Unsupervised Learning of Models for Object Recognition

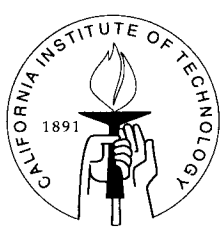
Thesis by

Markus Weber

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy



California Institute of Technology

Pasadena, California

2000

(Submitted May 1, 2000)

Für

Nathalie, Grégoire und Maximilian

Acknowledgements

I am most indebted to my wife, Nathalie, who was on my side throughout this entire adventure. She has been extremely supportive and has accepted a great deal of hardship and sacrifices to give me the necessary time to complete this project. This is her accomplishment as well!

At Caltech, my foremost thanks go to my advisor, Pietro Perona, who was always very supportive with regards to scientific and personal matters. I am also very grateful to Max Welling, my closest collaborator over several years. He was always available for discussions and advice. This work would not have been possible without him. I would like to thank my fellow graduate students, Arrigo Benedetti, Enrico di Bernardo, Jean-Yves Bouguet, Xiaolin Feng, Luis Goncalves, Mario Munich, and Yang Song, for making the Vision Lab at Caltech a very exciting and pleasant place to work. Wolfgang Einhäuser and Rob Fergus helped implement many of the ideas presented here during their time as summer students at Caltech.

I am grateful to the members of my thesis committee, Yaser Abu-Mostafa, Mark Konishi, Demetri Psaltis, and Mike Burl. Mike also helped me to get started on this project and, over the years, provided me with a wealth of useful comments.

Abstract

A method is presented to learn object class models from unlabeled and unsegmented cluttered scenes for the purpose of visual object recognition. The variability across a class of objects is modeled in a principled way, treating objects as flexible constellations of rigid parts (features). Variability is represented by a joint probability density function (pdf) on the shape of the constellation and the output of part detectors. Corresponding "constellation models" can be learned in a completely unsupervised fashion. In a first stage, the learning method automatically identifies distinctive parts in the training set by applying a clustering algorithm to patterns selected by an interest operator. It then learns the statistical shape model using expectation maximization. Mixtures of constellation models can be defined and applied to "discover" object categories in an unsupervised manner. The method achieves very good classification results on human faces, cars, leaves, handwritten letters, and cartoon characters.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Recognizing Objects in Sensory Data

Great efforts have been undertaken over the last half century to build machines that can see. A great number of fundamental problems have been solved. Examples of such problems are the detection of edges in images, the computation of basic flow fields representing motion throughout a sequence of images, or the point-by-point registration of a pair of stereoscopic images. At the same time, one cannot escape the impression that the biggest problems are still far ahead of us.

Central to understanding images is the problem of recognizing objects in images. Humans can recognize objects effortlessly and are rarely even aware of the changes in an object's appearance that occur, for example, due to changes in viewing direction or a shadow being cast across the object. We also readily group instances of objects, such as cars, faces, shoes, or houses into a single object category and forget about the differences between the individual members. At the same time, we can still discriminate on a sub-categorical level, stating, for example, "Here comes *Peter*, walking *my* dog!"[1] On the other hand, everyone who has ever dealt with a computer has inevitably experienced that even the smallest change in the information provided to a computer can, and often does, make all the difference in the world. Teaching a machine to recognize objects is all about teaching it which differences in the raw image information matter and which don't. The focus of this thesis is on the problem of learning principled representations of objects automatically from sensory signals, and on how such representations can be used to detect objects.

What exactly do we mean by "objects" and "object classes or categories?" It is not easy to give a formal definition, since the term is used in a very broad manner. We consider an object to be a part of, or token in, a sensory signal. The precise representation of the object within the signal can undergo changes such as scaling, translation, or other deformations, or it can be contaminated by noise or be partially occluded. These changes give rise to an entire collection or class of signals which can all still be associated with the original object. Objects in signals often correspond to physical objects in the real world environment from which the signals have been recorded. This is

[1] Identifying objects as members of categories, such as cars or dogs, is often refered to as *object detection*, while identifying individual members is refered to as *object recognition*. We will adhere to this interpretation.

the case for objects in images of natural scenes. However, objects can also be defined solely in the universe of signals. For example, a pattern in a recording of a birdsong or a in recording of neural activity might be defined as an object. Classes of objects are collections of objects that are similar. The similarity might be based on high-level cognitive principles, e.g., in the case of the class of chairs. In this case a recognition system might have to embody or develop some understanding of these concepts in order to identify objects of the class. In a more tractable scenario, the similarity largely manifests itself at the level of the signal representation. For example, portions of the signals of two different objects from the same class could be identical.

We used the term "sensory signal" above, since we believe that our methods are not restricted to image data. They have been successfully applied by Burl et al. to the problem of recognition of handwritten characters and words, based on a representation of the ink trace in terms of point coordinates over time. We have also had initial success in classifying neural spike trains representing information about odors in the olfactory bulb of the locust.

Why is *learning* an important component in this context? There are two fundamentally different approaches to building recognition systems. One is to design every aspect and detail of the system before it is put to use. Under the second approach, instead of designing a complete solution, one would endow the system with some prior knowledge about the problem at hand, as well as some mechanism that will allow it to extract, from some collection of *training signals*, the necessary remaining information to finally solve the problem. This concept is refered to as learning. The amount of learning can vary from a single parameter being adjusted to a complete representation of the problem setting being acquired. The extreme case would be a system that embodies a simple method of comparing signals and a huge memory to store every signal that might possibly be encoutered. It could then recognize objects by simply looking up the answer to the given problem in memory, using the input image as an index into the list of answers. Since such large amounts of memory are unavailable for most problems, and since training of such a system would require tutoring until all likely signals have been encountered at least once, this procedure is rather impractical. Especially since many simple rules, such as invariance relations with respect to changes in overall brightness in an image, can easily be incorporated into a recognition system. Learning methods have proved very successful in computer vision and many other fields. Some of the most successful methods for face detection, for example, make extensive use of learning techniques. They need to be trained on a set of a few hundred images, before they become powerful recognition tools.

In this work, we are particularly interested in learning methods that do not require any supervi-

sion by a human operator during the learning phase.

1.1.1 Applications

Applications for recognition systems are manifold and their number is steadily growing with the rapid improvements in signal acquisition and storage capabilities. Different types of sensors are almost omnipresent and provide an abundance of data that needs to be processed. A recent space shuttle mission to map the surface of the earth produced more than 12 Terabytes of data, corresponding to over 20,000 CDs or approximately the content of the Library of Congress. More and more large databases of texts and images are becoming available publically on the internet. CNN plans to make its entire archive searchable over the internet, while the *New York Times* has already done so. There is no possibility of processing such large amounts of data by hand. Therefore, automatic tools that understand or at least extract useful information from images are needed. Object recognition would be key.

Interaction with machines is often difficult and cumbersome, if not impossible, unless these machines understand something about their environment, in particular the objects and creatures they interact with. In order to navigate in any environment, vision is the most important sensory modality. In interactions between humans, visual input is almost as important as auditory information. This suggests that vision should play a crucial role in human-machine interactions. Examples include computer interfaces in general, computer games, interactions with appliances or automobiles, as well as the problem of surveillance. Automobiles now have vision-based systems that drive autonomously or at least assist the driver in keeping up with the changes in the outside environment. It is of great importance that these systems be able to detect objects such as roads, pedestrians, traffic signs, and other vehicles.

Another rapidly expanding area in computer vision is medical image processing. Here, machines that can accurately detect objects such as bones, organs, boundaries between different types of tissues, or tumors would be of great practical usefulness. Such biological objects lend themselves particularly well to the approach we have chosen to model object classes.

1.2 Related Work

In this section, we review some of the methods proposed for object recognition that are related to ours. Other widely used but less closely related approaches include linear subspace meth-

ods, often based on principal component analysis (PCA) [Hot33], such as [KS90, TP91, MN95, BFP+94, BAS+98]. Limitations due to the linearity of this approach have been pointed out by Bichsel [Bic96]. Important are also techniques based on geometric hashing, in particular by Wolfson [Wol90].

1.2.1 Modeling Objects as Collections of Parts

Fischler and Elschlager [FE73] proposed first, in the computer vision literature, to model objects as spatially deformable collections of rigid parts. However, their efforts to use such models for recognition were only moderately successful, as a practical and powerful detection method was lacking. The question of how to cope with background clutter was also not addressed. Furthermore, deformations in the part configurations were modeled through spring forces in a rather ad hoc fashion. Yuille extended their method using deformable templates as object parts [Yui91]. The templates themselves are based on snake-like representation; local deformations are thus modeled through an ad hoc energy function as well.

A more rigorous and principled approach based on a similar decomposition of objects into parts has been introduced by Burl et al. [BLP95, LBP95b, Bur97, BLP96]. In this body of work, the problem of background clutter is addressed, successful detection methods are proposed, and spatial deformations are handled in a probabilistic, principled manner. The work presented here is a direct continuation of these efforts. We have introduced some changes to the model and detection method and developed unsupervised learning algorithms for estimating the models from data.

The work by Amit et al. [AK93, AG99] is also closely related to ours. Objects are modeled as two-level hierarchies of parts, with constellations of simple edge elements at the lowest level. Deformations are modeled by allowing the edge elements to freely move inside a certain region. Triangulated graphs are formed such that groups of edge elements are used as parts on the top level. The parts are learned by evaluating exhaustively all possible constellations with respect to their discriminatory power. The whole learning process requires the training images to be brought into correspondence and is thus more restrictive than our approach.

The alignment method by Huttenlocher and Ullman [HU87] also embodies a similar approach to object detection. Points of an object model are projected into the image and brought into correspondence with image features. Only rigid objects are considered in this framework. Rigoutsos and Hummel [RH95] have proposed a method based on probabilistic hashing, which can be interpreted as a computation of probabilities of configurations of image features. Shortcomings are that the part

positions are assumed to be independent, as well as practical problems with the implementation of hashing.

Pope and Lowe [PL94, PL95] also model the global probability distribution of part posigions, together with local part appearance. Their approach is not as rigorous as that of Burl et al., since positions are assumed to be independent. Pope and Lowe have, however, associated more information to an object part, such as orientation and scale. In principle, this is possible with our approach as well, but we have not yet experimented with this idea.

1.2.2 Other Deformable Models

Deformability of objects has also been modeled without decomposing objects into specific parts. Work along these lines has been done by von der Malsburg et al. [LVB+93, WvdM93, WvdM96]. In this approach, objects are represented through points on a deformable grid. Points are matched to local image regions through a distance measure based on a bank of Gabor filters (JETS). The grid deformations are modeled using an ad hoc energy function. A challenge for this approach is to cope with background clutter and occlusion.

Lanitis et al. model shape deformations through eigenmodes and local features using snakes [LTCA95, CT95]. It is not clear how the problems of cluttered background, occlusion or learning would be addressed. The same research group has recently put forward the concept of active appearance models [ETC98], which is similar to the morphing-based technique of Bichsel [Bic96]. With active appearance models, the appearance of the entire object is modeled as a large deformable template using linear subspace methods. Representing different sources of deformation (shape, lighting, aging) along orthogonal directions in parameter space, one can synthesize and then match instances of objects with image. This model also does not address occlusion. It is furthermore computationally rather complex and no principled way exists to use this model for detection.

1.2.3 Face Detection

Probably the most prominent and most frequently addressed application of object recognition is detecting and recognizing human faces in images. Recently, several systems have been presented, which are able to detect faces from several viewing directions in cluttered scenes with high accuracy.

Fischler and Elschlager [FE73] originally applied their method to face recognition. However, no tests were done on larger databases. Baron [Bar81] devised a method for face recognition based on

correlation templates that can be seen as rigid parts. The geometry of part position was, however, not modeled, and the method was not applied to face *detection*. Regions on the faces had to be selected in each image by a human operator. The technique has been improved by Brunelli and Poggiog [BP93], who use templates defined for gradient images. They register face images for data acquisition by first identifying the positions of the eyes. This strategy is different from our approach, since it uses extensive prior knowlegde (supervision). For the same reason, it cannot not easily be generalized to other object classes.

Rowley et al. [RBK95, RBK98] have proposed a system to detect frontal views of faces based on a neural network. Sun and Poggio's [SP98] method relies on an approximation of the probability density function of faces in image space using Gaussian mixture models. Both methods require a very large set of segmented and properly aligned training images. The same requirements hold for the method introduced by Schneiderman and Kanade [SK98], which also represents the probability density of face images, in this case using explicit histograms. Their system is able to recognize faces from frontal and profile views. Recent improvements [SK00] and further experiments also show promising performance on the problem of detecting cars in cluttered scenes with a limited amount of occlusion. A novel type of learning algorithm, the support vector machine (SVM) [CV95, Vap95], has recently been applied to the problem of face detection [OGF97] and object recognition [PRV98]. The performance and limitations are very similar to the above methods. A large amount of training data are needed and objects need to be segmented. The problem of occlusion is also not addressed in a principled way.

The object model of v. der Malsburg et al., mentioned above, as well as a variation, in which the regular grid is replaced by a graph tailored to the object shape were used for face recognition in [WvdM96, WFKv97]. Graph deformation are still handled through an ad hoc energy function and no background clutter is assumed.

1.2.4 Learning

The problem of learning object models from unlabeled and unsegmented images has been neglected in most instances. It is often assumed that the images contain a constant background, or correspondence needs to be established by intervention of a human operator, whose task might be to hand-label feature points in the images.

A first step towards unsupervised learning methods is the automatic computation of correspondence between training images. This problem has been addressed recently by Walker et al.

[WCT98]. However, in this work no background clutter was present.

A learning problem similar to ours is addressed by Baumberg and Hogg [BH94]. The authors train a flexible shape model of pedestrians in motion. However, a static camera is required in order to subtract the background, thus obtaining a segmented image with little additional effort.

1.3 Outline

This thesis is divided into two parts. In the first part, we discuss object models, in particular the *constellation model*, while the second part is dedicated to learning these models. In Chapter 2 we motivate our approach to modeling objects and we describe the basic generative image model introduced by Burl et al. A more abstract version of this model, including some modifications, is introduced in Chapter 3. In Chapter 4 we describe how such models can be used to solve common detection tasks. In the beginning of the second part, we provide an overview of the learning problem and our approach in Chapter 5. Chapter 6 is dedicated to the problem of part selection for constellation models and Chapter 7 covers the problem of estimating the parameters underlying the probabilistic model. An extension of our model to a mixture model is provided in Chapter 8, together with a corresponding learning algorithm. In Chapter 9, we present results of detection experiments performed on images of human faces, cars, leaves, and other data sets. A conclusion and outlook are given in Chapter 10. The appendix includes brief reviews of standard methods applied in the thesis.

Part I

Object Recognition with Constellation Models

Chapter 2 Global Geometry and Local Photometry

2.1 Introduction

In this chapter, we provide a more concrete definition of the notion of "object class," which we already introduced in the previous chapter. In order to recognize objects as members of classes, one has to deal with the, possibly large, variability of the signals across a class. We propose a principled strategy to model important sources of variability explicitly. It is based on the idea of decomposing objects into components. Before we introduce our model based on "global geometry and local photometry," we will illustrate how this approach allows us to address the different sources of variability.

We will then derive the theoretically optimal detector based on this object class model. Unfortunately this detector is computationally too demanding for most practical recognition problems. Therefore, we will introduce, in the next chapter, an abstract version of our model which can be implemented more efficiently.

Most ideas included in this chapter are based on previous work by Burl, Leung, and Perona [Bur97, LBP95a].

2.2 Motivation

If we study the visual world surrounding us, we discover many objects that are visually similar. A few examples are human faces, bicycles, hands, cars, coffee mugs, telephones, shoes, pens, chairs, staplers, houses, tree leaves, flowers of the same species, and bottles. All these objects could be thought of as instantiations of *object classes*. Some classes can only be defined based on high-level cognitive concepts. The class of chairs, for example, could be defined in a rather abstract way as "anything man-made for a single person to sit on." As a consequence, two chairs are often visually extremely dissimilar. Just about any particular feature one might choose, in order to devise a more concrete definition of this object class, such as an arm rest, back rest, or a leg, is not present in many chairs. There is not much hope to model such abstract classes based solely on visual appearance. However, there is an intermediate range of classes, e.g., human faces, leaves of trees, or cars, the

members of which are more consistently visually similar, yet not identical. It is the problem of identifying members of those classes that we address here (see Figure 2.1).

If we are asked to give a definition of, say, the class of human faces, it comes naturally to do so by describing a decomposition of a prototypical face in terms of distinctive features or parts. A face is composed of two eyes, a nose, etc. Similar decompositions suggest themselves for cars (in terms of wheels, body, headlights, bumper,...) or hands (five fingers...). This idea of decomposing objects into components[1] was introduced to computer vision research almost three decades ago by Fischler and Elschlager [FE73].

Although it is difficult to give a formal definition, a "part" shall be understood as a subset of an object satisfying four key requirements:

1. The parts of one object correspond to parts in other objects belonging to the same class.
2. Corresponding parts are similar; more so than objects belonging to the same class are similar.
3. Detectors can be devised that can detect the part with some degree of success.
4. A part has geometrical properties, such as a spatial position, often a scale, an orientation, a spatial extent, a velocity.

Since it is often possible to model and detect parts based on the underlying photometric pattern, we refer to this portion of the model as *local photometry*.

A complete description of an object cannot be given by simply listing all its parts. It is also important how the parts are arranged spatially. We refer to this information as the *global geometry* of an arrangements of parts. From this global perspective, parts are reduced to a set of attributes, such as type, position, orientation, etc.

2.2.1 Addressing Signal Variability Explicitly

Describing objects as a collection of parts is not only a natural approach, but it also allows to account for signal variability in a principled way. We will soon show how exactly one could represent the variability in the geometry of an arrangement of parts and in the appearance of parts themselves using probability models. First, however, we will further illustrate some benefits of such an approach, by considering the major sources of variability.

[1] We will use the words "component," "part," or "feature" interchangeably throughout this work.

(a) (b) (c)

Figure 2.1: Different data may be represented as 'parts' and 'shape'. (a) Sequences of spikes produced by neurons in the mushroom body of the locust in response to different odors. Either individual action potentials, or bursts of action potentials could represent the parts. The mutual distance between such parts is the 'shape'. (b) A human face may be thought of as a collection of textured patches in a specific mutual position. (c) The human body in motion may be thought of as a collection of textured patches, each having a position and velocity.

Variability due to Absence of Features

Not all objects of an object class need to be composed of the exact same set of parts. For example, not all motorcars have the same number of wheels, headlights or doors, not all human faces are decorated with beards, hair, or eyeglasses, photographs of hands do not always show the same number of fingers. If we design a model which explicitly handles the event of part absence, be it due to occlusion or other causes, then we can limit our representation to the visible portion of the object and have gained a significant advantage over many methods which need to represent such information implicitly.

Variability due to Deformations

When we compare a non-rigid object before and after it has undergone a physical deformation, as in Figure 2.2, or two geometrically different instances of a class of rigid objects, we find that the *local* differences in appearance are small compared to the *global* change. It might therefore be possible to model the local changes with a simple model, or even to neglect them. The deformation can then be described explicitly by indicating the positions of the local patterns in each image.

We would expect that there is an optimal size for the local patterns, since patterns which are very small are not very distinctive and are thus easily confused with other patterns. On the other hand, the assumption that a part is a constant and, therfore, rigid pattern will become less justifiable with increasing part size.

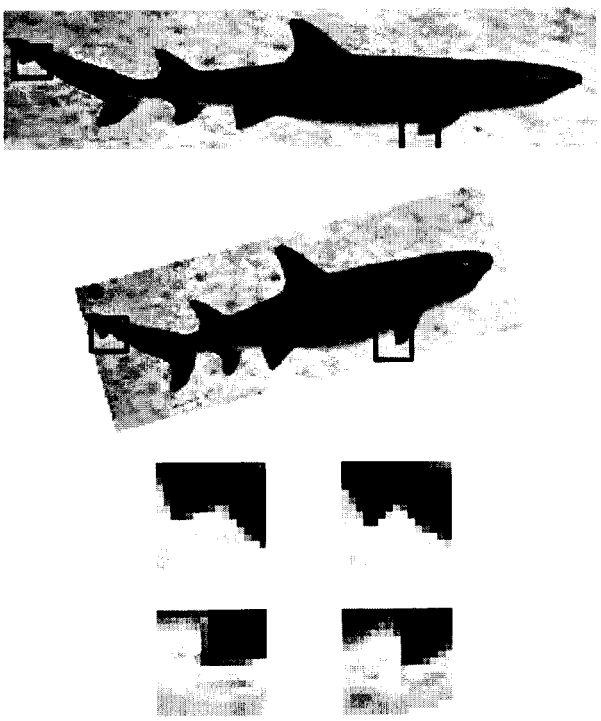


Figure 2.2: The top image has been compressed along the horizontal direction and then rotated, in order to produce the image in the center. This transformation distorts the global appearance, such that pixel-wise matching does not reveal that both images belong to the same class ("sharks"). However, local features (bottom) remain similar and correspondence can be established through simple matching. The two patches on the left correspond to the features extracted from the top image (as marked by dark squares), the ones on the right correspond to the center image. Global appearance could therefore be modeled explicitly using patterns of constant local appearance which change position under the transformations.

Variability Due to Pose Changes

Deformations of a signal due to changes in pose of the underlying objects can be considered a special case of the geometric deformations just mentioned. If pose changes in three dimensions are over a limited range, it would still be possible to use a simple model to explain the resulting changes in local photometry. The global geometry model might in this case be based on explicit knowledge about the three-dimensional structure of objects as well as allowed transformations, e.g., under a certain camera model.

Variability Due to Background Clutter

We can decompose an entire signal comprised of an object of interest and background clutter into a collection of components. We can then assign individual components to either foreground or background and model their actual appearance independently.

Variability Due to Lighting

When an entire scene undergoes changes in lighting conditions, the brightness patterns across an object often change in a complex fashion (see Figure 2.3). By concentrating on a local neighborhood on the object's surface, we might find changes in lighting conditions that can be accounted for by a much simpler model than would be needed to represent the global change. It might even be possible to eliminate those local changes through simple normalization steps.

2.3 Generative Probabilistic Models

All models discussed in this work are generative probabilistic models. They are based on probability density functions (pdfs), representing the set of possible signals corresponding to the class to be modeled. The attribute "generative" refers to the possibility to produce an actual signal (or a simplified version thereof) by drawing values for the random variables underlying the model. In order to solve recognition problems, this type of model can also be used for signal analysis, through a process called inference. Starting from a given signal, we try to *infer* the values of the underlying random variables in order to evaluate the probability that the signal could have been produced by the model. If this probability is large, we also say that the model "explains" the signal well.

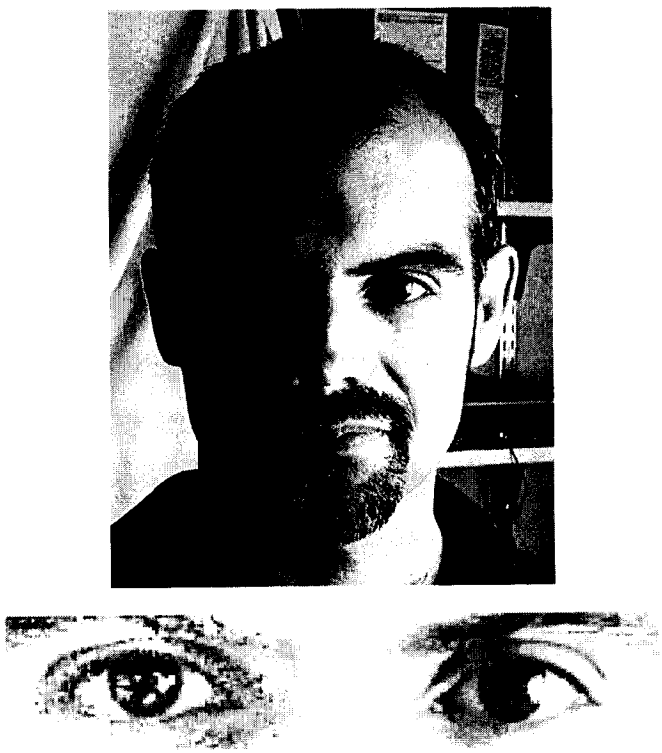


Figure 2.3: The top picture shows a human face lit from the right. The resulting difference in brightness between the left and right half of the face render detection challenging. Below we show two local regions (left and right eye) of the face that have been normalized separately with respect to brightness and contrast; pixel noise is more pronounced in the left patch due to the limited dynamical range of the camera used. The similarity of the two patches after normalization illustrates that representing objects based on *local* photometric patterns allows to compensate for changes in lighting conditions more easily.

Representation vs. Discrimination

From the above ideas one might conclude that a good model is a model that explains (represents) a class of signals well. However, by focusing on representation, we are addressing our final goal of recognition or detection only indirectly. We do not know which details of the signals we are interested in are "worth" representing well, until we know what signals this class will ultimately have to be discriminated against. While the most prominent or salient features might end up being well represented, they might not be the most distinctive features when it comes to discriminating between the class to be modeled and other signals. A fundamentally different approach therefore is to directly focus on discrimination instead of representation. [2] Fukunaga [Fuk90] discusses this choice and concludes that discrimination is important when a particular object, e.g. an individual human face, needs to be recognized amongst similar ones, while representation is useful when fundamentally different classes have to distinguished, such as faces from shoes.

One advantage of the representation-based approach is that we do not need to know beforehand what other classes our class of interest (let us call it C_1) will have to be distinguished from. We therefore also do not require any training examples from the possibly large set of other classes when building a model. A disadvantage—besides the fact that we might model irrelevant aspects of the class C_1—is that we might end up solving a more general problem than the problem we are ultimately interested in. This is generally not a good idea, as pointed out by Vapnik [Vap95]. Solving the more general problem of representation might require more training data than to directly establish a decision boundary in signal space. Such a boundary, dividing the space into a region corresponding C_1 and a second region corresponding to all other signals, could possibly be modeled with fewer parameters than the entire pdf corresponding to the signals in C_1. Furthermore, estimation of probability densities is generally an ill-posed problem that will require some form of regularization.

A significant disadvantage of the discriminative approach, on the other hand, is that even if we had enough training data for the signals not belonging to C_1, we could only construct a classifier that would take a binary decision of the sort "signal does/does not belong to C_1." This is fine if all we are interested in is the class C_1. If we want to assign objects to many different classes, we would need to build one classifier for each class, discriminating signals of that class against every other

[2] The representation based approach is also called *generative, sampling, causal*, or *class-conditional*, while the discrimination based approach is refered to as *diagnostic* or *predictive*. A discussion of the implications of each approach can be found in [Jor95].

possible signal. A prominent example of a discriminative approach is the support vector machine (SVM) classifier, brought about by the work on structural risk minimization by Vapnik [Vap95]. Support vector machines have been used, e.g., by Pontil et al. [PRV98] to solve the problem of recognizing the 100 objects contained in the COIL database [NNM96]. An impressive number of 496 classifiers, each able only to distinguish between two classes, needed to be trained for this task.

Since our ultimate goal *is* to distinguish between a larger number of different classes, and since we are less interested in recognizing individual objects within a class, we have chosen to focus on representation.

2.4 A Simple Model of Global Geometry and Local Photometry

We are presenting in the following a concrete generative model for two-dimensional signals (such as images), based on two-dimensional parts. The probabilistic formulation is due to Burl et al. [BLP95, BLP96], while the basic idea has been introduced by Fischler and Elschlager [FE73].

Consider an object consisting of N photometric parts $P_i(x, y)$, each occuring in the signal at a particular spatial location (x_i, y_i). The parts P_i can be thought of as "patches" that are placed down at the appropriate positions. Mathematically, the image T of an object is given by:

$$T(x, y) = \sum_{i=1}^{N} P_i(x - x_i,\ y - y_i) \tag{2.1}$$

For convenience, we will assume, that the $P_i(x, y)$ are defined for any pair (x, y), but are non-zero only inside a relatively small neighborhood around $(0, 0)$.

Let $\mathbf{x}$ be the vector describing the positions of the object parts, i.e.

$$\mathbf{x} = \begin{pmatrix} x_1 & x_2 & \dots & x_N & y_1 & y_2 & \dots & y_N \end{pmatrix}^T. \tag{2.2}$$

An *object class* can now be defined as the set of objects induced by a set of vectors $\{\mathbf{x}_k\}$. In particular, we assume that the part positions are distributed according to a joint probability density $p_{\mathbf{x}}(\mathbf{x})$. We will designate the resulting object class as $\mathcal{T}$. To generate an object from this class, we first generate a random vector $\mathbf{x}$ according to the density $p_{\mathbf{X}}(\mathbf{x})$. Since this vector determines the part positions, we simply place the corresponding pattern P_i at each of these positions.

Note that no assumption about $p_{\mathbf{x}}(\mathbf{x})$ is made at this time. It should be clear, however, that

through $p_{\mathbf{x}}(\mathbf{x})$ we can control properties of the object class, such as the range of meaningful object *shapes*, as well as tolerable ranges of certain transformations, such as rotation, scaling and translation.

2.4.1 Detectors for the Class $\mathcal{T}$

The basic detection problem can be stated as follows: given an image, $\mathcal{I}$, determine whether the image contains an instance from $\mathcal{T}$ (hypothesis ω_1) or whether the image is background-only (hypothesis ω_0). In the following section, we will directly derive the optimal detector for the signal class $\mathcal{T}$, starting from the pixel image $\mathcal{I}$.

Derivation of the Optimal Detector

If we take a standard MAP[3] approach, the optimal decision statistic is given by the likelihood ratio

$$\Lambda = \frac{p(\mathcal{I}|\omega_1)}{p(\mathcal{I}|\omega_0)} \tag{2.3}$$

We can rewrite the numerator by conditioning on the spatial positions $\mathbf{x}$ of the object parts. Hence,

$$\Lambda = \frac{\sum_{\mathbf{x}} p(\mathcal{I}|\mathbf{x},\omega_1)\cdot p(\mathbf{x}|\omega_1)}{p(\mathcal{I}|\omega_0)} \tag{2.4}$$

where the summation goes over all possible configurations of the object parts. Assuming that parts do not overlap, we can divide the image into $N+1$ regions, $\mathcal{I}^0, \mathcal{I}^1, \ldots, \mathcal{I}^N$, where $\mathcal{I}^i$ is an image which is equal to $\mathcal{I}$ in the area occupied by the non-zero portion of part P_i (positioned according to $\mathbf{x}$) and zero otherwise. $\mathcal{I}^0$ denotes the background. Assuming furthermore that the background is independent across regions, we obtain

$$\Lambda = \frac{\sum_{\mathbf{x}} \prod_{i=0}^{N} p(\mathcal{I}^i|\mathbf{x},\omega_1)\cdot p(\mathbf{x}|\omega_1)}{p(\mathcal{I}|\omega_0)} \tag{2.5}$$

$$= \sum_{\mathbf{x}} \left[\prod_{i=1}^{N} \frac{p(\mathcal{I}^i|\mathbf{x},\omega_1)}{p(\mathcal{I}^i|\omega_0)}\right]\cdot p(\mathbf{x}|\omega_1) \tag{2.6}$$

$$= \sum_{\mathbf{x}} \left[\prod_{i=1}^{N} \lambda_i(x_i,y_i)\right]\cdot p(\mathbf{x}|\omega_1) \tag{2.7}$$

[3]MAP stands for *maximum a posteriori probability* and is a basic concept of Bayes decision theory.

Here, the $\lambda_i(x_i, y_i) = \frac{p(\mathcal{I}^i|\mathbf{x},\omega_1)}{p(\mathcal{I}^i|\omega_0)}$ can be interpreted as likelihood ratios expressing the likelihood of part P_i being present in the image at location (x_i, y_i). Note that $\lambda_0(x, y)$ is equal to one, under the hypothesis that the statistics of the background region do not depend on the presence or absence of the object.

We can specialize this derivation by introducing a particular part detection method. For example, assuming that the object is embedded in white Gaussian noise, we can substitute Gaussian class conditional densities and obtain

$$\lambda_i = \frac{\mathcal{N}\left(\mathcal{I}^i;\, \mu_{\mathbf{x}}, \sigma^2\mathbf{I}\right)}{\mathcal{N}\left(\mathcal{I}^i;\, \mathbf{0}, \sigma^2\mathbf{I}\right)} \tag{2.8}$$

Here, $\mu_{\mathbf{x}}$ is the object with parts positioned at $\mathbf{x}$, $\mathbf{0}$ shall denote a vector of zeros and $\mathbf{I}$ is the identity matrix. Expanding the Gaussian densities and combining terms yields:

$$\begin{aligned} \lambda_i &= \exp\left(\frac{\mu_{\mathbf{x}}^T\mathcal{I}^i}{\sigma^2} - \frac{\mu_{\mathbf{x}}^T\mu_{\mathbf{x}}}{2\sigma^2}\right) \\ &= \exp\left(-\frac{\mu_{\mathbf{x}}^T\mu_{\mathbf{x}}}{2\sigma^2}\right) \cdot \exp\left(\frac{\mu_{\mathbf{x}}^T\mathcal{I}^i}{\sigma^2}\right) \\ &= c \cdot \exp\left(\frac{\mu_{\mathbf{x}}^T\mathcal{I}^i}{\sigma^2}\right) \end{aligned} \tag{2.9}$$

where σ^2 is the variance of the pixel noise and c depends only on the energy in the object image and is therefore constant independent of $\mathbf{x}$, provided the parts do not overlap. Equation (2.9) simply restates the well known fact that matched filtering is the optimal part detection strategy under this noise model. Writing A_i for the response image obtained by correlating part i with the image $\mathcal{I}$ and normalizing by σ^2, we finally obtain

$$\Lambda = c \cdot \sum_{\mathbf{x}} \left[\prod_{i=1}^{N} \exp\left(A_i(x_i, y_i)\right)\right] \cdot p(\mathbf{x}) \tag{2.10}$$

The constant c does not affect the form of the decision rule, so we will omit it from our subsequent equations.

Independent Part Positions

If the part positions are independent, $p(\mathbf{x})$ can also be expressed as a product

$$p(\mathbf{x}) = \prod_{i=1}^{N} p_i(x_i, y_i) \tag{2.11}$$

Thus, we have

$$\Lambda = \sum_{\mathbf{x}} \left[\prod_{i=1}^{N} \lambda_i(x_i, y_i) p_i(x_i, y_i) \right]$$

For the special case of additive white Gaussian noise, we obtain

$$\begin{aligned} \Lambda &= \sum_{\mathbf{x}} \left[\prod_{i=1}^{N} \exp\left(A_i(x_i, y_i)\right) p_i(x_i, y_i) \right] \\ &= \sum_{\mathbf{x}} \left[\prod_{i=1}^{N} \exp\left(A_i(x_i, y_i) + \log p_i(x_i, y_i)\right) \right] \\ &= \prod_{i=1}^{N} \left[\sum_{(x_i, y_i)} \exp\left(A_i(x_i, y_i) + \log p_i(x_i, y_i)\right) \right] \end{aligned} \tag{2.12}$$

Thus, we need to compute the correlation response image (normalized by σ^2) for each object part. To this image, we add the log probability that the part will occur at a given spatial position, take the exponential, and sum over the whole image. This process is repeated for each object part. Finally, the product of scores over all the object parts yields the likelihood ratio.

Note, that the detector is not invariant to translation, rotation, and scaling since the term $p_i(x_i, y_i)$ includes information about the absolute coordinates of the parts.

Jointly Distributed Part Positions

If the part positions are *not independent*, we must introduce an approximation since summing over all *combinations* of part positions as in (2.7) is infeasible. The basic idea—similar to a winner-take-all strategy—is to assume that the summation is dominated by one term corresponding to a specific combination $\mathbf{x}_0$ of the part positions. With this assumption, we have

$$\Lambda \approx \Lambda_0 = \prod_{i=1}^{N} \lambda_i(x_i, y_i) \cdot p(\mathbf{x}_0)$$

$$\log \Lambda_0 = \sum_{i=1}^{N} \log \lambda_i(x_i, y_i) + \log p(\mathbf{x}_0) \tag{2.13}$$

and in the case of additive white Gaussian noise

$$\log \Lambda_0 = \left(\sum_{i=1}^{N} A_i(x_{0i}, y_{0i}) \right) + \log p(\mathbf{x}_0) \tag{2.14}$$

The strategy now is to find a set of part positions such that the matched filter responses are high and the overall configuration of the parts is consistent with $p(\mathbf{x}|\omega_1)$. Again, the resulting detector is not invariant to translation, rotation, and scaling.

2.5 Conclusion

We have reviewed the concept of global geometry and local photometry and demonstrated how it can be used to model the different sources of variability underlying an object class in a principled way. We have also argued that generative probabilistic models, with their underlying focus on signal *representation*, are a reasonable choice to address the problems we are interested in. A simple image model of this type, due to Burl et al., has been presented and the corresponding optimal detector has been derived. This detector is still impractical for two reasons. Firstly, it requires to examine an intractably large number of possible joint part positions, secondly it is not invariant with respect to translation, rotation, or scale. Although a solution to the first problem has been proposed, it is unclear how it should be implemented. We will therefore reexamine the modeling problem in the following chapter from a slightly more abstract point of view. We will revisit the model presented in this chapter in Section 9.7, where we experiment with different part detection methods.

Chapter 3 The Constellation Model

3.1 Introduction

In this chapter, we introduce our part constellation model, which is an abstraction of the image model discussed in the previous chapter. We are not concerned with the specific low-level representation (such as the pixels) of signals and parts, instead we regard the entire signal as a collection of parts, some of which correspond to background clutter and some of which correspond to an object. Such a representation could be obtained by feeding the signals through a part detection process which only retains the positions at which parts of a certain type have been localized. This model is a modified version of the model used by Burl [Bur97]. We also use a Gaussian density to describe the object part positions, but we model background clutter differently.

A summary of the notation introduced in this chapter is provided at the end in Table 3.1.

3.2 Constellation Model

3.2.1 Foreground Objects

As before, we will call "object" a token or pattern that we will want to recognize, e.g., a shoe in an image or the response of a neuron to an olfactory stimulus. In this section, we introduce the portion of our model corresponding to the target object, to which we refer as the *foreground.* How we model background clutter will be discussed in the next section.

We use a variable, $\mathcal{O}$, to represent the fact that an object is present ($\mathcal{O} = 1$) or absent ($\mathcal{O} = 0$) in a given signal. We could allow this variable to take on larger values in the case where several objects are present. However, we do pursue this idea any further in the following. We will consider $\mathcal{O}$ to be a random variable and, as a shorthand, we will write $\mathcal{O}_i$ for the event $\mathcal{O} = i$.

We denote by F the number of parts in the object model. Since an object might contain several copies of the same part, such as a car with four or more identical wheels, each part is attributed to one of $T \leq F$ *part types*. This information is represented through the vector, $\mathbf{t}$, of length F with $t_f \in \{1, \ldots, T\}$ indicating the type of object part f. Aside from the type, we may associate further attributes with a part, such as its position, orientation, scale, or confidence of detection. The exact

set of attributes will depend on the application at hand and the desired complexity of the model. For example, when modeling neural spike trains, one might want to define a burst part and choose the time of onset and duration of a burst as its two main attributes. Certain information, such as orientation, might be represented either as an attribute or by creating different part (sub-)types for each range of possible values. We used this idea for line segments in [FW]. From here on, we assume that the only attributes are the type of part and its point-location in the signal.

In our experiments with images, we characterized parts by type and position only. For an object consisting of F parts, the two-dimensional part positions are represented as a $2F$-vector

$$\mathbf{x} = (x_1, x_2, \ldots, x_F, y_1, y_2, \ldots, y_F)^T \tag{3.1}$$

where the pair (x_f, y_f) designates the position of part f in the image.

Since we also want to model events such as occlusion, or the failure of part detectors to account for parts, as well as the possibility that not all instantiations of an object class exhibit all parts, we need to represent the fact that certain parts are not present in a given signal. For this we use a binary vector $\mathbf{d}$, of length F. If d_f is equal to zero, then part f is absent, while $d_f = 1$ means it is present.

To turn this representation into a probabilistic model, we consider $\mathbf{d}$ and $\mathbf{x}$ to be random variables. Hence, we need to define probability distributions over these variables.

If the number of parts, F, is small, then the vector $\mathbf{d}$ can only take on a relative small number of values – 2^F, to be precise. In this case, we can afford to represent $p(\mathbf{d}|\mathcal{O}_1)$ explicitly as a table of length 2^F, containing *joint* probabilities for the d_f. One reason to assume dependence between these variables is that parts in close spatial proximity might have a higher probability of being occluded simultaneously than parts that are further apart. If, on the other hand, we want to model the d_f as independent, or if F is large, we can provide a table of length F, containing for every d_f, the probability $p(d_f = 1)$, i.e. the probability that the corresponding part is present. If the object is not contained in the signal, no parts can possibly be present. Therefore we choose

$$p(\mathbf{d}|\mathcal{O}_0) = \begin{cases} 1 & \text{if } \mathbf{d} = \mathbf{0} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

$$\tag{3.3}$$

When defining probabilities over part positions, an important question is whether we want to consider the domain of $\mathbf{x}$ to be continuous or discrete. In most practical applications, $\mathbf{x}$ will only

take on discrete values, such as pixel locations in images or time-samples in one-dimensional signals such as neural spike trains. However, it will be much more convenient to assume a continuous range of values for these variables, as will become evident later. We thus introduce a probability density function $p_{\text{fg}}(\mathbf{x})$ to which we will refer as the *foreground density*. We will most of the time assume this density to be Gaussian:

$$p_{\text{fg}}(\mathbf{x}) = \mathcal{G}(\mathbf{x}|\mu, \Sigma). \tag{3.4}$$

Here, one has the liberty to model dependencies between the part positions by allowing for a full covariance matrix, or to limit the number of degrees of freedom of the model by choosing, e.g., a diagonal covariance matrix. Note that the true underlying p_{fg} is in many cases a complicated density, since $\mathbf{x}$ encodes intrinsic object properties, such as *shape*, together with the *pose* of an object. We will discuss in Section 3.3 how to address this problem.

When we are dealing with missing or undetected object parts, $p_{\text{fg}}(\mathbf{x})$ cannot readily be evaluated, because some dimensions of $\mathbf{x}$ are unknown. In this case, we can split $\mathbf{x}$ into an observed and a missing portion,

$$\mathbf{x}^T = (\mathbf{x}^{\text{o}T}\mathbf{x}^{\text{m}T}).$$

We can then evaluate $\mathbf{x}^{\text{o}}$ under the marginalized foreground density, which we obtain by integrating out the missing dimensions,

$$p_{\text{fg}}(\mathbf{x}^{\text{o}}) = \int p_{\text{fg}}(\mathbf{x})\, d\mathbf{x}^{\text{m}}.$$

In the case of a Gaussian foreground density this is achieved simply by eliminating the missing dimensions from the mean and covariance matrix.

In principle, one could consider joint probabilities even between $\mathbf{d}$ and $\mathbf{x}$, modeling, e.g., the fact that certain part detectors fail when an object in an image is in a particular pose, such as upside-down.

3.2.2 Background Clutter

Aside from the object of interest, a signal might also contain other objects or noise. We will refer to all parts of the signal that are not associated with the target object as the *background*. Adhering to our view of signals as a collections of parts, there are two main reasons why we need to accomodate for the presence of background. The first is that some part of the target object might also be present in other objects. This is especially true if we use very generic parts, such as line crossings in images.

The second reason is that a part detector that we might apply to search for the object could mistake some portion of the surrounding background signal for a part of the target object, i.e. it could produce a *false detection*. We need to be able to deal with these background parts, since, in a typical recognition problem, it is not known which part detections, if any, belong to the target object and which to the background. In any case, a truly generative model should be able to "explain" the background.

We make two important assumptions about the statistics of the background:

- Background parts are independent of the foreground object.
- Background parts are independent of each other.

These two assumptions allow us to derive a statistical background model separately from the foreground model given above. Below we provide a complete derivation of the background model for the important case of spatially uniformly distributed background parts.

Spatially Uniform Background Noise

If we have little knowledge about the process that gives rise to background parts, we should assume that those parts can appear anywhere in the signal with equal probability. We can then derive the exact probability of generating a particular background signal. To simplify things, we provide here a derivation for a single part type and a one-dimensional signal. Furthermore, we limit ourselves to *position* as the only part attribute.

An intuitive understanding of the background process can be achieved by first considering a discrete signal. In the discrete, one-dimensional case, we can think of a background signal as D sections or *bins* of length dx, such that the total length of the signal is

$$L = D\,dx.$$

Each bin can generate a background part with the same probability q. The probability of generating a particular background signal is then simply the probability of generating n parts *at specific locations*, which is

$$p(n) = p_{\text{bino}}(n|q, D)\,\frac{1}{D(D-1)\cdots(D-n+1)}. \tag{3.5}$$

Here, $p_{\text{bino}}(n|q, D)$ denotes the binomial distribution, which yields the probability of generating a total of n parts in *any position*. The second factor is simply the number of different ways n

indistinguishable parts can be filled into D bins. Note that the average number of background parts is $M = qD$, as is expected for a binomial distribution.

The solution for the case of continuous signals can now be otained by taking the limit

$$D \to \infty, \quad q \to 0, \quad dx \to 0$$

under the constraints

$$qD = M = const., \quad Ddx = L$$

At the same time, we have to divide (3.5) by dx^n. Making use of the fact that, in the limit, the binomial distribution will become a Poisson distribution, we obtain

$$\begin{aligned} p(n) &= \lim \left[p_{\text{bino}}(n|q,D) \frac{1}{D(D-1)\cdots(D-n+1)} \left(\frac{D}{L}\right)^n \right] \\ &= \frac{1}{L^n} p_{\text{Poiss}}(n|M) \lim \frac{D^n}{D(D-1)\cdots(D-n+1)} \\ &= \frac{1}{L^n} p_{\text{Poiss}}(n|M) \end{aligned}$$

As one might have expected, our final result still only depends on the *number* of background parts. Similarly to (3.5), we can think of the Poisson distribution as providing the probability to find n parts in *any* set of positions, while the first factor adjusts for the case of *specific* positions, which, in this case, are uniformly distributed over the length, L, of the signal.

Note that the only parameter of this background model is the average number of background parts, M.

Since we assumed independence between the background parts, this result is trivially extended to the case of multiple part types, by introducing a vector, $\mathbf{M}$. In the case of two-dimensional signals, such as images, we only need to adjust the parameter L, to represent the area of the signal support.

3.2.3 Complete Model

We are now ready to assemble the building blocks developed above in order to derive a probability model of for the complete signal, consisting of foreground and background. We again assume that the only part attribute is position. In order to describe one instance of a signal, we need to provide the number and positions of all parts (foreground and background) for each type of part as well as

the information as to which parts belong to the foreground. This information can be summarized in two variables. Let

$$\mathcal{X} = \begin{pmatrix} x_{11}x_{12},\ldots,x_{1N_1} \\ x_{21}x_{22},\ldots,x_{2N_2} \\ \vdots \\ x_{T1}x_{T2},\ldots,x_{TN_T} \end{pmatrix}$$

be a "matrix" containing the positions of all parts. Thus, the τ^{th} row contains the locations of all parts of type τ. Every $x_{\tau j}$ is a vector, the length of which depends on the signal dimensionality. Furthermore, let $\mathbf{h}$ be a vector of length F, with h_f indicating which, if any, of the parts in row t_f of $\mathcal{X}$ corresponds to the foreground part f.[1] Specifically, $h_f = 0$ shall mean that part f is absent, while $h_f > 0$ shall convey that the location of foreground part f is stored in $x_{t_f h_f}$. We also call $\mathbf{h}$ a hypothesis, for reasons to become evident later.

Given this rather compact representation of a signal, the probabilistic model comes down to the exact specification of the joint density

$$p(\mathcal{X}, \mathbf{h}|\mathcal{O}). \tag{3.6}$$

In order to write out this density in terms of the components derived in the previous sections, it is helpful to understand how exactly one signal can be generated in a step-by-step fashion:

1. Decide how many background parts are present for each type τ, $\tau = 1,\ldots,T$, using $p(n_\tau)$.
2. Decide if an object is present in the signal or not, based on some prior $p(\mathcal{O})$. If no object is present, set $\mathbf{h} = \mathbf{d} = \mathbf{0}$ and go to step 6.
3. Decide which parts of the foreground object are present, using $p(\mathbf{d}|\mathcal{O}_1)$.
4. Draw h_f for all f with $d_f = 1$ from a uniform distribution over $\{1,\ldots,n_{t_f}+1\}$.[2]
5. Generate positions for the foreground parts, using p_{fg}, and store those positions in $\mathcal{X}$ according to the indices in $\mathbf{h}$.
6. Generate positions for the background parts for all unoccupied entries of $\mathcal{X}$ from a uniform density. Background part positions are assumed independent.

[1] In the case of several parts of the same type, we allow here that the same part in the signal be assigned simultaneously to two different parts in the object model. This could represent two model parts that overlap exactly in the signal. If, for a given problem, this is not a reasonable assumption, it is still possible to exclude this case by assigning a zero probability to the corresponding constellation through the foreground density.

[2] If several, say l, parts of the same type are used, the first one is drawn with equal probability from $\{1,\ldots,n_{t_f}+l\}$. The next part is then placed with equal probability in any of the *remaining unoccupied* entries, and so forth.

Note that the order of these steps is important, since most steps depend on information generated in earlier steps. We can translate this process into a factorization of $p(\mathcal{X}, \mathbf{h}, \mathcal{O})$, obtaining

$$p(\mathcal{X}, \mathbf{h}|\mathcal{O}) = p(\mathcal{X}|\mathbf{h}, \mathbf{n})\, p(\mathbf{h}|\mathbf{n}, \mathbf{d})\, p(\mathbf{n})\, p(\mathbf{d}|\mathcal{O}). \tag{3.7}$$

The variables $\mathbf{n}$ and $\mathbf{d}$ are uniquely determined by $\mathcal{X}$ and $\mathbf{h}$, hence $p(\mathcal{X}, \mathbf{h}, \mathbf{n}, \mathbf{d}) = p(\mathcal{X}, \mathbf{h})$.

One could argue that $p(\mathbf{n})$ should also be conditioned on $\mathcal{O}$, since the area occupied by the background clutter is smaller when an object is present. However, given that false detections might also occur on the object itself, we assume that the same area can be used in both cases.

The density $p(\mathbf{h}|\mathbf{n}, \mathbf{d})$ is modeled by

$$p(\mathbf{h}|\mathbf{n}, \mathbf{d}) = \begin{cases} \prod_{f=1}^{F} N_{t_f}^{-d_f} & \text{if } \mathbf{h} \text{ is compatible with } \mathbf{n} \text{ and } \mathbf{d} \\ 0 & \text{otherwise.} \end{cases}$$

This expresses the fact that all consistent hypotheses, of which there are $\prod_{f=1}^{F} n_{t_f}^{d_f}$, are equally likely in the absence of information on the part locations. Substituting this and the model densities derived in the previous sections yields

$$p(\mathcal{X}, \mathbf{h}|\mathcal{O}) = p_{\text{fg}}(\mathcal{X}(\mathbf{h}))\; p_{\text{bg}}(\mathcal{X}(\overline{\mathbf{h}}))\; \prod_{f=1}^{F} N_{t_f}^{-d_f}\; \prod_{\tau=1}^{T} p_{\text{Poiss}}(n_\tau|M_\tau)\; p(\mathbf{d}|\mathcal{O}). \tag{3.8}$$

Here, $p_{\text{fg}}(\mathcal{X}(\mathbf{h}))$ shall mean extracting those part positions from $\mathcal{X}$ that are corresponding to the foreground object – according to $\mathbf{h}$ – and evaluating those positions under the foreground density. Similarly, $p_{\text{bg}}(\mathcal{X}(\overline{\mathbf{h}}))$ is computed by evaluating all positions *not* labeled as foreground by $\mathbf{h}$ under the background density.

Gaussian Foreground and Uniform Background Densities

Substituting a Gaussian foreground density and a uniform background density, one obtains

$$p(\mathcal{X}, \mathbf{h}|\mathcal{O}) = \frac{\mathcal{G}(\mathcal{X}(\mathbf{h})|\mu, \Sigma)}{\prod_{\tau=1}^{T} A^{n_\tau}}\; \prod_{f=1}^{F} N_{t_f}^{-d_f}\; \prod_{\tau=1}^{T} p_{\text{Poiss}}(n_\tau|M_\tau)\; p(\mathbf{d}|\mathcal{O}), \tag{3.9}$$

where $\mathbf{M}$ is a T-vector and M_τ is the expected number of background parts of type τ. Furthermore, A is the extent (length/area, depending on dimensionality) of the signal.

3.3 Invariance

In the preceeding sections, we have built our model upon a description of part constellations based on absolute part positions. As a consequence, the probability of observing a certain signal including an object, $p(\mathcal{X}, \mathbf{h}|\mathcal{O}_1)$, will change when the foreground object in the signal undergoes even simple transformations such as translation or scaling. Although small transformations of this kind can to some extent be accomodated through the foreground density, p_{fg}, as generic geometric variability, a more principled approach is to explicitly model them as well.

Bookstein [Boo84, Boo86] has suggested removing translation, rotation, and scaling from point constellations by transforming the constellation such that two points are sent to a fixed position. The information thus removed can be interpreted as the *pose* of the constellation, while the positions (after normalization) of the remaining points define the *shape*. Burl et al. [BLP95] have used this idea to develop a constellation model that can be used for detection invariant with respect to translation, rotation, and scale of an object.

We illustrate in the following section the implications for our model of a translation invariant represention of constellations, since we will later make use of this idea for model learning.

3.3.1 Translation

Translation can be removed from a point constellation by using a simplified version of Bookstein coordinates, where only one point is sent to a fixed location by a rigid translation of the entire constellation. Consider a vector of component positions

$$\mathbf{x} = (x_1, x_2, \ldots, x_F, y_1, y_2, \ldots, y_F)^T. \tag{3.10}$$

If we describe the positions relative to the first component, we obtain

$$\mathbf{x}' = (0, x_2 - x_1, \ldots, x_F - x_1, 0, y_2 - y_1, \ldots, y_F - y_1)^T. \tag{3.11}$$

In doing this, we have translated the entire constellation, such that the first component is moved to the point $(0, 0)$. We define the position $\tau = (x_1, y_1)$ to be the position of the entire constellation prior to normalization. The two zero-entries in $\mathbf{x}'$ can be dropped to yield a vector, $\mathbf{x}^*$, of dimension

$2F - 2$. The entire procedure can be expressed as a linear transform

$$\mathbf{x}^* = W\mathbf{x}, \tag{3.12}$$

where

$$W = \begin{pmatrix} A & \underline{\underline{0}} \\ \underline{\underline{0}} & A \end{pmatrix}, \quad \text{with} \quad A = \begin{pmatrix} -1 & 1 & 0 & \cdots & \cdots & 0 \\ -1 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & & & & & \vdots \\ -1 & 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}. \tag{3.13}$$

Following Burl [Bur97], we refer to the vector space of the $\mathbf{x}$ as *figure space* and to the "translation-free" representation as *figure space**.

With these insights, we can modify our constellation model in order to handle translation explicitly. This is done by changing the foreground density as follows

$$p_{\text{fg}}(\mathbf{x}) \quad \rightarrow \quad p_{\text{fg}}(\tau, \mathbf{x}^*) = p_\tau(\tau)\, p^*(\mathbf{x}^*). \tag{3.14}$$

This reflects the split of $\mathbf{x}$ into the position, τ, and the translation-free portion, $\mathbf{x}^*$, assuming that the two are statistically independent. In order to obtain true translation invariance, a uniform density over the size of the signal support, A, should be substituted for p_τ. If we have previously modeled p_{fg} as a Gaussian density, the corresponding invariant model is obtained as

$$p_{\text{fg}}(\mathbf{x}) = \mathcal{G}(\mathbf{x}|\mu, \Sigma) \quad \rightarrow \quad p_{\text{fg}}(\tau, \mathbf{x}^*) = \frac{1}{A}\, \mathcal{G}(\mathbf{x}^*|W\mu, W\Sigma W^T). \tag{3.15}$$

A further complication that needs to be addressed is the problem of missing parts. Above, we used the first part to define the position of a constellation and to eliminate translation. If the first part is missing, this is obviously not possible. In the case where the figure space density is assumed Gaussian, a simple solution exists. In a first step, we integrate out the dimensions of the Gaussian foreground density (parametrized through μ and Σ) that correspond to the missing parts. This will leave us with another, lower-dimensional Gaussian, which we can represent through the decimated parameters $\overline{\mu}$ and $\overline{\Sigma}$. We can then choose *any* of the present parts as a new reference and construct a matrix, W', accordingly. For the translation invariant model, p_τ does not need to be changed. The

resulting foreground density is

$$p_{\text{fg}}(\tau, \mathbf{x}^*) = \frac{1}{A}\, \mathcal{G}(\mathbf{x}^* | W'\mu', W'\Sigma' W'^T). \tag{3.16}$$

This means that we only need one set of parameters, μ and Σ, in order to evaluate any constellation with an arbitrary set of missing parts under a translation invariant foreground density. In practice, even if constellations are well described in figure space* by a Gaussian density, the true figure space density might be far from Gaussian. As we will discuss in Section 7.5, it is still possible to learn an equivalent Gaussian figure space density, if only to serve as a convenient way to represent the figure space* density.

3.3.2 Translation, Rotation, and Scale (TRS)

As mentioned above, Burl et al. have gone beyond translation invariance by mapping constellations to Bookstein shape coordinates—a representation that is invariant with respect to translation, rotation, and scale (TRS). If part positions are distributed according to a Gaussian pdf in figure space, the resulting density in *shape space* will be a Dryden-Mardia density. This density can be computed in closed form and it can also be represented simply by the parameters of the corresponding Gaussian in figure space. A disadvantage of this approach is that it is computationally expensive to evalute Dryden-Mardia densities.

It is important to realize that a large part of the advantage of TRS invariance brought about by a shape space representation can only be exploited if the part detectors are themselves invariant with respect to rotation and scale. Why is this? Let us suppose that part detectors are not TRS invariant. In this case, they will have to be applied to an input image at different scales and orientations. However, if this is necessary, we could also imagine a model which is invariant only with respect to translation, while the input image is presented to the model at different scales and orientations. The cost for part detection would be the same in this case.

Alternatively, the following procedure could be used to achieve the same result. Assuming that detectors are not invariant, we can retain the scale and orientation at which each part detection occurred, together with its location. Every detection would thus be represented through a four-dimensional vector, and a Gaussian foreground density, p_{fg}, could be defined over all $4F$ dimensions in figure space. If we now have to evaluate a particular constellation under the foreground density, we can normalize the constellation with respect to scale and orientation by using the scale

and orientation information stored with the reference part (used to eliminate translation). Once the constellation is normalized, it can be evaluated under the non-invariant foreground density, p_{fg}.[3] Since we need to consider partial constellations, it is important that any detector be able to indicate the scale and orientation of its detections. Therefore, no detector can be TRS invariant. This approach would also require that the detectors can report scale and orientation very accurately, since a small error in scale or orientation can affect the positions of the other candidates in the constellation significantly, when the orientation and scale information is used to normalize the entire constellation.

3.4 Conclusion

We have presented a revised version of the more abstract constellation model introduced by Burl et al. In particular, we have improved the way in which background clutter is modeled. The issue of translation, scale, and rotation invariance has been addressed. We have shown how our model can be used in a translation invariant way, and we have argued, in Section 3.3.2, that there is an alternative to the shape space approach used by Burl et al. to achieve rotation and scale invariance. Both approaches should be compared experimentally.

[3] Since the $4F$-dimensional Gaussian, p_{fg}, is expressed in absolute figure space coordinates, it would have to be transformed, depending on which model part is used as reference, similar to the translation invariant case (see Equation 3.15).

Variable	Size	Explanation
$\mathcal{O}$	scalar	class label ($\mathcal{O} = 0$: no object present, $\mathcal{O} = 1$: one object present)
T	scalar	number of part types
F	scalar	number of parts in model
$\mathbf{t}$	F	$t_f \in \{1, \ldots, T\}$ is type of model part f
$\mathbf{x}$	$2F$	vector of part positions (x and y)
$\mathbf{x}^{\mathrm{o}}$	variable	vector of *observed* part's positions
$\mathbf{x}^{\mathrm{m}}$	variable	vector of *missing* part's positions
$\mathbf{d}$	F	binary vector, indicating part presence
$p_{\mathrm{fg}}(\mathbf{x})$	pdf	pdf over foreground part positions
p_{bg}	pdf	pdf over background part positions
$\mathcal{X}$	$T \times ?$	row τ contains N_τ part candidates of type τ
$\mathbf{h}$	F	hypothesis vector; h_f is zero or index into row f of $\mathcal{X}$
$\mathbf{N}$	T	N_τ is the *total* number of candidates of type τ found in signal
$\mathbf{n}$	T	n_τ is the number of *background* candidates of type τ found in signal
$\mathbf{M}$	T	M_τ is the average number of background candidates of type τ
μ	$2F$	mean of Gaussian foreground density
Σ	$2F \times 2F$	covariance matrix of Gaussian foreground density

Table 3.1: Overview of the notation introduced in this chapter

Chapter 4 Object Recognition with Constellation Models

4.1 Introduction

In Chapter 3 we presented a probabilistic data model that allowed us to represent signals composed of a foreground object, in turn decomposed into parts, and background noise. We are now going to discuss how this model can be used to solve common object recognition problems. The advantage of a generative probabilistic model is that it fits easily within the context of standard statistical pattern recognition techniques, such as the standard Bayes decision theory.

The two most important problems we are interested in are *object detection* and *object localization*. Object detection refers to the problem of deciding if an object or an instance of an object class is present in a given signal. Object localization is the problem of determining which parts of the signal correspond to the target object. We could add the problem of "object counting," in which one has to determine the exact number of objects in the signal. However, since we have explained our probabilistic model for at most one object per image, we are not going to discuss the counting problem in the following.

Our basic detection technique is based on the one by Burl et al. However, we do not use a hypothesis selection framework as motivation. Burl addressed the localization problem, however the detection problem was not convincingly solved. No common detection threshold could be applied across signals with a different number of part detections. We solve the two problems in two slightly different ways. We also introduce an efficient search method at the end of this chapter.

4.2 Detection and Localization

4.2.1 Part Detection

Our motivation to introduce a model where objects are decomposed into parts is twofold. In the previous chapter, we showed how this representation allows us to model geometric variability and occlusion in a principled way. We will now see how it facilitates recognition as well.

In a first stage we detect individual object parts using detectors specialized for each type of part. Once a part has been detected, we retain its position in the signal and possibly other attributes such

as the detection confidence and the scale at which the part was detected. We call these detections part "candidates." The candidate information is then used as input for the next decision stage, which we describe in the following sections.

In the concrete model described throughout the previous chapter, we store only the *positions* of the detected parts in the variable $\mathcal{X}$ and do not retain any further information from the input signal.

4.2.2 Object Detection

We treat the detection problem as a statistical classification problem and employ the standard *maximum a posteriori probability* (MAP) technique to solve it (see [DH73] for an introduction to Bayes' decision theory). We therefore assume that the universe of all possible signals is divided into two classes: The class $\mathcal{O}_1$ of signals that contain an instance of the target object and the class $\mathcal{O}_0$ of signals that consist of background noise only. The detection problem then is to decide if a given, previously unseen signal belongs to class $\mathcal{O}_1$ or to class $\mathcal{O}_0$.

Using Bayes rule, the MAP technique reduces the detection problem to the evaluation of the following *likelihood ratio*

$$R = \frac{p(\mathcal{O}_1|\mathcal{X})}{p(\mathcal{O}_0|\mathcal{X})} = \frac{p(\mathcal{X}|\mathcal{O}_1)\, p(\mathcal{O}_1)}{p(\mathcal{X}|\mathcal{O}_0)\, p(\mathcal{O}_0)} \tag{4.1}$$

involving the so-called *class-conditional densities* $p(\mathcal{X}|\mathcal{O}_1)$ and $p(\mathcal{X}|\mathcal{O}_0)$. We now explain how to compute this ratio.

Computing Class-Conditional Densities

The class-conditional densities have been formulated as functions of $\mathcal{X}$ only, and not of $\mathbf{h}$. Why is this? This question touches upon a fundamental problem we need to address, which is that the only data we can actually compute from a previously unknown signal is $\mathcal{X}$. This is, in the language of pattern classification, the only measurement we perform on our items (signals) to be classified. Hence the omission of $\mathbf{h}$ from the class-conditional densities. We have what is called a *hidden data* problem. In our case, $\mathbf{h}$ is a hidden or unobserved variable. We will talk about hidden variables at length in the context of expectation maximization in Chapter 7. For now, we can adopt a conceptually simple solution which consists in *marginalizing* our joint density, $p(\mathcal{X}, \mathbf{h}|\mathcal{O})$, onto $\mathcal{X}$, the only *observed* data. We therefore need to integrate out (or better: "sum out") the unobserved

variable, $\mathbf{h}$:

$$p(\mathcal{X}|\mathcal{O}) = \sum_{\mathbf{h}\in\mathcal{H}} p(\mathcal{X}, \mathbf{h}|\mathcal{O}) = \sum_{\mathbf{h}\in\mathcal{H}} p(\mathcal{X}|\mathbf{h}, \mathbf{n})\; p(\mathbf{h}|\mathbf{n}, \mathbf{d})\; p(\mathbf{n})\; p(\mathbf{d}|\mathcal{O}). \tag{4.2}$$

Here, $\mathcal{H}$ refers to the set of all hypotheses consistent with $\mathcal{X}$. We can simply define our joint probability to be zero for all inconsistent hypotheses, such as those with any index h_f larger than the corresponding number of part candidates detected for that part type. Note that the variables $\mathbf{n}$ and $\mathbf{d}$ are obtained as simple functions of $\mathbf{h}$ and, as such, are completely determined for a given $\mathbf{h}$. Since we know $\mathcal{X}$, we know the *total* number, N_τ, of part detections for every type, τ. By subtracting the number of foreground parts according to $\mathbf{h}$, we obtain $\mathbf{n}$. Finally, $\mathbf{d}$ is simply given as $\text{sign}(\mathbf{h})$.[1]

It is, in principle, not difficult to compute the sum over $\mathcal{H}$ explicitly. However, in practice, the number of hypotheses is often too large. This number is equal to $\prod_{f=1}^{F}(N_{t_f}+1)$. For example, in the case of a model with 6 parts and 10 candidates per part, this number is larger than 10^6.[2]

Solution of the Detection Problem

With the insights from the preceeding sections, the solution of the detection problem is now straightforward. From (4.1) and (4.2) we know that we need to compute the likelihood ratio

$$R = \frac{p(\mathcal{X}|\mathcal{O}_1)}{p(\mathcal{X}|\mathcal{O}_0)} = \frac{\sum_{\mathbf{h}\in\mathcal{H}} p(\mathcal{X}|\mathbf{h}, \mathbf{n})\; p(\mathbf{h}|\mathbf{n}, \mathbf{d})\; p(\mathbf{n})\; p(\mathbf{d}|\mathcal{O}_1)}{p(\mathcal{X}|\mathbf{h}_0, \mathbf{n})\; p(\mathbf{h}_0|\mathbf{n}, \mathbf{d})\; p(\mathbf{n})\; p(\mathbf{d}|\mathcal{O}_0)} \tag{4.3}$$

for every signal to be classified. Note that, in the denominator, we have set $\mathbf{h} = \mathbf{h}_0 \stackrel{\text{def}}{=} \mathbf{0}$. We call $\mathbf{h}_0$ the *zero hypothesis*. This hypothesis indicates that all parts in the signal are background parts. We do not need to consider any other hypothesis in the denominator, since $p(\mathbf{d}(\mathbf{h})|\mathcal{O}_0) = 0$, for all $\mathbf{h} \neq \mathbf{h}_0$, according to (3.3). Also note that the sum in the numerator *does* includes $\mathbf{h}_0$. This is because the object might still be part of the signal, even if none of its parts have been detected.

Gaussian Foreground and Background Densities

In the case of our concrete model from the last chapter (Section 3.2.3), we obtain the following likelihood ratio,

[1] Taken element-wise, with $\text{sign}(0) \stackrel{\text{def}}{=} 0$

[2] Our experiments have shown that we are able to evaluate about 10^5 hypotheses per second in a typical detection scenario. We discuss methods to speed up this computation in Section 4.4

$$
\begin{aligned}
R &= \frac{\sum_{\mathbf{h}\in\mathcal{H}} \mathcal{G}(\mathcal{X}(\mathbf{h})|\mu,\Sigma) \prod_{\tau=1}^{T} A^{-n_\tau} \prod_{f=1}^{F} N_{t_f}^{-d_f} \prod_{\tau=1}^{T} p_{\text{Poiss}}(n_\tau|M_\tau)\, p(\mathbf{d}|\mathcal{O}_1)}{\prod_{\tau=1}^{T} A^{-N_\tau} \prod_{\tau=1}^{T} p_{\text{Poiss}}(N_\tau|M_\tau)\, p(\mathbf{d}|\mathcal{O}_0)} \\
&= \sum_{\mathbf{h}\in\mathcal{H}} \frac{\mathcal{G}(\mathcal{X}(\mathbf{h})|\mu,\Sigma)}{\prod_{\tau=1}^{T} A^{n_\tau - N_\tau}} \prod_{f=1}^{F} N_{t_f}^{-d_f} \prod_{\tau=1}^{T} \frac{p_{\text{Poiss}}(n_\tau|M_\tau)}{p_{\text{Poiss}}(N_\tau|M_\tau)} p(\mathbf{d}|\mathcal{O}_1) \\
&= \sum_{\mathbf{h}\in\mathcal{H}} \frac{\mathcal{G}(\mathcal{X}(\mathbf{h})|\mu,\Sigma)}{A^{-\Delta}} \prod_{\tau=1}^{T} M_\tau^{-\delta_\tau} \frac{N_\tau!}{N_\tau^{\delta_\tau} n_\tau!}\, p(\mathbf{d}|\mathcal{O}_1), \qquad (4.4)
\end{aligned}
$$

where we have defined δ_τ as the number of hypothesized foreground parts of type τ. And $\Delta \stackrel{\text{def}}{=} \sum_{\tau=1}^{T} \delta_\tau$ as the total number of hypothesized foreground parts.

Uniform Foreground and Background Densities

As a sanity check, we can now compute R for a uniform foreground density, and obtain

$$
R = \sum_{\mathbf{h}\in\mathcal{H}} \prod_{f=1}^{F} N_{t_f}^{-d_f} \prod_{\tau=1}^{T} M_\tau^{-\delta_\tau} \frac{N_\tau!}{n_\tau!}\, p(\mathbf{d}|\mathcal{O}_1). \qquad (4.5)
$$

By rearranging the order of summation and summing over the domain, $\mathcal{D}$, of possible values of $\mathbf{d}$, we obtain

$$
\begin{aligned}
R &= \sum_{d\in\mathcal{D}} \sum_{\mathbf{h}\in\mathcal{H}_d} \prod_{f=1}^{F} N_{t_f}^{-d_f} \prod_{\tau=1}^{T} M_\tau^{-\delta_\tau} \frac{N_\tau!}{n_\tau!}\, p(\mathbf{d}|\mathcal{O}_1) \\
&= \sum_{d\in\mathcal{D}} \prod_{\tau=1}^{T} \frac{N_\tau(N_\tau-1)\cdots(N_\tau-\delta_\tau+1)}{M_\tau^{\delta_\tau}}\, p(\mathbf{d}|\mathcal{O}_1) \qquad (4.6)
\end{aligned}
$$

This represents a classifier that relies only on counting the number of part candidates in the signal. In order to see how this works, we can assume that only one part of each type is included in the model. We then have

$$
R = \sum_{d\in\mathcal{D}} \prod_{f=1}^{F} \left(\frac{N_f}{M_f}\right)^{d_f} p(\mathbf{d}|\mathcal{O}_1). \qquad (4.7)
$$

Now, for a given hypothesized vector, $\mathbf{d}$, only those parts contribute to the ratio which are hypothesized to be present ($d_f = 1$). Each of these parts will "vote" in favor of class $\mathcal{O} = 1$, if more candidates have been found than appear on average in the background. It will vote for class $\mathcal{O} = 0$,

if less than average have been found. Summing over all possible values of $\mathbf{d}$ will compute the expectation of the votes with respect to $p(\mathbf{d}|\mathcal{O})$.

4.2.3 Localization

The second important recognition problem we are interested in is the problem of localizing an object within a signal. A concrete example is the problem of finding a face, a car, or any other physical object within an image.

This problem could be addressed after the presence of a foreground object in the signal has already been established using the method described in the previous section. Or one might have prior knowledge and be certain that an object is present.

After applying part detectors, a signal is reduced to a collection of part candidates. Hence, the localization problem becomes to identify those candidates that correspond to the foreground object. We have explored two different ways to achieve this. For now we will assume that we are looking for exactly one object.

Probably the most straightforward translation of the question "Which part candidates are most likely to belong to the foreground?" is to maximize the probability of the hypothesis vector, given the observed data. Since the question only makes sense if we assume that an object is present, we also condition on $\mathcal{O}_1$ and compute

$$\arg\max_{\mathbf{h}} p(\mathbf{h}|\mathcal{X}, \mathcal{O}_1). \tag{4.8}$$

Since this probability is proportional to the class-conditional density, $p(\mathbf{h}, \mathcal{X}|\mathcal{O}_1)$, which we can compute for any $\mathbf{h}$ using Equation 4.2, there are no further conceptual hurdles. We only need to generate all possible hypotheses and retain the one which yields the highest probability. The vector $\mathbf{h}$ will then point directly to the most likely foreground parts. Note that in this solution some or even all of the foreground parts might end up being declared missing. This method is equivalent to Burl's. It can only be applied if we know that an object is present. If the number of part candidates is different, comparing $\max_{\mathbf{h}} p(\mathbf{h}|\mathcal{X}, \mathcal{O}_1)$ across signals is not meaningful. Thus, no common threshold can be set to use this measure for detection.

A slightly different approach to the localization problem is to ask the question whether a *single part candidate* belongs to the foreground or to the background. The probability that candidate i of

part type τ is part of the foreground object is computed as follows

$$p((\mathcal{X})_{\tau,i} \in \text{foreground}) = \frac{\sum_{f \in \mathcal{F}(\tau)} \sum_{\mathbf{h} \in \mathcal{H}} \left(\delta_{h_f,i}\, p(\mathcal{X}, \mathbf{h}|\mathcal{O}_1)\right)}{\sum_{\mathbf{h} \in \mathcal{H}} p(\mathcal{X}, \mathbf{h}|\mathcal{O}_1)},$$

where we use $\mathcal{F}(\tau)$ to denote the set of all model parts that are of type τ and δ is the Kronecker-delta.

Finally, note that, using either of the above approaches, we cannot easily obtain the exact portion of the signal occupied by the foreground. This is, because in our model an object can be "more than the sum of its parts," i.e. it is likely that there are areas of the object which are not covered by any part. Therefore, if this information is important, some further work needs to be done in order to model the exact outline of an object as a function of its part positions and, possibly, other part attributes.

4.3 Computational Complexity

We are interested in the computational complexity of our method as a function of the number of parts in the model and the particular choice of foreground and background density. We can implement the formulae in equations 4.4 or 4.8 in such a way that we need to generate each hypothesis exactly once. If we assume that our model is composed of F parts and that we find exactly N part candidates in every signal, the number of hypothesis to be computed is $(N+1)^F = O(a^F)$. The most costly computations that need to be carried out for each hypothesis are, in the case of Gaussian or uniform geometry densities, matrix computations, involving matrices of size linear in F. All of these computations, such as the inversion of the covariance matrix, which is needed to evaluate Gaussian densities, have a computational cost in $O(F^k)$, for some small k. Therefore, the important limiting criterion is the exponential factor of $O(a^F)$. Hence, we conclude that the overall cost is exponential in the number of model parts. Note that it does not matter if several parts are of the same type or not.

In practice, we found that models with six parts can still be used for reasonably fast detection (with a computation time in the order of seconds). However, the exponential growth in F is a severe limitation, and there is no hope, at present, to use models with, say, ten features or more.

These findings motivate the search for more efficient detection methods, such as the one described in the next section.

4.4 Efficient Evaluation of Hypothesis Probability Density

When we use our model for detection or localization or when we learn constellation models (later in this thesis), we will often have to sum over or search through the entire set of hypotheses associated with one signal. In this section, we describe a way to do such computations more efficiently, by visiting hypotheses in a particular order and only computing the terms that actually change from one hypothesis to the next. At the same time, this procedure can be used to exclude a large set of hypotheses that do not contribute significantly to the computation at hand.

We first reformulate $p(\mathbf{h}, \mathcal{X}|\mathcal{O})$, in order to separate terms depending on individual model parts. We then show how this view can be used to render the hypothesis evaluation more efficient.

4.4.1 Reformulating $p(\mathbf{h}, \mathcal{X}|\mathcal{O})$

It will prove advantageous in the next subsection to break down $p(\mathbf{h}, \mathcal{X}|\mathcal{O})$ into separate terms according to their dependence on particular indices h_f. For simplicity we assume that only one part per type is present in the model. For every $\mathbf{h}$ we need to compute

$$p(\mathbf{h}, \mathcal{X}|\mathcal{O}) = p_{\text{fg}}(\mathcal{X}(\mathbf{h})) \prod_{f=1}^{F} p_{\text{Poiss}}(n_f) \prod_{f=1}^{F} A^{-n_f} \prod_{f=1}^{F} N_f^{-d_f}\, p(\mathbf{d}|\mathcal{O}).$$

For the null hypothesis, $\mathbf{h}_0$, we obtain

$$p(\mathbf{h}_0, \mathcal{X}|\mathcal{O}) = \prod_{f=1}^{F} \frac{p_{\text{Poiss}}(N_f)}{A^{N_f}} p(\mathbf{d}_0).$$

Using the fact that $\frac{p_{\text{Poiss}}(N)}{p_{\text{Poiss}}(N-1)} = \frac{M}{N}$, we can factor out the null hypothesis in the case of an arbitrary $\mathbf{d}$ and write

$$\begin{aligned} p(\mathbf{h}, \mathcal{X}|\mathcal{O}) &= \frac{p(\mathbf{d}|\mathcal{O})}{p(\mathbf{d}_0|\mathcal{O})} p_{\text{fg}}(\mathcal{X}(\mathbf{h})) \prod_{f=1}^{F} \left(\frac{A}{N_f}\right)^{d_f} \prod_{f=1}^{F} \left[\frac{p_{\text{Poiss}}(N_f)}{A^{N_f}} \left(\frac{N_f}{M_f}\right)^{d_f}\right] p(\mathbf{d}_0|\mathcal{O}) \\ &= \frac{p(\mathbf{d}|\mathcal{O})}{p(\mathbf{d}_0|\mathcal{O})} p_{\text{fg}}(\mathcal{X}(\mathbf{h})) \prod_{f=1}^{F} \left(\frac{A}{M_f}\right)^{d_f} \prod_{f=1}^{F} \frac{p_{\text{Poiss}}(N_f)}{A^{N_f}} p(\mathbf{d}_0|\mathcal{O}). \end{aligned}$$

We can also factor p_{fg} by conditioning part positions on the positions of parts corresponding to a smaller index, f. If a part is missing, we replace the corresponding term with one and omit further

conditioning on this part. We obtain

$$p(\mathbf{h}, \mathcal{X}|\mathcal{O}) = \frac{p(\mathbf{d}|\mathcal{O})}{p(\mathbf{d}_0|\mathcal{O})} \prod_{f=1}^{F} \left(p_{\text{fg}}(x_{h_f}|x_{h_1} \dots x_{h_{f-1}}) \frac{A}{M_f} \right)^{d_f} \prod_{f=1}^{F} \frac{p_{\text{Poiss}}(N_f)}{A^{N_f}} p(\mathbf{d}_0|\mathcal{O}).$$

If $p(\mathbf{d}|\mathcal{O})$ is assumed independent, it can be factored as well and we obtain

$$\begin{aligned} p(\mathbf{h}, \mathcal{X}|\mathcal{O}) &= \prod_{f=1}^{F} \left(p_{\text{fg}}(x_{h_f}|x_{h_1} \dots x_{h_{f-1}}) \frac{A}{M_f} \frac{p(d_f = 1)}{p(d_f = 0)} \right)^{d_f} \prod_{f=1}^{F} \frac{p_{\text{Poiss}}(N_f)}{A^{N_f}} p(d_f = 0) \\ &= \prod_{f=1}^{F} \left(p_{\text{fg}}(x_{h_f}|x_{h_1} \dots x_{h_{f-1}}) \alpha_f \right)^{d_f} K. \end{aligned}$$

Where K and $\alpha_f \stackrel{\text{def}}{=} \frac{A}{M_f} \frac{p(d_f=1)}{p(d_f=0)}$ are constant across hypotheses of one image.

Since we assume a joint Gaussian for p_{fg}, we have a simple dependence of the parameters in the factorization:

$$p_{\text{fg}}(x_f|x_1 \dots x_{f-1}) = G(x_f|\mu_f(x_1 \dots x_{f-1}), \Sigma_f) \stackrel{\text{def}}{=} G_f(x_f).$$

Note that due to the nature of the Gaussian density, only the mean of $p_{\text{fg}}(x_f)$ will depend on the candidate points with lower index, not the covariance matrix!

4.5 Simplification of Hypothesis Generation

Detection, localization, and the parameter estimation method we will later use to learn models all require that the entire set of hypotheses be generated and some computation involving $p(\mathbf{h}, \mathcal{X}|\mathcal{O})$ be carried out. We assume that we generate the hypothesis in a particular order, using $\mathbf{h}$ as a counter in such a way that the highest index, h_F, is increased at every step. When it reaches the maximum number of candidates, N_f, for the corresponding part, a "carry over" occurs and h_F will be reset to zero. At the same time the next lower index, h_{F-1} will be increased by one, and so forth. It follows that the first few indices of $\mathbf{h}$ are only updated very rarely.

Assuming that the first ϕ entries of a hypothesis $\mathbf{h}$ have assigned candidate positions or are missing, we can compute an upper bound on $p(\mathbf{h}, \mathcal{X}|\mathcal{O})$ that holds independently of which parts, if

any, are chosen for $h_{\phi+1} \ldots h_F$.

$$\begin{aligned} p(\mathbf{h}, \mathcal{X}|\mathcal{O}) &= \prod_{f=1}^{\phi} (G_f \alpha_f)^{d_f} K \cdot \prod_{f=\phi+1}^{F} (G_f \alpha_f)^{d_f} \\ &\leq \prod_{f=1}^{\phi} (G_f \alpha_f)^{d_f} K \cdot \prod_{f=\phi+1}^{F} \max_{x_f}(G_f(x_f)\alpha_f, 1). \end{aligned}$$

If we note that $\max_{x_f} G_f$ is simply $\frac{1}{(2\pi)^F \sqrt{|\Sigma_f|}} \stackrel{\text{def}}{=} D_f$, we obtain

$$p(\mathbf{h}, \mathcal{X}|\mathcal{O}) \leq \prod_{f=1}^{\phi} (G_f \alpha_f)^{d_f} K \cdot \prod_{f=\phi+1}^{F} \max(D_f \alpha_f, 1).$$

Since D_f does not depend on $(x_1 \ldots x_{f-1})$, the last factor also needs to be computed only once per image. We can now introduce a threshold which indicates the minimum value that $p(\mathbf{h}, \mathcal{X}|\mathcal{O})$ must take on, for the corresponding hypothesis to be included in the computation at hand. By examining a certain "prefix" – a choice for the first ϕ candidates – we can decide if all hypotheses starting with this prefix are to be rejected. The variable ϕ can be dynamical, in order to reject or accept hypotheses with prefixes of any length.

4.5.1 Implementation of Flexible Search Regions

The rejection of negligible hypotheses can be implemented through a look-up table, Ξ, which stores all entities which are constant throughout one signal. Thus,

$$\Xi(\mathbf{d}, \phi) = K \cdot \prod_{f=1}^{\phi} \alpha_f^{d_f} \max_{b_{\phi+1,\ldots,F}} \prod_{f=\phi+1}^{F} (\alpha_f D_f)^{d_f}.$$

Here, the maximum is taken over all b with the first ϕ bits fixed to the first ϕ bits of the table index.

If we now choose a threshold, T, indicating the minimum value of $p(\mathbf{h}, \mathcal{X}|\mathcal{O})$ for a hypothesis to be worthy of consideration, we can perform the necessary check as follows

$$\prod_{f=1}^{\phi} G_f^{d_f} \geq \frac{T}{\Xi(\mathbf{d}, \phi)}.$$

If any partial hypothesis with the first ϕ parts candidates chosen does not fulfill the above inequality, all hypotheses whith the same "prefix" can be rejected.

In the case of a joint density $p(\mathbf{d}|\mathcal{O})$ we can simply refrain from factoring $p(\mathbf{d}|\mathcal{O})$ and use instead

$$\Xi(\mathbf{d},\phi) = K \cdot \prod_{f=1}^{\phi} \beta_f^{d_f} \max_{b_{\phi+1},\ldots,F} p(\mathbf{d}|\mathcal{O}) \prod_{f=\phi+1}^{F} (\beta_f D_f)^{d_f},$$

with $\beta_f \stackrel{\text{def}}{=} \frac{A}{M_f}$.

If, during hypothesis generation, the first ϕ candidates of a hypothesis have been assigned, the above method provides search regions for the remaining candidates, depending on the chosen threshold T and the exact positions of the already assigned candidates. The advantage is that an exact upper is chosen for the probability of a rejected hypothesis. Since it is simple to keep track of the exact number of rejected hypotheses, error bars can be computed a posteriori for the entire computation at hand.

4.6 Conclusion

We have shown how the problem of object detection can be solved with a standard MAP approach. The solution involves a summation over all possible hypotheses. The localization problem can be solved by generating all hypotheses, in order to find the one maximizing the conditional probability $p(\mathbf{h}|\mathcal{X},\mathcal{O}_1)$. This is very similar to the solution proposed by Burl et al. Both problems could be solve simultaneously, generating each hypothesis exactly once. Since the number of hypotheses grows exponentially with the number of model features, F, we need to find more efficient ways to solve the detection and localization problem. One such method has been presented in this chapter. It can be used to explicitly exlude all hypotheses with a probability below a given threshold. This allows to solve the localization problem exactly, whithout the need to generate all hypotheses. For example, when experimenting with models comprising six features, less than 1% of the total number of hypotheses needed to be computed. For the detection problem, we can compute an accurate approximation of the sum over all hypotheses, which will slightly underestimate the correct value. However, an upper bound on the correct value can be computed as well.

Part II

Learning Constellation Models

Chapter 5 Introduction to Model Learning

5.1 Introduction

We have discussed the constellation model for object classes and how it can be used to solve recognition problems. The underlying fundamental ideas, as well as some details of the model, were previously introduced by others. This second part of the thesis is dedicated to the problem of learning object class-models from examples of signals with minimal supervision, a problem that humans seem to be able to solve with relatively little effort in many cases, but that was mostly unsolved and rarely addressed in the machine vision literature.

In this short chapter, we place our particular problem in the context of learning theory, paying particular attention to the differences between *supervised* and *unsupervised* learning. We then give an overview of our chosen approach, before explaining our method, which we have published in [WWP00b, WEWP00], in detail throughout the following chapters.

5.2 Model Learning

We can see the problem of learning constellation models from two slightly different perspectives.

From a practical standpoint, one is interested in constructing a model for a given detection task at hand. In this case, the class of objects to be detected is known and example signals containing such objects are likely to be available. The task then is to define a set of discriminative and informative parts and to determine the entire set of remaining model parameters – such as the parameters of the pdf defining the global geometry – while the ultimate goal is to build the most useful or powerful system for the problem at hand. We refer to this scenario as the *weakly supervised* setting, since it is assumed that all of the training signals contain an instance of the object class of interest, although the learning process itself might not require any further intervention by a human operator. Figure 5.1 illustrates the weakly supervised setting.

The second perspective is motivated by a more fundamental research interest, although important practical applications exist also here, such as the problem of organizing and searching large databases of images. It is a very intriguing idea to develop a method which would be able to dis-

cover everything there is to know about the world "from scratch." This would include learning what classes of objects there are, and what the defining criteria for class membership are, all without any prior knowledge. If this is to be done based on training data, we would obviously not be allowed to provide any labels whatsoever as to the content of the training signals. We therefore call this the *strictly unsupervised* scenario. Note that this includes the case where the training data consist of signals corresponding to only a single object class, mixed with signals containing only background clutter, while the exact mixing ratio is unknown.

There are two reasons why it is generally desirable to automate the learning process as much as possible. Firstly, it requires skill and time of a human operator to make the necessary design decisions. Secondly, while it is certainly possible for an operator to assist the learning process, e.g., by outlining the target objects in training signals, such choices will influence the final performance of the recognition system in a very complex way, into which even the most skilled operator has only limited insight. As an example, consider the difficult question of the size and scale (sampling resolution) of parts to be used. Hence, a system designed manually is likely to be inferior in performance to the best one. Therefore, it is our declared goal to minimize user intervention in the learning process.

5.3 The Context of Statistical Learning Theory

One reason why we formulated our object class model as a probabilistic model is that this allows us to make use of concepts and tools from the field of statistical learning theory. In order to construct specific models based on a collection of training signals we face the following problems.

Detector Design Part detectors need to be designed or trained based on examples.

Part/Detector Selection The detectors corresponding to the most stable and informative parts need to be chosen, since only a limited number of detectors can be included in the final model.

Parameter Estimation The remaining parameters of the model need to be estimated, given the chosen set of detectors.

Performance Evaluation The performance of the final model needs to be evaluated, or it should at least be predictable to some extent.

A part detector is really an entire object recognition system of its own. In our work we rely on the well known technique of normalized correlation with a single template to detect parts of a given

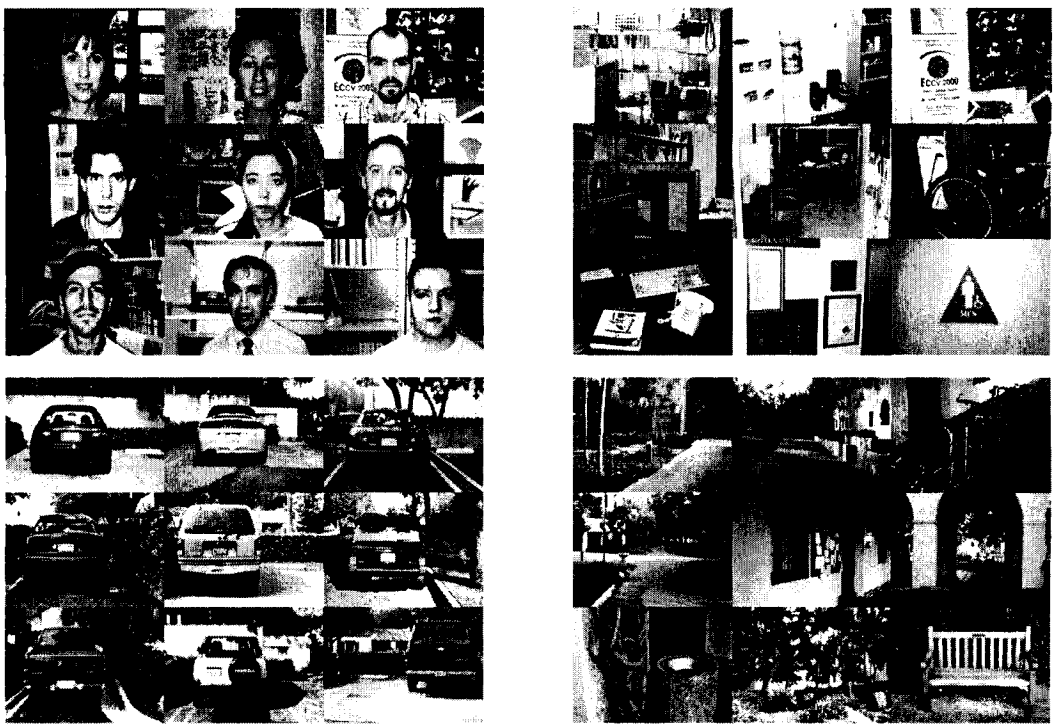


Figure 5.1: Illustration of the *weakly supervised* scenario. Which objects appear consistently in the left images? Can a machine learn to recognize instances of the two object classes (*faces* and *cars*), if the only further information provided is a set of images that do *not* contain any instances of the object classes?

type. The remaining problem then, which we address in the next chapter, is to select or synthesize templates for the detectors, based on the training images.

Our problem of part selection is very similar to the standard problem of feature selection as it is commonly encountered when training neural networks and similar learning models. We will consider some standard methods developed in this context in the next chapter. In deciding on the parts a model will comprise, we decide on the amount and type of information to be extracted from signals. This step is crucial, since it will implicitly set an upper bound on the achievable recognition performance. Furthermore, the *complexity* of a model will largely depend on the *number* of parts used. Comparing and choosing between models of different complexity or explanatory power is the standard problem of *model selection*. The fundamental question is how complex a model we can "afford," in order not to *overfit* a given, limited amount of training data. This topic has generally received less attention and the only convincing framework developed to date is the *Bayesian* approach (see, e.g., [Mac92] for a brief introduction). However, we have not yet adapted this approach to our problem and use instead some simple measures that allow us to choose the right model complexity for a given amount of training data.

Once detectors are chosen, estimating the parameters of a constellation model means to estimate probability density functions – the very essence of statistical learning theory. Since we will restrict our presentation of model learning to the case of Gaussian foreground densities, we can apply the theory developed for *parametric density estimation*, in particular the *maximum likelihood* (ML) framework. Aside from some problems inherent to ML (such as it being ill-defined), we will have to cope with its limited usefulness when comparing potential models in light of the training data. ML is only meaningful when considering one *specific* set of data in order to choose one pdf (the best "fit") out of a single class of model densities. As a consequence, ML per se will not be a meaningful criterion to compare two models with different part detectors, even if the total number of detectors is the same in both models. This is because different detectors essentially perform different measurements on the training data resulting in a different data set to be used for density estimation.

A globally optimal solution to the entire learning problem can only be found by considering all above problems jointly. Consider, for example, that when choosing detectors, we do not know how reliably and with how much spatial variability the underlying object parts occur throughout the object class. Unfortunately, no framework exists to date that would provide a unified solution to our problem. We propose to use an iterative technique to solve the part selection and parameter

estimation problem jointly.

Before we present our concrete approach to the learning problem, we will have a closer look at the important notion of unsupervised learning, upon which we already touched in the previous section.

5.3.1 Supervised vs. Unsupervised Learning

We have motivated the distinction of different degrees of supervision from the practical perspective of operator intervention. However, the notions of supervised and unsupervised learning are deeply rooted in learning theory. Supervised learning is defined as learning in the presence of ground truth, i.e. when training data are *labeled* with respect to their class membership. Unsupervised learning, on the other hand, is learning from unlabeled data. In the context of classification, unsupervised learning is equivalent to the problem of *clustering*. The goal of clustering is to divide a set of unlabeled data into classes, or *clusters*. One possibility to perform unsupervised clustering is to directly fit a model of a multi-modal probability density, such as a mixture of Gaussians, to the training data. In principle, it is not difficult to extend any probability model to a mixture model. We apply this idea to constellation models in Chapter 8. We also face an unsupervised clustering problem in Chapter 6, when we select patterns to train part detectors.

Probably the most fundamental difference between supervised and unsupervised learning is that unsupervised learning can only focus on *representing* the training data as well as possible. If we have performed clustering in an unsupervised fashion and later want to build a classifier based on the clusters we found, we can only hope that this classifier will perform well. Since we have no access to ground truth, we cannot validate the classifier or otherwise predict its performance. It is therefore crucial that the model we use to represent the training data be accurate, so that we can rely on the results of the clustering process. In supervised learning, on the other hand, we can focus directly on *discriminating* between the different classes in the training set and if a sufficient amount of data is available, the classification error on the training set will predict the classification error on unseen data well.

Where do our above problems, the "weakly unsupervised" and the "strictly unsupervised," fit into the classical picture? On one hand, our weakly supervised setting corresponds to a standard supervised setting since, in stating that all training signals contain a foreground object, we have labeled these images as members of class $\mathcal{O}_1$ (the class of foreground images). One might be tempted to collect a second set of images which contain only background clutter, i.e. samples of

class $\mathcal{O}_0$, and present both sets to a supervised learning method. However, this is not possible. Although the foreground images each contain an object, they also contain background clutter – in even larger proportion. Since we do not know which is which, the set of samples from $\mathcal{O}_1$ alone, already embodies the entire learning problem! Without labels that would assign each *part candidate* to either foreground or background, the problem is in fact almost completely unsupervised. Aside from the part positions and types, the only additional information we have, is that there can be at most one object in each signal. This is, however, a very weak constraint. The conclusion then is that our problem sits somewhere in between the classical supervised and unsupervised case. It is very similar to a standard unsupervised clustering problem, where no class labels are available, but some additional constraint is provided, such as the maximum number of data points that can be assigned to one of the clusters. We therefore believe that the term "weakly supervised" is also meaningful in the standard context and we will continue to use it. It should now be easy to see that our strictly supervised problem really is a problem of unsupervised learning in the classical sense, since no additional information is available in this case.

5.4 Proposed Approach to Model Learning

We propose to the solve the learning problem as illustrated in Figure 5.2. Since we assumed that no part detectors are provided, we first need to construct those detectors. We therefore start from the assumption that any pattern in any training image could serve as a prototype for a model part. We reduce this huge set of candidates by first filtering out "interesting," highly textured patterns based on a standard interest operator and then performing an unsupervised clustering step. We refer to these steps as "part pre-selection." They will produce about 100–200 clusters of patterns. We then choose the average pattern of each cluster as a template for a detector based on normalized correlation. One could also imagine to train more elaborate detectors, using the patterns of each cluster as training data. The set of detectors, or parts, pre-selected in this way is now of a manageable size for standard feature selection methods to be applied. Since we can only use a limited number of parts in the final model, we use a selection method which iteratively evaluates and updates small, promising sets of parts, to which we refer as *model configurations*. We already mentioned that the final part selection cannot be done optimally without considering a completely trained model. This means that the ultimate goodness criterion for any given choice of parts is the performance of the constellation model built with these parts. We therfore learn an entire constellation model at

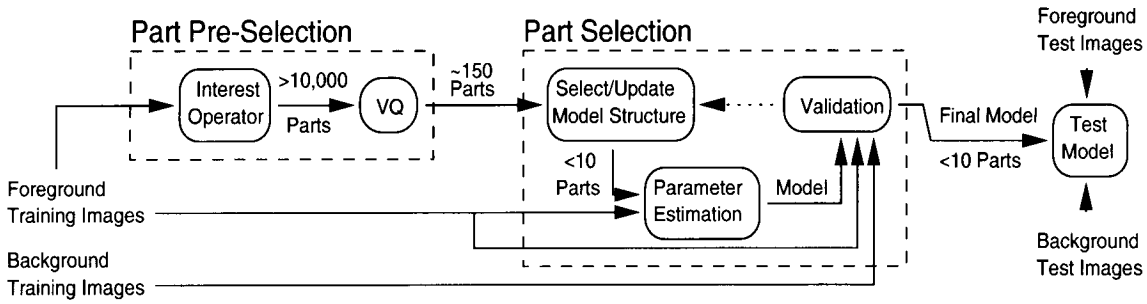


Figure 5.2: Block diagram of our method. "Foreground images" are images of class $\mathcal{O}_1$, containing the target objects embedded in cluttered background. "Background images" are of class $\mathcal{O}_0$ and contain background clutter only.

every iteration of the part selection process. For a given choice of parts, the remaining model parameters are estimated using *expectation maximization*. This step implicitly addresses the problem of deciding which portions of the trainig images contain objects and which clutter. Since this is an unsupervised learning problem in itself, a representation based method is needed. Expectation maximization (EM) is precisely such a method; it fits our probability model to the training data based on the maximum likelihood criterion. We will need to evaluate each complete constellation model as to its expected classification performance. This can be done in several ways. One possibility is to set aside part of the input signals before the parameter estimation, combine them with a set of signals that do not contain any objects, and compute the classification error on this *validation set*. We do not believe that this requirement for an additional set of signals with background clutter is a serious restriction, nor does it fundamentally increase the degree of supervision. One could assume that these images are selected at random from some large set of images that is "incorporated" into the learning method once and forever. We explore other possibilities of evaluating a trained model in Chapter 7.

Our part selection process in discussed in the following chapter. The parameter estimation method is presented in Chapter 7.

Chapter 6 Part Selection

6.1 Introduction

Selecting a set of parts for a constellation model is equivalent to selecting a set of detectors that will be used to identify those parts in signals. For some detectors, it is possible to determine what a prototypical part would "look like," while for others this notion might not be meaningful. This is because such detectors might have been trained on a large and possibly diverse set of training patterns or they might have been designed from first principles, as is often the case, e.g., for edge or corner finders.

Although we could easily incorporate pre-designed part detectors into our learning method, we start here from the premise that such detectors are not available and that the parts themselves need to be identified in the training signals in an unsupervised manner. Hence, we start from the simple, if intimidating, perspective that *any* pattern in *any* training image could serve as a prototype for a model part. Depending on the sophistication of the part detectors a single prototype could already define a detector, e.g., if matched filters are used. For more elaborate detection schemes an entire training set of patterns might have to be extracted from the training signals. So far, we have only explored the former idea, using normalized correlation as detection method.

We select the best set of part detectors in two stages. In a preselection stage, the extremeley large set of potential correlation templates is reduced using an interest operator followed by a clustering step. This is done without considering the detection performance of the detectors. Small sets of the remaining templates are then evaluated, in the second selection stage, as to their joint performance within a fully trained constellation model. Thus model parameters need to be estimated at every iteration of the second stage.

6.2 Part Selection

6.2.1 Generic vs. Specialized Parts

If we envision a "universal" object recognition system, capable of identifying a very large number of object classes, it appears highly inefficient to endow such a system with specialized part detectors

for each class. One reason for this is that part detection is computationally expensive. A more sensible approach would therefore be to establish a collection of *generic* parts, such as corners or line endings in images. Different combinations of those parts could then be used as building blocks to represent objects of very different classes. For the purpose of detecting or recognition, it would be sufficient to apply the entire set of generic detectors only once to a given input signal. We have taken such an approach when we detected cars in images based on simple line features [FW]. In order to identify line segments in images, we used a generic line finding method.

The method presented below produces a set of parts which is biased towards the foreground object class to be learned. The reason for this lies only in the choice of training signals, which, in our experiments all contain instances of a single object class. When applied to a homogenous set of very different objects, our method should be capable of producing a set of generic parts, although we have not experimented with this idea.

6.3 Automatic Part Selection

6.3.1 Approach

The problem of part selection for constellation models is very similar to the classical problem of *feature selection* addressed in the pattern recognition literature, e.g., in [Bis95, Kit78, FPHK94]. We will argue in Section 6.3.4 that important results of this research are applicable to our case. In the standard feature selection scenario, features correspond to measurements to be performed on the objects or patterns one seeks to classify. The goal of feature selection is to select, out of a large set of features, an optimal subset in the sense that the chosen combination of features provides the subsequent classification process with a maximum of information relevant to the classification problem. This problem is challenging for two reasons. Firstly, a particular feature might be extremeley useful in connection with some features, while it is of no use when combined with others. For example, the best pair of features does not have to contain the single best feature [Cov74]. Secondly, the only way of assessing the quality of a selection of features is typically to actually make use of them by training and testing under the classification method for which they are considered.

In order to formalize our problem, let Y be a given set of features or parts, with cardinality d and let F represent the desired number of parts in the selected subset X, $X \subseteq Y$. We then need to define a criterion function $J(X)$ to guide the selection process. We assume that a greater value of J indicates a better subset X. A typical choice is $J(X) = 1 - P_e$, where P_e is the probability

of a classification error of the final classifier. Note that this makes the part choice dependent on some training and test data and thus susceptible to problems such as overfitting. It has been shown [CvC77] that in general no selection method short of testing all $\binom{d}{m}$ possible combinations[1] and choosing the one with maximum J is guaranteed to produce the optimal subset. For most applications, including ours, this strategy is entirely infeasible. If we take the – rather optimistic – position that we are able to evaluate close to 5000 different part choices through training and testing of a constellation model, we could then, for example, choose the best four out of only 20 feature candidates through exhaustive testing. On the other hand, a single training image might already provide us with more than 10^4 part candidates. Although more efficient alternative methods have been proposed, none can handle this large a number of candidates. We therefore introduce two further steps to reduce the number of candidates before we consider some standard feature selection strategies.

The first step is to preselect "interesting" parts using a standard interest operator that selects highly textured regions in the training signals. The second step groups part candidates using a clustering method. It produces about 100–200 clusters. Each cluster could be used to train a part detector. The approach we have taken is to simply retain the mean pattern of each cluster to be used as a template for normalized grayscale correlation.

We have tried the procedures explained in the remainder of this chapter only on image data.

6.3.2 Preselection of Patterns of Interest

In the first selection step, we identify *points of interest* in training images (see Fig. 6.1). This is done using the interest operator proposed by Förstner [HS93], which is capable of detecting corner points, intersections of two or more lines, as well as center points of circular patterns. This step produces about 150 part candidates per training image.

6.3.3 Clustering of Interesting Patterns

The set of patterns preselected by the interest operator will contain many subsets of similar patterns, in particular if every training image contains an instance of the object class of interest. Similar patterns might also arise from the background clutter, especially when the pattern size is small.

In order to identify similar patterns we employ a clustering method (a similar procedure was used by Leung and Malik in [LM99]). Although clustering is a standard procedure in the field of

[1] For simplicity, and to maintain the analogy with standard feature selection, we assume for now that each part type is chosen at most once.

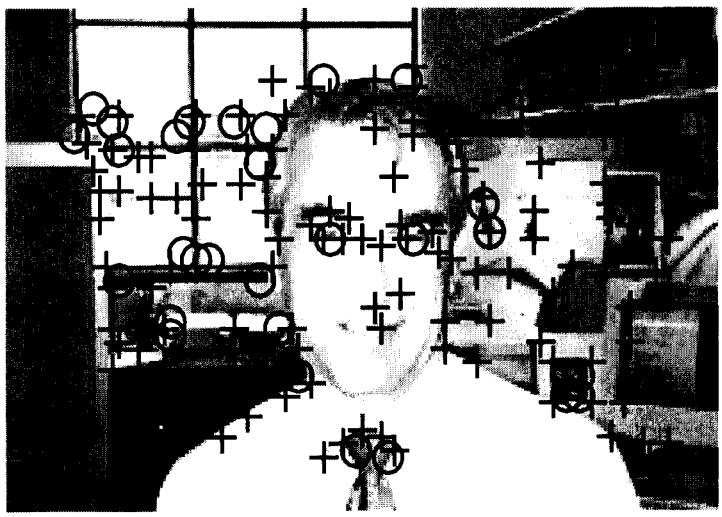


Figure 6.1: Points of interest identified on an unsegmented training image of a human face in cluttered background. We used Förstner's method, which can identify corner type (marked with '+') and circular (marked with 'o') patterns. This selection step reduces the set of potential parts from 40,000 to about 200 per training image.

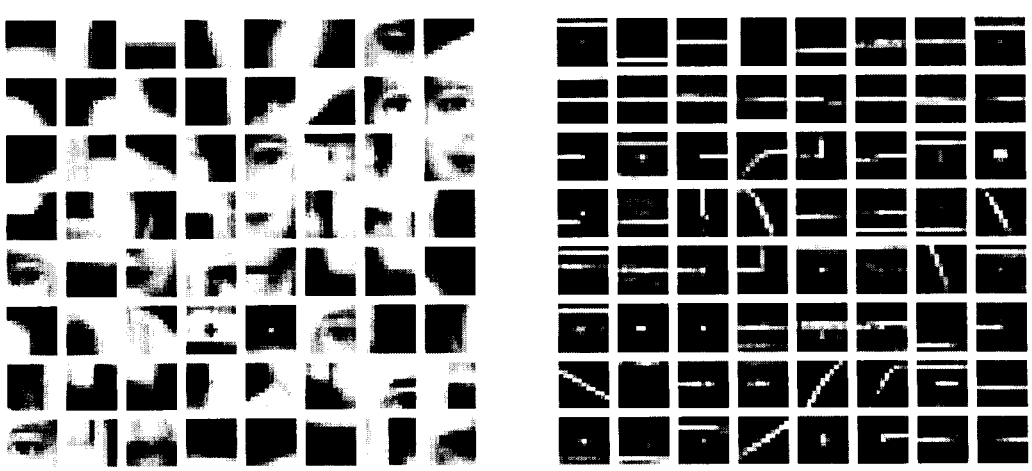


Figure 6.2: A sample of the patterns obtained using k-means clustering of small image patches is shown for faces (center) and cars (right). The car images were high-pass filtered before the part selection process. The total number of patterns selected were 81 for faces and 80 for cars.

unsupervised learning, no principled approach exists that addresses all related issues in a satisfactory manner. Even the definition of a *cluster* is quite fuzzy, and in most applications no true or correct clustering solution exists. The goal is to group data in such a way that the scatter within a group is minimized while the scatter between clusters is maximized. To this end different goodness criteria can be chosen. Many algorithms require the number of clusters to be determined beforehand. Finding this number is a notoriously difficult problem. Jain et al. provide the following "state-of-the-art" procedure for iterative partitional clustering:

1. Select an initial partition with K clusters. Repeat steps 2 through 5 until the cluster membership stabilizes.
2. Generate a new partition by assigning each pattern to its closest cluster center.
3. Compute new cluster centers as the centroids of the clusters.
4. Repeat steps 2 and 3 until an optimum value for the criterion function is found.
5. Adjust the number of clusters by merging and splitting existing clusters or by removing small, or outlier, clusters.

The above procedure could, in principle, find an optimal number of clusters automatically. Further heuristics are needed to determine how to split and merge clusters. Steps 2 and 3 correspond to the very efficient and widely used k-means clustering algorithm [DH73].

We have been able to obtain very good results with the following simplified "one-pass" method:

1. Select a random initial partition with K clusters. We set $K = 150$ in our experiments.
2. Generate a new partition by assigning each pattern to its closest cluster center. We use euclidian distance in pattern space.
3. Compute new cluster centers as the centroids of the clusters.
4. Repeat steps 2 and 3 until cluster membership is stable.
5. Split clusters for which $\mathrm{trace}(C) > T_1$, where C is the covariance matrix of the points in the cluster and T_1 is some appropriate threshold.
6. Merge two clusters, i and j, if $||\mu_i - \tau(\mu_j)|| < T_2$, where $\tau()$ is a small translation in an arbitrary direction.

7. Remove clusters with less than T_3 members. A typical choice for T_3 would be one half the number of training signals.

As with the "standard" algorithms, heuristics are needed to choose the thresholds, T_1, T_2, and T_3, for this procedure. We tuned these parameters, such that, in our experiments with image data, 80–120 clusters were produced for a set of 100 training images. Due to the initial restriction to points of interest the set of preselected patterns exhibits interesting structure, as is shown in Figure 6.2. Some parts, such as human eyes, can be readily identified. Other parts, such as simple corners, result as averages of larger clusters, often containing thousands of patterns.

This procedure dramatically reduces the number of candidate parts. However, both, parts corresponding to the background and foreground portions of the training images are still present.

Choice of Part Size

As with part selection in general, the choice of part size is also an empirical one. When the size is chosen too small, the parts tend not to be informative enough and too many false alarms are produced. When the size is too large, part detectors do not work well across different objects – they are to specialized.

These findings are best illustrated with our multi-resolution experiment in Section 9.4, where parts are selected automatically across different scales. There, an intermediate part size was produced automatically.

6.3.4 Final Part Selection Stage

Jain et al. have evaluated [JZ97] several feature selection methods that attempt to avoid an exhaustive search:

Best Individual Features Evaluate all d features individually and select the F best individual features. As confirmed by our experiments, this method is known to produce poor results.

Sequential Forward Selection (SFS) Start with the single best feature and add features, one at a time, which maximize J in combination with the already selected ones. We have successfully tried this method.

Sequential Backward Selection (SBS) Start with all features and successively delete one feature at a time. This method is infeasible for us, since we cannot train models with a large number

of parts.

Add-l-Take-Away-r Enlarge the selection by l features using SFS, then eliminate r features using SBS. This is our method of choice, with $l = r = 1$.

Sequential Forward (Backward) Floating Search (SFFS or SBFS) A generalization of the previous method where l and r are determined in an automatic fashion.

Branch and Bound This method is guaranteed to find the optimal subset if a monotonicity property (see below) is satisfied. Since the complexity might still be exponential in d and the gain over SFFS (below) is small in most cases, we have not implemented this method.

We have experimented with most of the above methods. Since the complexity of our learning algorithm growths exponentially with the number of parts (see Section 7.6.1), we are not able to train models with more than 6 parts in a reasonable amount of time. This excludes the SBS and SBFS methods for our problem, while the "Add-l-Take-Away-r" and SFFS methods can only be used for very small l.

We implemented the *best individual features* strategy by first ranking all parts based on their classification performance when used alone. Since we do not know the position of the foreground objects in the training images, our models need to be invariant to translation. For a single part model this means that no information whatsoever can be extracted from the position of the part in the signal and *presence* of the part becomes the only criterion for classification. Hence, it is straightforward to evaluate the single part performance by sorting a test set of signals according to the number of detections. We found that models based on the best F parts chosen in such a manner rarely produced a good classification performance. This result supports the findings, for example in [Cov74], that single part performance is not a good criterion for part selection.

A slightly more sophisticated strategy is *sequential forward selection*. Starting from the best single part, one adds one part at a time, by exhaustively trying all parts in order to find the one that most improves performance. We were able to obtain significantly better models with this method.

The best performing method in our case was the Add-l-Take-Away-r procedure, which we use for $l = r = 1$, due to our limitation to few parts. We initialize the choice, X, with a random selection of F parts. We then tentatively replace a single part by a randomly chosen one and test the model performance. If it increases the replacement part is kept. We repeat this step until no further improvements are possible.

The only method that can find the optimal part set without an exhaustive search is the widely used *branch and bound* method. Since it is known to often produce only marginal improvements over SFFS [JDM00], and since its worst case complexity is still exponential, we have not yet implemented this method. However, we will briefly discuss why it could, in principle, be applied to our problem. The branch-and-bound method requires the following monotonicity criterion to hold

$$J(X^+) \geq J(X) \qquad \text{if} \quad X \subset X^+.$$

It essentially means that adding parts to a model can only increase, but never decrease, the performance of the model. This criterion is fullfilled in a great number of problems and the gain in efficiency of the branch-and-bound method stems from the fact that, if a given set, X', has been shown to be inferior to the current best choice, then no subset of X' needs to be tested (see [Bis95] for an illustration). The gain is thus more significant when larger sets need to be selected. This is another reason why we have not implemented this method.

It is nevertheless useful to realize that the above monotonicity criterion holds for our constellation models. In order to show this, we will demonstrate that a given constellation model, M, with F parts, can implement any model M' that is based on a subset of the parts of M. This can be achieved through modification of M by "switching off" the parts that are not used in M'. Let M^* refer to this modified model. A part can be switched off, or eliminated, by setting the corresponding probability that this part is present in the signal to zero. One therefore needs to modify the probability mass table $p(\mathbf{d})$ (see Section 3.2.1). This will eliminate the part in question from the foreground model and its related geometry information will become superfluous. During detection, hypothesis involving the part will not have to be considered any longer, since they would be assigned a probability of zero through $p(\mathbf{d})$. However, the part will still be present in the background portion of M^*. This can have two possible consequences. Either, if the part was contributing to the recognition performance of M, it might still aid in classification through its mere presence in the signal, which is exploited through the background model of M^*, or it will not influence the classification results. In either case the performance of M^* is not inferior to that of M' and, hence, the monotonicty criterion will hold.

A different possibility to render the selection process more efficient and to ultimately allow for learning of larger models is to replace the objective function $J(X)$ by a simpler one. It is generally not a good idea to use a different classification method for feature selection than the one used in

the final classifier, since the performance of a simpler classifier is rarley a good predictor of the final performance. However, this risk might be outweighed in our case by the possible gain from a larger number of parts. A candidate for a simpler function J is the classification performance of a "geometry-free" constellation model. Most of the computational cost of training a model is due to the iterative learning of the geometry, which involves considering many possible hypothesis of foreground candidates (as explained in the next chapter). This cost could be eliminated by evaluating a choice of parts only with respect to its "counting performance" (see Section 4.2.2). The only information used for classification would then be the *joint presence* of candidates for the parts in question. Similarly to the procedure used above to establish the single part performance, a set of parts could be evaluated very rapidly, simply by counting the number of candidates for each part type.

Number of Model Parts

An important question that the selection methods discussed above do not answer is with how many parts to endow our model. Although, as argued in the previous section, a model with more parts performs better in theory, there are two important reasons why one might want to limit the number of parts. The first is that a large number of parts increases the computational cost of the detection process, since the number of hypothesis that need to be considered grows as $O(a^F)$. The second reason is that a larger model will require a larger set of training examples in order to yield a good *generalization* performance.

6.4 Conclusion

We have presented our part preselection method, based on a standard interest operator and a k-means clustering step, as well as the final selection procedure, which involves building and testing complete constellation models. Drawing the analogy to standard feature selection methods, we have argued that part selection cannot be done without evaluating a large number of potential part choices. Although our proposed method does not need to evaluate all possible choices, the number of which grows at least exponentially with the number of model parts (for small F and large N), this is a severe limitation. It would therefore be useful to find another criterion that could be used to evaluate the quality of a choice of parts. However, even if such a criterion exists, it will almost certainly be sub-optimal. It is therefore not clear if a net gain in classification performance will

result for a given amount of training time, even if a less complex criterion will allow to learn models with more parts.

Chapter 7 Parameter Estimation

7.1 Introduction

In this chapter, we address the problem of estimating the probability density functions underlying the constellation model for a given set of model parts. In Section 7.4, we derive update rules for our estimation method without any invariance assumptions. Translation invariance is addressed in Section 7.5. We only treat the case of a Gaussian foreground density in this chapter. The assumptions about available information are consistent with the "weakly supervised" setting. A solution to the "strictly unsupervised" problem is discussed in the next chapter.

7.2 Problem

We assume that a set of training signals, $\{S_i\}$, is provided, each containing one instance of the object class to be learned, embedded in background clutter. Furthermore, a set of part detectors has been chosen and is assumed fixed throughout this chapter. We can then cast the model learning problem formally as the estimation of the parameters, θ, of the model density

$$p(\mathcal{X}, \mathbf{h}|\mathcal{O}_1, \theta), \tag{7.1}$$

from the set of training signals. The parameters to be estimated, $\theta = \{\mu,\ \Sigma,\ p(\mathbf{d}),\ \mathbf{M}\}$, are the mean vector, μ, and covariance matrix, Σ, of the density of foreground part positions, the probability mass table, $p(\mathbf{d})$, governing the presence of parts, as well as the average number, $\mathbf{M}$, of background detections for each part type. Note that it is sufficient to only estimate the class conditional density for $\mathcal{O}_1$, as indicated above. Since the background model is already included in this density, the class density for $\mathcal{O}_0$ is obtained by replacing $p(\mathbf{d}|\mathcal{O}_1)$ with $p(\mathbf{d}|\mathcal{O}_0)$, which does not contain any parameters. We will not explicitly write the conditioning on $\mathcal{O}_1$ in the remainder of this chapter.

Supervised Setting

Although we are ultimately interested in an unsupervised learning method, we briefly sketch how the learning problem could be approached in a completely supervised setting. As discussed in Chapter 5, our understanding of a supervised learning scheme is that information about the position of the object and, possibly, individual object features in the training signals is provided. If such information is available, for example in the form of a complete segmentation of the training signals into foreground object and background clutter, the solution to our problem is relatively straightforward. We could obtain samples of the model density by applying the part detectors to each training signal, S_i, producing a set of detections, $\mathcal{X}_i$. Since it is known which detections correspond to the foreground, an index $\mathbf{h}_i$ could be formed accordingly. Based on these samples, the model parameters can then be estimated as follows. By averaging the number of background detections across training signals, $\mathbf{M}$ can be computed. Using only the foreground detections, labeled as such by $\mathbf{h}$, one can estimate the parameters of p_{fg}, and $p(\mathbf{d})$.

"Wildcard" Parts

A problem will arise under the above supervised approach if a single part detector produces multiple detections in the foreground. In this case, one would not know which ones to use, if any, in order to estimate p_{fg} and $p(\mathbf{d})$. A natural way to proceed would be to treat the false foreground detections as background noise. However, this would still require intervention of a supervisor who would have to (1) decide which detections are the "true" foreground detections and to (2) label each individual detection accordingly. Solving the second problem can be extremely tedious, while the first one is ill-defined. Assume, for example, that, in the case of face detection, a particular part detector responds well to the corner of an eye on some images, and to the corner of an eyebrow on others. It could be advantageous to exploit this ambiguity and to declare the different physical features to be the correct match in each case, instead of insisting that either eye or eyebrow corners should be detected, but not both. To determine if allowing for such *wild card* features is ultimately going to benefit the performance of the detection system, one would have to take into consideration the effects on all model prameters. For example, allowing a single model part to match different physical features will tend to increase the amount of geometric variability that needs to be accounted for.

These difficulties further motivate our search for an unsupervised solution to the learning problem that would automatically assign detections to foreground or background in an optimal way, such

that the most discriminative and informative locations on the objects will be implicitly declared the "true" foreground parts.

Unsupervised Setting

In the weakly and strictly unsupervised settings, the only information that can be extracted from a training signal, S_i, is the set of detections, $\mathcal{X}_i$. One does not have any access to the value of $\mathbf{h}_i$. We have encountered the same *hidden data* problem in Section 4.2.2, when trying to solve the detection task. There, we could simply integrate out the hidden information. In order to solve the learning problem, a more sophisticated approach is needed.

7.3 Approach

Although our problem might seem significantly more challenging than a typical hidden data problem, we will show in the following that it can be solved with the method of *expectation maximization* (EM). This method allows us to estimate iteratively the parameters of the model density as well as conditional probability densities over the missing data. Once the model has been fit, we can estimate the missing information, for example, by considering expected values of the missing data under the model density.

As we mentioned in Chapter 5, estimating the model parameters from a set of images of which we know that an object is present in each instance, is still an unsupervised problem, since individual part candidates do not carry any foreground or background label. We can therefore only focus on *representing* the training images as well as possible with our constellation model.

The standard criterion for parametric density estimation is the *likelihood* of the model parameters, given the training data. EM uses this criterion as well.

7.4 Parameter Estimation with EM

In the previous section, we described our learning problem as a maximum likelihood density estimation problem, with the additional complication of hidden data. The EM algorithm has been proposed[DLR76] to solve precisely this type of problem. Hence, we will use it to produce maximum likelihood estimates of the model parameters, $\theta = \{\mu,\ \Sigma,\ p(\mathbf{d}),\ \mathbf{M}\}$. We proceed in standard

EM fashion by maximizing the likelihood of the observed data, given the model parameters

$$L(\{\mathcal{X}_i\}|\theta) = \sum_{i=1}^{I} \log \sum_{\mathbf{h}_i} \int p(\mathcal{X}_i, \mathbf{x}_i^{\mathrm{m}}, \mathbf{h}_i|\theta)\, d\mathbf{x}_i^{\mathrm{m}}.$$

We have explicitly integrated out the entire set of hidden data which also includes the positions, $\mathbf{x}^{\mathrm{m}}$, of the foreground parts that are declared missing under a given hypothesis. It is difficult to tackle this optimization problem directly, in particular since it involves logarithms of sums and integrals. The EM algorithm provides an indirect solution by iteratively maximizing a sequence of functions

$$Q(\tilde{\theta}|\theta) = \sum_{i=1}^{I} E[\log p(\mathcal{X}_i, \mathbf{x}_i^{\mathrm{m}}, \mathbf{h}_i|\tilde{\theta})],$$

where $E[.]$ refers to the expectation with respect to the posterior $p(\mathbf{h}_i, \mathbf{x}_i^{\mathrm{m}}|\mathcal{X}_i, \theta)$. Throughout this section, a tilde denotes parameters we are optimizing for, while the absence of a tilde implies that the values from the previous iteration are substituted. EM theory guarantees that subsequent maximization of the $Q(\tilde{\theta}|\theta)$ converges to a local maximum of L.

The EM algorithm is divided into an 'E-step' and an 'M-step.' In the E-step, the posterior density is computed, while the M-step maximizes Q based on the posterior. We now derive update rules that will be used in the M-step. The parameters we need to consider are those of the Gaussian governing the distribution of the foreground parts, μ and Σ, the table representing $p(\mathbf{d})$ and the parameter, $\mathbf{M}$, governing the background densities. It will be helpful to decompose Q into four parts, following the factorization in Equation 3.7.

$$\begin{aligned} Q(\tilde{\theta}|\theta) &= Q_1(\tilde{\theta}|\theta) + Q_2(\tilde{\theta}|\theta) + Q_3(\tilde{\theta}|\theta) + Q_4 \\ &= \sum_{i=1}^{I} E[\log p(\mathbf{n}_i|\theta)] + \sum_{i=1}^{I} E[\log p(\mathbf{d}_i|\theta)] + \sum_{i=1}^{I} E[\log p(\mathcal{X}_i, \mathbf{x}_i^{\mathrm{m}}|\mathbf{h}_i, \mathbf{n}_i, \theta)] \\ &+ \sum_{i=1}^{I} E[\log p(\mathbf{h}_i|\mathbf{n}_i, \mathbf{d}_i)] \end{aligned} \tag{7.2}$$

Only the first three terms depend on the model parameters.

Update Rule for μ

Since only Q_3 depends on $\tilde{\mu}$, taking the derivative of the expected likelihood yields

$$\frac{\partial}{\partial \tilde{\mu}} Q_3(\tilde{\theta}|\theta) = \sum_{i=1}^{I} E\left[\tilde{\Sigma}^{-1}(\mathbf{x}_i - \tilde{\mu})\right],$$

where $\mathbf{x}^T = (\mathbf{x}^{oT}\, \mathbf{x}^{mT})$ according to our definition above. Setting the derivative to zero yields the following update rule

$$\tilde{\mu} = \frac{1}{I} \sum_{i=1}^{I} E[\mathbf{x}_i]. \tag{7.3}$$

Update Rule for Σ

Similarly, we obtain for the derivative with respect to the inverse covariance matrix

$$\frac{\partial}{\partial \tilde{\Sigma}^{-1}} Q_3(\tilde{\theta}|\theta) = \sum_{i=1}^{I} E\left[\frac{1}{2}\tilde{\Sigma} - \frac{1}{2}(\mathbf{x}_i - \tilde{\mu})(\mathbf{x}_i - \tilde{\mu})^T\right].$$

Equating with zero leads to

$$\tilde{\Sigma} = \frac{1}{I} \sum_{i=1}^{I} E[(\mathbf{x}_i - \tilde{\mu})(\mathbf{x}_i - \tilde{\mu})^T] = \frac{1}{I} \sum_{i=1}^{I} E[\mathbf{x}_i \mathbf{x}_i{}^T] - \tilde{\mu}\tilde{\mu}^T. \tag{7.4}$$

Update Rule for $p(\mathbf{d})$

To find the update rule for the 2^F probability masses of $p(\mathbf{d})$, we need to consider $Q_2(\tilde{\theta}|\theta)$, the only term depending on these parameters. Taking the derivative with respect to $\tilde{p}(\bar{\mathbf{d}})$, the probability of observing one *specific* vector, $\bar{\mathbf{d}}$, we obtain

$$\frac{\partial}{\partial \tilde{p}(\bar{\mathbf{d}})} Q_2(\tilde{\theta}|\theta) = \sum_{i=1}^{I} \frac{E[\delta_{\mathbf{d},\bar{\mathbf{d}}}]}{\tilde{p}(\bar{\mathbf{d}})},$$

where δ shall denote the Kronecker delta. Imposing the constraint $\sum_{\bar{\mathbf{d}}} \tilde{p}(\bar{\mathbf{d}}) = 1$, for instance by adding a Lagrange multiplier term, we find the following update rule for $\tilde{p}(\bar{\mathbf{d}})$,

$$\tilde{p}(\bar{\mathbf{d}}) = \frac{1}{I} \sum_{i=1}^{I} E[\delta_{\mathbf{d},\bar{\mathbf{d}}}].$$

Update Rule for M

Finally, we notice that $Q_1(\tilde{\theta}|\theta)$ is the only term containing information about the mean number, M_f, of background points per part type. Differentiating $Q_1(\tilde{\theta}|\theta)$ with respect to $\tilde{\mathbf{M}}$ we find,

$$\frac{\partial}{\partial \tilde{\mathbf{M}}} Q_1(\tilde{\theta}|\theta) = \sum_{i=1}^{I} \frac{E[\mathbf{n}_i]}{\tilde{\mathbf{M}}} - I.$$

Equating with zero leads to the intuitively appealing result

$$\tilde{\mathbf{M}} = \frac{1}{I} \sum_{i=1}^{I} E[\mathbf{n}_i].$$

Computing the Sufficient Statistics

All update rules derived above are expressed in terms of *sufficient statistics*, $E[\mathbf{x}]$, $E[\mathbf{x}\mathbf{x}^T]$, $E[\delta_{\mathbf{d},\bar{\mathbf{d}}}]$ and $E[\mathbf{n}]$. These statistics represent expectations with respect to the posterior density and are calculated during the E-step of the EM algorithm. The posterior density can be written as

$$p(\mathbf{h}_i, \mathbf{x}_i^{\mathrm{m}}|\mathcal{X}_i, \theta) = \frac{p(\mathbf{h}_i, \mathbf{x}_i^{\mathrm{m}}, \mathcal{X}_i|\theta)}{\sum_{\mathbf{h}_i \in \mathcal{H}_\mathbf{d}} \int p(\mathbf{h}_i, \mathbf{x}_i^{\mathrm{m}}, \mathcal{X}_i|\theta)\, d\mathbf{x}_i^{\mathrm{m}}}.$$

The denominator in the expression above, which is equal to $p(\mathcal{X}_i)$, is calculated by explicitly generating and summing over all hypotheses, while integrating out[1] the missing data of each hypothesis. The expectations are calculated in a similar fashion: $E[\delta_{\mathbf{d},\bar{\mathbf{d}}}]$ is calculated by summing only over those hypotheses consistent with $\bar{\mathbf{d}}$ and dividing by $p(\mathcal{X}_i)$. Similarly, $E[\mathbf{n}_i]$ is calculated by averaging $\mathbf{n}(\mathbf{h})$ over all hypotheses. The case of $E[\mathbf{x}]$ is slightly more complicated. For every hypothesis we regroup $\mathbf{x}^T = (\mathbf{x}^{\mathrm{o}T}\ \mathbf{x}^{\mathrm{m}T})$ and note that $E[\mathbf{x}^{\mathrm{o}}] = \mathbf{x}^{\mathrm{o}}$. For $E[\mathbf{x}^{\mathrm{m}}]$ one needs to calculate

$$E[\mathbf{x}^{\mathrm{m}}] = \int \mathbf{x}^{\mathrm{m}}\, G(\mathbf{x}|\mu, \Sigma)\, d\mathbf{x}^{\mathrm{m}} = \mu^m + \Sigma^{mo}\Sigma^{oo\,-1}(\mathbf{x}^{\mathrm{o}} - \mu^o),$$

where we defined[2]

$$\mu = \begin{pmatrix} \mu^o \\ \mu^m \end{pmatrix} \qquad \Sigma = \begin{pmatrix} \Sigma^{oo} & \Sigma^{om} \\ \Sigma^{mo} & \Sigma^{mm} \end{pmatrix}.$$

[1] Integrating out dimensions of a Gaussian is simply done by deleting the means and covariances of those dimensions and multiplying by the suitable normalization constant.

[2] Splitting $\mathbf{x}$, μ, and Σ in this way requires sorting the dimensions such that the observed data come first. We do not deal with this problem here explicitly, in order to avoid further clutter in the notation.

Summing over all hypothesis and dividing by $p(\mathcal{X}_i)$ establishes the result. Finally we need to calculate

$$E[\mathbf{x}\mathbf{x}^T] = \begin{pmatrix} \mathbf{x}^{\mathrm{o}}\mathbf{x}^{\mathrm{o}T} & \mathbf{x}^{\mathrm{o}}E[\mathbf{x}^{\mathrm{m}}]^T \\ E[\mathbf{x}^{\mathrm{m}}]\mathbf{x}^{\mathrm{o}T} & E[\mathbf{x}^{\mathrm{m}}\mathbf{x}^{\mathrm{m}T}] \end{pmatrix}.$$

Here, only the part $E[\mathbf{x}^{\mathrm{m}}\mathbf{x}^{\mathrm{m}T}]$ has not yet been considered. Integrating out the missing dimensions, $\mathbf{x}^{\mathrm{m}}$, now involves

$$E[\mathbf{x}^{\mathrm{m}}\mathbf{x}^{\mathrm{m}T}] = \int \mathbf{x}^{\mathrm{m}}\mathbf{x}^{\mathrm{m}T}\, G(\mathbf{x}|\mu,\Sigma)\, d\mathbf{x}^{\mathrm{m}} = \Sigma^{mm} - \Sigma^{mo}\Sigma^{oo-1}\Sigma^{moT} + E[\mathbf{x}^{\mathrm{m}}]E[\mathbf{x}^{\mathrm{m}}]^T.$$

Looping through all possible hypotheses and dividing by $p(\mathcal{X}_i)$ again provides the desired result. This concludes the E-step of the EM algorithm.

7.5 Invariant Learning

We have already discussed translation-invariant constellation models in Chapter 3, where we introduced the notions of figure space and figure space*. We will show in the following how translation invariant models can be learned from training examples "contaminated" by arbitrary translations.

7.5.1 Translation-Invariant Learning

We have seen, in Section 3.3.1, how a translation invariant constellation model can be obtained starting from a model with a Gaussian foreground density in figure space. It was shown how the corresponding figure space* density is computed by applying a linear transform, W, to the figure space density (Equation 3.13).

Even if a Gaussian is an appropriate model for the figure space* density, the true figure space density—or at least the one underlying our training examples—might not be Gaussian at all. An important question then is how we can learn the parameters of the Gaussian figure space* density from the non-Gaussian figure space data, in order to obtain a translation invariant model.

If we assume that the figure space* density is Gaussian, but the figure space is not, then the reason for this discrepancy can only lie in the *positions* of the objects in figure space, or better: in the pdf that governs these positions. Hence, we can think of our problem as being caused by the following process. Signals are generated using a Gaussian figure space density, $p_{\mathrm{fg}} = \mathcal{G}(\mathbf{x}|\mu,\Sigma)$, translations from an arbitrary density $p_\tau(\tau)$ are then added to the objects' positions, resulting in

some non-Gaussian figure space density, $\hat{p}_{\text{fg}}$. The latter might even be highly non-Gaussian, e.g., if a multi-modal p_τ is used. It is now interesting to see that there is no need to actually recover the original parameters μ and Σ. We can fit a Gaussian, which we shall parametrize by μ' and Σ', to the "contaminated" data. Although this might result in a very bad fit, all effects of p_τ are nullified when we now transform μ' and Σ' to figure space*.

To show this formally, we can compute explicitly the mean and covariance matrix of the translated data,

$$\mu' = E\left[\mathbf{x} + B\tau\right] \quad \text{and} \quad \Sigma' = E\left[(\mathbf{x} + B\tau - \mu')(\mathbf{x} + B\tau - \mu')^T\right], \tag{7.5}$$

with

$$\mathbf{x} \sim p_{\text{fg}}(\mathbf{x}) = \mathcal{G}(\mathbf{x}|\mu, \Sigma), \quad \tau \sim p_\tau(\tau) \quad \text{and} \quad B = \begin{pmatrix} 1 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1 & \cdots & 1 \end{pmatrix}^T,$$

a $2F \times 2$ matrix, which expands the vector $\tau = (\tau_x \tau_y)^T$ such that τ_x is added to the x-components of all part positions and τ_y to the y-components. We can elect an arbitrary part as reference and compute the parameters of the figure space* density corresponding to $\mathcal{G}(\mathbf{x}|\mu', \Sigma')$, obtaining

$$W\mu' = E\left[W\mathbf{x} + WB\tau\right] \tag{7.6}$$

and

$$W\Sigma'W^T = E\left[\left(W\mathbf{x} + WB\tau - W\mu'\right)\left(W\mathbf{x} + WB\tau - W\mu'\right)^T\right]. \tag{7.7}$$

It is easy to see that WB is a $2F \times 2$ matrix of zeros, independent of the part that has been chosen as reference when constructing W. Therefore, $WB\tau = \mathbf{0}$, and we finally obtain

$$W\mu' = W\,E\left[\mathbf{x}\right] = W\mu \quad \text{and} \quad W\Sigma'W^T = W\,E\left[(\mathbf{x} - \mu)(\mathbf{x} - \mu)^T\right]W^T = W\Sigma W^T. \tag{7.8}$$

This proves that *any* Gaussian, fit to data containing translations from *any* arbitrary density, p_τ, can serve as an equivalent parametrization of the underlying figure space* density.

These findings impact our EM learning rules in two ways. Firstly, the foreground density, p_{fg}, is used in computing the posterior density, $p(\mathbf{h}_i, \mathbf{x}_i^{\text{m}}|\mathcal{X}_i, \theta)$. Thus, when learning a translation invariant model, we need to use the translation invariant version of p_{fg} (e.g. as in Equation 3.16)

when computing the posterior. Secondly, since we now know that the computation of the mean and covariance of p_{fg} is not affected by arbitrary translations of the training data, we have the liberty to add or remove any translation to a given constellation, $\mathbf{x}_i$, when applying the update rules (7.3) and (7.4).

We can therefore conclude that only one minor change, namely the replacement of p_{fg} by the translation invariant version in the computation of the posterior, is needed in order to learn a translation invariant constellation model with a Gaussian figure space* density.

7.6 Analysis

In order to assess the practical usefulness of the parameter estimation method derived above, the following questions are of interest:

Computational Complexity What is the computational cost of the algorithm as a function of the number of parts in the model, the number of training images, and the number of part candidates detected in the training images?

Local Maxima How likely is it that the method converges to a local maximum of the objective function and how does this effect the performance of the learned model?

Amount of Training Data How many training data are needed to avoid overfitting? How does the amount depend on the model complexity and the structure of the data ("difficulty" of the problem)?

Performance What is the classification performance of a learned model learned? Which measures can be used to predict it?

We provide a formal analysis of the first question before we discuss a set of experiments conducted on artificial data, which will elucidate the remaining issues.

7.6.1 Computational Complexity

The structure of the implementation of our learning method is similar to the algorithm used for detection. Therefore, most findings from Section 4.3, where we briefly analyzed the computational cost of detection and localization, apply also for learning. Let us first consider parameter estimation without part selection. From the EM update rules we see that the computational cost is linear in the

number of training samples. Furthermore, to compute the sufficient statistics, we need to consider each hypothesis exactly once, as was the case for detection. The cost for one hypothesis is at most exponential for the Gaussian densities considered here. The cost for one iteration of EM is therefore in $O(a^F)$. It is difficult to gauge the number of EM iterations needed, as a function of I, F, or the number of candidates per signal. However, this is not even so important, since it is almost certainly not overwhelming the exponential complexity in F, as confirmed in our experiments.

Regarding the iterative part selection procedure, we have not done a detailed analysis. The conclusions from our experiments are that the number of necessary iterations is about linear in the number of pre-selected candidate detectors and sub-linear in the number of model parts.

The limiting factor is thus, as for detection, the exponential complexity in F.

7.6.2 Experiments with Synthetic Data

The following experiments serve only to illustrate the behavior of the EM algorithm used to estimate the model parameters. Since synthetic data is used, the question of optimal part selection does not arise. All experiments were conducted in the translation invariant setting.

Data Generation

In all experiments, we generated data artificially from a known model. We provided explicit values for the number of model part types and the number of parts per type, the detection probabilities, $p(\mathbf{d})$, the average number of background detections, $\mathbf{M}$, and the range of the eigenvalues of the covariance matrix underlying the Gaussian foreground density. The actual values for mean vector and (full) covariance matrix were then chosen randomly. Figure 7.1 depicts a foreground density obtained in this fashion and a set of training signals generated from the resulting model.

Measuring Goodness of Representation

One reason why ML is used as a criterion for density estimation, is that it is equivalent to minimizing the Kullback-Leibler divergence between the model density, $\hat{p}(x|\theta)$, and the true data density, $p(x)$. We will briefly illustrate this. The Kullback-Leibler divergence is defined as

$$D(p(x), \hat{p}(x|\theta)) = \int p(x) \log \frac{p(x)}{\hat{p}(x|\theta)} \, dx = H(x) - \int p(x) \log \hat{p}(x|\theta) \, dx, \tag{7.9}$$

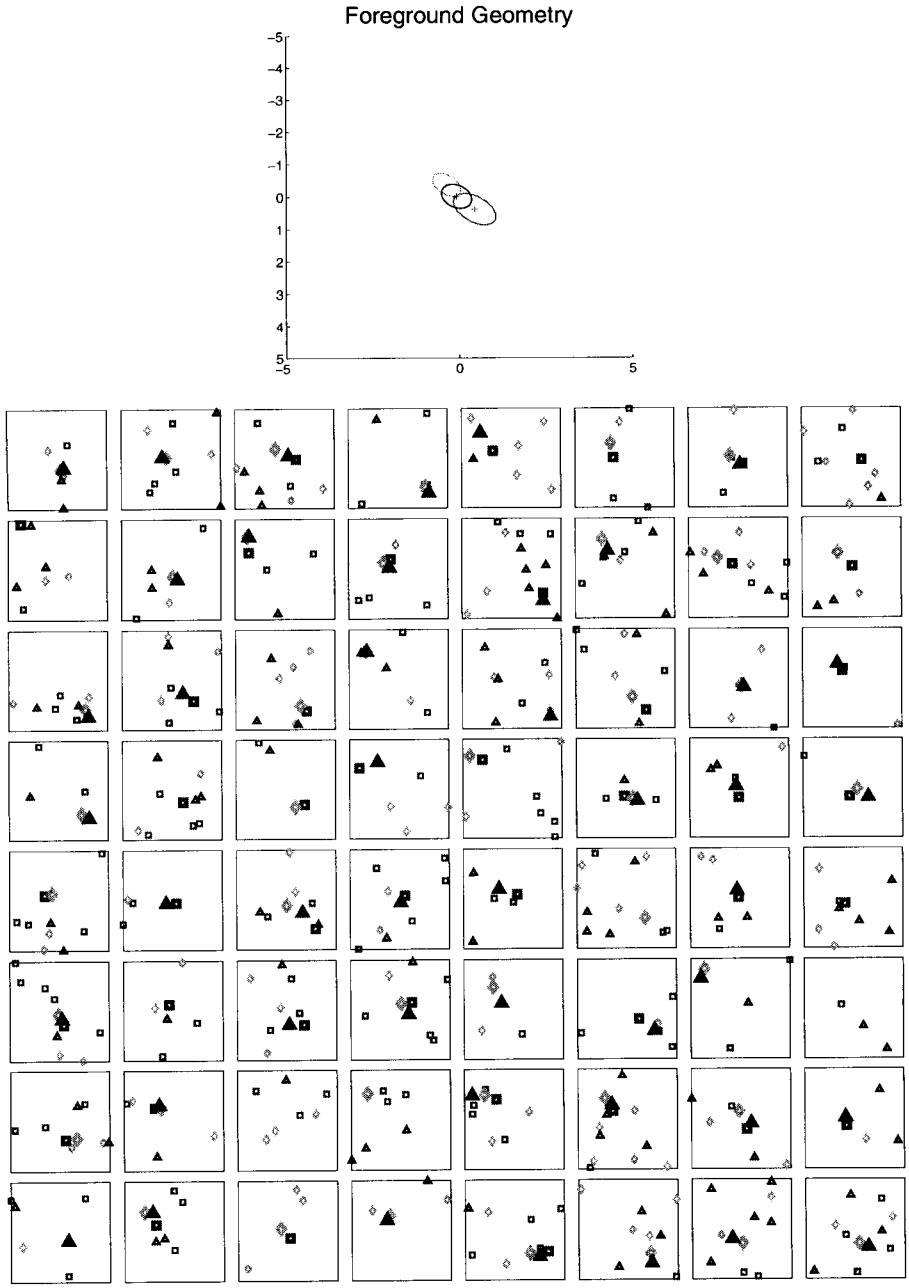


Figure 7.1: A randomly generated foreground density is depicted (top) for a model with three features of different types. Crosses indicate mean part positions. The absolute positions are not meaningful, since the model is translation invariant. Hence, the object could appear with equal probability at any given location in the image. The ellipses indicate a one standard deviation distance from the mean. Below we show 64 images generated with the corresponding model. Parameters were: $p(d_f) = 0.7$ indepently for all features, $M_f = 2$. Symbols are (from top left to bottom right according to top figure): $\diamond$, $\triangle$, $\square$. Foreground features are shown with bold symbols.

where $H(x)$ is the entropy of $p(x)$ and does not depend on θ. Since we do not have access to $p(x)$, and since solving the integral might be difficult, we assume that we have obtained an i.i.d. set of N samples, $\{x_i\}$, and approximate

$$\int p(x) \log \hat{p}(x|\theta)\, dx \approx \frac{1}{I} \sum_{i=1}^{I} \log \hat{p}(x_i|\theta).$$

The term on the right is simply the log-likelihood normalized by the number of samples. It is now important to see that the normalization allows us to compare the likelihood across data sets of different sizes. A slight complication in our case is the fact that each training signal might contribute a different number of degrees of freedom to the total set of training data, since the number of candidate parts per signal varies. We therefore divide by the total number of *candidate detections* in the training set, instead of the number of training signals, I. We define

$$N_{DOF} \stackrel{\text{def}}{=} \sum_{i=1}^{I} \sum_{\tau=1}^{T} N_{\tau}^{(i)}$$

and our normalized likelihood as

$$\hat{L} \stackrel{\text{def}}{=} \frac{1}{N_{DOF}} \sum_{i=1}^{I} \log \hat{p}(x_i|\theta).$$

Unless otherwise noted, we will assume this measure in the following when we refer to the likelihood.

Local Maxima and Initialization of EM

As we have mentioned in the derivation of our parameter estimation method, the EM algorithm is only guaranteed to converge to a *local* maximum of the likelihood function. This is a notorious problem, encountered even in simple incarnations of the algorithm, such as the estimation of Gaussian mixture densities. Which maximum the EM algorithm converges to depends on the initial parameters. We initialize all entries of the mean vector of the foreground density with random numbers drawn from a uniform distribution over half the size of the signal support. We discovered that it is important not to choose an initial variance that is too small. We therefore initialize the covariance matrix with $\sigma_0^2\mathbf{I}$, where $\mathbf{I}$ is the identity matrix and σ_0 is chosen to be about half the size of the signal support as well.

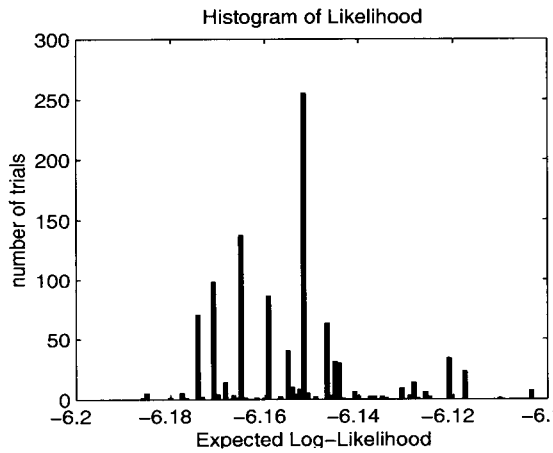


Figure 7.2: A histogram showing the final log-likelihood to which the EM algorithm converged after 10^3 random restarts. Results were obtained with a three-feature model where all three features were of the same type. Parameters were: $p(d_f) = 0.6$ (independent), $M_f = 2$, $\sigma_0 = 1.5$.

No principled method exists to address the problem of local maxima. The only solution is to restart the EM algorithm multiple times with different initial parameter values. After extensive experimentation, we concluded that local maxima are a rare phenomenon in our problem setting. We also confirm this later for real data. Local maxima only arise when the input data is very ambiguous. By this we mean that at least several of the following conditions hold: (1) The detection probability, $p(\mathbf{d})$, is low (e.g., around 0.6 for most parts); (2) only few (less than 50) training signals are available; (3) a significant number (2 or more) of background parts is present on average per image and model part; (4) the geometrical variability is high; and (5) many model parts are of the same type. As can be seen in Figure 7.2, local maxima are clearly found when all of these conditions hold. We also provide, in Figure 7.1, examples of models for which local maxima were observed. In the case of three part models, they only arise when at least two parts are of the same type, or when the detection probabilities are very low and many background detections are present.

Overfitting

In order to investigate the amount of overfitting for a given number of training examples we conducted a series of experiments for four different scenarios. Each scenario corresponds to a "true" model, the parameters of which we chose. Data were then generated from the true model density and used to learn new models. The parameters used in the four scenarios are provided in Table 7.2. We first generated a synthetic training set composed of I signals of class $\mathcal{O}_1$ (signals containing an

No Local Maxima Observed				
# part types	$p(d_f)$	M_f	$\sigma_{\max}$	N
3	0.95	1	1.5	30
3	0.6	2	1.5	30
3	0.5	3	1.5	30
2	0.9	1	0.4	50
1	0.9	1	0.4	100

Local Maxima Observed				
# part types	$p(d_f)$	M_f	$\sigma_{\max}$	N
3	0.5	5	1.5	30
1	0.9	1	1.5	30
1	0.9	1	1.5	50
1	0.9	1	0.4	50
1	0.6	1	0.4	100

Table 7.1: The two tables give examples of parameters where local maxima were not/were observed. Models with three parts were learned in all cases. The detection probability was set to the same indicated value for all parts and was independent across parts. M_f was the same for all part types. $\sigma_{\max}$ quantifies the geometrical variability as the standard deviation along the principal axis of the foreground covariance matrix. The size of the signal support was 10×10 units.

object). We also generated two test sets of size I, one with signals containing objects, the other with background only (corresponding class $\mathcal{O}_0$). After training a model by running EM on the training set, we computed the normalized likelihood, $\hat{L}$, for the training and test sets of class $\mathcal{O}_1$. We then used the learned model to classify the training and test set against the set of background signals. This provided us with a classification error (test and training). In order to obtain expected values of the likelihood and error measures, we repeated this procedure 500 times for the same model, generating different data sets each time. Results are shown in Figure 7.3, as a function of the data set size. The discrepancy between the two likelihoods is an indicator for the amount of overfitting, as is the difference between test and training error.

One important finding is that the average between test and training values predicts the asymptotic (for an infinite number of training data) likelihood and classification error very well. The difference between test and training error can be used to estimate the amount of training data needed for a given model complexity. For example, in order to confine the test error to within 1% of the training error, about 100 training signals are needed for three-part models, while at least 200 signals are needed for five parts. It is furthermore surprising that the standard deviation of the likelihood

Model	Parts	M_f	$p(d_f)$	σ_0
1	A B C	0.2	1	0.2
2	A A B	1	0.6	1.5
3	A A B C	1	0.6	1.5
4	A A B C D	1.5	0.8	1.5

Table 7.2: This table shows the parameters for the four models used in the overfitting experiments. The result are reported in Figure 7.3. 'A' to 'D' symbolize different part types; for example, "A A B" refers to a model with three parts, two of which are of the same type.

measure is smaller or equal to the difference between test and training value, while the standard deviation of the classification errors is often significantly larger than the difference between test and training. Overall, the variability in the classification error is quite significant. This should be investigated further.

The above findings provide a useful guidance for experiments on real data.

Predicting Classification Performance

In order to do part selection, we need to evaluate the performance of a constellation model as a classifier, once the parameters have been estimated. In the introduction, we proposed to do this using a validation data set. Such a validation set would consist of signals from both classes, $\mathcal{O}_1$ and $\mathcal{O}_0$. The classification error on this set would serve as the objective for part selection. Does the use of a validation set render our method more supervised? We already know that whatever objects are supposed to be learned are present in the training data. We could therefore set aside a portion of this data for validation. All we need in addition is a set of images containing only background clutter. We could endow our learning method with a large set of arbitrary images once and forever. If we randomly select of small subset for validation, the probability that the object to be learned is included in this set is negligible. We therefore argue that no additional supervision is introduced through the validation step.

An important problem regarding validation is the confidence associated with the classification error. Consider the following example, taken from [DH73]. Suppose that the validation error is zero on 50 validation samples. We could then conclude with 95% confidence, that the true error lies below 8%. In order to conclude that it lies below 2%, we would need 250 samples. Or, if we measure a validation error of 10% on 250 samples, we could conclude that the true error lies between 7% and 15%. This shows that a large validation set is needed in order to arrive at reasonably reliable

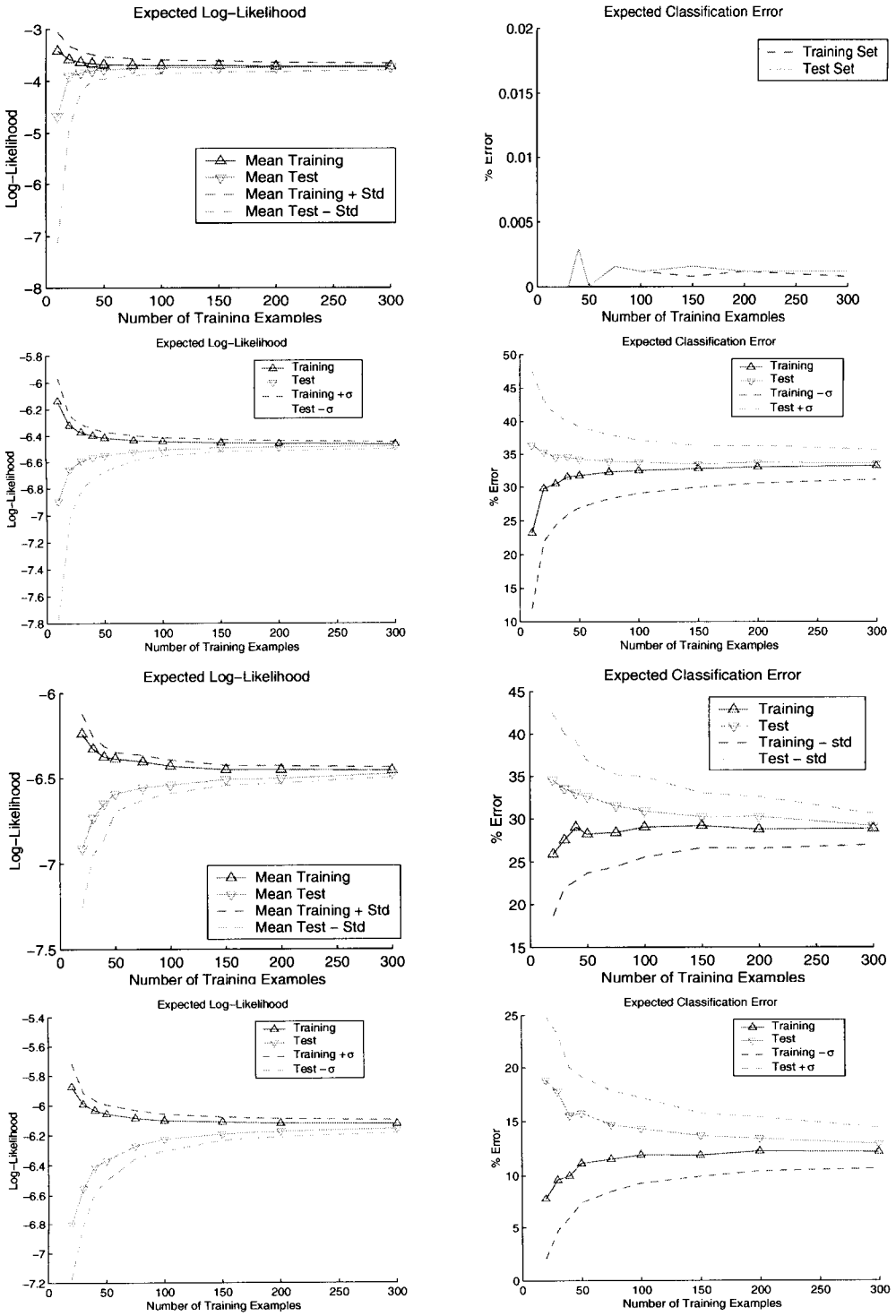


Figure 7.3: Results of the overfitting experiments. Each row corresponds to one of four scenarios (see Table 7.2). In the left column, we plot the normalized likelihood, $\hat{L}$, computed for training and test set, as a function of the number of training examples, I. Note that the abscissae indicate the size of the set of class $\mathcal{O}_1$ used for training. The test sets used to compute classification errors (right column) consisted of I samples of *each class* ($\mathcal{O}_1$ and $\mathcal{O}_0$). All values are averages taken over 500 randomly generated data. Dashed lines are drawn one standard deviation away from the average values.

conclusions.

This motivates a search for other possibilities to predict classification performance. One alternative is to compute the classification error directly from the learned model. This can be done by sampling from a given model by generating random foreground and background signals according to the model parameters. When we classifiy these images using the MAP method, we obtain an estimate of the smallest achievable classifcation error, which we call E_{opt}. The disadvantage of using this measure is that it is overoptimistic, if the model does not fit reality well. The advantage is that no data need to be set aside for validation.[3] The two reasons why a learned model might not fit reality well are that (1) the class of constellation models corresponds to a simplified view of the world, or (2) a particular constellation model learned might not be the best fit, e.g., due to overfitting. It seems then, that we have a trade-off. If we use all our training data for parameter estimation, we might not suffer from (2), but from (1). If we reserve part of the training data and perform validation, we might suffer from (2) but not from (1). In order to investigate this idea further, we performed the following experiment. We learned constellation models from real data (images of faces), recorded the validation error and computed E_{opt}. We also computed the area under the corresponding ROC curve.[4] The results are shown in Figure 7.4. This experiment demonstrates that E_{opt} can be used as a criterion for the part selection process. The best model selected according to E_{opt} is also a very good model regarding the validation error. We also see that the ROC area measure is a slightly better predictor since it is less noisy. We use this measure in the experiments in Chapter 9 as the criterion for part selection. However, the plots also show that E_{opt} underestimates the validation error by almost a factor of two for two-part models and more than a factor of two for three-part models. It can therefore not be trusted to predict the absolute performance.

Finally, one might also wonder if there is any connection between the likelihood (or at least the normalized likelihood, $\hat{L}$, introduced above) and the classification error. We have already argued before, that there is no a priori reason for a dependency between these two quantities. Likelihood measures the fit between the training data and the model. It is therefore concerned with representing the class conditional density well across the entire input space. The classification performance,

[3]Theoretically, we could also perform validation on the training data using a leave-one-out or cross-validation method. However, the computational cost would be extremely large, given that we need to run EM parameter estimation for every split of the training data.

[4]We use this measure, because it averages the detection performance over all possible detection thresholds. It is therefore less prone to noise than the error for any particular threshold, such as the one yielding equal error rate on both classes, which we use to report absolute classification performance. Note, however, that the area measure is *not* equal to the fraction of correctly classified signals.

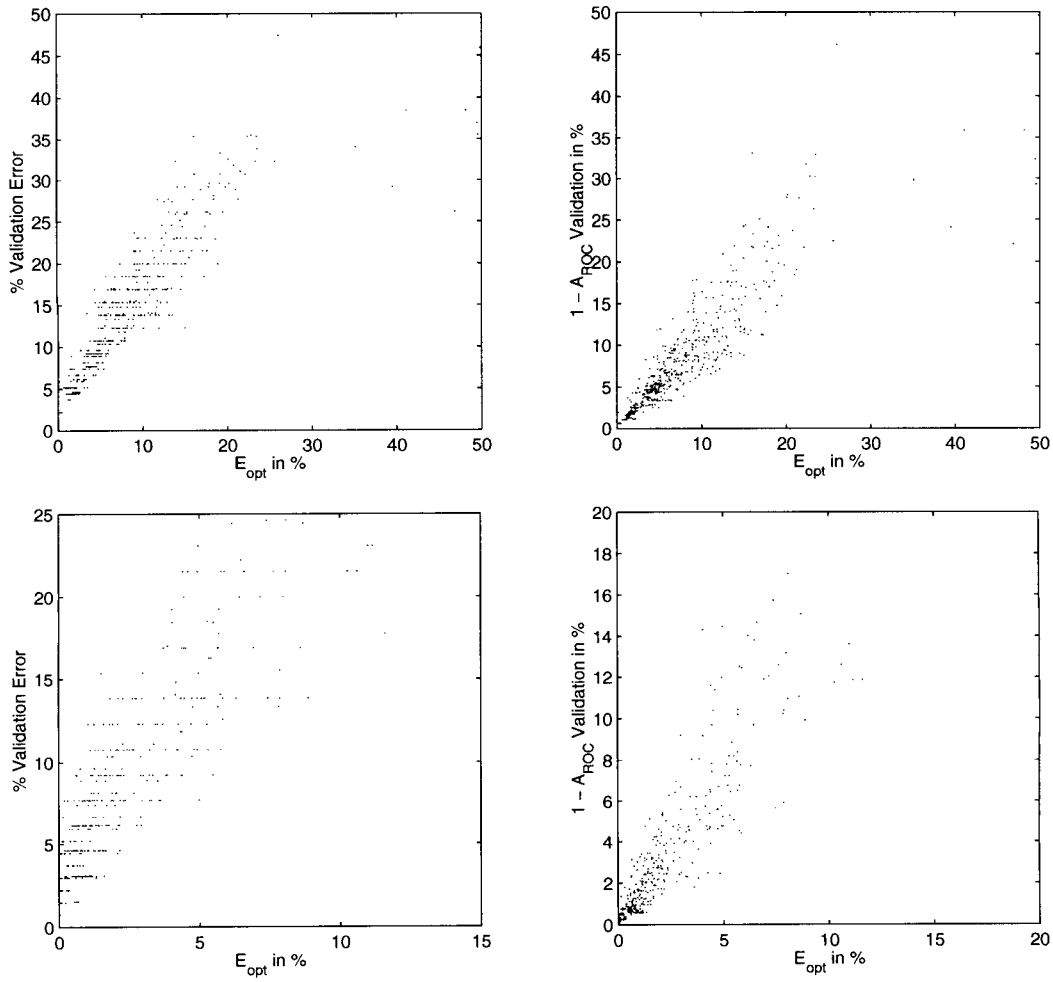


Figure 7.4: The relationship between the classification error, E_{opt}, derived directly from models trained on real data, and the corresponding validation error (left column), as well as the area under the corresponding ROC curve (right column). We learned constellation models of images of human faces. We recorded the validation error and computed E_{opt} (by generating 5000 samples from the model density) throughout the part selection process. Thus, every point in the plot corresponds to a fully trained model and a particular, possibly sub-optimal, choice of parts. The top row is based on two-part models, the bottom one on three-part models.

however, only depends on the final decision boundary derived from the model. In fitting the model, we implicitly model the prior data density $p(\mathcal{X}, \mathbf{h})$. What is relevant for classification is only the posterior, $p(\mathcal{O}|\mathcal{X}, \mathbf{h})$.

Although this is true in general, the situation might be different for our particular model, consisting of a Gaussian foreground density and uniform background density. To understand intuitively why this might be the case, consider first the problem of fitting a single Gaussian class conditional density to some set of samples. If the pdf underlying the samples is really a Gaussian, then we will obtain a perfect fit, as the number of samples grows toward infinity. This means that the KL-divergence will become zero and, therefore, $\hat{L}$ will be equal to the entropy of the density (see Equation 7.9 above). Now consider introducing a second class conditional density that is uniform,[5] as well as equal priors, for simplicity. It is now easy to see that, everything else being equal, a smaller variance of the Gaussian class will reduce the overall classification error, since the overlap between the two densities will decrease as the variance decreases. In this case, then, the classification error turns out to be a monotonically decreasing function of the normalized likelihood.

Although it seems that a higher entropy will always improve classification results when discriminating against a uniform background density, even for non-Gaussian foreground densities, it is not clear if this intuition will carry over to our full model, which contains additional parameters. We have not attempted to prove this link formally, however Figure 7.5 shows results from a simulation that does confirm the dependence, in the case of complete constellation models. Varying all model parameters, we could determine the best model in terms of classification error based purely on $\hat{L}$. The plots also show that there is not one-to-one mapping between the two quantities. In particular, the dependence is rather loose for models with different average numbers of background candidates. This issue needs further investigation.

7.7 Learning the Exact Outline of an Object

So far, we have only learned about the portions of objects that are covered by the model parts. This will not allow us to identify the exact outline of an object in the signal, i.e. to fully segment the entire signal into foreground and background. Such a segmentation could be learned using the following idea.

[5] We can limit ourselves to a finite region in data space, such that the uniform density is non-zero, and choose the variance of the Gaussian density such that most of the probability mass is confined within the finite region.

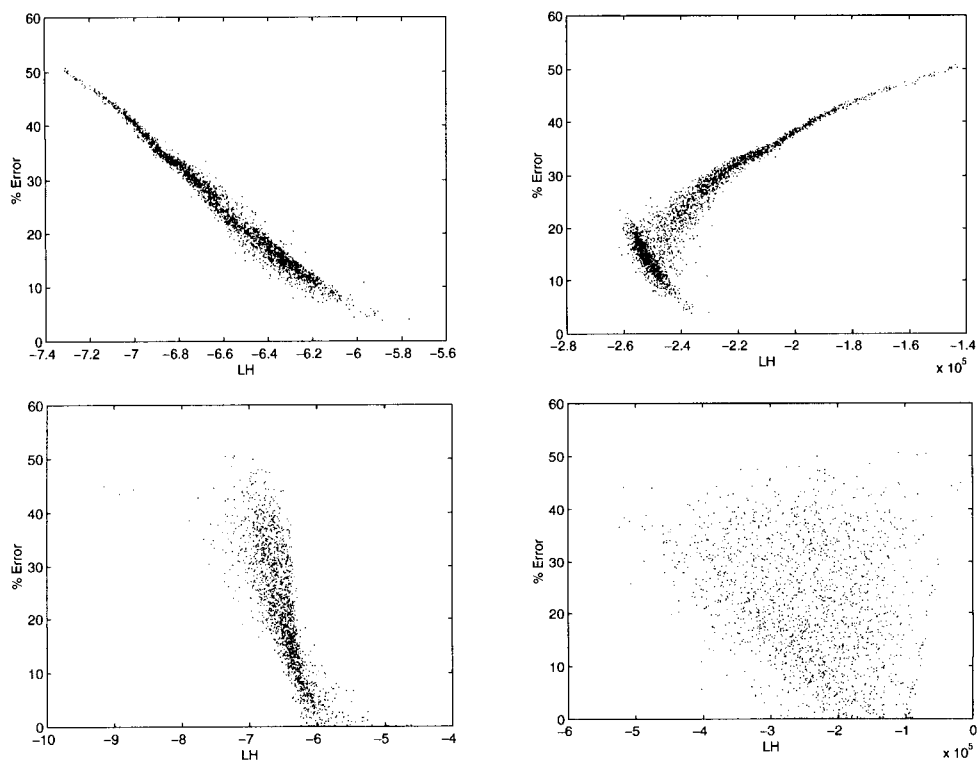


Figure 7.5: Analyzing the relationship between likelihood and classification error with synthetic data. The plots in the top row were produced by generating 2500 models with random μ, Σ, and $p(\mathbf{d})$, and a constant average number of background detections, $\mathbf{M}$. For the plots on the bottom, we held μ, Σ, and $p(\mathbf{d})$ fixed, and generated models with different random values for $\mathbf{M}$. The normalized likelihood, $\hat{L}$, and MAP classification error, E_{opt}, of each model were computed (left column) based on a set of 10^4 foreground signals and the same number of background signals randomly generated from the density corresponding to each model. To illustrate that normalization is essential, the right column shows the likelihood without normalization.

Starting from a learned constellation model, we identify in each training signal the most likely positions of the model parts. Using these positions, we define a grid of dense correspondence across all training images. By looking at the similarity across signals at each grid position, we can obtain a *foreground probability map*, which encodes the likelihood that a certain grid position corresponds to the foreground. During detection, this map could be warped and "anchored" to an input image at the most likely foreground positions. It would then tell us for each pixel in the signal how likely it is that this pixel belongs to the foreground. We have not yet tested such a procedure.

7.8 Implementational Issues

7.8.1 Efficient Evaluation of Hypotheses

The computations that need to be carried out for parameter estimation are very similar to those for detection or localization. In all cases, we need to generate the entire set of hypotheses and compute probabilities under the model density for each hypothesis. The speed-up method described in Section 4.4 should therefore be useful for learning as well. Although we have indeed used it successfully, there are two caveats. The first is that the number of hypotheses that can be excluded depends strongly on the variance of the foreground density. Since the parameter estimation needs to initialized with a large variance, no hypothesis can be excluded in the beginning. This is also clear intuitively: No hypothesis should be ignored right from the start, because we have no information whatsoever as to which part candidates are important and which are not. Later in the estimation, the variance typically shrinks and a speed-up will manifest itself.

The second caveat is that there is a risk of the learning process becoming unstable if we discarded hypotheses. When candidates far away from the mean are ignored, the variance will shrink. This might lead to more candidates being ignored and a vicious circle might establish itself. We found, however, that this can be avoided by setting the rejection threshold to a small value. A large portion of the hypotheses can still be rejected in this way.

There is a possibility to increase the gain further when models with many parts are to be learned. If we have already trained a model with, say, three parts, we can learn a fourth part without re-initializing the variance for *all* parts to a large value. The variance corresponding to the three previously learned features can be kept at the value to which it has already converged.

7.8.2 Further Issues

One further problem concerning spatial variability should be mentioned. We found out that the parameter estimation procedure works best when the size of the signal support, A, is slightly greater (about 20%) than the correct value. If the correct value is chosen, the EM algorithm sometimes rapidly converges to a solution in which all part candidates are assigned to the background. We believe that the reason for this is that a Gaussian foreground density is not a reasonable model when the variance is large, since a lot of probability mass will fall beyond the limits of the signal. The Gaussian then cannot "compete" against the uniform background density that does not suffer from this problem. Setting the signal area to a larger value will attenuate the background density and encourage assigning candidates to the foreground. One could consider adding a correction term to the foreground density as a function of the variance.

7.9 Summary and Conclusions

We proposed to solve the problem of estimating the parameters of constellation models from unlabeled training signals with the EM algorithm. Under this approach, individual part candidates in the training signals are assigned to either the foreground or the background in a "soft," probabilistic fashion. In computing this assignment, the EM algorithm implicitly solves the problem of identifying the objects to be learned in the training signals, while the model paramters are estimated simultaneously.

The update rules for EM involve sums over the set of all possible hypotheses, similar to the computations needed for detection (see 4.2.2). The complexity of the learning algorithm is therefore also exponential in the number of model parts. Hence, we are currently not able to learn models with more than a few (about six) parts.

We have shown that only a minor modifications to the learning method are required in order to learn models in a translation invariant fashion.

Two problems should be investigated further. It is not clear what the most efficient and reliable method is to predict the classification error of a learned model. We have proposed to use either validation data set or the optimal Bayes error derived directly from the model. Both measures perform comparably when used as criterion for part selection. The relationship between the classification error and some form of normalized likelihood should be analyzed. Secondly, in synthetic experiments, we observed a relatively large variance of the test and training errors, when models were

learned from different training sets generated from a unique "true" model. This cannot be explained simply by the uncertainty due to sampling from the underlying binomial distribution around the true, unknown classification error. Neither should it be due to local maxima encountered by the EM algorithm, since those have been shown not to occur in most of the scenarios tested.

The parameter estimation method proposed in this chapter was designed for the "weakly supervised" setting, in which it is assumed that all training images contain an instance of the class to be learned. However, as we discovered during our experiments with real image data (see Section 9.2), the method does still work if an unknown ratio (up to 50%) of the training images do *not* contain an object of interest. We performed this experiment without learning the prior ratio of signals of class $\mathcal{O}_1$ vs. $\mathcal{O}_0$ in the training data. This possibility should be added to the learning algorithm.

Chapter 8 Mixtures of Constellation Models

8.1 Introduction

Although the constellation models we have studied so far can be used to solve difficult visual recognition problems, they are by no means adequate to capture the full complexity of the world surrounding us. So far, we have only represented geometric deformations in the two-dimensional image plane through a single Gaussian density. Although large changes in the three-dimensional pose of objects can, in principle, be handled without an explicit three-dimensional model, the resulting probability density of the projections of object part positions is certainly more complex than what can be captured in two dimensions with a uni-modal model.

Similarly, it might be possible to build part detectors that can detect object parts from any viewing direction in three dimensions, but those would also have to be more sophisticated than the matched filters we have used in our experiments. It is likely that such powerful detectors would have to be more tailored to the specific object parts they detect. Therefore, many detectors might be necessary in order to model different objects. This would add significantly to the complexity of a system designed to recognize a large number of objects.

Another problem for which our simple models might not be adequate is the problem of representing different sub-categories of an object class. Such categories might not share the same parts or they might have a distinct geometry, resulting, again, in a multi-modal probability density.

In order to address the above insufficiencies, we discuss in this chapter the concept of mixtures of constellation models that we have introduced in [WWP00a]. It will allow us to combine sub-models with different parts and distinct geometry in a single model.

8.2 Complex Variability

The concept of mixture models may be illustrated using the analogy of "divide and conquer." If a simple model is not sufficiently flexible to represent a large amount of variability in a set of signals, we may break down the set into smaller subsets. The variability across a single subset might then be manageable for a unimodal model. As a consequence, we will need several simple models to

represent the entire set of signals. Such an approach could be labeled a "competitive" strategy, since only one component model is allowed to take responsibility for any given signal. An example would be a model for images of mammals where one mixture component is used to model dogs, another for cats, and so on.

This way of splitting the representation between different component models is not the only possible approach. One could also imagine a setting where every signal is explained *simultaneously* by a set of component models. Each component model might then "specialize" in a localized portion of the signal. Following our example above, one component could contribute the heads of the animals, another the body. Individual models could also contribute non-local partial signals in an additive fashion. For example, one model could provide the raw visual scene while another adds lighting information, such as shadows. These strategies could be seen as a "collaborative" way of dividing the work between component models.

A similar view on competitive vs. collaborative learning has also been put forward in [HSG98]. A discussion of these concepts in a more general context can be found in [JJNH91].

The approach we have taken here is of the competitive kind. It is therefore also similar to the popular "mixture of experts" methods, first introduced in [HW89], where a separate "gating network" is used to divide the input space into regions, each of which is assigned to a component model. This division is learned simultaneously with the component models. We do not use a separate gating network. As in the case of simple Gaussian mixture models, the input space is divided in a "soft" way, which is represented implicitly through the responsibilities that the component models take on for each input signal. Since it is difficult to determine the optimal subdivision a priori, this information is learned also in our case. This is done by providing a training set comprised of examples across the whole specturm of variability. Using the extension of our unsupervised learning method provided below, we will be able to "discover" categories in the training data automatically. Similarly, it might be possible to learn *view-based* models for visual recognition together with an optimal partitioning of the range of possible viewing directions.

8.3 Theory and Learning of Mixture Models

In this section, we provide the theory extending our basic constellation model to a mixture model, including the necessary extensions of our learning algorithm. Object detection with mixutre models is covered in the following section.

8.3.1 Definition of the Mixture Model

It is possible to construct mixtures of constellation models with different degrees of "overlap." The most straightforward approach might be to keep the models completely independent. Thus, every component model would have its own set of parts, a foreground, and a background model. On the other hand, it might sometimes be desirable to share a common background model across components. This would reduce overall model complexity and the number of training data needed. Finally, it is also conceivable to only replace the foreground density p_{fg} with, say, a Gaussian mixture model. The set of parts and the background model would then be the same across all mixture components. The approach we have taken is to use a common background model, while putting no restrictions on the number of parts shared across the foreground portions of the components. The parts to be used in a particular foreground model can thus be chosen from a common pool, while the shared background model incorporates the entire pool.

We shall denote the number of components by Ω. We can then write the complete mixture model as

$$p(\mathcal{X}, \mathbf{h}|\mathcal{O}) = \sum_{\omega=1}^{\Omega} p(\mathcal{X}, \mathbf{h}|\omega, \mathcal{O}) p(\omega|\mathcal{O}).$$

The conditional density $p(\mathcal{X}, \mathbf{h}|\omega, \mathcal{O})$ represents component model ω. Every component model is based on its own set of foreground parameters, explaining how the object parts in $\mathcal{X}$ can be arranged spatially in an image. However, not every component needs to make use of all part types. As a consequence, a given component model will *always* assign all part candidates of the types not used in that component to the background, even if an object is present. This restriction is expressed through $p(\mathbf{d}|\mathcal{O})$, as explained below.

The probabilities $p(\omega|\mathcal{O}_1)$ express our a priori expectation of explaining signals that contain an object with component model ω. A signal in $\mathcal{O}_0$ is always equally well explained be any component model, since the background model is the same. Therefore, the priors $p(\omega|\mathcal{O}_0)$ have no influence and are simply assumed to be equal.

In analogy with Equation 3.7, the factorized pdf of the mixture model can be written as follows,

$$p(\mathcal{X}, \mathbf{h}, \omega|\mathcal{O}) = p(\mathcal{X}, \mathbf{h}, \mathbf{n}, \mathbf{d}, \omega|\mathcal{O}) = p(\mathcal{X}|\mathbf{h}, \mathbf{n}, \omega)\, p(\mathbf{h}|\mathbf{n}, \mathbf{d})\, p(\mathbf{n})\, p(\mathbf{d}|\omega, \mathcal{O})\, p(\omega|\mathcal{O}). \quad (8.1)$$

Since the background model is common to all components, the probability density over the

number of background detections, $p(\mathbf{n})$, does not depend on ω. Thus, we use

$$p(\mathbf{n}) = \prod_{t=1}^{T} \frac{1}{n_t!}(M_t)^{n_t} e^{-M_t},$$

as in the basic model.

The density $p(\mathbf{h}|\mathbf{n}, \mathbf{d})$ is also the same as before:

$$p(\mathbf{h}|\mathbf{n}, \mathbf{d}) = \begin{cases} \prod_{f=1}^{F} N_{t_f}^{-d_f} & \text{if } \mathbf{h} \text{ is compatible with } \mathbf{n} \text{ and } \mathbf{d} \\ 0 & \text{otherwise.} \end{cases}.$$

The density over $\mathbf{d}$, on the other hand, will differ across component models. This reflects the fact that different components might use different parts out of the common pool. The entries of $\mathbf{d}$ corresponding to parts *not* used in model ω, can thus only take on the value zero. The individual probability tables $p(\mathbf{d}|\omega)$ can still be modeled to represent either joint or independent detection probabilities across part types.

Finally, we use

$$p(\mathcal{X}|\mathbf{h}, \mathbf{n}, \omega) = \int p_{\text{fg}}\left((\mathcal{X}(\mathbf{h})^T \mathbf{x}^{\text{m}T})^T|\omega\right) d\mathbf{x}^{\text{m}}\, p_{\text{bg}}(\mathcal{X}(\overline{\mathbf{h}})),$$

where $\mathcal{X}(\mathbf{h})$ denotes, again, the positions of the foreground parts, extracted from $\mathcal{X}$ according to the indices in $\mathbf{h}$, and $\mathcal{X}(\overline{\mathbf{h}})$ denotes the positions of all remaining parts thus assigned to the background. Note that the foreground model depends on ω while the background does not. As in our experiments with the basic model, we choose $p_{\text{fg}}(\mathbf{x}|\omega)$ to be joint Gaussian with mean μ_ω and covariance Σ_ω.

The positions of the background detections are modeled by a uniform density,

$$p_{\text{bg}}(\mathcal{X}(\overline{\mathbf{h}})) = \prod_{t=1}^{T} \frac{1}{A^{n_t}},$$

where A is the total image area.

8.3.2 EM Learning Rules for Mixture Models

Since the learning algorithm for the mixture model is similar to the one developed in detail in Chapter 7, we provide here only the necessary changes in the derivation of the EM algorithm and the resulting update rules.

For a given set of I training signals, the likelihood of the entire set of observed data, $\{\mathcal{X}_i\}$, can be computed as

$$L(\{\mathcal{X}_i\}|\theta) = \sum_{i=1}^{I} \log \sum_{\mathbf{h}_i,\omega_i} \int p(\mathcal{X}_i, \mathbf{x}_i^{\mathrm{m}}, \mathbf{h}_i, \omega_i|\theta)\, d\mathbf{x}_i^{\mathrm{m}}, \tag{8.2}$$

where we have dropped the conditioning on $\mathcal{O}_1$, which is from now on assumed implicitly. The likelihood is to be maximized with respect to the parameters $\theta = \{\mu_\omega, \Sigma_\omega, p(\mathbf{d}|\omega), \mathbf{M}, p(\omega)\}$, if we assume that the prior probabilities, $p(\omega)$, are estimated from the training data as well. The cost functions optimized by EM are now:

$$Q(\tilde{\theta}|\theta) = \sum_{i=1}^{I} E[\log p(\mathcal{X}_i, \mathbf{x}_i^{\mathrm{m}}, \mathbf{h}_i, \omega_i|\tilde{\theta})]. \tag{8.3}$$

Taking the derivative of $Q(\tilde{\theta}|\theta)$ with respect to the parameters and equating to zero produces the following update rules,

$$\tilde{\mu}_\omega = \frac{\sum_{i=1}^{I} p(\omega|\mathcal{X}_i)\, E_\omega[\mathbf{x}_i]}{\sum_{i=1}^{I} p(\omega|\mathcal{X}_i)}, \tag{8.4}$$

$$\tilde{\Sigma}_\omega = \frac{\sum_{i=1}^{I} p(\omega|\mathcal{X}_i)\, E_\omega[\mathbf{x}_i \mathbf{x}_i{}^T]}{\sum_{i=1}^{I} p(\omega|\mathcal{X}_i)} - \tilde{\mu}_\omega \tilde{\mu}_\omega^T, \tag{8.5}$$

$$\tilde{p}(\bar{\mathbf{d}}|\omega) = \frac{\sum_{i=1}^{I} p(\omega|\mathcal{X}_i)\, E_\omega[\delta_{\mathbf{d}_i,\bar{\mathbf{d}}}]}{\sum_{i=1}^{I} p(\omega|\mathcal{X}_i)}, \tag{8.6}$$

$$\tilde{\mathbf{M}} = \frac{1}{N} \sum_{i=1}^{I} \sum_{\omega} p(\omega|\mathcal{X}_i)\, E_\omega[\mathbf{n}_i], \tag{8.7}$$

$$\tilde{p}(\omega) = \frac{1}{N} \sum_{i=1}^{I} p(\omega|\mathcal{X}_i), \tag{8.8}$$

where $\mathbf{x}^T = (\mathbf{x}^{\mathrm{o}T} \mathbf{x}^{\mathrm{m}T})$ and $E_\omega[.]$ denotes taking the expectation with respect to the posterior density $p(\mathbf{h}_i, \mathbf{x}_i^{\mathrm{m}}|\mathcal{X}_i, \omega, \theta)$. These update rules constitute the M-step.

The E-step amounts to the calulation of the sufficient statistics $E_\omega[\mathbf{x}]$, $E_\omega[\mathbf{x}\mathbf{x}^T]$, $E_\omega[\delta_{\mathbf{d},\bar{\mathbf{d}}}]$, $E_\omega[\mathbf{n}]$ and the posterior density,

$$p(\omega|\mathcal{X}_i) = \frac{p(\mathcal{X}_i|\omega)\, p(\omega)}{\sum_\omega p(\mathcal{X}_i|\omega)\, p(\omega)}, \tag{8.9}$$

with

$$p(\mathcal{X}_i|\omega) = \sum_{\mathbf{h}_i} \int p(\mathbf{h}_i, \mathbf{x}_i^{\mathrm{m}}, \mathcal{X}_i|\omega, \theta)\, d\mathbf{x}_i^{\mathrm{m}}.$$

We compute $p(\mathcal{X}_i|\omega)$ as follows. Summing over all hypotheses consistent with the observed data, we integrate out the missing data by eliminating the appropriate dimensions from the Gaussian foreground pdf. We calculate $\mathbf{d}(\mathbf{h})$ and $\mathbf{n}(\mathbf{h})$ and insert them into the joint density. The expectations of the statistics are calculated in a similar fashion. $E_\omega[\delta_{\mathbf{d},\bar{\mathbf{d}}}]$ is calculated by summing only over those hypotheses consistent with $\bar{\mathbf{d}}$ in the numerator and dividing by $p(\mathcal{X}_i|\omega)$. Similarly, $E_\omega[\mathbf{n}]$ is calculated by averaging $\mathbf{n}(\mathbf{h})$ over all hypotheses. For $E_\omega[\mathbf{x}]^T = (\mathbf{x}^{\mathrm{o}T} E_\omega[\mathbf{x}^{\mathrm{m}}]^T)$ we need

$$E_\omega[\mathbf{x}^{\mathrm{m}}] = \int \mathbf{x}^{\mathrm{m}}\, G(\mathbf{x}|\mu_\omega, \Sigma_\omega)\, d\mathbf{x}^{\mathrm{m}} = \mu_\omega^m + \Sigma_\omega^{mo}\Sigma_\omega^{oo\,-1}(\mathbf{x}^{\mathrm{o}} - \mu_\omega^o),$$

where we defined ${\mu_\omega}^T = ({\mu_\omega^o}^T {\mu_\omega^m}^T)$ and a similar decomposition for Σ_ω, as in Section 7.4. For the calculation of $E_\omega[\mathbf{x}\mathbf{x}^T]$ we need the following result

$$E[\mathbf{x}^{\mathrm{m}}\mathbf{x}^{\mathrm{m}T}] = \Sigma_\omega^{mm} - \Sigma_\omega^{mo}\Sigma_\omega^{oo\,-1}\Sigma_\omega^{moT} + E_\omega[\mathbf{x}^{\mathrm{m}}]E_\omega[\mathbf{x}^{\mathrm{m}}]^T.$$

8.3.3 Selection of Parts for Mixture Models

The methods we use to select parts for mixture models are essentially the same used for the basic model (see Chapter 6). There, we had to try out different subsets $X \subseteq Y$, where Y was the complete set of available parts and X was the set of parts to be used in one model. In the case of mixture models, X is the joint set of parts used across all mixture components (the "pool"). Additionally, we also need to specify which parts should be used by each model component and, therefore, different mixture models can be built from the same choice of parts. As with the basic model, there is no way around learning a complete model from a given selection of parts, X, in order to obtain the goodness function, $J(X)$.

It might seem that there is a possible gain by training the mixture components separately, in order to find the best choice of parts for each component. However, this would require a separate training set for each component model and, thus, the decision as to what each component should "specialize" on would have to be taken beforehand. We would lose one of the big advantages of our learning algorithm, which is to determine automatically how the responsibility of each component model is optimally allocated throughout the set of possible signals. We would also introduce the need for additional supervision.

8.4 Object Detection

The detection paradigm we use for mixture models is also the same as for the single component model, introduced in Section 4.2.2. The MAP approach leads to the likelihood ratio, R, which, for mixture models, is computed as follows,

$$R = \frac{p(\mathcal{O}_1|\mathcal{X})}{p(\mathcal{O}_0|\mathcal{X})} \propto \frac{\sum_{\mathbf{h},\omega} p(\mathcal{X}, \mathbf{h}, \omega|\mathcal{O}_1)}{\sum_{\omega} p(\mathcal{X}, \mathbf{h}_0, \omega|\mathcal{O}_0)} = \frac{\sum_{\mathbf{h},\omega} p(\mathcal{X}, \mathbf{h}, \omega|\mathcal{O}_1)}{p(\mathcal{X}, \mathbf{h}_0|\mathcal{O}_0)}. \tag{8.10}$$

The sum in the numerator includes all hypotheses, also the null hypothesis, since the object could be present but remain undetected by any part detector. In the denominator, the only consistent hypothesis to explain "object absent" is the null hypothesis.

8.5 Automatic Discovery of Categories

Once a mixture model has been learned from a diverse training set, we might be interested in finding out what the individual mixture components have specialized on. Unfortunately, there is no direct way to obtain this information. One possibly solution is to simply inspect the parts and foreground geometries adopted by each component, hoping that it will be easy to deduce from this information what the component is "tuned" to. Another possibility is to assign each of the training images – or some other set of images – to one model component based on the posterior, $p(\omega|\mathcal{X})$ (computed according to Equation 8.9). We then need to inspect all signals assigned to a given component, in order to see what they might have in common.

There is no guarantee that the tuning of the components will be interesting or meaningful. The goal of our method is to represent the joint set of training signals best by building categories or clusters, such that the members are visually as similar as possible. All this, of course, only within the limitations of our constellation model. Hence, it might happen that we feed our learning method with images of cats and dogs, hoping to learn these two categories, while the tuning of the resulting components is according to color, size, length of ears, or some other unexpected criterion. If this is the case, the obvious visual similarities probably do not reflect the categories we expected to evolve. A more sophisticated model and/or learning method is needed, which is able to "look beyond" the obvious similarties, maybe even using additional, non-visual information.

Note that it might still be possible, in the above example, to build constellation models that are able to discriminate well between cats and dogs. The tuning according to the two categories could

be enforced, e.g., by leaning a single constellation model for each category separately. However, this appears to be outside the scope of our unsupervised philosophy.

8.6 Conclusion

We addressed the problem of inhomogeneous data sets and automatic discovery of categories by introducing an extension of our basic constellation model to mixture models. The component of a mixture model "compete" for the explanation of a given signal, since the mixture model is set up such that a signal can only be generated by a single component. Mixture components share a common background model, but each comoponent embodies a different foreground model and can have a different set of parts. We presented a straight-forward extension of our learning method to mixture models. Object detection is performed by "integrating" the probabilities of each mixture component having generated the given signal. In principle, mixture models can be used to automatically discover categories in an inhomogeneous data set. Such categories could be entirely different object classes or components of a complex, multimodal class density.

Chapter 9 Experiments

9.1 Overview

In this chapter, we report the results of several series of learning and classification experiments performed on real images. We used data of human faces, cars, leaves, and cartoon strips. In the first experiment, we learned models on most of the data sets and report the resulting classification performance. This experiment corresponds to the "weakly unsupervised" setting. The second experiment was done to demonstrate that our model can handle partial occlusion of the signals used for learning and testing. The third experiment is dedicated to the problem of learning models with parts of different sizes detected at different scales. In the fourth experiment we learn view-based models of human heads over a 90° range of viewing directions around the vertical axis. Experiment 5 is dedicated to mixture models and learning in the "strictly unsupervised" setting. We learn and test on different views of human heads and cars, and on different species of leaves. The last experiment is not concerned with learning. It addresses the issue of hard vs. soft detection of parts.

9.2 Experiment 1—Weakly Supervised Setting

The experiments discussed in this section are intended to demonstrate the general feasibility of our learning method when applied to unmarked and unsegmented cluttered signals. The task was to select parts and to learn models from a set of training images, each containing one instance of the respective class of objects. This task corresponds to the "weakly supervised" setting, as defined in Chapter 5. A second set, comprised of background images, was provided for the validation step underlying the part selection method. After learning, performance was evaluated on an independent test set, which consisted of an equal number of foreground and background images. Each image had to be assigned to class $\mathcal{O}_1$ or $\mathcal{O}_0$.

Instead of using a fixed classification threshold, we computed *receiver operating characteristics* (ROC). As the criterion for part selection, we used the area under the ROC curve, computed on the validation set.[1] However, since this measure underestimates the true classification error, we also

[1] This measure averages the detection rate across all possible detection thresholds and is thus less sensitive to sampling

report the absolute error rate on the test set, for a decision threshold yielding equal error on both classes.

All experiments were performed with the translation invariant version of our model. A Gaussian foreground density with full covariance matrix was used in all cases.

9.2.1 Data Sets

We used four different sets of images in this experiment. All images were acquired with a digital camera at a resolution of 896×592 pixels; they were converted to a grayscale representation and downsampled to a resolution of 223×147 pixels. No images were discarded by hand prior to the experiments.

Cars (Experiment 1a)

We used 200 images of back views of cars, collected on public streets and parking lots. About 50% of the image areas was covered by background clutter. Examples are shown in Figure 9.1. Since the version of the model learning algorithm used was invariant only to translation, particular care was taken to keep the scale and orientation of the cars constant across images. We also took 200 images of background scenes from the same environment. We photographed vehicles of different absolute sizes, colors and types, such as sedans, sport utility vehicles, and pick-up trucks.

The car images were high-pass filtered with a difference of Gaussian filter (DOG) in order to promote invariance with respect to different colors of cars and different lighting conditions (see Fig. 9.5).

Human Faces (Experiment 1b)

We collected images containing frontal views of human faces of 30 subjects. The majority (about 80%) of the images was taken in an indoors office/laboratory environment, the remaining images were taken outdoors, in front of buildings and natural scenes. Corresponding background images were taken reflecting this indoors/outdoors ratio. We took five images per person and a total of 200 background images. Half of these images were used for training, and the other half for testing. No

noise. The measure is equal to 1 for perfect classification and equal to 0.5 in the worst case (chance level). It grows monotonically with the rate of correct classifications, which is the reason why we use it as a criterion for part selection. However, it cannot generally be interpreted as a classification error. We found that it underestimated the classification error in all experiments.

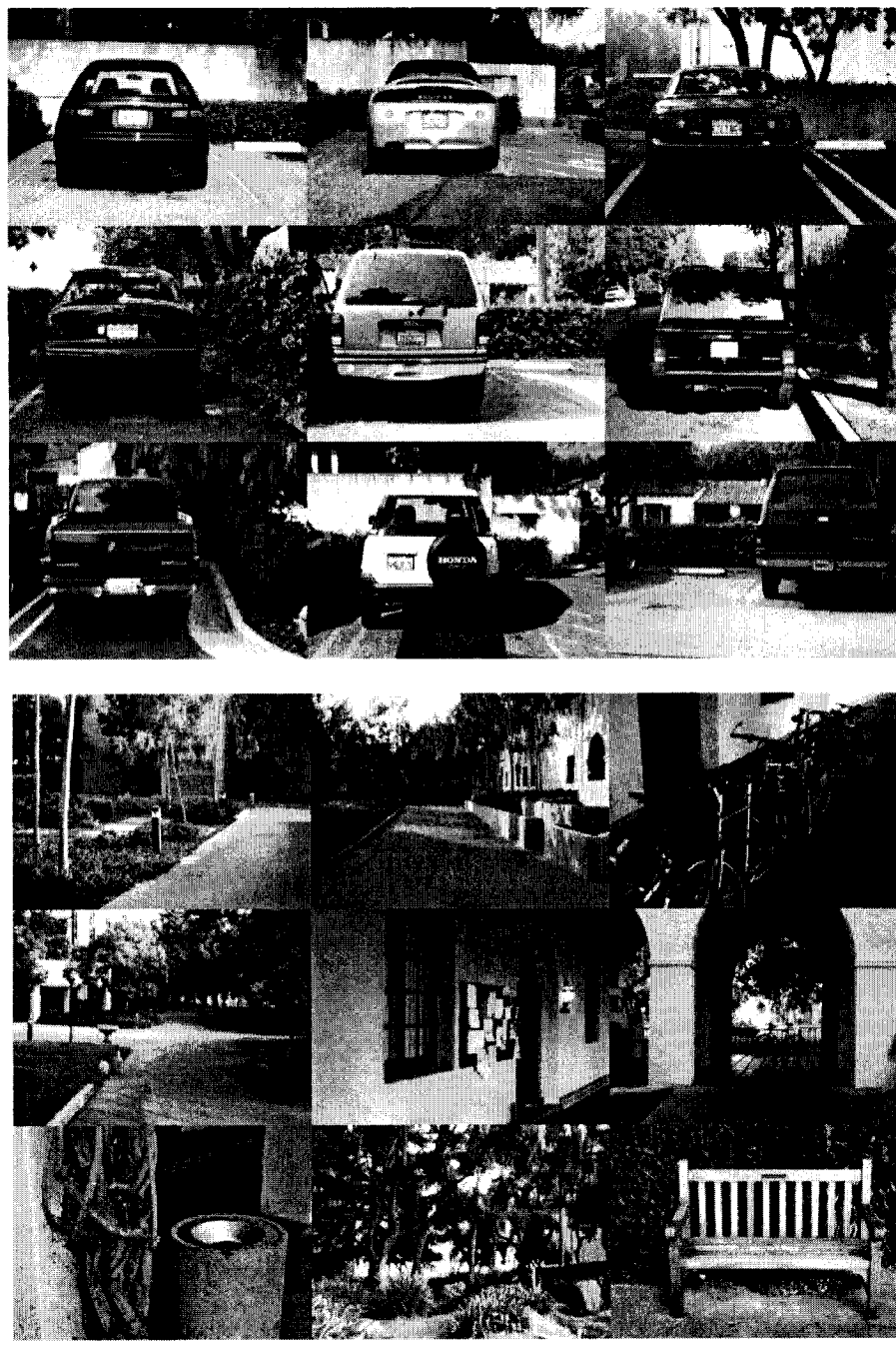


Figure 9.1: Examples of back views of cars (top) and corresponding background images (bottom), used in Experiment 1a.

individual was present in both sets. Orientation and scale of the faces were also kept constant across images. Examples are shown in Figure 9.2.

Letters in Cartoon Strips (Experiment 1c)

We used "Dilbert" cartoon strips of the type shown in Figure 9.3. For this experiment, we concentrated on the text portions of the cartoons, which are composed of hand-written capital letters. The task was to learn a model for a given letter (we report here only the results for the letter 'E'). The data set for class $\mathcal{O}_1$ was created by extracting windows containing approximately 15 letters from the text portion of the cartoons, including the letter to be learned. For $\mathcal{O}_0$ we extracted windows that did not contain the letter. We performed a similar experiment in [WWP99], where we designed part detectors manually.

Cartoon Figures (Experiment 1d)

For this experiment, we extracted image patches containing individual cartoon characters from the same strips used in Experiment 1c. Examples are shown in Figure 9.4. The goal was to recognize the main character, "Dilbert," amongst the other characters. Note that Dilbert is seen from different viewing directions, at different sizes, and with partial occlusion. A total of 77 examples was available for Dilbert, compared to 125 examples for the other characters. The size of the image patches was 101×101 pixels.

Preprocessing

9.2.2 Results

It is important to note that the car and cartoon letter data were used during the development stage of our method. The human faces and Dilbert heads were added only later. We applied our method to these data without changing any parameters.

We learned models with 2, 3, 4, and 5 parts. In order to assess the expected performance in the presence of local maxima, we learned each model 100 times, except for the case of 5 parts, where only 10 models were learned. In Experiment 1d, we used a cross-validation framework to deal with the limited amount of available data. We split the data into five sets of equal size, trained on four sets and computed the test error on the remaining set. This was repeated five times, in order to obtain a test error for each set. The final test error was computed as the average over the five sets.

Figure 9.2: Examples of frontal views of human faces (top) and corresponding background images (bottom), used in Experiment 1b.

Figure 9.3: Example of a cartoon strip used in Experiment 1c.

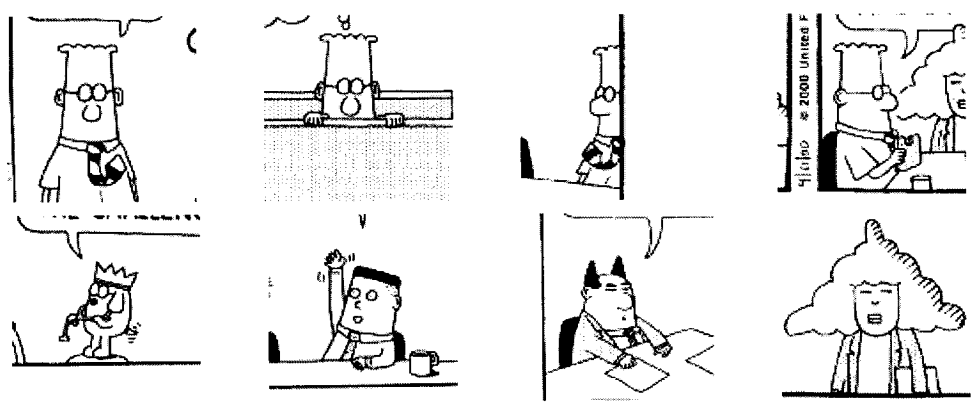


Figure 9.4: Examples of "Dilbert" data set. The top row shows different views of the main character, "Dilbert," extracted from cartoons, such as the one shown in Figure 9.3. In the corresponding experiment the Dilbert character had to be distinguished from other characters, such as the ones in the bottom row.

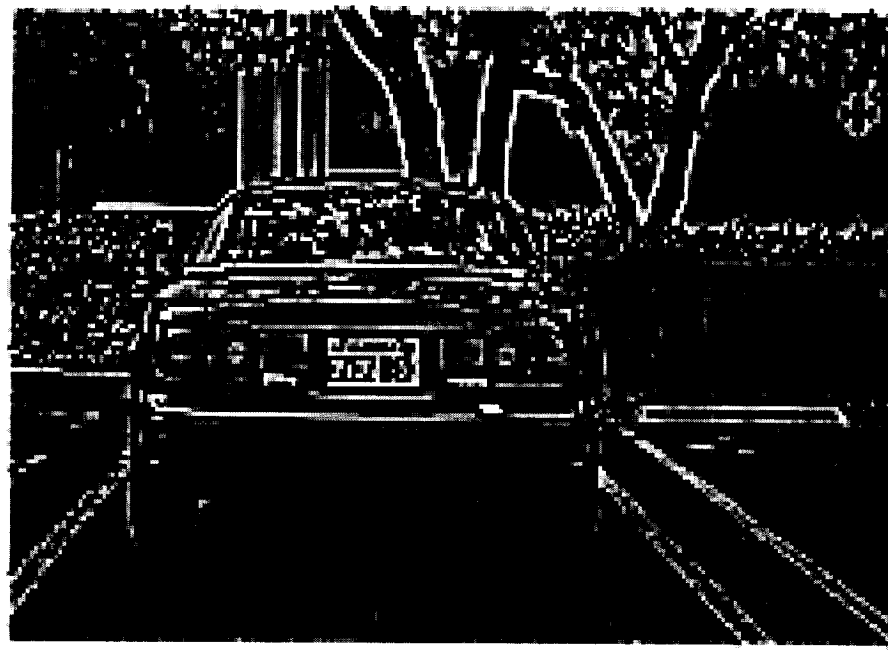


Figure 9.5: Example of a car image after high-pass filtering.

Our algorithms were implemented in Matlab, with subroutines for the EM algorithm implemented in 'C'. The total learning time on a PC with 450MHz Pentium processor was less than one hour for two-part models and about two days for models with five parts. The computation time for the parameter estimation stage was between 5 seconds for two parts and 2 minutes for five parts. The number of different part choices evaluated varied from 200 for two-part models to about 2500 for five-part models.

In order to determine the optimal part size we experimented with sizes from 3×3 up to 13×13 pixels and found 11×11 pixels to yield the best results on faces and cars, while 5×5 was chosen for the cartoon letter experiment.

Figures 9.6, 9.7, 9.8, and 9.9, show examples of models for each data set. Across different numbers of parts, the smallest average test error was 6% in the case of faces (four parts), 13% for cars (five parts), 6% for the letters (four parts), and 15% for Dilbert's head.

In Figure 9.10, we take a closer look at the performance as a function of the number of parts. We use the area under the ROC curve to measure performance. As we increased the number of model parts, the test performance increased in all cases, except for face models with five parts, which showed a significant decrease in performance. The discrepancy between test and training perfor-

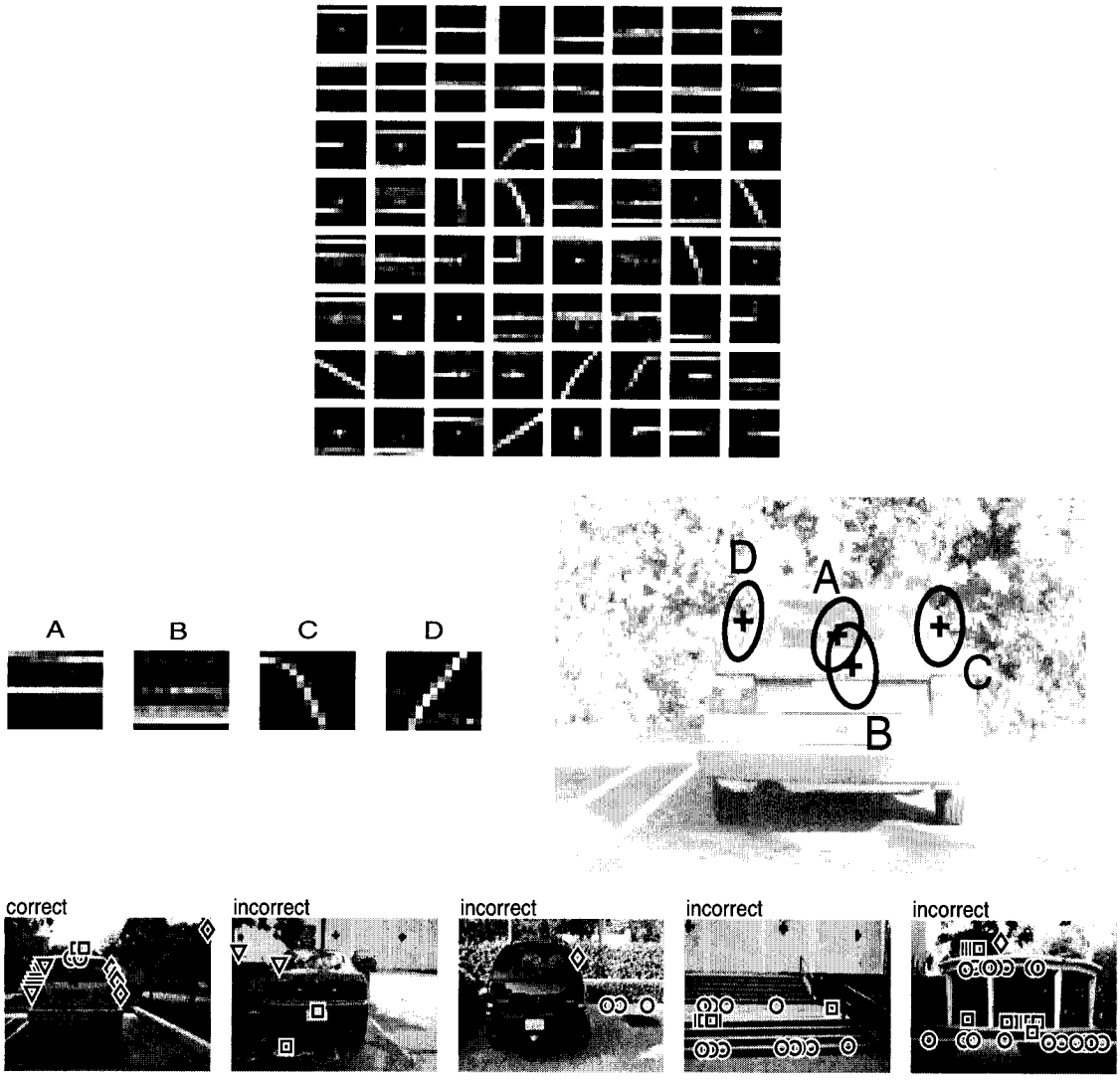


Figure 9.6: Results of Experiment 1a. On the top, a sample of a total of 80 part candidates obtained through k-means clustering is shown. The part size was set to 11×11 pixels. The center row shows the automatically selected parts for a four-part model and the corresponding Gaussian foreground density. Ellipses are drawn at a one-standard deviation distance from mean part positions. The alignment between image and model was done by hand for illustrative purposes, since the models are translation invariant. Two sets of 100 foreground and 100 background images were used for training and testing. The bottom row shows examples of correctly and incorrectly classified images from the test set. Part labels are: $\bigcirc$ = 'A', $\square$ = 'B', $\diamond$ = 'C', $\bigtriangledown$ = 'D'. The smallest average test error was 13%, obtained for models with five parts.

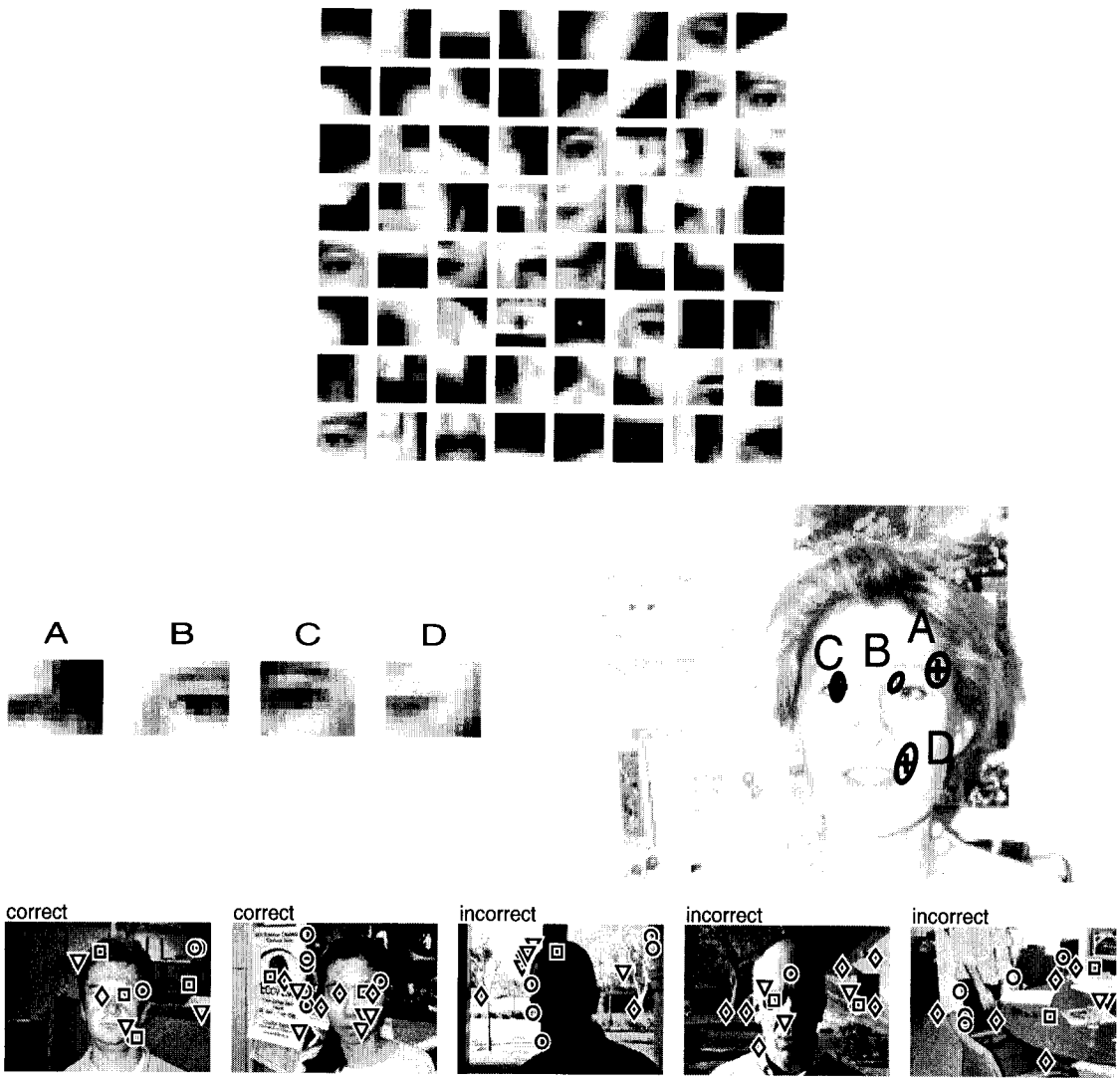


Figure 9.7: Results of Experiment 1b. On the top, a sample of a total of 81 part candidates, obtained through k-means clustering, is shown. The part size was set to 11×11 pixels. The center row shows the parts selected automatically for a four-part model and the corresponding Gaussian foreground density. The bottom row shows examples of correctly and incorrectly classified images from the test set. Part labels are: $\bigcirc$ = 'A', $\square$ = 'B', $\diamond$ = 'C', $\bigtriangledown$ = 'D'. The smallest average classification error was 6%, obtained for four-part models.

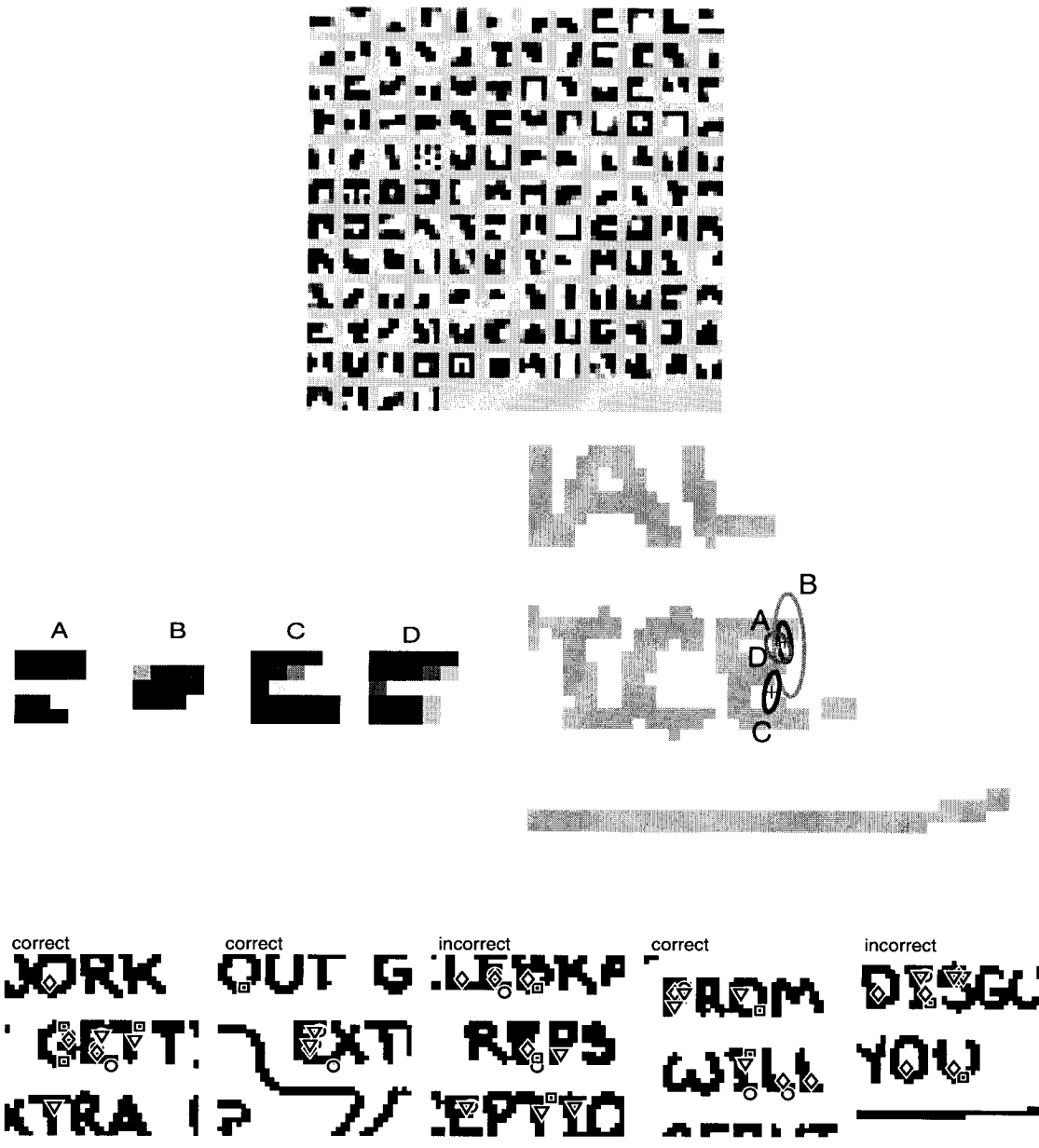


Figure 9.8: Results of Experiment 1c. On the top, 136 part candidates obtained through k-means clustering are shown. The part size was set to 5×5 pixels. The center row shows the parts selected automatically for a four-part model for the letter 'E' and the corresponding Gaussian foreground density. The bottom row shows examples of correctly and incorrectly classified images from the test set. The left three contain the letter 'E' while the two on the right do not. Part labels are: $\bigcirc$ = 'A', $\square$ = 'B', $\diamond$ = 'C', $\bigtriangledown$ = 'D'. The average test error observed was 6%, for four-part models.

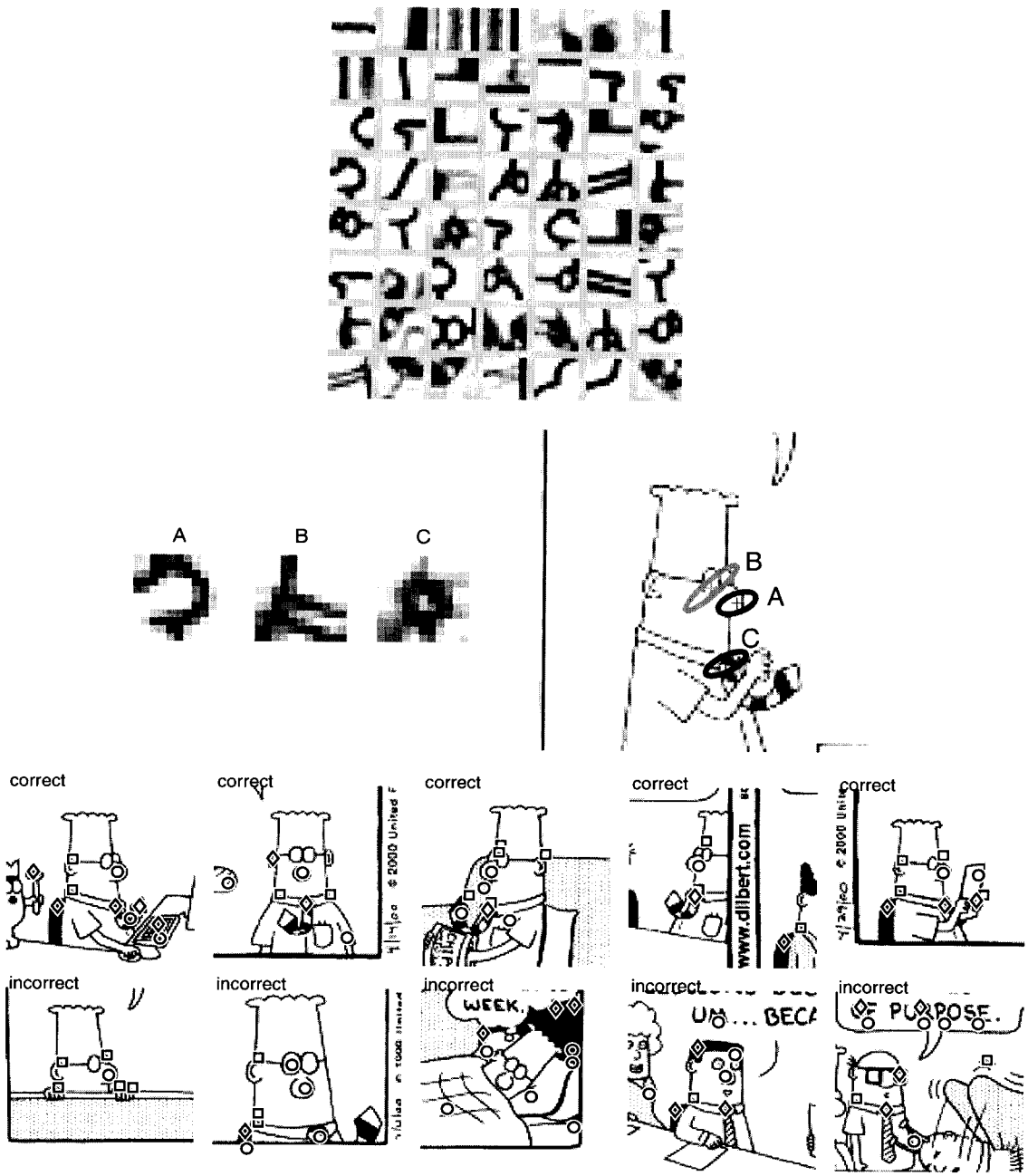


Figure 9.9: Results of Experiment 1d. On the top, a sample of the total of 121 preselected parts is shown. The part size was set to 11×11 pixels. The center row shows the parts selected automatically for a three-part model for Dilbert's head and the corresponding Gaussian foreground density. The bottom two rows shows examples of correctly and incorrectly classified images from the test set. Part labels are: $\bigcirc$ = 'A', $\square$ = 'B', $\diamond$ = 'C'. The average test error observed was 15% for four-part models.

mance, which can be observed in the plots, indicates that we suffered from moderate overfitting.

In Chapter 7 we introduced the concept of "wildcard parts." While analyzing the learned models, we confirmed that this phenomenon does indeed arise, as is illustrated in Figure 9.11. The possibility of wildcard parts was exploited in particular by models with few parts, by assigning larger variance to such parts and, in turn, allowing for several possible matches with "true" underlying object features. The three-part model in Figure 9.11 assigned a large variance along the vertical direction to a part which often matches the car's upper and lower rear window edges as well as portions of the bumper and license plate. Note that in the similar model with four features, as shown in Figure 9.6, two separate parts with lower variance were dedicated to model the same set of underlying physical features.

9.3 Experiment 2—Occlusion

The purpose of this experiment was to demonstrate the possibility of performing learning and detection when objects are partially obscured. Since our detection and learning methods were explicitly designed to handle this problem, it is important to verify this capability experimentally. The setup for this experiment was essentially the same as in in Experiment 1b, except that the faces were artificially occluded in every training and test image.

9.3.1 Training and Test Images

This experiment is based on the same images of human faces used in Experiment 1b. We artificially occluded each face for testing and training in the following way. First, the exact position of the face was determined by manually selecting, in each image, the centroid of the face. This point was typically slightly above the tip of the nose. The centroid was then chosen as the center of an imaginary circle with a radius sufficiently large to cover the entire head in all instances. We then replaced a segment (wedge) of this circle covering an angular range, $[\alpha, \alpha + \theta]$, with an occluding pattern. The orientation, α, of the occluder was chosen randomly with uniform probability over the range $[0, 2\pi]$. The occluding pattern was selected at random from a set of images containing only background clutter from the same environment in which the original pictures were taken. We chose this particular occlusion procedure because we believe that a realistic occluder would have to extend across the object from some point outside of the region occupied by the object. Examples of occluded images for $\theta = \pi/2$, the amount of occlusion used in the experiment, are shown in

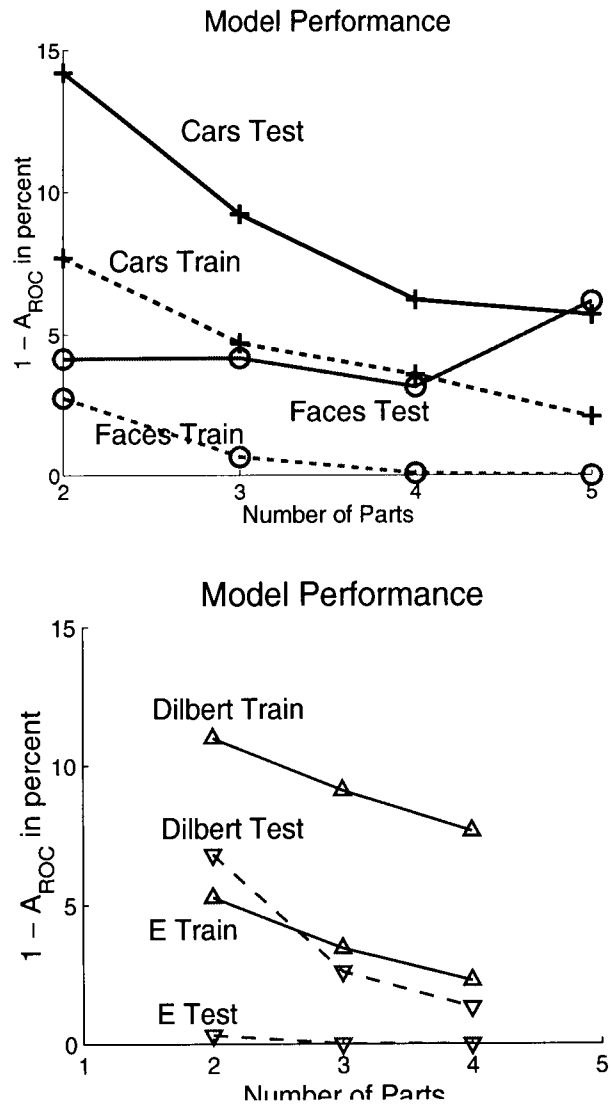


Figure 9.10: The top plot summarizes the classification performance observed in Experiments 1a and 1b for different numbers of model parts. We show average training and testing performance measured as $1 - A_{ROC}$, where A_{ROC} is the area under the corresponding ROC curve. For both models, one observes moderate overfitting. For faces, the smallest test error occurs at 4 parts. Hence, for the given amount of training data, this is the optimal number of parts. For cars, 5 or more parts should be used. The bottom plot shows the performance in Experiments 1c and 1d, where we learned models for the letter 'E' in "Dilbert" cartoons, as well as Dilbert's head. Overfitting can be observed also here. Four or more parts should be used for both data sets.

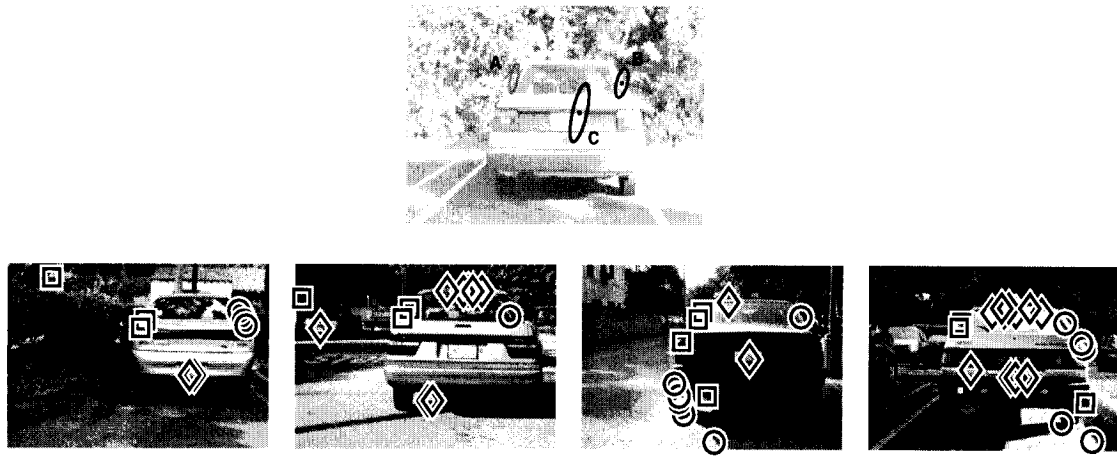


Figure 9.11: Use of "wildcard parts:" The three-part model on top correctly classified the four images (markers are: ◯ = 'A', □ = 'B', ◇ = 'C'). Note that the middle part (C) exhibits a high variance along the vertical direction. It matches several locations in the images, such as the bumper, license plate and roof. In our probabilistic framework, no decision is made as to the *correct match*. Rather, evidence is accumulated across all possible matches. Use of such wildcard parts was observed particularly for models with few parts.

Figure 9.12.

Results

We used exactly the same part selection and learning procedure as in Experiment 1b. We evaluated the performance of the learning and detection method using our standard paradigm of binary classification of a test set of images into the classes "object present" and "object absent." In Experiment 1b we had obtained the best classification results with a part size of 11×11 pixels. Since we suspected that larger parts would be more susceptible to detector failure under occlusion, we performed this experiment with part of sizes 11×11 as well as 9×9. Part selection for this experiment was performed on the occluded images as well.

Figure 9.13 summarizes the results of this experiment. We observed a clear decrease in absolute detection performance from 6% error in the unoccluded setting with a four-part model, to 18% error, in the occluded setting. Overfitting was rather significant in this experiment, with a difference between test and trainig error of 5% (three parts) up to 10% (five parts). This suggests that the asymptotic classification error (for large training sets) lies closer to 10% than the observed 18%. No model size (number of parts) was clearly superior. Although the performance was about identical for parts of size 9×9 pixels and 11×11 pixels, the comparison with Experiment 1, where we

Figure 9.12: Examples of the occluded faces used in Experiment 2. An angular portion of $\theta = \pi/2$ (one quadrant) was obscured by a randomly selected background pattern in each training and test image.

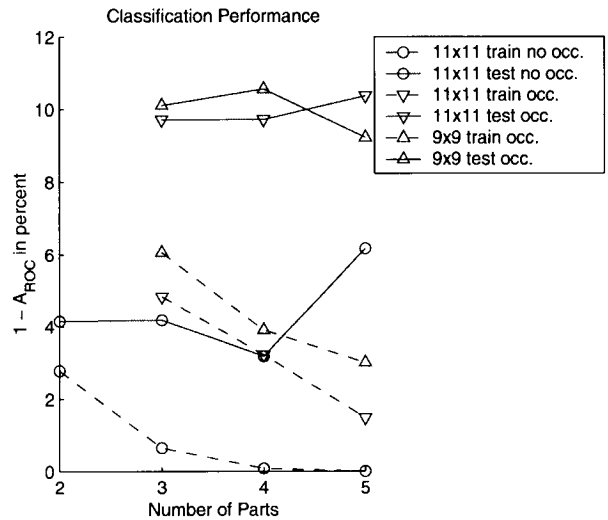
observed an advantage of the latter size over the fomer, reveals that the models with larger parts suffered a more significant degradation of performance.

9.4 Experiment 3—Multi-Scale Models

In all previous experiments, we used model parts selected at a single spatial resolution. There is no fundamental reason that this has to be the case. On the contrary, it could even be advantageous to combine parts across different scales in the same model. We tried this approach on the data set of human faces (used in Experiment 1b).

For each training image we computed a Gaussian image pyramid by downsampling the image five times by a factor of $1/\sqrt{2}$ (see Figure 9.14. Part preselection was then done by combining, in one large set, all patterns selected around points of interested across all scales, before performing k-means clustering. The resulting set contained 440 patterns and was thus about four times as large as in the single-scale experiments. This slowed down the part selection process significantly.

We only learned models with three and four parts (see Figure 9.15). Overfitting was more



occlusion	part size	number of parts	% training error	% test error
no	11×11	2	6	6
no	11×11	3	4	9
no	11×11	4	2	6
no	11×11	5	0	12
yes	11×11	3	12	18
yes	11×11	4	10	18
yes	11×11	5	8	20
yes	9×9	3	14	19
yes	9×9	4	11	20
yes	9×9	5	8	18

Figure 9.13: Results of Experiment 2 (occluded human faces). The top diagram shows the classification performance (measured as $1 - A_{ROC}$, where A_{ROC} is the area under the corresponding ROC curve), for detection of occluded human faces with part sizes of 9×9 and 11×11 pixels and non-occluded faces with part size 11×11 pixels. The table below shows absolute classification errors.

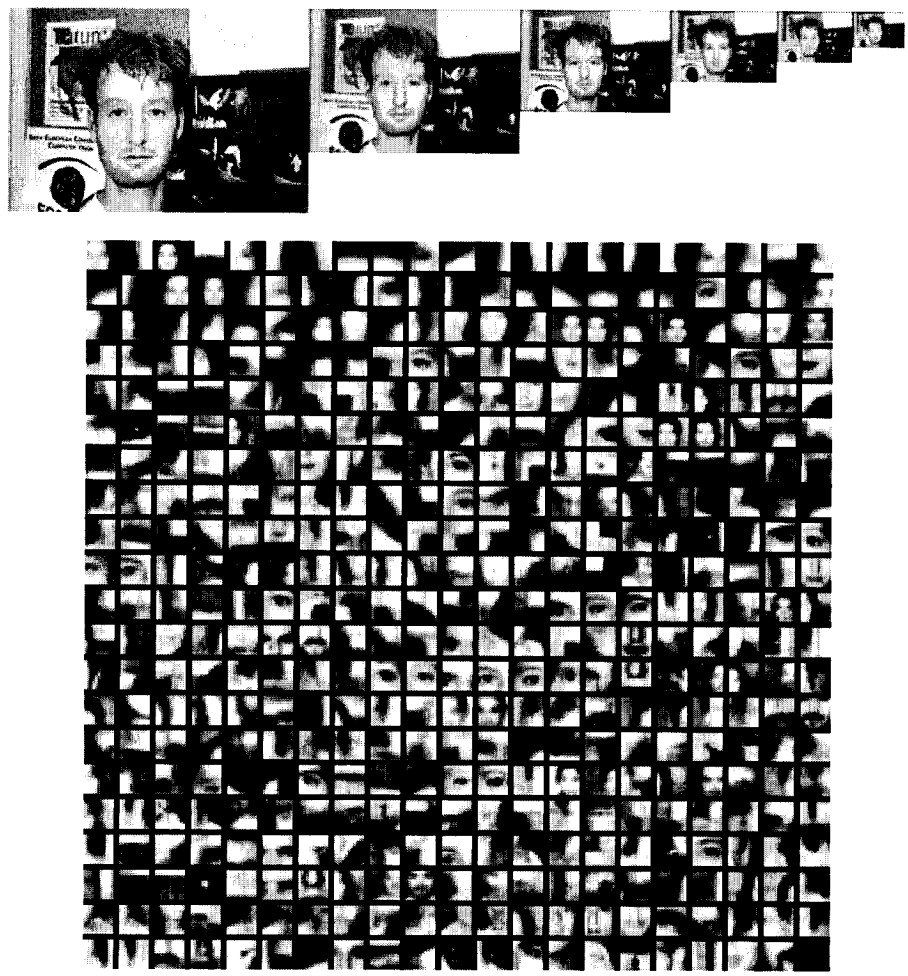


Figure 9.14: Multi-scale part selection for Experiment 3. Top images: A Gaussian image pyramid obtained by subsampling the leftmost image in five times by a factor of $1/\sqrt{2}$, after applying a Gaussian smoothing filter. On the bottom we show the entire set of 440 parts of size 11×11 pixels, preselected across all six scales. The patterns are sorted by cluster size, in descending order from top left to bottom right.

pronounced in this experiment than in Experiment 1, which is illustrated by the fact that models with a zero training error were frequently produced (such as the one on the right in the figure). At the same time, the test error was about twice as large as in Experiment 1 (12% vs. 6%). We observed a smaller spatial variance in the foreground density in this experiment (the ellipses in this plot are drawn at *two* standard devations from the mean), which could explain the larger test error. We found that parts on scale 2 were chosen most of the time, as in the left model in the figure. This scale corresponds to the part size which we have found to work best in Experiment 1 (see choice of parts in Figure 9.7). Note that one part (B) of the right model is a low-resolution representation of the entire face. In order to decide whether the multi-resolution approach is beneficial, this experiment should be repeated with a larger training set.

9.5 Experiment 4—View-Based Models of Human Heads

In this experiment we investigated the possibility of learning *view-based* models for situations with a significant variation in the three-dimensional viewing direction. In particular, we used images of human heads rotated around the vertical axis over a range of more the 90°. Models were learned and tested for different ranges of viewing angles to asses the limitations of single models and to search for optimal combinations of models.

9.5.1 Training and Test Images

In order to produce a large set of images with different but known head orientations, a sufficient number of subjects, as well as different, cluttered backgrounds, we resorted to a synthetic blending of head images with background scenes. Subjects were photographed in front of a blue background, facing 9 different viewing directions ($-15°, 0°(=$ frontal), $15°, \ldots, 90°(=$ profile), $105°$). The background was then subtracted from the images (which were converted to grayscale) and replaced with entirely white or black regions to produce training images for parameter estimation (see Fig. 9.16, top), as well as with random scenes to produce validation and test images (see Fig. 9.16, bottom).

The idea underlying the pure black and white backgrounds is that points on the silhouette of the face might be useful features, if there is a reasonable chance that the face is seen against a fairly uniform background that is either darker or brighter than the head itself. Since we used normalized correlation for part detection, it is sufficient that the background be *slightly* different in brightness

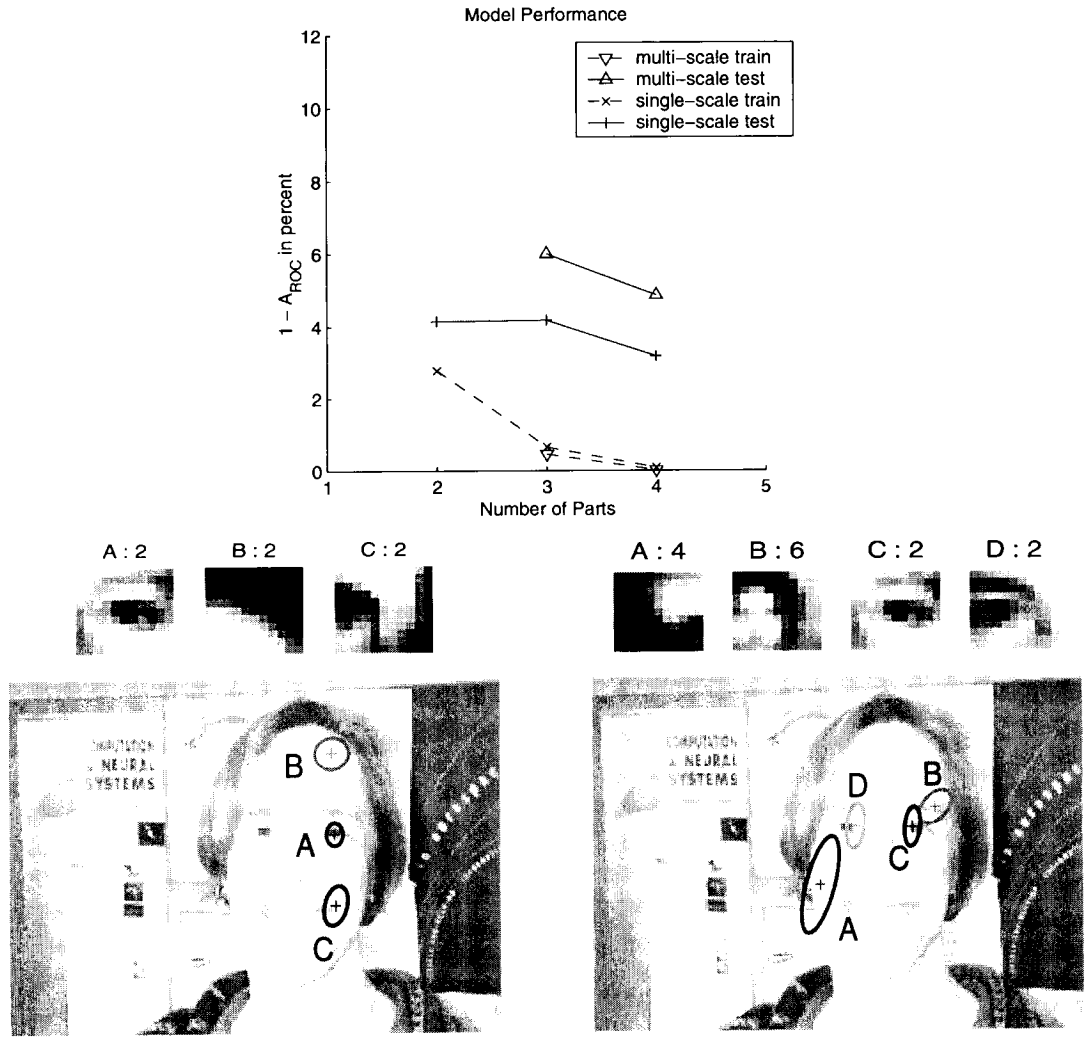


Figure 9.15: Results of Experiment 3 – Multi-scale part learning. The top plot shows a comparison of the performance between single-scale (results taken from Experiment 1) and multi-scale models for different numbers of model parts, based on the ROC area measure. The smallest classification test error observed in the multi-scale experiment was 12%, for four parts. Also shown are two models with three and four parts. Ellipses are drawn at *two* standard deviations from the mean part positions. The numbers behind the part labels indicate the scale at which the parts were chosen, where 1 corresponds to the leftmost image of the Gaussian pyramid shown in Figure 9.14 (top) and 6 to the rightmost.

from the head, since any such difference will be amplified through the normalization. We used black *and* white backgrounds to avoid biasing the models toward dark or bright backgrounds and to encourage the use of non-silhouette features.

The background scenes not used for the blending were selected as negative examples in the classification task. Overall, we used images of 22 subjects. This set was divided into two non-overlapping, equally large sets for training and testing. Four pictures were taken of each subject at every viewing direction.

The set of background scenes contained 150 pictures of landscapes, outdoor scenes with buildings and cars, as well as indoor scenes of office and laboratory environments. This set was divided into two sets of 75 pictures for training and testing.

Results

In three separate experiments (4a – 4c) we trained models on single views, a viewing range of 30° (centered around 0°, 45° and 90°) and the total range from 0° to 90°.

We preselected parts of size 11 × 11 pixels for each viewing direction independently (see Fig. 9.17).

In order to limit the computation time and the amount of overfitting, we only trained models with three parts.

The results from all three experiments are summarized in the tuning curves shown in Figure 9.18, which were obtained by testing each model across the entire range of viewing directions. The models showed good detection performance (about 90% correct) across a significant range of viewing directions. The best performance and generalization across novel viewing angles were obtained in Experiment 4b, where models were trained on three adjacent viewing directions. We found this ability to generalize across viewing directions remarkable. It was due to a choice of very versatile parts comprised, e.g., of features along the hairline. This finding is further illustrated in Experiment 5, were we trained mixture models on the same data set of human heads.

Some examples of detections and failures are shown in Figure 9.19.

It is furthermore interesting that the model trained over the entire range of viewing angles performed consistently worse than the model trained over the 30° to 60° range. There is no simple explanation for this, since the same model could have been learned, in principle, in both cases. The reason therefore has to be sought in the dynamics of the learning method, in particular, the part selection process, which seems to perform differently for the different training sets.

Figure 9.16: Examples of images used to learn view-based models. The top images were used for component preselection and EM parameter estimation. The center and bottom images were used for validation and testing. The bottom images were created through synthetic blending of the top images with background images, such as those shown in the center.

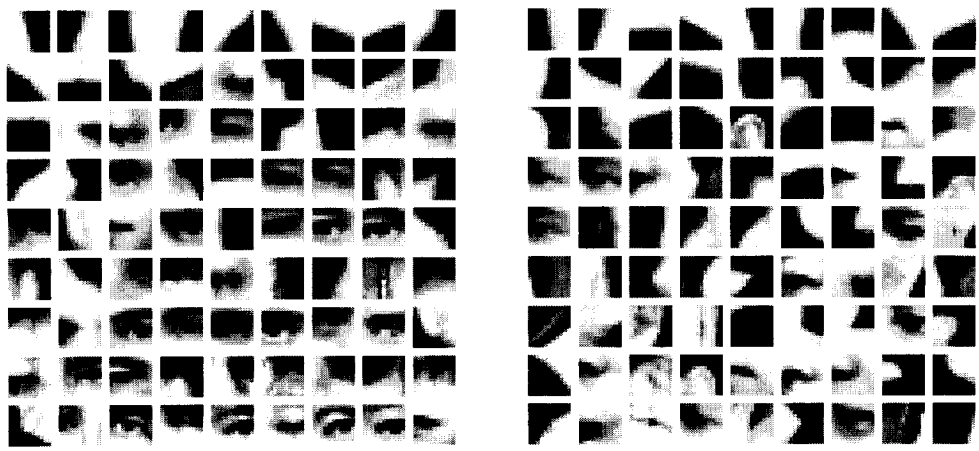


Figure 9.17: Samples of parts preselected independently for each viewing direction. The left set was obtained from frontal views, the right one from semi-profile (45°) views. Since the heads were presented against a constant background (black and white), the preselected parts are not "contaminated" by patterns stemming from background clutter. This considerably reduces the computation time for the final part selection.

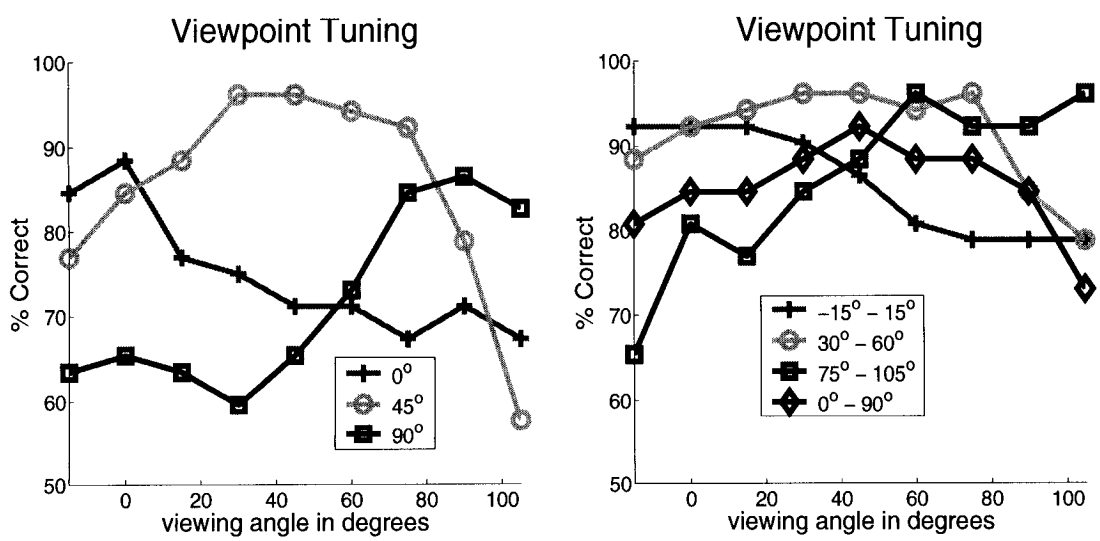


Figure 9.18: Tuning curves showing test performance across all viewing angles for models learned in Experiment 4a (left) and Experiments 4b and 4c (right).

Figure 9.19: Examples of correctly and incorrectly classified images for models trained on 0° (left), 45° (center), and 90° (right) views. Models are comprised of three parts. Part candidates are labeled accordingly with three different markers. Markersize indicates the probability of a part candidate to correspond to the foreground object. This information can be used to *locate* the heads. Classification errors result mostly from failure to detect enough parts, due to large tilt/slant or scale difference.

9.6 Experiment 5—Automatic Discovery of Categories

The experiments discussed in this section (Experiments 5a—5c) were designed to test the mixture model introduced in Chapter 8. We therefore used image data with a high degree of variability for which we suspected an underlying multi-modal signal distribution. We used images of cars seen from the rear and the side, images of human heads (as in Experiment 4) and images of leaves from three species of trees. The setup for learning mixture models was exactly the same as for single-component models.

The important questions we were interested in are, firstly, to what extent the use of mixture models increases classification performance on data sets with high variability, and, secondly, in which way the "responsibility" of the component models is distributed across the entire set of signals. In particular, we wanted to test whether the learning algorithm for mixture models would discover meaningful sub-classes or categories within the provided data sets. We therefore analyzed the tuning characteristics of the trained mixture models.

In a further experiment (5d) we took a more supervised approach and investigated the possibility of training individual mixture components separately on pre-defined categories.

9.6.1 Data Sets

Back and Side Views of Cars

We used the same images of rear views of cars as in Experiment 1a. In addition, we collected images of cars taken from the left side. Figure 9.24 (center) shows examples of the latter. All car images were high-pass filtered as in Experiment 1a.

Human Heads

The data set of human heads was identical to the one used in Experiment 4.

Leaves

We photographed leaves (from three species of trees) in front of cluttered indoor laboratory scenes. The three species were similar in brightness and reflectance but varied significantly in shape (see Fig. 9.20). We made an effort to keep the size constant across species. The leave images were also high-pass filtered.

Figure 9.20: Examples of the image database of leaves (top) and the corresponding background images (bottom).

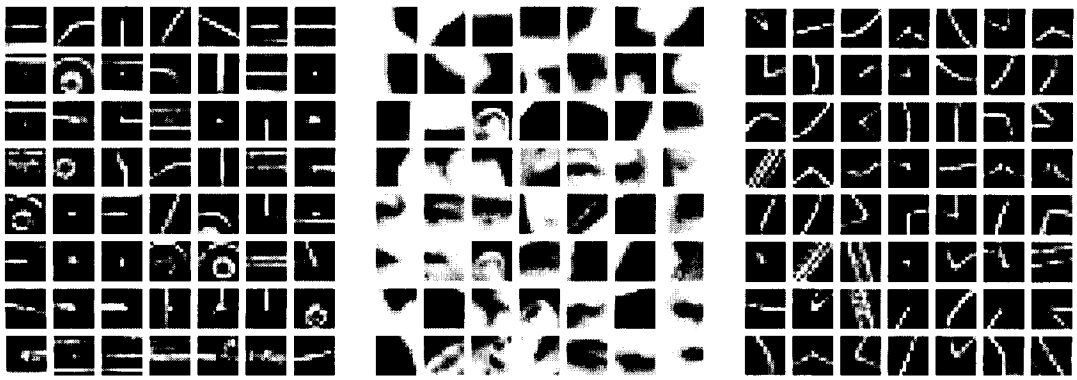


Figure 9.21: preselected parts are shown for cars (left), heads (center), and leaves (right). The leaf and car images were high-pass filtered. The total number of patterns selected was 175 for heads (across a 90° range of viewing directions), 89 for cars (across two orthogonal viewing directions), and 81 for leaves (across all three species).

9.6.2 Results

We performened the part preselection on the combined sets of images (see Fig 9.21).

We trained models with 1 and 2 mixture components on the car data, with 1, 2, and 3 components on leaves and with 1, 2, and 4 components on the heads data set. The larger models (3 components for leaves and 4 for heads) were only trained to investigate their tuning characteristics. In these cases, we therefore did not set aside data for testing but rather used all available data for training. The individual component models always comprised three parts.

Figure 9.22 shows the anatomy of a model with four mixture components trained on the set of human heads.

In Table 9.1 we provide an overview of the classification performance obtained in all experiments. We included the results of Experiment 5d, in which we trained individual single-component models separately on side and back views of cars. We then merged the two single-component models into one two-component model, assigning each model equal prior responsibility ($p(\omega_1) = p(\omega_2) = 0.5$) and evaluated the performance on the test set. The two-component model obtained in this way performs clearly worse than the two-component models trained on the entire, unlabeled data set. Hence, it appears that training components simultaneously within a mixture models is crucial.

We investigated the tuning properties of the individual component models by assigning every training image to that particular component, ω_k, for which the "responsibility," $p(\omega_k|\mathcal{X})$, was maximal. If no component had a responsibility of more than 80%, we assigned the corresponding image

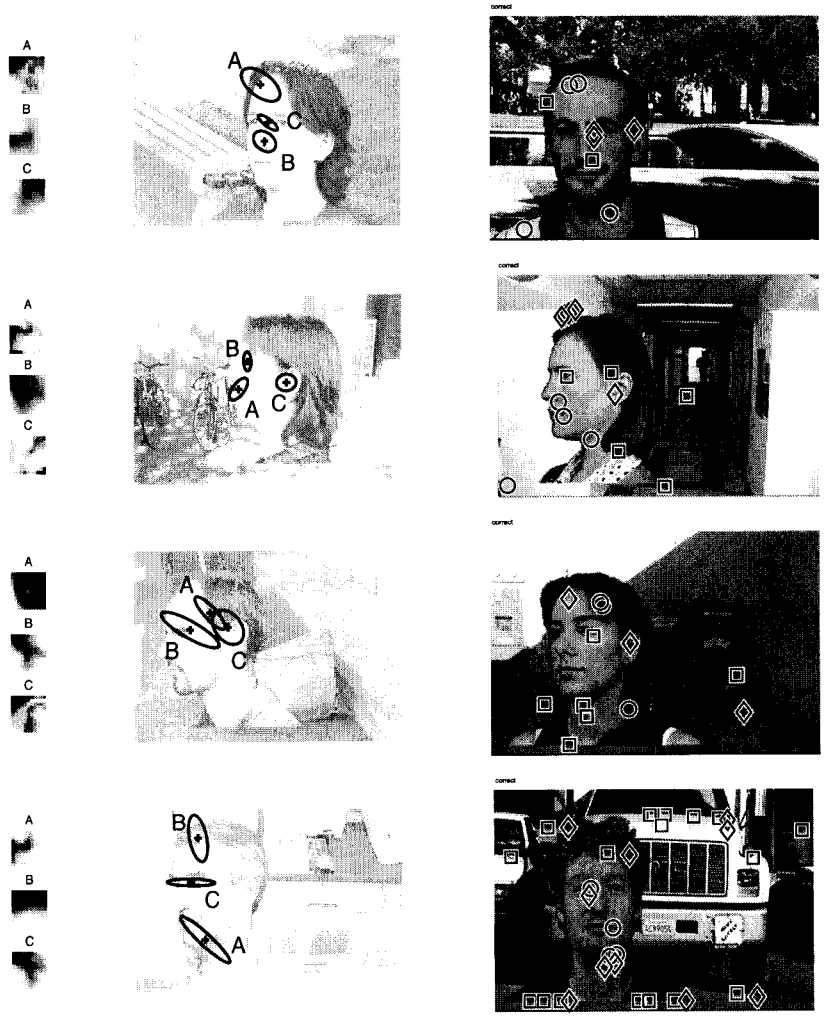


Figure 9.22: Analysis of a four-component mixture model with three parts per component. This model was trained on the *entire* set of available data. We show the three parts selected for each component (left) by the training algorithm. They have been superimposed on an arbitrary image for which the corresponding component had a high "responsibility." On the right we show a second arbitrarily selected image, also with a high responsibility of the corresponding component. To reduce clutter, detected candidate part locations are shown *only* for the three part types of the respective component (o = 'A', □ = 'B', ◇ = 'C'). Note that the model components are able to represent the heads over a fairly large range of viewing directions (about 45°), due to the stability of the chosen parts. This effect was even more pronounced when we trained models with only two components over a 90° viewing range. In this case, single components were often able to detect a head over the entire range. Hence, the data set could be considered not difficult enough for the mixture models.

	Cars		Heads			Leaves		
# components	1	2	1	2	4*	1	2	3*
train	17%	12%	14%	13%	5%	8%	8%	6%
test	18%	16%	14%	13%	n/a	16%	16%	n/a
ZFA-DR	46%	54%	57%	61%	n/a	59%	60%	n/a
Exp. 5d train	13%[1] / 12%[2]	n/a						
Exp. 5d test	18%[1] / 14%[2]	24%						

Table 9.1: Test and training performance for the detection experiments. We show the total misclassification error, obtained at the decision treshold where an equal number of positive and negative images is misclassified. We also report the detection rate at zero false alarms (ZFA-DR). This performance is important for applications such as digital library searches. *Models with three (leaves) and four (heads) components were trained on *all* available data for the tuning experiments. Therefore no test error is available. [1]refers to the performance (test and train) when trained on back views only, [2] is for side views only.

to an "undecided" category. We were able to observe clear tuning characterstics for all three data sets. In the case of leaves, only a model with three components showed tuning characteristics, but not with two. In the head experiment we only observed tuning for models with four components. Figure 9.23 shows histograms illustrating these findings.

While further analyzing the tuning of the head model we made the interesting observation that components were also tuned to the differences in head *size*, which remained in the data despite our efforts keep the scale constant across images. As can be seen in Figure 9.23, the components 1 and 2, as well as 3 and 4, have a similar orientation tuning. However, they differ in their preferred head size. The difference in head sizes in the data becomes evident, e.g., when considering the images in the fourth row of Figure 9.19. On the other hand, this tuning to head size could only occur since the individual component models were able to detect heads over an astonishingly wide range of orientations, as illustrated in Figure 9.22, and already found in Experiment 4.

9.7 Hard vs. Soft Part Detection

In this experiment, which is not concerned with learning, we investigated the benefits of "soft" part detection. By this we mean that no hard threshold is applied for part detection. Instead, the actual detector output, which can be interpreted as the likelihood that a part is present at a given location, is retained and later incorporated into the hypothesis evaluation function.

The idea for this experiment is due to Burl [Bur97] and it is based on the model that we discussed

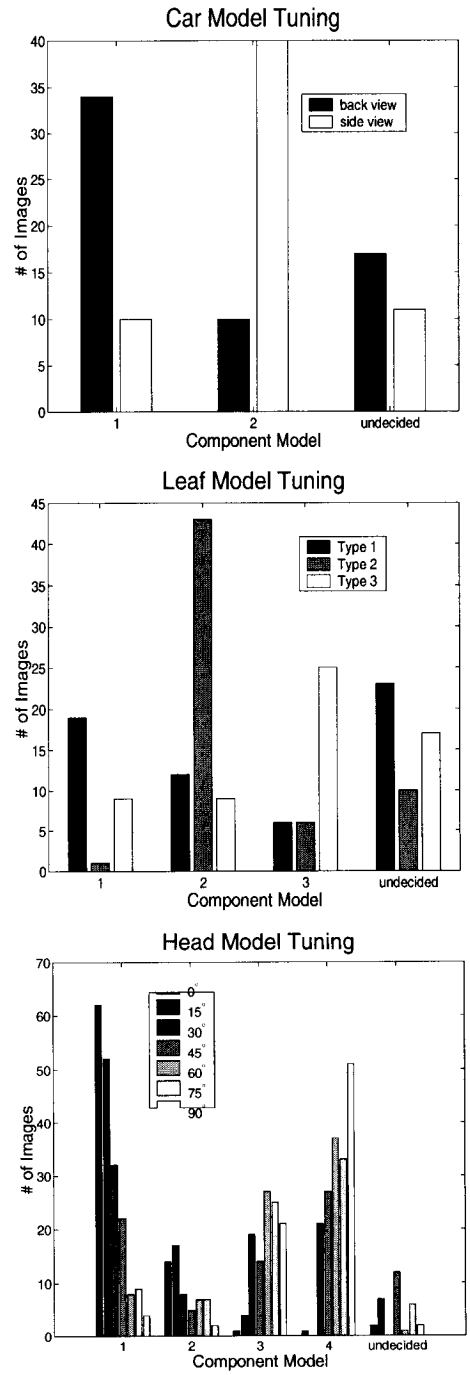


Figure 9.23: Histograms showing tuning properties of the mixture models. For every component model the number of images for which it is "responsible" is plotted. The right-most group of bars show "undecided" images, where no model reached a value of more than 80% probability. The number of components was two (cars), three (leaves) and four (heads). Head models were found to be partially tuned to head orientation but also to small remaining differences in head sizes in our data sets.

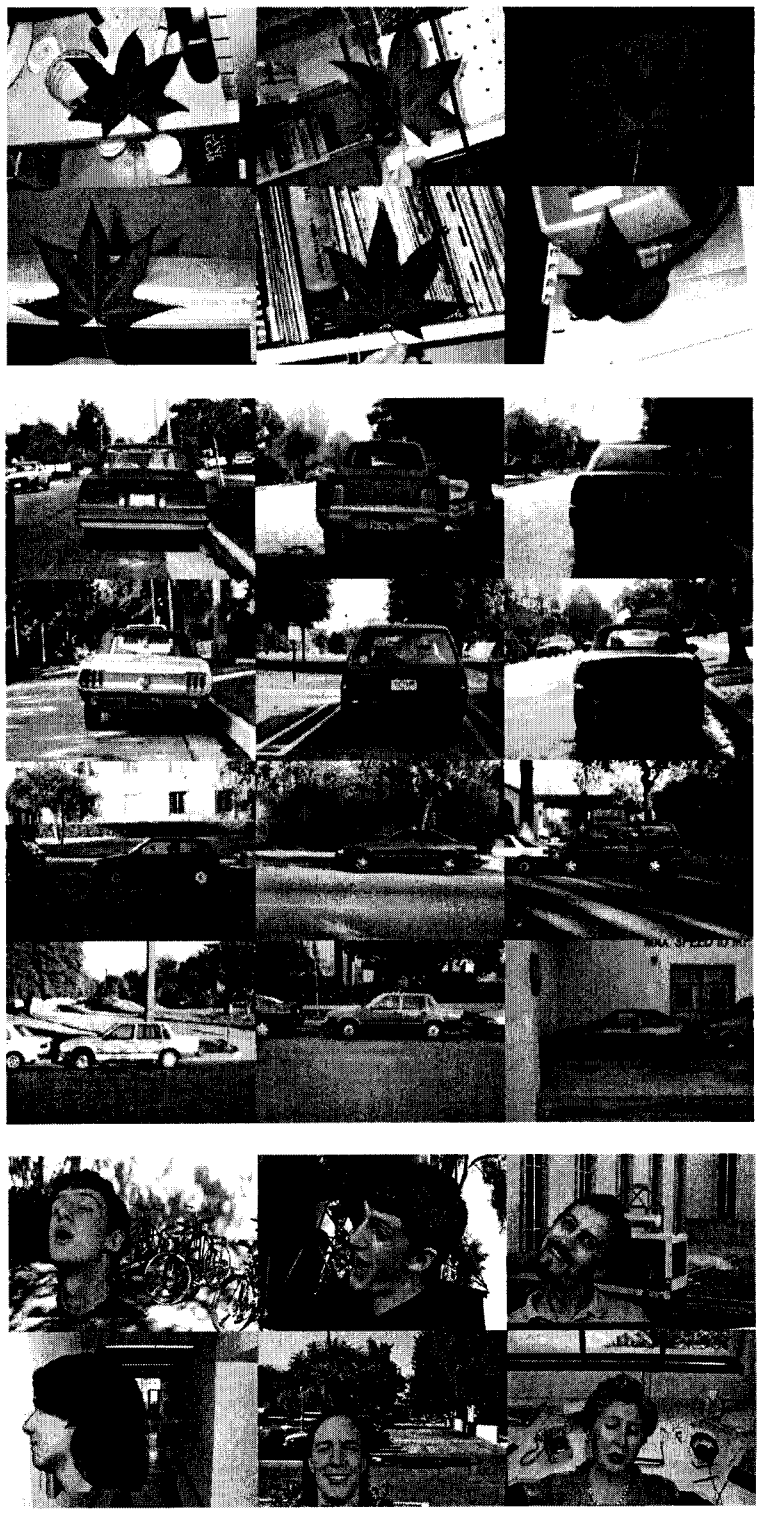


Figure 9.24: Examples of misclassified images from the three test sets.

in Chapter 2. We also evaluate hypotheses in shape space, using Bookstein's shape coordinates. This is different from the constellation models considered so far, which were only translation invariant.

9.7.1 TRS-Invariant Approximation to the Optimal Detector

The approximate log-likelihood ratio given in (2.13) can readily be interpreted as a combination of two terms: the first term, $\sum A_i$, measures how well the hypothesized parts in the image match the actual model parts, while the second term, $p(\mathbf{x}_0)$, measures how well the hypothesized spatial arrangement matches the ideal model arrangement. The second term, the configuration match, is specified as a probability density over the absolute coordinates of the parts, which in practice is not useful since (a) there is no way to know or estimate this density and (b) this formulation does not provide any invariance to translation, rotation, or scaling (TRS-invariance).

We can make use of the theory developed in our previous work (see [BLP96] or [BWLPss]) to write down a TRS-invariant detector that closely follows the form of (2.13). In particular, we know how to factor the term $p(\mathbf{x}_0)$ into a part that depends purely on shape and a part that depends purely on pose:

$$p_{\mathbf{x}}(\mathbf{x}_0) = p_{\mathbf{u}}(\mathbf{u}_0(\mathbf{x}_0)) \cdot p_{\Theta}(\mathbf{\Theta_0}(\mathbf{X_0})). \tag{9.1}$$

Here, $\mathbf{u}$ denotes the *shape* of the constellation and the vector $\mathbf{\Theta}$ captures the pose parameters. Computing $\mathbf{U}(\mathbf{x})$ corresponds to transforming a constellation $\mathbf{x}$ in the image to so-called *shape space* by mapping two part positions (the *base-line pair*) to fixed reference positions. In shape space, the positions of the remaining $N-2$ parts define the shape of the configuration, written as

$$\mathbf{u} = \begin{bmatrix} u_3 & u_4 & \dots & u_N & v_3 & v_4 & \dots & v_N \end{bmatrix}^T \tag{9.2}$$

If $p_{\mathbf{x}}(\mathbf{x})$ is a joint Gaussian density, then the shape density, $p_{\mathbf{u}}(\mathbf{u})$, can be computed in closed form as shown by Dryden and Mardia [DM91]. This established, we can obtain the TRS-invariant detector by dropping the pose information completely and working with shape variables $\mathbf{u}_0$, instead of figure space variables $\mathbf{x}_0$. The resulting log-likelihood ratio is then

$$\log \Lambda_1 = \sum_{i=1}^{N} A_i(x_{0i}, y_{0i}) + K \cdot \log \frac{p_U(\mathbf{u}_0|\omega_1)}{p_U(\mathbf{u}_0|\omega_0)}. \tag{9.3}$$

The shape likelihood ratio, rather than just $p_{\mathbf{u}}(\mathbf{u}_0)$, is used in place of $p_{\mathbf{x}}(\mathbf{x}_0)$ to provide invariance to the choice of baseline features. The likelihood ratio also assigns lower scores to configurations that have higher probabilities of accidental occurrence. The factor of K provides a weighted trade-off between the part match and shape match terms, since the units of measurement for the two terms will no longer agree. (The proper setting for this value can be estimated from training data).

An object hypothesis is now just a set of N coordinates specifying the (hypothesized) spatial positions of the object parts. Any hypothesis can be assigned a score based on (9.3). It is no longer the case that hypotheses must consist only of points corresponding to the best part matches. The trade-off between having the parts match well and having the shape match well may imply that it is better to accept a slightly worse part match in favor of a better shape match or vice versa.

We do not have a procedure for finding the hypothesis that optimizes $\log \Lambda_1$. One heuristic approach, $\mathcal{A}_1$, is to identify candidate part locations at maxima of the part detector responses and combine these into hypotheses using the conditional search procedure described in [BLP96]. However, instead of discarding the response values, these should be summed and combined with the shape likelihood. In this approach, the emphasis is on finding the best part matches and accepting whatever spatial configuration occurs. There is no guarantee that the procedure will maximize $\log \Lambda_1$.

Figure 9.25 illustrates the gain of approach $\mathcal{A}_1$ over hard detection. The two components of the goodness function (sum of responses and shape log-likelihood) can be seen as dimensions in a two-dimensional space. Evaluating the goodness function is equivalent to projecting the data onto a particular direction, which is determined by the trade-off factor K. Fisher's linear discriminant (FLD) (see, e.g., [Rip96]) provides us with the direction that maximizes the separability of the two classes. If the sum of the detector responses had no discriminative power, the value of K would tend toward infinity. This would correspond to a horizontal line in the figure. The advantage of soft detection is further illustrated in Fig. 9.26.

A second approach, $\mathcal{A}_2$, is to insist on the best shape match and accept whatever part matches occur. This method is roughly equivalent to using a rigid matched filter for the entire object, but applying it at multiple orientations and scales.

Finally, we tested a third approach, $\mathcal{A}_3$, that intuitively seems appealing. Candidate part locations are identified as before in $\mathcal{A}_1$ at local maxima in the part response image. From pairs of candidate parts, the locations of the other parts are estimated to provide an initial hypothesis. (So far, this is equivalent to using a fixed-shape template anchored at the two baseline points). From the

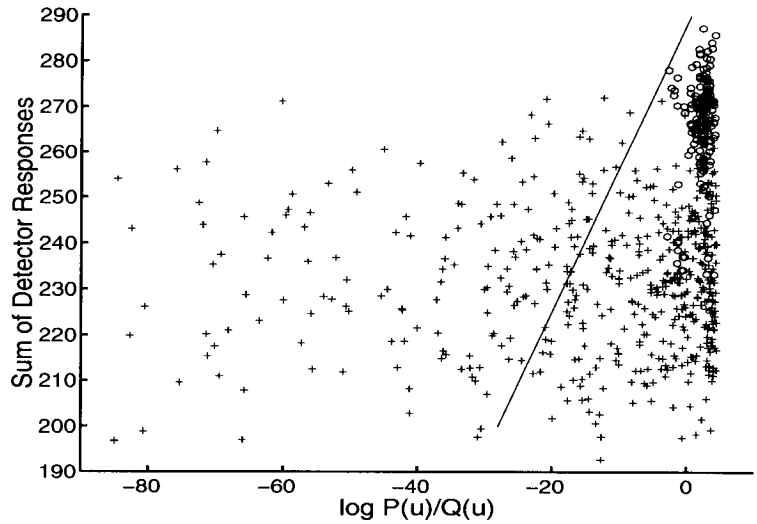
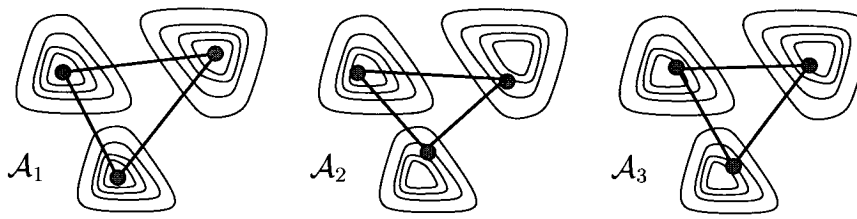


Figure 9.25: Illustration of the advantage of *soft* detection. The sum of the detector outputs is plotted against the shape log-likelihood, for a set of face (o) and background (+) samples. Also shown is a line onto which the data should be projected (derived by Fisher's linear discriminant method).

Figure 9.26: Two constellations of candidates for part locations are shown. The background constellation (black 'x') yields a greater shape likelihood value than the correct hypothesis (white '+'). However, when the detector response values are taken into consideration, the correct hypothesis *will* score higher.



part resp.:	optimal	part resp.:	suboptimal	part resp.:	suboptimal
shape:	suboptimal	shape:	optimal	shape:	suboptimal
combined:	suboptimal	combined:	suboptimal	combined:	optimal

Figure 9.27: Pictorial illustration of the three approaches $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$ discussed in the text. For each approach we show a set of three contours that represents the superposition of response functions from three part detectors. With approach $\mathcal{A}_1$ the detector responses are optimal, but the combination of responses and shape is suboptimal. With approach $\mathcal{A}_2$ the shape likelihood is optimal, but the combination is still suboptimal. Only under approach $\mathcal{A}_3$ is the combined likelihood function optimized by seeking a compromise between contributions from the detector responses and shape.

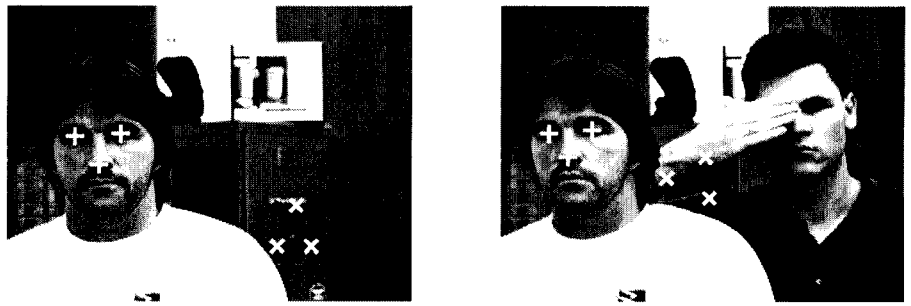
initial hypothesis, however, a gradient-style search is employed to find a local maximum of $\log \Lambda_1$. Individual part positions are pulled by two forces. One force tries to maximize the response value while the other force tries to improve the shape of the configuration.

9.7.2 Experiments

We conducted a series of experiments aimed at evaluating the improvements over hard detection of object parts, brought about by the different approaches described in the previous section. To test our method, we chose the problem of detecting faces from frontal views. A grayscale image sequence of 400 frames was acquired from a person performing head movements and facial expressions in front of a cluttered background. The images were 320×240 pixels in size, while the face occupied a region of approximately 40 pixels in height. Our face model was comprised of five parts, namely eyes, nose tip, and mouth corners.

For the part detectors we applied a correlation based method—similar to a matched filter—acting not on the grayscale image, but on a transformed version of the image that characterizes the dominant local orientation. We found this method, which we previously described in [BWLPss], to be more robust against variations in illumination than grayscale correlation. The part detectors were trained on images of a second person. In order to establish ground truth for the part locations, each frame of the sequence was hand-labeled.

Prior to the experiment, shape statistics had been collected from the face of a third person by



	Best Correct	Best False
Σ Responses	101.1	93.7
Shape Log-LH.	1.457	-0.096
Weighted Total	101.5	93.7

	Best Correct	Best False
Σ Responses	96.4	94.8
Shape Log-LH.	3.460	-3.530
Weighted Total	97.5	93.7

Figure 9.28: Examples from the sequence of 400 frames used in the experiments. The highest scoring correct and incorrect constellations are shown for each frame. The tables give the values for shape log-likelihood, sum of detector responses as well as overall goodness function.

fitting a joint Gaussian density with full covariance matrix to data extracted from a sequence of 150 images, taken under a semi-controlled pose as discussed in [BLP96].

Soft Detection vs. Hard Detection

In the first experiment, we found that using the five features described above, recognition on our test sequence under the hard detection paradigm was almost perfect, making it difficult to demonstrate any further improvements. Therefore, in order to render the task more challenging, we based the following experiments on the upper three features (eyes and nose tip) only. In this setting, approach $\mathcal{A}_1$, i.e. combining the part responses with the shape likelihood without any further effort to maximize the overall goodness function (9.3), yields a significant increase in recognition performance. This result is illustrated in Fig. 9.29, where ROC (*receiver operating characteristics*) curves are shown for the hard detection method as well as for approach $\mathcal{A}_1$.

Gradient Descent Optimization

Approach $\mathcal{A}_3$ was tested in a second experiment by performing a gradient descent maximization of the goodness criteria with respect to the hypothesized part positions in the image. There are two potential benefits of doing this: improved detection performance and improved localization accuracy of the part positions. A cubic spline interpolation of the detector response images was

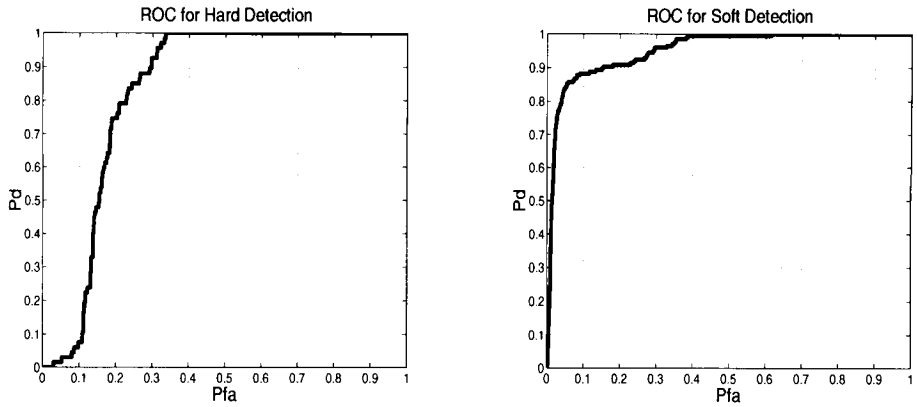


Figure 9.29: The two ROC curves show the performance of hard vs. soft detection of features as a trade-off between detection probability, P_{d}, and probability of false alarm, P_{fa}. The soft detection method $\mathcal{A}_1$ clearly outperforms the hard detection strategy, especially in the low false-alarm range.

used in order to provide the minimization algorithm with a continuous and differentiable objective function. Local maxima of the detector response maps were used as initial estimates for the part positions. We found that, on average, optimal part positions were found within a distance of less than one pixel from the initial positions.

Fig. 9.30 shows the detection performance of the method before and after optimization of (9.3). There does not seem to be any noticeable improvement over approach $\mathcal{A}_1$. This result is somewhat surprising, but not entirely counterintuitive. This is because by optimizing the goodness criteria, we are improving the score of the constellations from both classes, ω_1 and ω_0. It is not clear that, on average, we are achieving a better separation of the classes in terms of their respective distribution of the goodness criteria. From a different perspective, this is a positive result, because the gradient descent optimization is computationally very expensive, whereas we have already been able to develop a 2 Hz real-time implementation of approach $\mathcal{A}_1$.

Since our part detectors did not exhibit a significant localization error for the test data at hand, we have not been able to determine whether approach $\mathcal{A}_3$ might provide improved localization accuracy.

9.8 Summary and Conclusion

We reported results of a number of different learning and recognition experiments. The feasibility of our method was demonstrated using data sets of cars, human faces, handwritten letters, and faces

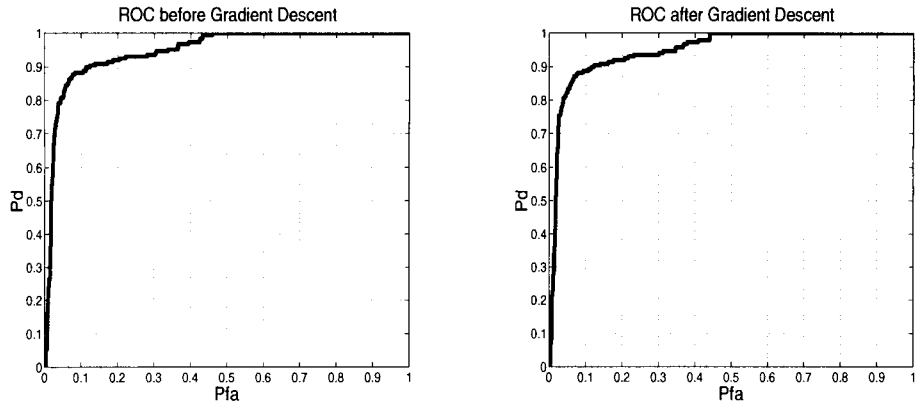


Figure 9.30: The ROC performance does not significantly improve after gradient descent optimization of the goodness criteria.

of cartoon characters. We successfully learned models with 2 to 5 parts. The detection performance was above 90% correct for faces and letters, and above 85% correct for cars and cartoon faces. These errors are a conservative estimate of the achievable performance for larger amounts of training data, since we observed overfitting in all experiments. The running time of the learning method was between 1 hour and 1 day on a standard PC, depending on the number of model parts. The detection performance on occluded human faces (25% of face area occluded) was above 80% correct. We demonstrated that learning is possible in a multi-resolution framework. Due to significant overfitting we did not observe an improvement of the final detection performance when parts were selected across different scales. We demonstrated that learning and detection are possible across a range of viewing directions using a view-based approach on a set of images of human heads, observed over a range of more than 90° of rotation around the vertical axis. Using mixture models, we demonstrated that sub-categories of the training data might be learned in an unsupervised way. We found that components of mixture models exhibit tuning to viewing directions—for images of cars and human heads—and to sub-classes—different species contained in a set of images of leaves. This approach could also be used to simultaneously learn a number of different classes without labeling the training images accordingly prior to learning.

In a separate experiment, we demonstrated that soft detection of parts can improve the detection performance. We also investigated the possibility to trade off part detection confidence with the fit of the global geometry model, when parts are detected in a soft way. Exploiting this trade-off using gradient descent optimization did not noticeably improve the detection performance.

Chapter 10 Conclusion

10.1 Summary

We have reviewed the probabilistic model of *global geometry and local photometry* introduced by Burl et al. [BLP95, BLP96, Bur97, BWP98], under which objects are modeled as flexible constellations of rigid, photometric parts. The deformations of the geometry of part positions are modeled with probability density functions. The more abstract *constellation model* has been introduced, which incorporates some changes over Burl's model. In particular, we have modified the modeling of background detections and proposed a method to use the model for object detection. With Burl's method it was previously only possible to *localize* objects within signals, i.e., the presence of objcts had to be assumed or otherwise established.

The main contribution of this work, however, is a learning algorithm for constellation models. The proposed algorithm can be used in a completely unsupervised way. Provided with a set of training signals, it identifies the class of objects to be learned while it selects parts and estimates the underlying model geometry. Part selection is done in several stages. First, "patterns of interest" are selected in the training signals with a standard interest operator. The patterns are then clustered using the k-means algorithm. This produces a set of 100–200 clusters that could serve as training sets for part detectors. We use simple detectors based on normalized correlation, for which we retain the average pattern of each cluster as a template. The optimal set of a few detectors is then chosen with a method similar to standard feature-selection techniques. The criterion for this choice is the classification performance of the resulting constellation model. For a given choice of parts, the parameters of the model geometry and the detection statistics are estimated using the expectation maximization (EM) algorithm. The EM algorithm is typically used to estimate probability densities in situations where part of the data are not observable. In our case, the unobservable data are the assignments of detected part candidates to either the *foreground*, i.e. an instance of the object class to be learned, or the *background*, i.e. clutter in the signal. This hidden data problem is intricate, because part candidates are not assigned to the foreground or background individually, but in groups corresponding to the parts of the constellation model. In addition, parts missing due to occlusion, natural absence, or detector failure need to be accounted for. Furthermore, our learning algorithm

is translation invariant, since we do not assume to know where in the training signals the objects of interest are located. We have also extended the model as well as the learning method to mixtures of constellation models. This allows us to represent and learn different object classes, as well as "multi-modal" object classes, i.e. classes with distinct sub-categories, simultaneously. This capability can be used to "discover" classes in an unlabeled set of data.

We presented results from several experiments performed on a variety of image sets. We learned and tested the detection performance of models for human faces, cars, leaves, hand-written letters, and cartoon figures. The data sets of leaves, faces, and cartoon figures were collected after the learning method had been finalized. Hence, no adjustments to the algorithms were made to accomodate these data sets. Detection performance was above 90% correct for faces and letters, and close to 90% for the other data sets. We used mixture models to learn representations of human heads across a large range of viewing directions, as well as of cars seen from the rear and from the side, and leaves of three different species. We confirmed that the mixture components of the resulting models were tuned to the different intuitive sub-categories. We also demonstrated that models can be learned and used in a multi-resolution framework. This allows for the optimal part size to be determined automatically. Robustness to partial occlusion of the objects in training and test data was demonstrated as well.

10.2 Limitations and Outlook

10.2.1 Modeling

The number of parts in our models is currently limited to about six. This is due to the exponential increase in computation cost for detection and localization, as well as model learning. Although we have presented an efficient technique that does not require the entire set of possible combinations of part candidates to be formed and evaluated during detection or learning, we are still not able to learn large models with a significant number of parts. Models with a large number of parts might perform well, even if individual parts are less informative, as demonstrated, for example, in the work by Amit and Geman [AG99]. This would allow for smaller parts to be used and the entire model would therefore be less fragile under partial occlusion. Therefore, even more efficient methods should be investigated for detection and learning.

Although we have improved the model of background clutter over Burl's earlier version, the Poisson model we currently use might still be inappropriate. This is because the assumption that

background detections are statistically independent is an oversimplification. We have observed that parts often appear in clusters. For example, if an image contains an area of a certain texture to which some detector is particularly responsive, a large number of detections will result, while the same detector might not respond at all on other images. The same effect was observed for detectors sensitive to edge features. When edges of the appropriate orientation were present, such detectors responded many times along the edges. Although we do not know to which extent this problem influences recognition performance, a more appropriate background model might be developed. Alternatively, one could attempt to define "macro" features based on sets of detections in the image.

Although we have experimented with models that incorporate parts of different spatial resolutions, we have not used such models to detect objects across multiple scales. If part detectors can be devised that report the scale at which parts are detected, it is possible to use the approach outlined in Section 3.3.2. Otherwise, one would have to resort to a shape space representation, as used in [BLP95].

10.2.2 Learning

Our part selection method can be improved on several counts. Firstly, the implementation of Förstner's interest operator is sub-optimal. Some patterns currently produced only have a large gradient along one direction. Those patterns can therefore not be located very reliably. This gives rise to problems such as the multiple detections along edges mentioned above. At the same time, it is unclear if such parts should really be excluded, since we found them to be used successfully in our learned models (see, e.g., the horizontal line feature in Figure 9.6).

The clustering stage is susceptible of improvement as well. It is important that this stage does not produce more clusters than necessary, since the subsequent part selection stage is computationally expensive. One should try to apply a clustering method which automatically splits and merges clusters, in order to arrive at the optimal number of clusters. A similar result could possibly also be achieved by adding a complexity term, such as a minimum description length (MDL) criterion. It would furthermore be beneficial to avoid producing clusters corresponding to patterns that differ only by small translations. We currently eliminate those after the clustering process. However, it might be possible to devise a translation invariant clustering method. Detector pre-selection could potentially also be improved by using a more elaborate technique to generate the correlation templates based on the clustering results. We currently compute the average pattern of each cluster. This leads to a loss of the information contained in higher spatial frequencies, in particular for large

clusters. One could try to find the *best representative* of a given cluster without averaging.

A different possibility is to integrate the part selection process and the parameter estimation method into one procedure. Under such an approach, the representation of individual parts could be refined iteratively, in conjunction with the other model parameters. It is not clear, however, how such a scheme could be initialized, and how transitions from a given set of parts to an entirely different, possibly superior one could be achieved.

Our criteria used to evaluate complete models in the final part selection process should be revisited. We have shown that, in our special case, a normalized likelihood term might be used to predict model performance. This should be investigated analytically. It is possible that a different normalization factor that would better account for the *effective* degrees of freedom in the training data can be defined – the individual part detections are not strictly independent samples of the same process.

Since we observed some amount of overfitting in most experiments, further experiments should be performed with more training data, in order to obtain a more accurate assessment of the achievable classification performance.

We have not yet evaluated the performance of models in object localization tasks. A related issue is the problem of learning and detecting the exact silhouette of objects, based on the positions of the model parts.

We have demonstrated that an object class model can be learned from a set of training signals, even if it is not known which proportion of training signals actually contain instances of the object class. This rises another question: How can we determine, if anything at all has been learned?

Bibliography

[AG99] Y. Amit and D. Geman. A computational model for visual selection. *Neural Computation*, 11(7):1691–1715, 1999.

[AK93] Y. Amit and A. Kong. Graphical templates for image matching. Technical Report 373, Department of Statistics, University of Chicago, August 1993.

[Bar81] R. J. Baron. Mechanism of human facial recognition. *Int. J. Man Machine Studies*, 15:137–178, 1981.

[BAS+98] M.C. Burl, L. Asker, P. Smyth, U. Fayyad, P. Perona, J. Aubele, and L. Crumpler. Learning to recognize volcanoes on Venus. *Machine Learning*, April 1998.

[BFP+94] M.C. Burl, U.M. Fayyad, P. Perona, P. Smyth, and M.P. Burl. Automating the hunt for volcanoes on Venus. In *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, 1994.

[BH94] A. Baumberg and D. Hogg. Learning flexible models from image sequences. In *Proc. 3rd Europ. Conf. Comput. Vision, J.-O. Eklundh (Ed.), LNCS-Series Vol. 800-801, Springer-Verlag*, pages 299–308, 1994.

[Bic96] M. Bichsel. Automatic interpolation and recognition of face images by morphing. In *FG96*, pages 128–129, Killington, Vermont, 1996.

[Bis95] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[BLP95] M.C. Burl, T.K. Leung, and P. Perona. Face localization via shape statistics. In *Int. Workshop on Automatic Face and Gesture Recognition*, 1995.

[BLP96] M.C. Burl, T.K. Leung, and P. Perona. Recognition of planar object classes. In *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, 1996.

[Boo84] F.L. Bookstein. A statistical method for biological shape comparison. *J. Theor. Biol.*, 107:475–520, 1984.

[Boo86] F.L. Bookstein. Size and shape spaces for landmark data in two dimensions. *Statistical Science*, 1(2):181–242, 1986.

[BP93] R. Brunelli and T. Poggio. Face recognition: Features versus templates. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(10):1042–1052, October 1993.

[Bur97] M.C. Burl. *Recognition of Visual Object Classes*. PhD thesis, Department of Electrical Engineering, California Institute of Technology, Pasadena, CA, 1997.

[BWLPss] M.C. Burl, M. Weber, T.K. Leung, and P. Perona. *From Segmentation to Interpretation and Back: Mathematical Methods in Computer Vision*, chapter Recognition of Visual Object Classes. Springer, in press.

[BWP98] M.C. Burl, M. Weber, and P. Perona. A probabilistic approach to object recognition using local photometry and global geometry. In *proc. ECCV'98*, pages 628–641, 1998.

[Cov74] T.M. Cover. The best two independent measurements are not the two best. *IEEE Transactions on Systems, Man, and Cybernetics*, 4:116–117, 1974.

[CT95] T.F. Cootes and C.J. Taylor. Combining point distribution models with shape models based on finite element analysis. *Image and Vision Computing*, 13(5):403–409, 1995.

[CV95] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:1–25, 1995.

[CvC77] T.M. Cover and J.M. van Campenhout. The possible orderings in the measurement selection problem. *IEEE Transactions on Systems, Man, and Cybernetics*, 7(9):657–661, Sept. 1977.

[DH73] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, Inc., 1973.

[DLR76] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical society* **B**, 39:1–38, 1976.

[DM91] I.L. Dryden and K.V. Mardia. General shape distributions in a plane. *Adv. Appl. Prob.*, 23:259–276, 1991.

[ETC98] G.J. Edwards, T.F.Cootes, and C.J.Taylor. Face recognition using active appearance models. In *Proc. 5th Europ. Conf. Comput. Vision, H. Burkhardt and B. Neumann (Eds.), LNCS-Series Vol. 1406–1407, Springer-Verlag*, pages 581–595, 1998.

[FE73] M.A. Fischler and R.A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computer*, c-22(1):67–92, Jan. 1973.

[FPHK94] F. Ferri, P. Pudil, M. Hatef, and J. Kittler. Comparative study of techniques for large-scale feature selection. In E.S. Gelsema and L.N. Kanal, editors, *Pattern Recognition in Practice IV*, pages 403–413. Elsevier Science B. V., 1994.

[Fuk90] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 2nd edition, 1990.

[FW] R. Fergus and M. Weber. Unsupervised learning of models for car recognition. Unpublished report.

[Hot33] H. Hotelling. Analysis of a complex of statistical variables into principal components. *J. Educ. Psych*, 24:417–441,498–520, 1933.

[HS93] R.M. Haralick and L.G. Shapiro. *Computer and Robot Vision II.* Addison-Wesley, 1993.

[HSG98] G. E. Hinton, B. Sallans, and Z. Ghahramani. A hierarchical community of experts. In M. I. Jordan, editor, *Learning in Graphical Models*. Kluwer Academic Publishers, Dordecht, The Netherlands, 1998.

[HU87] D.P. Huttenlocher and S. Ullman. Object recognition using alignment. In *Proc. 1st Int. Conf. Computer Vision*, pages 102–111, 1987.

[HW89] J. Hampshire and A. Waibel. The meta-pi network: Building distributed knowledge representations for robust pattern recognition. Technical Report CMU-CA-89-166, Carnegie Mellon University, Pittsburgh, PA, 1989.

[JDM00] A.K. Jain, R.P.W. Duin, and J.C. Mao. Statistical pattern recognition: A review. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(1):4–37, 2000.

[JJNH91] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[Jor95] M. I. Jordan. Why the logistic function? A tutorial discussion on probabilities and neural networks. Technical Report 9503, Massachusetts Institute of Technology, Computational Cognitive Science, August 1995.

[JZ97] A. Jain and D. Zongker. Feature selection: Evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153–158, Feb. 1997.

[Kit78] J. Kittler. Feature set search algorithms. In C.H.Chen, editor, *Pattern Recognition and Signal Processing*, pages 41–60. Sijthoff and Noordhoff, Alphen aan den Rijn, The Netherlands, 1978.

[KS90] M. Kirby and L. Sirovich. Applications of the Karhunen-Loeve procedure for the characterization of human faces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(1):103–108, Jan 1990.

[LBP95a] T.K. Leung, M.C. Burl, and P. Perona. Finding faces in cluttered scenes. In *Intl. Conf. on Computer Vision*, Cambridge, MA, 1995.

[LBP95b] T.K. Leung, M.C. Burl, and P. Perona. Finding faces in cluttered scenes using random labeled graph matching. *Proc. 5th Int. Conf. Computer Vision*, pages 637–644, June 1995.

[LM99] T.K. Leung and J. Malik. Reconizing surfaces using three-dimensional textons. In *Proc. 7th Int. Conf. Computer Vision*, pages 1010–1017, 1999.

[LTCA95] A. Lanitis, C.J. Taylor, T.F. Cootes, and T. Ahmed. Automatic interpretation of human faces and hand gestures using flexible models. In *International Workshop on Automatic Face- and Gesture-Recognition*, pages 90–103, 1995.

[LVB+93] M. Lades, J.C. Vorbruggen, J. Buhmann, J. Lange, C. v.d. Malsburg, R.P. Wurtz, and W. Konen. Distortion invariant object recognition in the dynamic link architecture. *IEEE Trans. Comput.*, 42(3):300–311, Mar 1993.

[Mac92] D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.

[MN95] Hiroshi Murase and Shree Nayar. Visual learning and recognition of 3-d objects from appearance. *Int J. of Comp. Vis.*, 14:5–24, 1995.

[NNM96] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (coil-100). Technical Report CUCS-006-96, Columbia University, February 1996.

[OGF97] E. Osuna, F. Girosi, and R. Freund. Training support vector machines: an application to face detection. In *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, pages 130–136, Puerto Rico, June 1997.

[PL94] Arthur R. Pope and David G. Lowe. Modeling positional uncertainty in object recognition. Technical report, Department of Computer Science, University of British Columbia, 1994. Technical Report # 94-32.

[PL95] Arthur R. Pope and David G. Lowe. Learning feature uncertainty models for object recognition. In *IEEE International Symposium on Computer Vision*, 1995.

[PRV98] M. Pontil, S. Rogai, and A. Verri. Recognizing 3-d objects with linear support vector machines. In *Proc. 5th Europ. Conf. Comput. Vision, H. Burkhardt and B. Neumann (Eds.), LNCS-Series Vol. 1406–1407, Springer-Verlag*, pages 469–483, 1998.

[RBK95] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Human face detection in visual scenes. source unknown, July 1995.

[RBK98] H.A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(1):23–38, Jan 1998.

[RH95] I. Rigoutsos and R. Hummel. A Bayesian approach to model matching with geometric hashing. *Comp. Vis. and Img. Understanding*, 62:11–26, Jul. 1995.

[Rip96] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[SK98] H. Schneiderman and T. Kanade. Probabilistic modeling of local appearance and spatial relationships for object recognition. In *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, pages 45–51, Santa Barbara, CA., 1998.

[SK00] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, 2000. to appear.

[SP98] K.-K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(1):39–51, Jan 1998.

[TP91] M. Turk and A. Pentland. Eigenfaces for recognition. *J. of Cognitive Neurosci.*, 3(1), 1991.

[Vap95] V.N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, 1st edition, 1995.

[WCT98] K. N. Walker, T. F. Cootes, and C. J. Taylor. Locating salient facial features. In *Int. Conf. on Automatic Face and Gesture Recognition*, Nara, Japan, 1998.

[WEWP00] M. Weber, W. Einhäuser, M. Welling, and P. Perona. Viewpoint-invariant learning and detection of human heads. In *Proc. 4th IEEE Int. Conf. Autom. Face and Gesture Recog., FG2000*, March 2000.

[WFKv97] L. Wiskott, J.M. Fellous, N. Kruger, and C. vonderMalsburg. Face recognition by elastic bunch graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(7):775–779, 1997.

[Wol90] H.J. Wolfson. Model-based object recognition by geometric hashing. In *Proc. 1^{st} Europ. Conf. Comput. Vision, LNCS-Series Vol. 427, Springer-Verlag*, pages 526–536, 1990.

[WvdM93] L. Wiskott and C. von der Malsburg. A neural system for the recognition of partially occluded objects in cluttered scenes. *Int. J. of Pattern Recognition and Artificial Intelligence*, 7(4):935–948, 1993.

[WvdM96] L. Wiskott and C. von der Malsburg. Recognizing faces by dynamic link matching. *Neuroimage*, 4(3):S14–S18, 1996.

[WWP99] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for visual object class recognition. In University of California Institute for Neural Computation, editor, *Proceedings of the 6th Joint Symposium on Neural Computation*, volume 9, pages 153–160, San Diego, May 1999.

[WWP00a] M. Weber, M. Welling, and P. Perona. Towards automatic discovery of object categories. In *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, June 2000.

[WWP00b] M. Weber, M. Welling, and P. Perona. Unsupervised learning of models for recognition. In *Proc. 6th Europ. Conf. Comp. Vis., ECCV2000*, June 2000.

[Yui91] A.L. Yuille. Deformable templates for face recognition. *J. of Cognitive Neurosci.*, 3(1):59–70, 1991.