24 to 36 hours to train, and the time to learn the parameters for this method is generally exponential in the number of parts P, so larger models are not feasible.

With the learning procedure described in this thesis, the amount of computation time to learn an object model is roughly linear in the number of parts. The learning is composed of two stages, a preprocessing step that initializes a small model and a set $\mathcal{J}$ of potential parts, and the incremental learning that adds these parts to the model. It generally takes about 10 minutes for the preprocessing step, but it is not optimized, so this compute time can be reduced. In general it takes about 1 to 3 minutes to add a new part to the model and run EM till convergence on the parameters. Learning an object model of 25 parts usually takes 30 minutes or less.

6.4 Results

There are a variety of measures that can be utilized to measure how effective an object model is as an object classifier. Given that we have learned parameters for $p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\theta)$, we can use the ratio $R = p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\theta)/p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h}_\emptyset)$ to recognize an image. If $R > \omega$, where ω is some threshold, then we classify the image as having the object class. If $R < \omega$ then the image does not contain an instance of the object class.

One of the most useful measures for evaluating an object recognition system is the Receiver-Operator curve (ROC), which measures how the true positive rate TP_ω varies with the false positive rate FP_ω when ω is varied. A true positive is when an image containing the object is identified as containing the object class. A false positive, on the other hand, is when an image is identified as containing the object class when it does not. The true positive rate is defined as

$$TP_\omega = \frac{\text{Number of true positives}}{\text{Number of images containing the object class}} \tag{6.1}$$

whereas the false positive rate is defined as

$$FP_\omega = \frac{\text{Number of false positives}}{\text{Number of images not containing the object class}} \tag{6.2}$$

It it fairly common to compare different object classifications systems by utilizing the ROC equal error rate, which is defined to be $TP_\omega = 1 - FP_\omega$ which is determined by varying the threshold ω. This measurement is meant to convey the tradeoff between classifying images containing the object class correctly and falsely classifying images that do not contain the image class as containing the image class.

6.4.1 Comparison to previous methods

Our comparisons are restricted to comparing our results to the results presented by Fergus *et al.*, since few approaches achieve the same amount of success on the datasets we use to validate our approach. The results that they present are primarily ROC equal error rates, with the exception of the dataset of cars viewed from the side.

With the dataset of cars viewed from the side, they utilize a recall-precision curve (RPC), which is similar to the ROC curve. The task, however, is not to classify an image but to locate the object class in the image. So, in order to utilize our model to accomplish this we output all distinct matchings $\mathbf{h}$ and the associated object centers $\mathbf{c}_{opt}$ where $p(\mathbf{A}, \mathbf{X}, \mathbf{S}|\mathbf{h}) > \omega$. A true positive is defined to be the case when $\mathbf{c}$ occurs within some r from an instance of the object class, and a false positive is when $\mathbf{c}$ does not occur within r of an instance of an object class. $Recall_\omega$ is similar to the true positive rate,

$$Recall_\omega = \frac{\text{Number of true positives}}{\text{Number of instances of the object class}} \tag{6.3}$$

The precision is defined to be

$$Precision_\omega = \frac{\text{Number of true positives}}{\text{Number of false positives + true positives}} \tag{6.4}$$

So, given this experimental setup the result reported for the dataset of cars viewed from the side is recall-precision equal error, which is the value of $Recall_\omega$ when $Recall_\omega = 1 - Precision_\omega$. More information on this type of experiment can be found in [1]

By the results in Table 6.1, it is unclear whether the approach proposed in this thesis is any better at object recognition than the one presented by Fergus *et al.*. However, this could very be well an artifact of the datasets. For the datasets in

Dataset	Ours	Fergus *et al.*
Motorbikes	90.2 (20 part model)	92.5 (6 part model)
Faces	94.7 (9 part model)	96.4 (6 part model)
Cars (Side)	90.5 (6 part model)	88.5 (6 part model)
Cars (Rear)	96.2 (20 part model)	90.3 (6 part model)

Table 6.1: ROC equal error results comparing our results to that of Fergus *et al.*

question, it turns out that most of the object classes can be recognized with just a few parts. For example, we found that a two part model with the two eyes was able to produce an ROC equal error rate of 0.8. As a result, learning models with more parts does not dramatically improve recognition results. This issue will be further discussed in the next section.

6.4.2 Incremental learning

The main contribution of this thesis is an approach to learning a part-based object recognition that is able to learn a large number of parts in a reasonable amount of time. To test the effectiveness of the learning approach, we trained models of varying sizes for each object class to determine the improvement or lack of improvement in object recognition results. The ROC curves in Figure 6.2 demonstrate that learning models with more object parts definitely improves results, although the improvement in some cases is not large. These ROC curves are the average obtained over 10 different splits of the dataset into testing set and training set.

Perhaps an easier way to assess how much having larger models improves recognition results is to look at the equal error rate for models of various sizes. In Figure 6.3 we present the average ROC equal error rate for models of various sizes. The first thing to note is that for all of the data sets the recognition results tend to peak at some point and performance tends to degrade or plateau as additional parts are added to the model.

The explanation for this is somewhat complicated but enlightening. An inspection of the datasets reveals that there is a portion of the each dataset that is quite different in character from the average image in the dataset. For example, in the face dataset, about 5 to 10 percent of the images contain an image of a face taken under lighting conditions that differ considerably from the rest of the dataset (see Figure 6.4). These faces appear much darker, and as a result there is very little contrast, which results in fewer feature detections on the face. In addition, the appearance of these features

are significantly different than features in similar regions on the face in the average image in the dataset. Now, in our initializing procedure, we select a sample set $\mathcal{J}$ of potential parts, where a feature can act as a potential part if it is involved in lots of image matches. As a result few potential parts would come from these images, and as a result the learning method would have a bias and thus neglect this segment of the dataset. A similar analysis can be applied to the motorbike dataset. In this case a majority of the motorbikes are against a light background. As a result, many of the parts the learning algorithm learned were on the outside of the motorbike, so they included the lighter blackground. However, for a small portion of the dataset the background is quite dark, and as a result these images are hard to recognize using

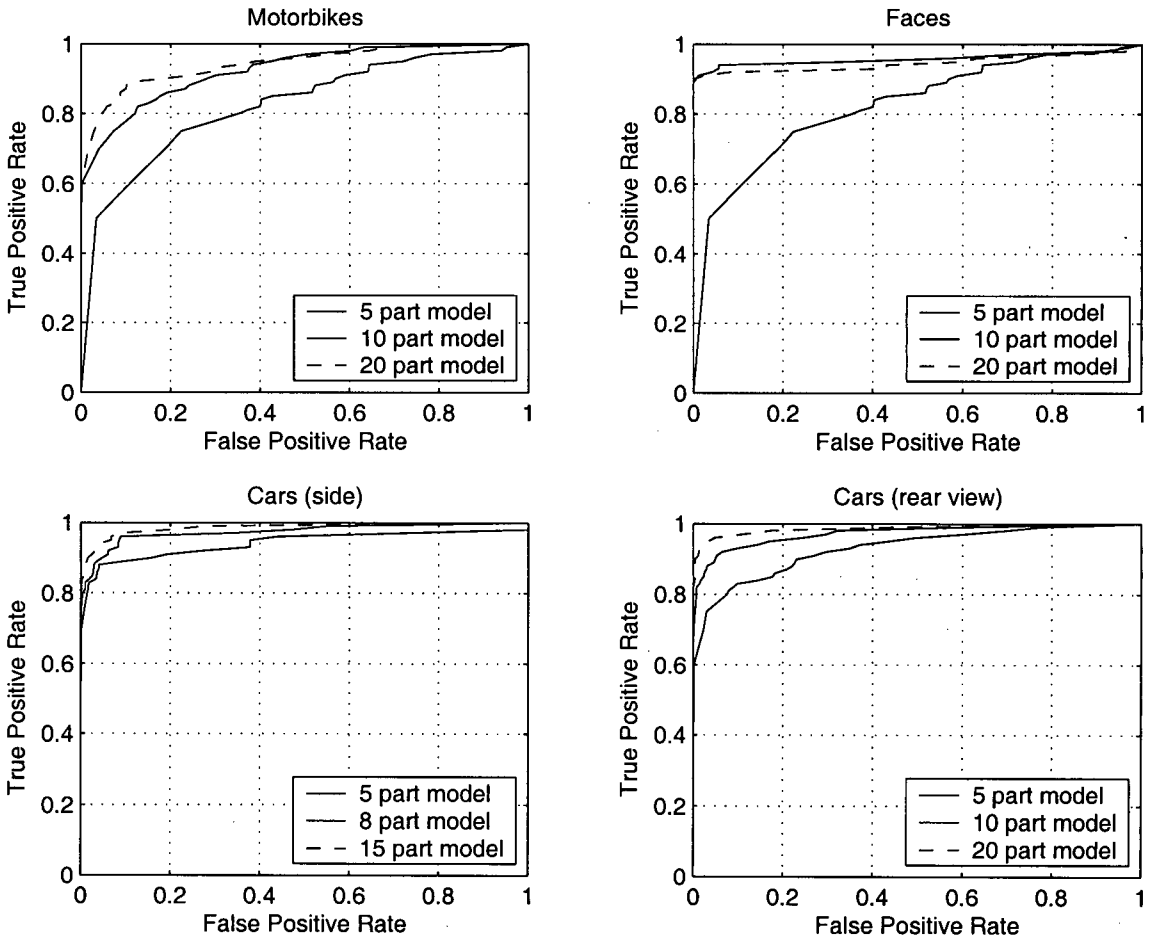


Figure 6.2: These ROC curves demonstrate that as the number of parts increase the ROC curve improves.

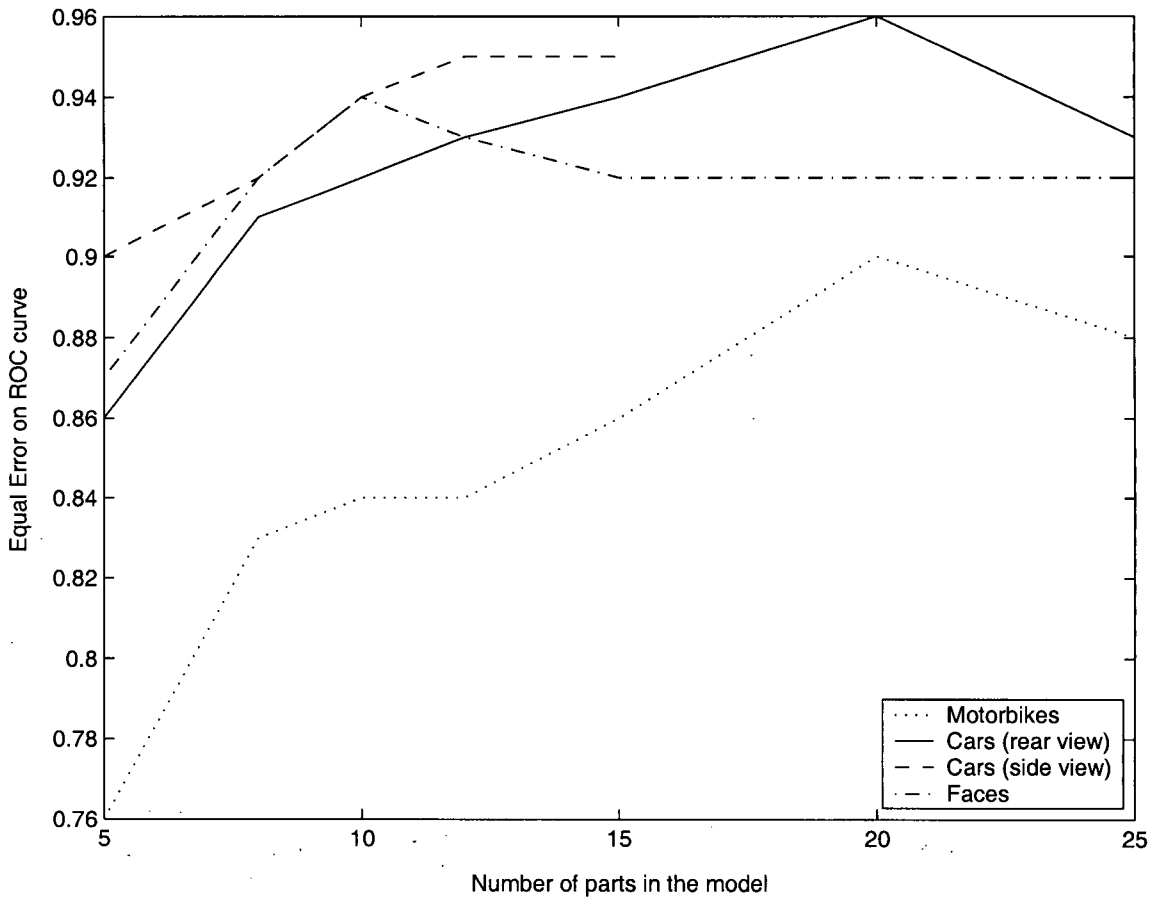


Figure 6.3: This figure outlines how the ROC equal error rate improves as more parts are added to the model. Note that for each model the curve seems to plateau, indicating that results don't improve with larger models. This is somewhat due to the nature of the datasets.

these parts. Moreover, since these images are only a small portion of the dataset, very few parts in the potential part set $\mathcal{J}$ would come from these images. As a result, few parts would ever be added to the motor bike model that could help in modelling these darker images.

6.4.3 Learned models

In Figures 6.5,6.7,6.6, 6.9, 6.8 are visualizations of the models that were learned by the approach outlined in this thesis. In each figure, the model is split into two parts,

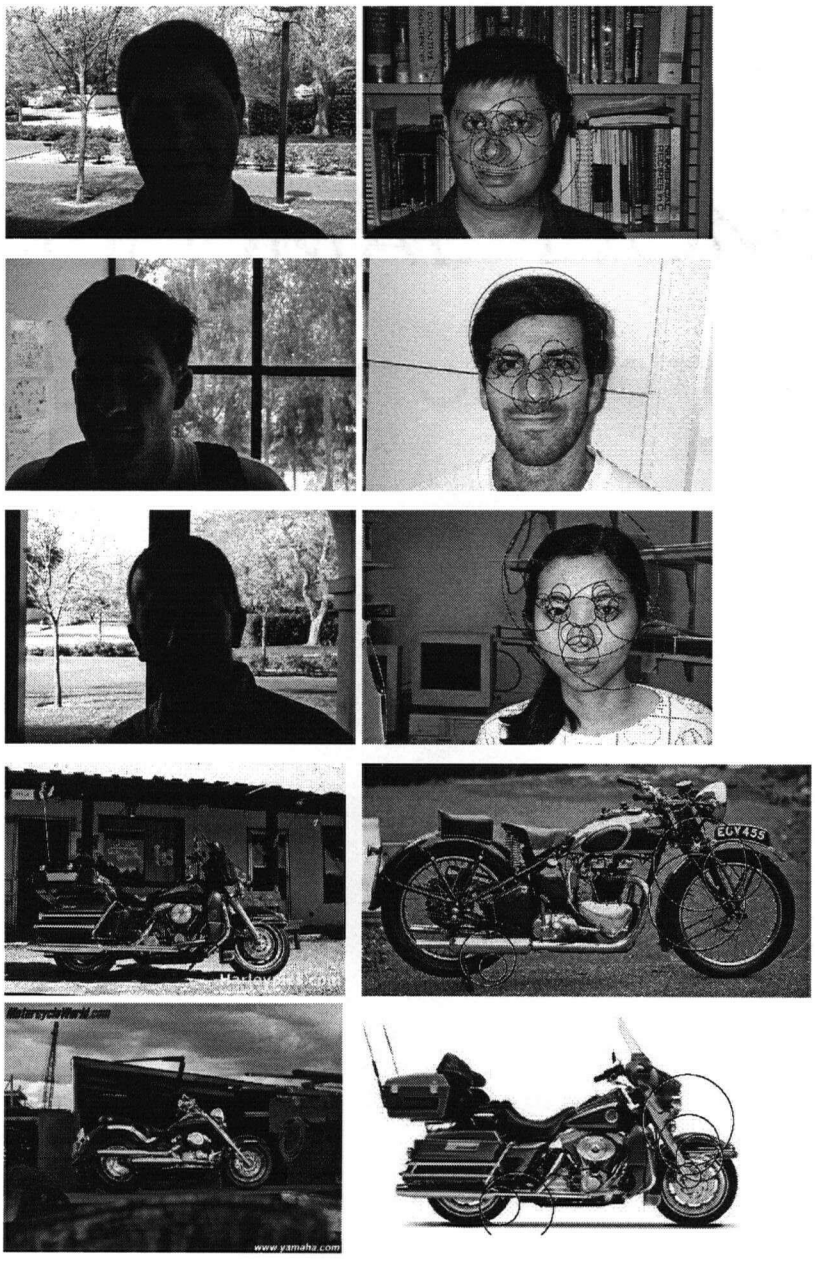


Figure 6.4: The first are images that are not recognized, where as the second column are easily recognized. The features in the optimal match are highlighted. Notice how in the first column the images are substantially different in character from those in the other column, indicating why they may be difficult to recognize.

the appearance model and the location of the parts model.

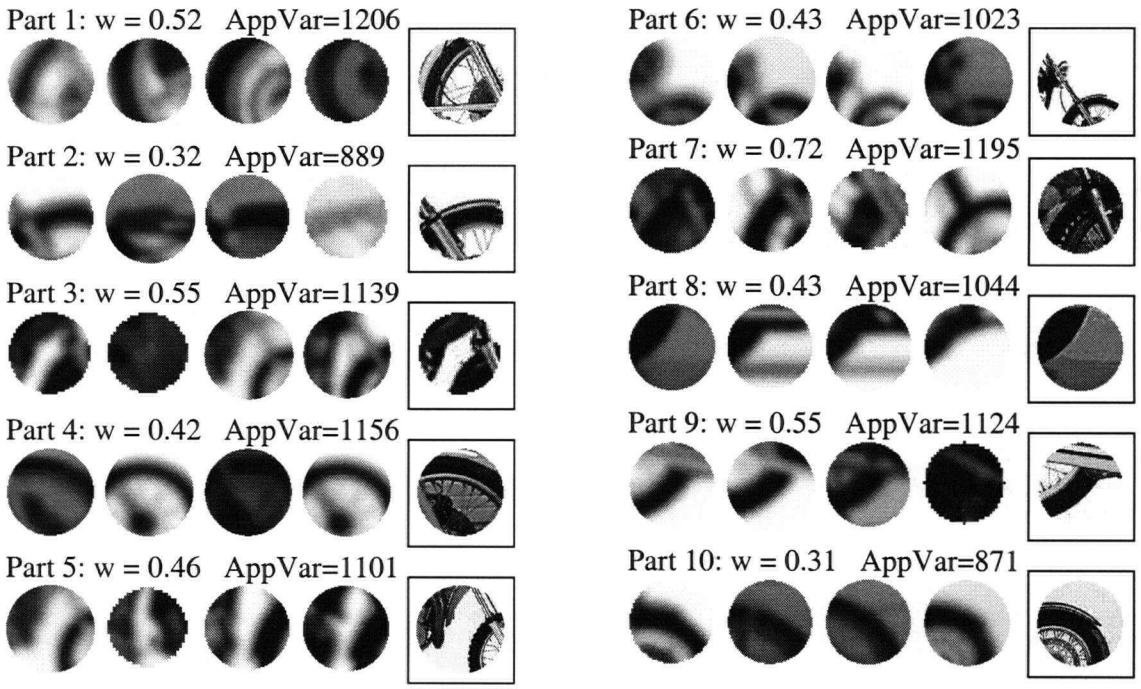
In the appearance section, for each part, is the blurred image of the top four features from the dataset that match the part based solely on appearance. The blurring is proportional to the scale of the feature, which is performed on the original image patch before encoding the feature descriptor $\mathbf{A}_f$. The 5th image patch is an example of a feature without the blurring. This is meant to illustrate which part of the object it comes from if it is not clear from the blurred image patches. Also with each part is the weight w_p for the part p, which is roughly the proportion of the dataset that this part appeared in in the training data. Also included is the AppVar which is $\sum_i^d \boldsymbol{\sigma}_{p,i}^{app}$, the sum of the standard deviations over all dimensions of the appearance for the part p. This is meant to illustrate how wide the variance is for the appearance of each part.

In the locations section is displayed various ellipses, where the centre of each ellipse corresponds to part p's location in model coordinate space. The x and y radius are $2\boldsymbol{\sigma}_p^{loc}$. Each ellipse is connected to a line that points to some image patch, which is an example of that feature from the dataset. The scale of the feature is represented by the scale of the image patch.

As can be seen in the figures, some interesting properties are picked up by the model. Of particular interest is the model of motorbikes, which are primarily described by the rounded curves in a particular geometric configuration, so it appears that it is modelling the wheels primarily. With the motorbikes it was generally the case that most of the parts were concentrated on the front wheel, even for large models. A partial explanation for this is that in the initialization procedure, the initial small model tended to be composed of parts from the front wheel. As a result, when we trying to add new parts to the model, there is a bias towards parts being somewhat close to the parts already in the model. The reason for this is that if a majority of matches in $\mathbf{h}$ in an image are in one particular area, then these tend to dictate where the center $\mathbf{c}_{opt}$ is. As a result, the location for any potential part that is far from the parts in the model thus far will appear to be more noisey, and so the increase in the likelihood when this part is added to the model will be less than it would have been if it were closer to the other parts already in the image. As a result, parts that are closer to the parts already learned for the model are preferred by the incremental learning procedure.

Another interesting thing to note is that many of the parts that were learned for these models were on the edge of the object in the image. This means that it is picking up the shape of the object, however, this relies on the background being a

particular type. For example, the cars viewed from the rear tended to be against a light grey background, usually a road scene, so the parts that included part of the background will not be detected in instances where a car is on a very dark or very light background. However, this is not a problem of the approach, but of the dataset. Clearly if the dataset has a regularity, then an unsupervised approach to learning parts will pick up on this. In order to deal with this problem, datasets should include objects against a great variety of background scenes so that parts that are learned do not rely on the fact that the object class is against the same type of background as in the training data.



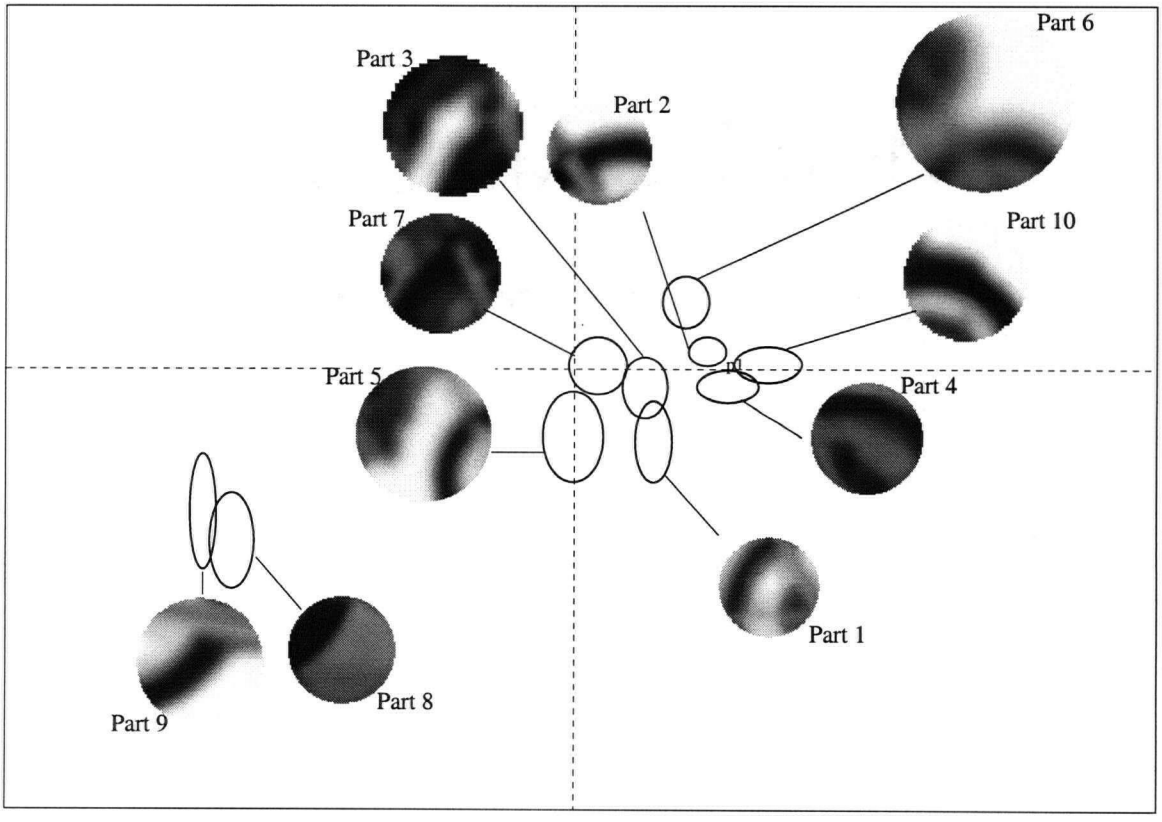


Figure 6.5: A 10 part model for motorbikes

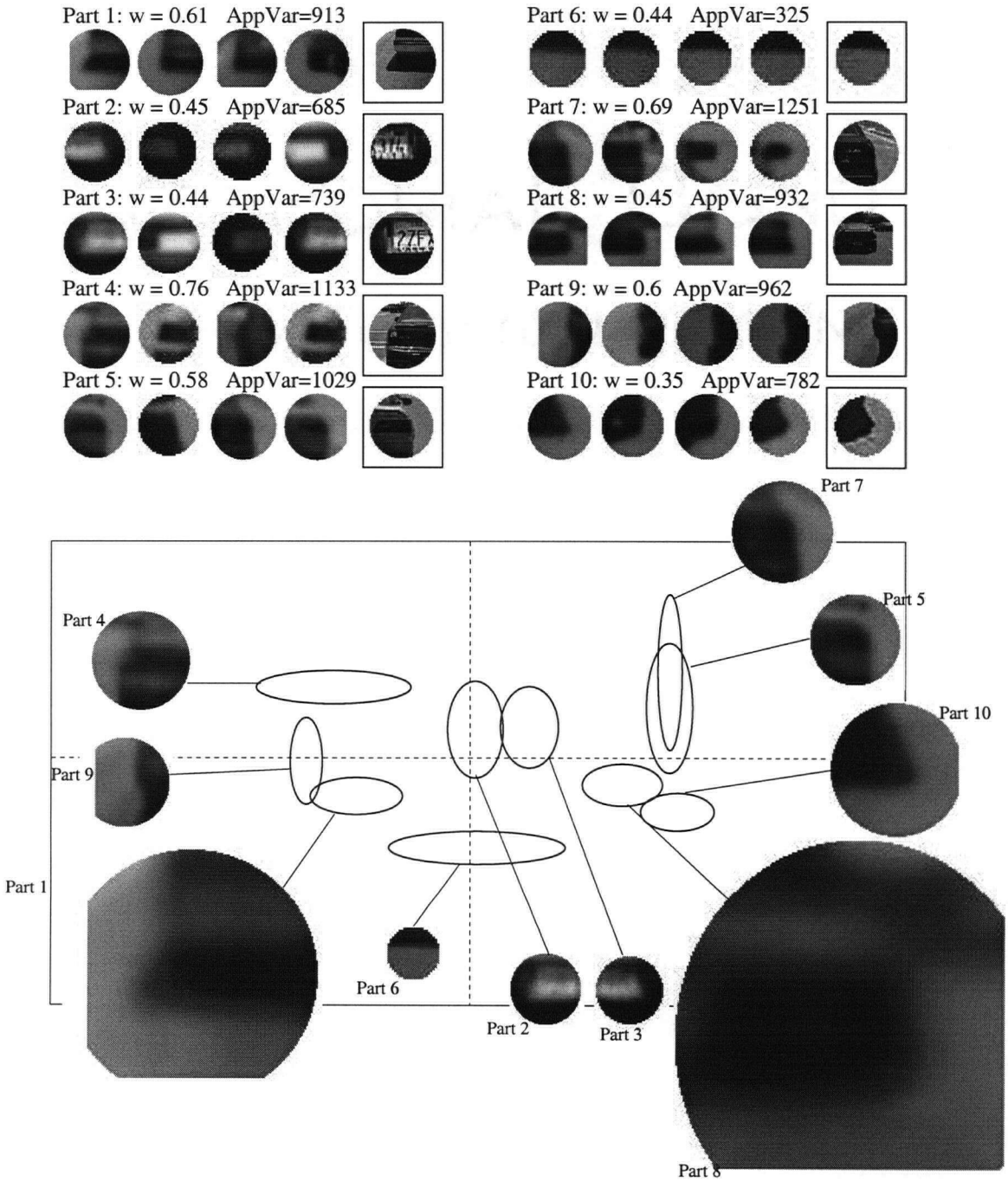


Figure 6.6: A 10 part model for cars viewed from the rear

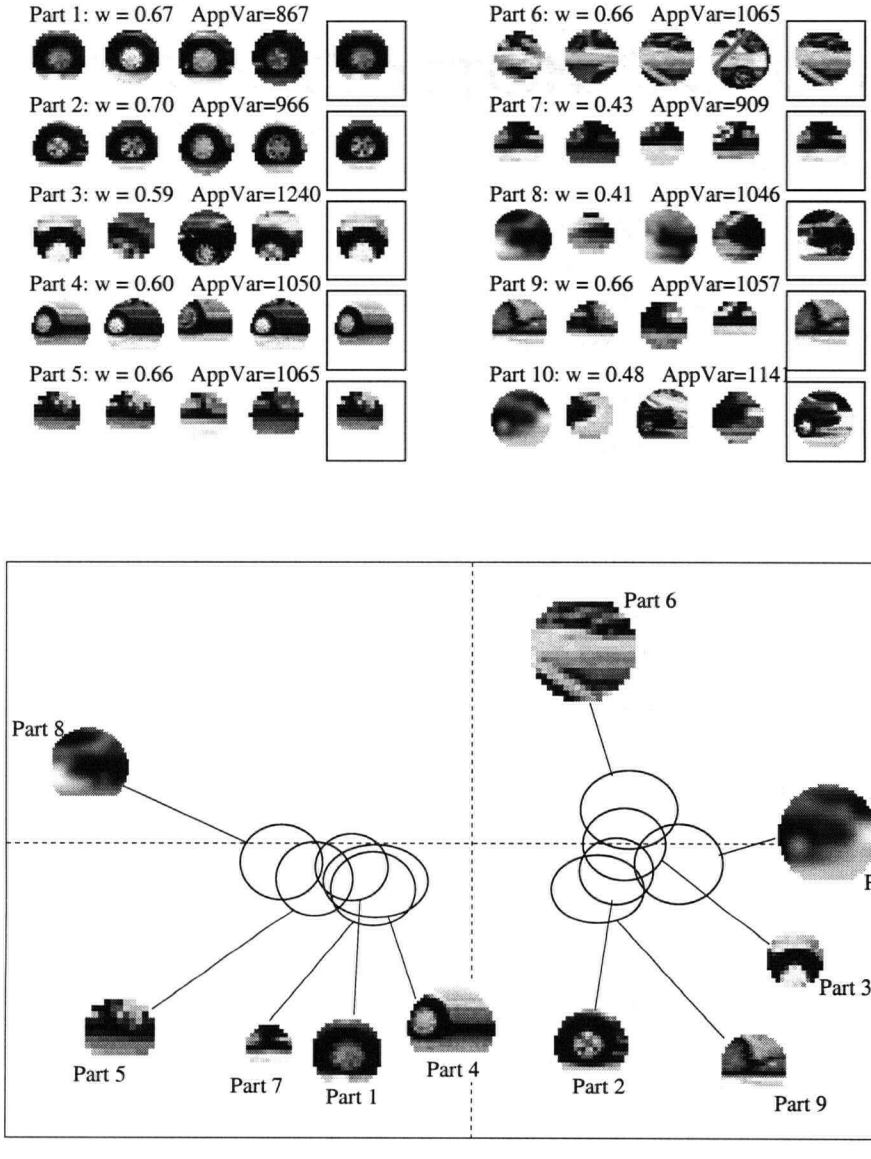


Figure 6.7: A 10 part model for cars viewed from the side

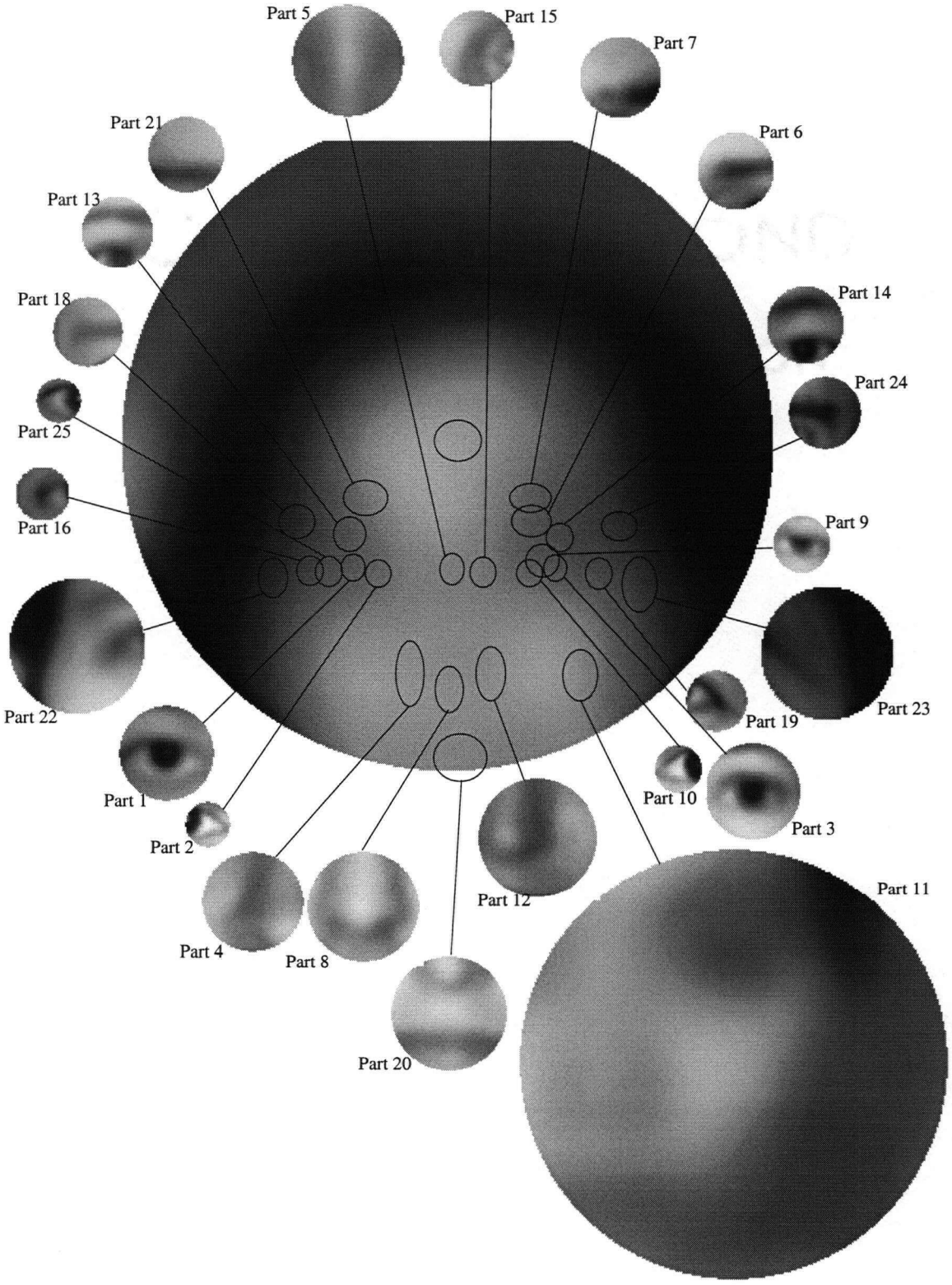


Figure 6.8: The locations for the parts in a 25 part model for faces

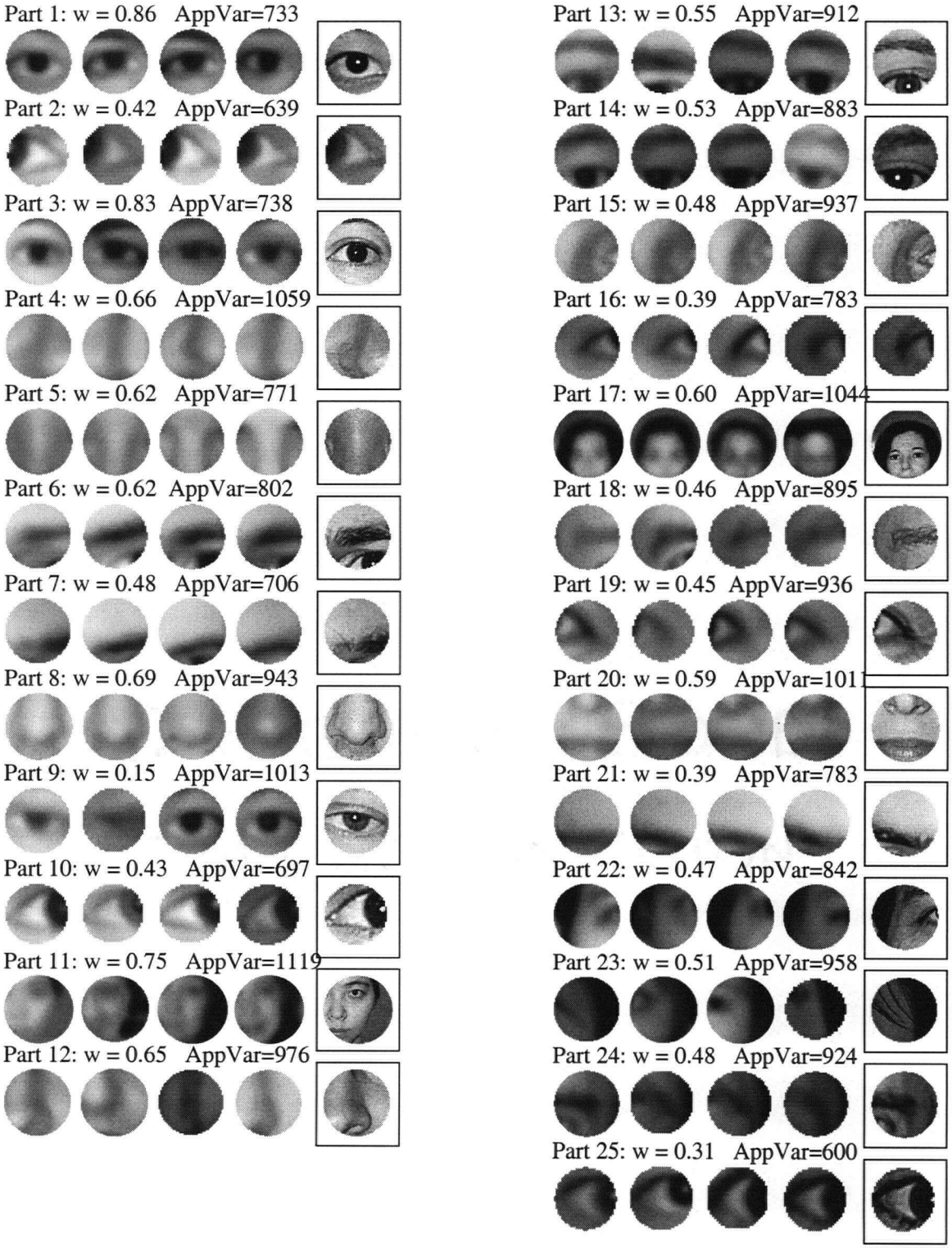


Figure 6.9: The appearance for parts in 25 part model for faces

Chapter 7

Conclusion and Future Work

There has been a great deal of attention focused on part-based approaches to object classification in recent research in computer vision, and some approaches have achieved a surprising amount of success. However, learning models with a large number of parts has been a particular challenge. One of the most successful approaches is that of Fergus *et al.*, who have developed a generative model for recognition that achieves excellent results on a variety of datasets, including cars, motorbikes, cats, faces, and air planes. The learning method that they present to learn the parameters for the model, however, requires an exponential amount of time to train as the number of parts increase. They cite a 6 part model takes about 24 hours to train. The reason for this is that in the learning algorithm they have to evaluate nearly all possible matchings in the initial stages, and the number of matchings increases exponentially with the number of parts in the model.

The primary contribution of this thesis is the extension of their generative model, and the development of a learning algorithm that can learn a large number of parts in a reasonable amount of time. In particular, we have developed an incremental learning algorithm where the model is initialized intelligently with a small number of parts, and parts are added to the model one at a time. By initializing the parameters of a small model, we don't need to evaluate a great number of matchings in order to update the parameters for the model. Moreover, by adding parts one at a time we can utilize information from previous iterations, so the number of matchings that need to be evaluated to update the parameters can be greatly reduced. The end result, as demonstrated in Chapter 6, is that we are able to learn models with a large number of parts, over 25, in much less than an hour. Moreover, there is no reason to suggest that larger models cannot be learned.

The recognition results, however, do not directly indicate that results improve a great deal with models with more parts. However, with the datasets on which we evaluated our approach, it turns out that a majority of the dataset can be recognized with only a few parts, and that the remaining are extremely difficult to recognize due to properties in these images that weren't present in a majority of the dataset.

As a result, recognition results like the ROC equal error rate are not necessarily a good indicator of the power of our method. In future we would like to evaluate our approach on more natural datasets, since this is more likely to reveal the advantage of having models with a large number of parts.

The learning method presented, however, has a number of pitfalls and interesting extensions that we would like to address in future work. The first is the improvement of the procedure for initialization of a small model and for the initialization of a sample set. One improvement is to remove the bias towards selecting parts that are close together in an object, for both the small model and for the sample set of potential parts. The effect of this bias can be see in the model for motorbikes where a majority of the parts learned were near the front wheel.

Another interesting improvement would be to be able to automatically split an object class into distinct subclasses in the cases where the subclasses are sufficiently distinct. For example, the class of vehicles is composed of motorbikes, cars, trains, but these classes do not share a large number of parts that would manifest themselves visually. It should be possible for the learning algorithm to detect if there is some segment of the dataset that it currently does not recognize and then try to create a separate object model for this segment.

In addition, when we select a new part to add to the dataset, we select it from the set of potential parts that was obtained before learning, and we select it so that it maximizes the likelihood of the data the most. The problem with this is that the learning algorithm will build a richer model primarily for the segment of the training data that it already recognizes very well. There is no constraint that learning algorithm has to be able to recognize the training data evenly. As a result, a common result is that the model learned can recognize a majority of the dataset very well, but cannot recognize a few images of the dataset at all. An interesting extension would be to impose a constraint, or bias the selection of a new part, so that the model can recognize each of the images in the dataset evenly. One approach to do this would be to instead select a potential part from the segment of training data that is not recognized well.

It should be noted, however, that the learning method presented in this thesis is not restricted to the model presented. In general, a part-based approach has to learn the parts of the model, and learning models with a large number of parts will be difficult regardless of the model. The key insight provided by this thesis is that it is not necessary to learn all the parts at once. In fact, by beginning with a smaller model and building a more complex model slowly it is possible to arrive at a good

model while avoiding the computational cost that is associated with learning all of the parts at once.

Bibliography

[1] S. Agarwal and D. Roth. Learning a sparse representation for object detection. In *European Conference on Computer Vision*, pages 113–1130, 2002.

[2] I. Biederman. Human image understanding: Recent research and theory. *Computer Vision, Graphics, and Image Understanding*, 32:29–73, 1985.

[3] M. C. Burl and P. Perona. Recognition of planar object classes. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1996.

[4] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(B):1–38, 1976.

[5] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2003.

[6] C. Harris and M. Stevens. A combined corner and edge detector. In *Fourth Alvey Vision Conference*, pages 147–151, 1988.

[7] S. Helmer and D.G. Lowe. Object recognition with many local features. In *Generative-Model Based Vision (held in conjunction with Proc. IEEE Conf. Computer Vision and Pattern Recognition)*, 2004.

[8] D. G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, pages 1150–1157, 1999.

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

[10] G. Mayraz and G. E. Hinton. Recognizing hand-written digits using hierarchical products of experts. In *Adv. in Neural Information Processessing Systems*, pages 953–959, 2000.

[11] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *European Conference on Computer Vision*, pages 128–142, 2002.

[12] H. Murase and S. K. Nayar. Visual learning and recognition of 3d objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, 1995.

[13] A. R. Pope and D. G. Lowe. Probabilistic models of appearance for 3-d object recognition. *International Journal of Computer Vision*, 40(2):149–167, 2000.

[14] C. Schmid and R. Mohr. Local greyvalue invariants for image retrieval. In *IEEE Trans. on Pattern Analysis and Machine Learning*, pages 530–534, 1997.

[15] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 2004.

[16] M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 586–591, 1991.

[17] S. Ullman and E. Sali. Object classification using a fragment-based representation. In T. Poggio S.W. Lee, H. H. Bulthoff, editor, *First IEEE International Biologically Motivated Computer Vision Workshop (BMCV)*, pages 73–87. Springer, 2000.

[18] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2001.

[19] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *European Conference on Computer Vision*, pages 18–32, 2000.