Combining deformable models and neural networks for handprinted digit recognition

Christopher K. I. Williams[†]
Department of Computer Science
University of Toronto

Technical Report CRG-TR-94-2
November 1994

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy

Requests for copies of this technical report should be addressed to:

The CRG Technical Report Secretary
Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4
CANADA
email: maureen@cs.utoronto.ca

† From September 1994 Chris Williams' mailing address is:

Department of Computer Science and Applied Mathematics
Aston University
Birmingham, B4 7ET
England
email: c.k.i.williams@aston.ac.uk

Abstract

Combining deformable models and neural networks for handprinted digit recognition

Christopher K. I. Williams

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto
1994

In this thesis I develop a method for recognizing isolated handprinted digits using trainable deformable models. Each digit is modelled by a cubic B-spline whose basic shape is defined by the "home" positions of the control points. A Gaussian distribution over displacements of the control points away from their home locations defines a probability distribution over shapes. The quality of the match of a spline model to an image is calculated as the likelihood of the data under a mixture of Gaussian "ink generators" placed along the length of the spline. Each spline model is adjusted to minimize an energy function that includes both the deformation energy of the model and the likelihood of the data, using a elastic matching procedure which is a generalization of the Expectation Maximization (EM) algorithm. I show that the matching procedure can be significantly speeded up by using a neural net to provide better starting points for the search.

The use of deformable models has a number of advantages. (1) After identifying the model most likely to have generated the data, the system not only produces a classification of the digit but also a rich description of the instantiation parameters. I have shown that these can be used to detect writing style consistency within a string of digits. (2) During the process of explaining the image, generative models can perform recognition-driven segmentation. (3) Unlike many other recognition schemes the method does not rely on some form of pre-normalization of input images, but can handle arbitrary scalings, translations and a limited degree of image rotation. The main disadvantage of the method is it requires much more computation than more standard optical character recognition techniques.

Acknowledgements

First of all I would like to thank my supervisor, Geoff Hinton, for his guidance and encouragement over the past six years. It has been a great privilege to have learned how to do research with someone as creative and generous as Geoff.

I also thank the other members of my committee, Allan Jepson, Michael LeBlanc, Yann LeCun, Rudi Mathon, Ken Sevcik and Demetri Terzopoulos for their help and constructive criticism, which has greatly improved this thesis. Peter Dayan, Tony Plate, Michael Revow and Rich Zemel read earlier versions of the thesis and made invaluable suggestions. In particular, Peter read the drafts amazingly quickly, yet his comments went right to the heart of the matter. I have also been fortunate to work closely with Michael Revow, and would like to thank him for being a hard-working and easy-going collaborator.

I thank Geoffrey Hinton and Michael Revow for permission to include figures and text fragments from our co-authored papers Hinton, Williams and Revow (1992a); Hinton, Williams and Revow (1992b); Williams, Revow and Hinton (1993) and Revow, Williams and Hinton (1993).

I have benefitted greatly from being a member of the Connectionist Research Group at the University of Toronto, and would like to thank the past and present members of the group including Sue Becker, Peter Dayan, Sid Fels, Conrad Galland, Radford Neal, Steve Nowlan, Sageev Oore, Tony Plate, Michael Revow, Evan Steeg, Drew van Camp and Rich Zemel for innumerable valuable discussions, and plenty of distractions as well. Thanks also to many others in the department and beyond the ivory tower for making Toronto a great place to be (even though I wish they could have conjured up a few decent hills nearby).

Several visitors, including David MacKay, Richard Durbin and Mike Mozer made valuable comments on parts of the work while it was in progress. I also thank James Stewart and Yin Zhan for providing code for the convex hull problem.

I gratefully acknowledge financial assistance from the Department of Computer Science, and support from the Ontario Information Technology Research Centre and NSERC via grants to Prof. G. Hinton.

And finally, I thank Barbara for her love and for the sacrifices she has made to create the time for me to write this thesis, and Kate, for just coming along.

Preface

This thesis describes a system which uses deformable models to classify handwritten digits, as found in ZIP codes. It achieves recognition rates comparable to the state of the art. Obtaining this level of performance has required working with large databases and the investigation of several variations on the basic system, and has required more person-years of research activity than mine alone. Prof. Geoffrey Hinton has been an enthusiastic supervisor of the work, and I have also collaborated closely with Dr. Michael Revow on parts of the work described in this thesis. I am responsible for the research on half of the contributions itemized in section 6.2.1, and have been involved collaboratively on the other half.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Object recognition is one of the most important problems in biological and computer vision. Although on occasions it would be better to evade an object coming towards you at high speed, rather than pondering if it is a bird, a plane, or Superman, most of the time recognizing and locating objects is a vital ability for organisms. For instance, it is needed in order to perform tasks like finding food, detecting a mate or fleeing from predators. At a more mundane level, you need to be able to recognize a certain class of objects to read this thesis.

The recognition of even a single object against a neutral background is a difficult problem, because transformations of the object will give rise to entirely different images. For example, a two-dimensional pattern such as a machine-printed character can be translated, scaled or rotated. The recognition of three-dimensional objects from a single image is more difficult because there are extra degrees of freedom in the imaging transformation, and because there is a loss of information when forming a 2-d image of a 3-d object.

Although the recognition of a single object is difficult, the kind of performance that we would like from a computer vision system goes far beyond this, to the recognition of scenes containing many objects. This is particularly challenging because the objects may be partially occluded so that none of them are totally visible. The presence of multiple objects greatly increases the difficulty of the task, as the recognition process must now also segment the image, i.e. decide which features belong to which objects. Current vision systems tend to be successful only in a highly constrained environment; for instance, recognizing a machine part in an industrial context. In this example the lighting conditions can be carefully controlled, and knowledge that there are only a small number of different parts can be exploited.

In this thesis I have focussed on recognizing handwritten digits, as found in ZIP codes. Although handwritten digit recognition is an easier task than general three-dimensional object recognition, it retains, albeit in reduced form, many of the problems associated with general computer vision such as variability in shape and pose, overlapping objects and both structured and unstructured noise. Handwritten digit recognition is a task of great practical importance for which there are standard databases that allow different approaches to be compared.

There are two main approaches to object recognition—*model-based vision* and *statistical pattern recognition*. In statistical pattern recognition, a set of features is extracted from the input so that the image is represented as a point in feature space, and classification boundaries are then constructed in this space. For example, given a binary image of a digit we might extract features like the number of holes in the image, the number of endpoints (an endpoint is the tip of a stroke), crossing points of two strokes etc. The choice of which features to use for a statistical pattern recognizer is critical to its success. Given a vector of these features, there is a wide variety of methods for building the classifiers, ranging from linear and quadratic parametric classifiers (which assume that each class can be modelled by a Gaussian distribution in feature space), through to memory-based non-parametric classifiers (e.g. k-nearest neighbour, kernel density estimation), decision trees and feedforward neural networks.

Statistical pattern recognition techniques usually aim to output only the label of the single object supplied as input. As the pattern recognizer operates in feature space, it does not construct a geometric model of the object and is therefore unable to output the instantiation parameters. It is difficult for statistical pattern recognizers to deal with multiple objects in the input. For example, the existence of two holes in the image should be a good clue for the presence of an "8". However, if the image is poorly segmented and actually contains two zeros, the same feature will be active. Despite these limitations statistical pattern recognition systems can be useful in a well constrained environment.

In model-based vision, an image is explained in terms of instantiated models of the objects which comprise it. This process requires the identification of the appropriate models, and instantiating them in the correct way so as to explain the data. For rigid objects the relevant instantiation parameters are the degrees of freedom in the imaging transformation, which maps the model into the image. In addition, there may be other parameters concerned with the rendering of the scene, such as colour, light sources etc. Instantiation parameters may also include binary variables that indicate the presence or absence of part of an object, such as a cap on a bottle. In order for a model to explain the data in an image, the features of the model should match corresponding features in the image. To achieve a good overall match we need to search for the instantiation parameters of the model that bring it into good correspondence with the image. Knowing the instantiation parameters can be important for further tasks; for example, in order to use a screwdriver to repair my car I have to first find it in my toolbox, but also understand how it is oriented so that I can pick it up and use it.

So far I have discussed rigid object models. However, some flexibility can be incorporated into an object-model; for example, hand-printed characters can take on a great variety of shapes, especially when they are produced by a diverse population of writers. This variability makes the use of rigid models impractical for hand-printed character recognition, and suggests that deformable models should be used instead. The flexibility of a deformable model means that it can represent a whole class of objects rather than a particular instance. The kind of flexibility required by the model is illustrated by the variability in handwritten 2's. Some examples have a large loop at the bottom left-hand corner, while others have a cusp (see figure 3.1). The deformable model should be able to capture this by being easily deformed in this mode of variation.

One reason why statistical pattern recognition is widely used is that the process of matching a model to an image is often a slow and difficult task, so that a system which simply extracts some features from the image will be much faster. Another problem with model-based vision is that a model is needed; for a particular manufactured part a CAD model may already be available, but if this is not the case model acquisition can be a difficult and time consuming process. It tends to be easier to build a statistical classifier using automatic methods.

We can formalize the model-based approach to vision in terms of a prior probability distribution $P(\mathbf{z})$ over the instantiation parameters $\mathbf{z}$, and an imaging model which defines the likelihood of the image $P(I|\mathbf{z})$ given an instantiated model. This sets up object recognition as a problem in signal detection theory, which is a well-founded probabilistic framework. By combining the prior and imaging models it is possible to generate images from the model. The procedure is to first choose $\mathbf{z}$ from $P(\mathbf{z})$, and then to generate an image I using $P(I|\mathbf{z})$. To analyze novel images we run the procedure backwards; this is known as "analysis by synthesis". We would expect to find a rather broad or non-informative prior for some of the instantiation parameters, such as translations or the scaling factor. The prior distribution on the deformation parameters will probably be tighter, otherwise the models would be too flexible.

A more geometrical interpretation is as follows; given an $m \times n$ pixellated image, we can think of a model with k degrees of freedom as sweeping out a k-dimensional (non-linear) manifold of ideal images in the mn-dimensional image space. The imaging model defines how expensive it is to move off the manifold, and the prior distribution defines how expensive it is to be at a particular point on the manifold. For a given test image we can search for the point that has the minimum combined cost. Roughly speaking, the best hypothesis (e.g. of digit identity) will have the lowest combined cost[1].

The probabilistic framework has been used by a number of authors. For example, Grenander *et al* (1991) have shown how to match a model of a hand to noisy images, and Mjolsness (1992) has demonstrated the matching of line-segment images derived from stereo pairs. The same framework has also been used by Durbin *et al* (1989) in their analysis of Durbin and Willshaw's (1987) elastic-net algorithm for the Travelling Salesman Problem. These authors have basically focussed on finding the best instantiation parameters given a particular model. In this thesis I investigate the use of deformable models for handwritten character recognition, which involves finding the best fit for each model and then performing *model comparison*. As there are only ten different digit classes it is feasible to explore all ten different ways of generating a test image.

In the rest of the thesis I explore the issues around

- the construction and refining of deformable models and imaging models for character recognition
- their performance on the digit recognition task
- efficient methods for matching the models to data (including the use of neural networks)
- exploiting the extra information available from the model-based approach (e.g. instantiation parameters)

[1] More details are given in section 2.1.

1.1 Thesis outline

Chapter 2 outlines the framework for object recognition using deformable models. The two key elements required are a probability distribution over model parameters, and a generative model that produces images given an instantiated model. The model that best explains the data can be found by evaluating the posterior probability for each model. I discuss the relationship of the signal detection theory framework to the Minimum Description Length principle and the Hough transform. Previous work on deformable models, imaging models and parameter-space search techniques is reviewed. I also discuss some previous approaches to the optical character recognition problem.

Chapter 3 gives the details of the system that has been built for the digit recognition task. The deformable digit models are constructed from cubic B-splines whose control points have a Gaussian distribution in an object-based frame. An affine transformation projects each model into the image-based frame. The imaging model is based on placing a mixture of Gaussians along the spline. To fit the models to data we use a generalization of the usual EM method for fitting a mixture of Gaussians which can deal with the affine transformation from object-based to image-based frame and constraints on the Gaussians due to the spline. I discuss how to improve the deformable models using training data, and how to improve the performance of the system using a simple post-processing neural network.

In **Chapter 4** I describe extensions to the basic system that (i) characterize the distribution of model deformations better by using either a full covariance matrix Gaussian or a mixture of Gaussians, (ii) improve the imaging model by explaining the white pixels as well as the black ones and (iii) show that there is style consistency within ZIP codes by computing the mutual information between pairs of digits produced by the same writer.

Chapter 5 describes how neural networks can be used to improve the search process. In the first set of experiments I show that neural networks can be successfully trained to predict the instantiation parameters of models given a normalized image as input, and so speed up the search. Another set of experiments deals with the prediction of correspondences between data points and points on the model using neural networks taking a local window of context as input. I discuss how these predictions can be used either in the search process or to instantiate a model.

Chapter 6 provides a discussion of related work, summarizes the major contributions of the thesis and describes directions for future research.

Chapter 2

Object Recognition

In section 2.1 I discuss the probabilistic framework for deformable model matching and model comparison. Links are made with the Minimum Description Length principle and Hough transforms. The relationship to template-based methods is discussed in section 2.1.4. Section 2.2 reviews previous work in the area concerning deformable models, imaging models and algorithms for matching models to data. Section 2.3 discusses previous work on optical character recognition.

2.1 Deformable models

The existence of continuous variability within a class of objects leads naturally to the notion of elastic matching between a template and a test image. Work based on this idea dates back to the early 1970's. For example, Ullmann (1972) discusses the idea of finding a distortion mapping from a test image to a stored template such that there is *correspondence* between like features. Widrow (1973) suggests the idea of using rubber templates to achieve flexible matches for a variety of natural objects and waveforms. Fischler and Elschlager (1973) described the matching of an elastic net of parts to images; the deformations allow the spatial relationships of the parts to be somewhat flexible.

These ideas can be formalized by regarding the matching process as a competition between a datafit term that encourages the deformation of the template in order to bring it into better correspondence with the test image, and a penalty term which opposes the deformation of the template. The aim of the matching process is then to find the deformation which minimizes the total "energy" E_{tot}

$$E_{tot} = E_{fit} + E_{def} \tag{2.1}$$

where E_{def} measures the template deformation and E_{fit} measures the misfit between the template and the test image. A scalar can be used to adjust the relative importance of these two terms. For example, weighting the deformation energy more strongly encourages less deformed solutions. Letting $\mathbf{z}$ denote the vector of displacements that describes the deformation of the template, we find that at a minimum of E_{tot}

$$\frac{\partial E_{tot}}{\partial \mathbf{z}} = 0 \qquad \text{which implies that} \qquad \frac{\partial E_{def}}{\partial \mathbf{z}} = -\frac{\partial E_{fit}}{\partial \mathbf{z}} \tag{2.2}$$

i.e. the forces due to the deformation and datafit terms are equal and opposite. This methodology has been used extensively, for example in Bajcsy *et al* (1983; concerning the matching of radiographic images), Witkin *et al* (1986; matching stereo and motion pairs), Bookstein (1978; searching for the smoothest diffeomorphism that maps "landmarks" in one frame into another) and Lades *et al* (1993; matching face images).

An important reinterpretation of equation 2.1 can be obtained by adopting a probabilistic viewpoint. Given an "energy function" $E(\mathbf{z})$ for a system, we can define a probability distribution using the Gibbs distribution from statistical physics $P(\mathbf{z}) \propto \exp(-E(\mathbf{z})/T)$, where T is the temperature of the system (see, for example, Waldram, 1985). Setting $T = 1$, equation 2.1 becomes $P_{tot} \propto P_{def} P_{fit}$. This can be understood in terms of Bayes' theorem

$$P(\mathbf{z}|I) = \frac{P(\mathbf{z})P(I|\mathbf{z})}{P(I)} \qquad \text{or} \qquad P(\mathbf{z}|I) \propto P(\mathbf{z})P(I|\mathbf{z}) \tag{2.3}$$

$P(I|\mathbf{z})$ is defined by the imaging model; if the test image I is only improbably generated from the deformed template then E_{fit} is large. $P(\mathbf{z})$ is a prior distribution on deformations, which might specify that small deformations are preferred. $P(\mathbf{z}|I)$ describes the posterior distribution for $\mathbf{z}$ given the image. We can now see that the global minimum of E_{tot} will correspond to the maximum a posteriori (MAP) estimator for $\mathbf{z}$, which is the highest peak (or mode) in the posterior distribution[1]. However, the probabilistic framework also makes it clear that there are more options than just finding a mode, such as sampling from the posterior distribution or using other estimators, e.g. the posterior mean.

A similar probabilistic viewpoint can also be taken on the "regularization" of many inverse problems that occur in computer vision (Poggio, Torre and Koch, 1985). For example, given sparse data about the height of a surface above a reference plane, we would like to approximate the surface, and hence predict the height of the surface at points where there are no measurements. Without any further assumptions, the problem is ill-posed; there are an infinite number of feasible surfaces. However, by using a prior model on the smoothness of the surface, it is possible to show that there are preferred solutions which are the local maxima of the posterior probability density. If the data-fit model and the prior are Gaussian, there will be a unique maximum, which is the MAP estimator (Terzopoulos, 1986). Szeliski (1989) extended this analysis by including the "error bars" of the reconstruction due to the sharpness of the posterior peak.

The simple Gaussian priors tend to smooth over real discontinuities, due, for example, to the edges of physical objects. Hence it was necessary to investigate more elaborate priors which allowed ridges or tears in the surface (Terzopoulos, 1986; Blake and Zisserman, 1987). Bottom-up region segmentation algorithms use a similar approach, by regarding the greyscale image intensity as analogous to surface height. The aim of segmentation algorithms is to partition the surface into regions that have internal coherence, and to create boundaries between different regions. An early example of this approach is the work of Geman and Geman (1984) , which involved the use of line processes to create the boundaries. They concentrated on finding the MAP solution by using simulated annealing. Leclerc (1989) also studied the segmentation problem; his

[1] MAP estimation is also known as penalized maximum likelihood estimation.

algorithm fits a piecewise polynomial approximation to the data, using a continuation method to search for the region boundaries.

The definition of prior probabilities in terms of the Gibbs distribution is useful, but it can be difficult to sample from the distribution or to obtain correctly normalized probabilities. It can be convenient to define an explicit prior probability distribution like a multivariate Gaussian on $\mathbf{z}$, rather than obtaining it implicitly through an energy function. Examples of explicit prior models are found in the work of Grenander and his co-workers (Knoerr, 1988; Grenander *et al*, 1991) and Cootes and Taylor (1992). These examples are discussed in more detail in section 2.2.

2.1.1 A probabilistic view of model-based object recognition

So far I have discussed the matching of one particular model to a test image. However, we also want to be able to use this framework to perform *model comparison*, i.e. to rank a number of hypotheses about what object(s) are present in the image. For example, in an isolated character recognition task we want to find which character model gives the best explanation of the data. This can be done by integrating the posterior distribution for each model over $\mathbf{z}$.

Let I be an image, M be a model and $\mathbf{z}$ be its instantiation parameters[2]. Then the likelihood of the image given model M is

$$P(I|M) = \int P(\mathbf{z}|M)P(I|M,\mathbf{z})d\mathbf{z} \tag{2.4}$$

To go from $P(I|M)$ to obtain $P(M|I)$, the probability of a model given an image, we use Bayes' theorem to calculate

$$P(M|I) = \frac{P(I|M)P(M)}{P(I)} \tag{2.5}$$

$P(M)$ expresses a prior belief in which models are most likely; it will often be uniform over a set of alternatives. For example, in digit recognition the prior is usually taken to be uniform over all ten digits. Assuming that the data was actually generated by one of the models, then $P(I) = \sum_j P(I|M_j)P(M_j)$, and $P(I)$ simply serves to normalize the $P(M_j|I)$'s so that they sum to one. Low values of $P(I|M_j)$ for all existing models may lead to the creation of a new model, for example in the case that a system trained on the digits encounters other symbols in its input.

Equation 2.4 is formally correct, but the evaluation of this integral is very computationally intensive if $\mathbf{z}$ has more than a few dimensions. Simple grid-based methods will typically miss the peaks of the integrand, unless the sampling is very dense. Hough-transform methods (see section 2.1.3 below) can be viewed as grid-based approaches; they are tractable in low dimensional search spaces. Monte Carlo methods for sampling the posterior distribution can also be used (for a review see Neal, 1993b). However it is often possible to approximate the integral, using the assumption that the integrand is strongly peaked around a global maximum (MAP) value $\mathbf{z}^*$. In this case, $P(I|M)$ can

[2] M may actually be made up of a number of individual models, in which case $\mathbf{z}$ is the concatenation of their individual instantiation parameters.

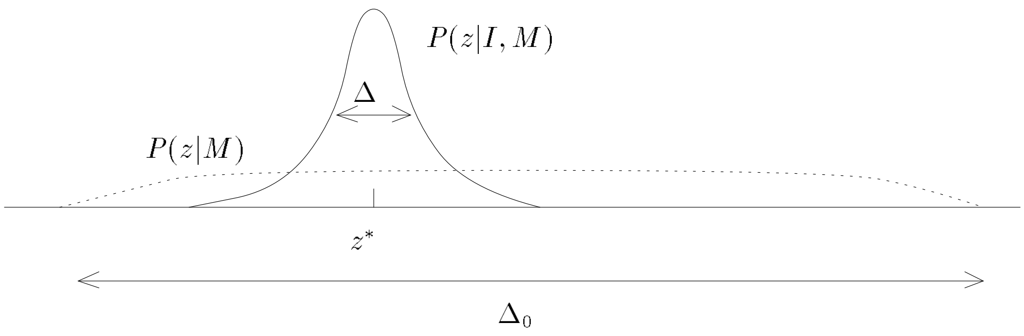


Figure 2.1: The Occam factor for a single parameter z. For a model M, if we assume that the prior is roughly uniform over some large interval Δ_0, then $P(z^*|M) \simeq 1/\Delta_0$. Under Laplace's approximation the posterior distribution has a single peak at z^* of width Δ. The Occam factor is $P(z^*|M)\Delta = \Delta/\Delta_0$. See MacKay (1992).

be approximated by the highest peak of the integrand times a volume factor $\Delta(\mathbf{z}^*|I,M)$, which measures the sharpness of the peak.

$$P(I|M) \simeq P(I|\mathbf{z}^*,M)P(\mathbf{z}^*|M)\Delta(\mathbf{z}^*|I,M) \tag{2.6}$$

By Taylor expanding $\log P(\mathbf{z},I|M)$ around $\mathbf{z}=\mathbf{z}^*$ to second order and using the fact that $\mathbf{z}^*$ is a local maximum, it can be shown that $\Delta(\mathbf{z}^*|I,M) = (2\pi)^{k/2}\det^{-1/2} H$, where $H = -\partial^2 \log P(\mathbf{z},I|M)/\partial z_i \partial z_j$ and k is the dimension of $\mathbf{z}$. This is known as Laplace's approximation[3]. In the statistical literature H is known as the (observed) Fisher information matrix. The integrand will probably have many local maxima; for the character recognition task we aim to find the global maximum by using techniques like continuation methods (see chapter 3).

Another way of looking at equation 2.6 is to group it into the $P(I|\mathbf{z}^*,M)$ term, which measures the fit of the model to the data at $\mathbf{z}=\mathbf{z}^*$, and the $P(\mathbf{z}^*|M)\Delta(\mathbf{z}|I,M)$ term. This second term is known as an Occam factor (Gull, 1988), and measures the ratio of the posterior to the prior accessible volume of M's parameter space (see figure 2.1 for details).

As an example of these ideas for model comparison, suppose we have an image comprised of black pixels in a "B" shape on a white background, and two alternative hypotheses to explain the data—one "B" shaped model or one "1" and one "3". If both hypotheses are aligned correctly, then the datafit is the same. The only difference in the costs comes from the specification of the translation/rotation parameters. If we assume that the width of the posterior peak for these parameters (see figure 2.1) is about the same for the "B", the "1" and the "3", then the "B" interpretation will be preferred as its Occam factor is larger[4].

2.1.2 Relationship to Minimum Description Length theory

The model comparison framework can also be viewed in coding terms; the models that best explain the data are also those in terms of which the data is cheapest to encode. The key idea is Shannon's optimal coding theorem (Shannon, 1948), which states that

[3]The use of Laplace's approximation for model comparison has been popularized in the neural networks field by MacKay (1992).

[4]If the Occam factor for the "B" is $\Delta/\Delta_0 < 1$, then the Occam factor for the combined "1" and "3" model is approximately $(\Delta/\Delta_0)^2$, which is smaller.

if there are a set of discrete alternatives x_i having corresponding probabilities P_i, then for an optimal coding scheme the expected code length for each x_i under a D-ary alphabet is given by $L(x_i) = -\log_D P_i$.[5]

The Minimum Description Length (MDL) principle (Rissanen, 1985) states that one should prefer models which can communicate the data in the shortest number of bits. Taking logs of equation 2.6 gives

$$-\log P(I|M) = -\log P(I|\mathbf{z}^*, M) - \log(P(\mathbf{z}^*|M)\Delta(\mathbf{z}^*|I, M)) \tag{2.7}$$

The first term in equation 2.7 is identified as the cost of encoding the image relative to the predictions of the model and the second term is the cost of encoding the instantiation parameters relative to the prior distribution. So the objective of maximizing $P(I|M)$ can be seen as finding the model which gives the shortest encoding of the data. If the natural logarithms are used, the description length is expressed in nats; if the logarithms are to base 2, the units are bits.

When dealing with the coding of real valued variables, we have to be careful about the precision to which values are sent. Usually we can handle this by dividing up each dimension into intervals of width δ, and assume that δ is sufficiently small so that we can make a rectangular approximation. Then the code length for a point in the neighbourhood of x_i will be $-\log(P(x_i)\delta) = -\log P(x_i) - \log \delta$, where $P(x_i)$ is the probability density at point x_i. However, the $\Delta(\mathbf{z}^*|I, M))$ term in equation 2.7 indicates that the precision to which $\mathbf{z}^*$ must be sent is not arbitrary, but is determined by the Occam factor. To understand this, consider the case where there are two models, and assume that the Occam factor for the first model is larger than that for the second. This means that there is a larger range of instantiation parameters for which the first model is a good predictor of the data as compared to the second, and thus the instantiation parameters of the first model need to be coded less precisely. This argument can be made precise using the idea of stochastic complexity (Rissanen, 1989), whereby we stochastically pick $\mathbf{z}$ from the posterior distribution $P(\mathbf{z}|I, M)$.

The MDL principle has been applied to a number of computer vision problems. Pentland (1989) studied the decomposition of binary images into parts. He defined a repertoire of blob-like parts in a grid of locations, and an objective function that included terms for the data misfit and a cost term for each instantiated part. By minimizing this objective function using a continuation method, he obtained segmentations of the images into parts.

Fua and Hanson (1989) discussed the ranking of hypotheses using the MDL principle, and applied it to the detection of buildings in aerial photographs. Their imaging model dealt with the encoding of edge and intensity information in the image.

Freeman (1994) has used this framework to compare possible surfaces in the shape from shading problem. By incorporating the volume factor, interpretations which require very specific parameter tuning (e.g light direction) are down-weighted compared to more generic views.

Zemel (1993) has used the MDL framework as a basis for unsupervised learning. By viewing unsupervised learning procedures as consisting of an encoding of the data under a particular scheme, plus a reconstruction term, he was able to generalize methods

[5] To actually attain this bound in practice it may be necessary to use large block lengths.

like clustering or principal components analysis to obtain factorial encodings from an ensemble of images.

2.1.3 Relationship to the Hough transform

In the Hough transform, the parameter space for each model is divided up into cells ("binned"), and then for each image feature a vote is added to each parameter space bin that could have produced that feature[6]. After collecting votes from all image features we then search for peaks in the parameter space accumulator array.

The Hough transform was originally designed for the detection of straight lines in images (Hough, 1962), and has been extended to cover a number of geometric shapes, notably conic sections such as circles, ellipses and parabolae. Ballard (1981) further extended the approach to arbitrary shapes with the Generalized Hough Transform (GHT)[7]. For shapes (such as conic sections) that can be expressed in the form $F(x, y; \mathbf{z}) = 0$, simply substituting in the values of x and y for a particular point leads to a constraint equation on the parameters $\mathbf{z}$. The bins that satisfy this equation are then incremented. For shapes not defined in this way, Ballard's GHT algorithm uses an "R-table", which describes the relationship between a point (and optionally its tangent) on the curve and a reference point of the object for a large number of points along the contour of the object.

The Hough transform can be viewed as a crude way of approximating the logarithm of the posterior distribution (Hunt *et al*, 1988; Yuille *et al*, 1990; Stephens, 1991)

$$\log P(\mathbf{z}, I|M) = \log P(\mathbf{z}|M) + \sum_i \log P(I_i|M, \mathbf{z}) \tag{2.8}$$

where I_i is the ith feature in the data and we have assumed that the features are conditionally independent (i.e. independent *given* M and $\mathbf{z}$). In the Hough transform the effect of the prior $P(\mathbf{z}|M)$ is usually ignored, although it would be easy to incorporate it by adding it on separately from the data terms. A principled Hough transform would compute the evidence from each feature in the data for every bin in Hough space, using a model for $P(I_i|M, \mathbf{z})$. Processing feature i would then increment the Hough bins by $\log P(I_i|M, \mathbf{z})$ integrated over the bin volume.

In contrast to Laplace's approximation described above, the Hough transform is able to capture multimodal structure in the posterior distribution (assuming that the discretization is fine enough). Multiple objects will show up as separate peaks in the parameter space array, as long as they are not too close together. The peak-hunting step in the Hough transform means that it is robust against spurious image data, as random image points are unlikely to contribute coherently to the accumulator counts. The disadvantages of the Hough transform relate primarily to the discretization of parameter space. It can be very expensive to set up the accumulator array, especially for parameter spaces that have more than two or three dimensions (Bellman's "curse of dimensionality").

[6] Each image feature will induce a constraint surface in parameter space, corresponding to the locus of parameter settings that are consistent with the feature. A bin will receive a vote if the constraint surface passes through the bin.

[7] For a review of Hough transform techniques see Illingworth and Kittler (1988).

Another problem with the Hough transform is that the votes from a single feature cover a large submanifold of the parameter space, which means that many bins have to be visited and incremented. To avoid this features can be grouped together in order to reduce the dimensionality of the submanifold in parameter space; for example Xu and Oja (1993) use enough features to obtain a single point in parameter space. If any of the data features are spurious, the prediction will be wrong, but these errors may be less disruptive than the noise created by using simpler features that vote over a large number of bins.

2.1.4 Other approaches to density modelling in image space

The generative model-based approach discussed above defines a probability density $P(I|M)$ in image space. Although the model-based framework is appealing, the search to find the minimum-cost point on the manifold can be difficult. One simple generative alternative is to use a template-based approach, where a probability density in the image space is defined as a Gaussian mixture distribution, with a centre at each template in the database. The templates are usually taken to be examples from a training set. Geometrically, we are approximating the manifold by dotting sample points on it. Essentially, classification now comes down to finding out the class of the exemplar which is closest to the test pattern; it is conceptually easy to find the closest template, although it may require a lot of computation.

An interesting half-way house between these two schemes is to use a number of linear models in the image space to represent each class (Hinton, Revow and Dayan, 1994). This corresponds to approximating the manifold with a number of linear subspaces, or "pancakes". Movement within the pancake is free, and we only pay for the perpendicular distance off the pancake. Each pancake can be viewed as a multivariate Gaussian which has large variances along directions within the subspace and small variances orthogonal to it. Each pancake can then cover a significant part of the image space. The closest point on a pancake to a test image can be found using linear algebra, i.e. the search problem is easy.

Both of these schemes are used with normalized images. Although it would be possible to resample the templates (plus direction vectors for the linear subspace method) "on-the-fly" so that they could be used with arbitrarily sized images, this would be very inefficient compared to normalizing the test image, because there are so many templates. There is also an important difference between the model-based approach and these others concerning images which contain multiple objects. Assuming that we can define an imaging model for multiple objects[8], then it is possible to evaluate $P(I|M)$ when the hypothesis M is composite. The template and pancake methods would be inferior because they superimpose images rather than models.

2.2 Previous Work

In this section I discuss previous work on three key aspects in the use of deformable models for object recognition—creating deformable models, creating an imaging model and iteratively improving the match of a model to an image. Discussion of the work of

[8] The imaging model may need to deal with occlusion.

Lanitis, Taylor and Cootes (1993) on using deformable models for object recognition is deferred to section 6.1, where it is compared with the approach developed in this thesis.

2.2.1 Probability distributions on models

In this section I classify and analyze the various types of deformable model that have been proposed in the literature. Most of the focus is on 2-d models, as this is most relevant to the scope of the thesis. The key idea is that deformable models have a number of internal parameters, and that a probability distribution over the parameters will induce a probability distribution over the shape of the object. Often the distribution of the parameters is chosen to have some simple form (e.g. a Gaussian), but it is also possible to derive distributions for more complex cases by using the Gibbs distribution to relate a deformation energy to a probability.

One disadvantage of using the Gibbs distribution is that although it is easy to obtain the relative probabilities of two configurations, it is harder to obtain their absolute (i.e. correctly normalized) probabilities. The best techniques currently available for sampling from the distribution use Markov chain Monte Carlo methods (e.g. Metropolis *et al*, 1953) which allow us to stochastically visit configurations with a frequency proportional to their probability (once the chain has reached equilibrium).

Articulated models are perhaps the simplest form of parameterized models. Rigid subparts are connected by rotary or sliding joints. Simple examples of this class of objects are a pair of scissors and a hand drill (where the circular disc component is free to rotate relative to the shaft). It is interesting to note that the probability distributions on joint configurations are not necessarily just the products of uniform distributions on each of the joint variables – for example robot arms should avoid self-intersecting positions, and for human hands it is not possible to control all of the fingers independently.

The idea behind the **elastic net of parts** (Fischler and Elschlager, 1973) is that there is a set of rigid parts whose spatial relations are somewhat variable. One of their examples deals with the fitting of a face model to images. The components of the model are hair, eyes, nose, mouth and sides of the face. These components are coupled together by springs which have non-zero rest-lengths; deformations of the parts away from the ideal shape incur a deformation energy penalty. A later example of this idea is the work by Yuille, Cohen and Hallinan (1989) on eye and mouth templates which combines an elastic nets of parts with globally deformable contour models (see below).

Deformable contours are another important class of models, which are particularly relevant to this thesis. They can be subdivided into open and closed contour models. Closed contour models delineate a region of the plane, and so it is possible to think of the shape as being defined by either the region or the bounding contour. For open contour models, only the contour description is available.

We first consider deformable contour models with **local** control. In most cases the contour will be a spline, consisting of control points and a procedure for generating a parametric curve. For instance, linear interpolation between control points is produced by using splines with linear basis functions. The location of a control point may either be specified relative to a fixed coordinate frame , or relative to other control point locations. For example, "snakes" (Kass, Witkin and Terzopoulos, 1987) specify the position of a control point relative to the previous control point in the sequence along the chain. If the position of each control point is chosen independently from the previous ones, the resulting curve will tend to have fine scale (high spatial frequency) corrugations. To remedy this Kass *et al* impose constraints on the curvature of a snake, which induces correlations between the offsets to produce a smoother curve.

Another example of the use of linear spline models is the work of Knoerr (1988) on modelling *Sassafras* leaves, using a chain of 160 points. The form of the probability density for $\mathbf{z}$, the vector of control point positions, is given by:

$$f(\mathbf{z}) = c \prod_{i=0}^{n-1} A_i \left(\frac{z_i - z_{i-1}}{\epsilon_i} \right) \prod_{i=0}^{n-1} B_i(z_i) \tag{2.9}$$

where c is a normalizing constant. This probability distribution is the product of independent distributions on each control point (the B_i's), with acceptor functions (the A_i's). If the A_i's and B_i's are Gaussians the effect of acceptor functions is to make the variation of $\mathbf{z}$ smoother than it would otherwise be. Grenander, Chow and Keenan (1991) also used this kind of distribution with 256 control points to model 2D hand shapes. Similar work has been done by Cootes and Taylor (1992) on Point Distribution Models. They hand-fitted point models to many examples, and then used Principal Component Analysis to determine a number of modes of variation for a Gaussian model. Approaches that *learn* the relevant deformation modes from data should be contrasted with the work of Pentland and Sclaroff (1991), who assume that the low-frequency normal modes of an elastic model of the shape will be useful in modelling observed deformations.

A relatively complex probability distribution involving correlations between control points is needed to avoid a wavy contour when using a simple spline. An alternative way to approach the problem is to use a simpler probability distribution but a more complex spline. For example, Cipolla and Blake (1990) use a cubic B-spline snake for their tracking work; the key idea is that the number of control points is far fewer than the number of points on the curve, which are generated by a linear combination of the control points. The models of handprinted digits used in this thesis are also based on B-spline models (see chapter 3). For example, to model a “2” character only eight control points were used, although the datafit measure is evaluated at sixty points along the contour.

The contour models discussed above have illustrated a trend for one control point to have an effect over more of the curve as the order of the spline is increased. This can be taken further, so that the control parameters have a **global** effect on the object. For example, algebraic curves $F(x, y; \mathbf{z}) = 0$ define a contour which is globally dependent on the parameters $\mathbf{z}$. A probability distribution on $\mathbf{z}$ will induce a distribution over shapes.

Taubin *et al* (1992) and Keren *et al* (1992) have used fourth-degree polynomials as contour models, and have shown how to restrict the parameter space in order to obtain bounded zero sets[9]. Superellipses (Gardiner, 1965) are a generalization of ellipses that have an extra shape parameter ϵ. By varying this parameter the resulting shape changes from a four-pointed star to a shape which is almost a rectangle. Superellipses can

[9] I.e the curves do not extend to infinity. For second-degree polynomials (conic sections), ellipses are bounded while parabolae and hyperbolae are not.

be generalized to superellipsoids in 3D. Terzopoulos and Metaxas (1990) have shown how model a more complex shape by adding local deformations to a globally defined superellipsoid.

As well as using curves which are defined algebraically, it is also possible to use parameteric approaches, such as "ribbons"[10] (Brooks, 1981) and templates with global shape parameters, e.g. Widrow's (1973) template for an "X" shaped chromosome which uses global control parameters for the length, width, curve and angle of each arm.

The probabilistic models discussed in this section have had a non-hierarchical structure. Hierarchical model decompositions (Marr and Nishihara, 1978) are clearly a useful way of structuring object knowledge. It is possible to consider building up more complicated deformable models (like those that might be required for Chinese character recognition) using hierarchically structured distributions based on part decompositions. Using a small number of primitives in different combinations is an efficient way to represent a large number of models.

It should be emphasized that the deformable models discussed above are not intended to allow dynamic simulation, only to represent probabilities across the class. Terzopoulos and Fleischer (1988) have discussed elastic (and inelastic) models that allow them to predict behaviour over time.

2.2.2 Imaging models

There are two kinds of datafit measures that have been used in the literature, which I will term *explanation* and *support* measures. The differences can be demonstrated by considering the example of computing the match between a contour model and a binary image, as might arise for a character recognition problem. One way to evaluate the match of the model and the data is to find the number of pixels where they are in agreement as a fraction of the total number. Whether this is the total number of data pixels or just the total number of model pixels is important. In the first case it is necessary to get agreement between the model and the data pixels at all of the black and white pixels in the image—i.e. the aim is to *explain* the data. In the second case it is only necessary to get agreement between the model and data pixels at all of the model locations—we are evaluating the *support* of the data for the model.

These two kinds of data-fit are very different. An explanation allows us to calculate the likelihood of the data given the model, and requires us to account for all of the data. A serious problem with the support measure is that there may be good support for a hypothesis that is a subset of the true cause; for example, a handwritten "9" will frequently contain a vertical stroke that would be well matched by a "1" digit. However, the "1" model would then fail to explain the curved stroke left at the top left of the image. The explanation/support distinction is reminiscent of Yuille's (1991) classification of "altruistic" and "selfish" templates. "Altruistic" templates try to cooperatively organize themselves to explain the whole image, whereas "selfish" templates try to grab the parts of the image they can explain and ignore the rest.

Some examples of generative image models are given below. For a dense array of data such as a pixellated image, the simplest assumption is that the value of each pixel is independently perturbed away from the predicted value according to some

[10] Ribbons are the 2D version of generalized cones (Binford, 1971).

known distribution. For example, in Geman and Geman (1984) and Grenander *et al* (1991), the perturbations are assumed to be Gaussian (modulo quantization). Note that this does not imply that the values of the pixels are independent, only that the perturbations are; the pixel values are conditionally independent. If there are p pixels in the image, the imaging model is a p-dimensional spherical Gaussian centered on the predicted mean. More generally, a probabilistic imaging model can be created using any complete decomposition of the image which has p degrees of freedom.

Mjolsness (1990) gives another example of a generative model, in this case for generating sparse data. He considers images made up of a set of points that were created by a "dot grammar". Each image dot is produced with Gaussian noise from a model dot. If we knew the correspondences between the image dots and the model generators it would be easy to calculate the likelihood of the data. When the correspondences are unknown, the likelihood of the data is obtained by summing over possible correspondences, but this is awkward analytically. By relaxing the constraints in this summation we can regard the data as having been generated from a mixture of Gaussians distribution, as suggested by Durbin *et al* (1989). This is essentially the imaging model that we use for the character recognition experiments, as described in section 3.2.

For computer vision problems it is often of interest to match at the level of line segments rather than pixels. In this case it is necessary to come up with a metric for comparing two different line segments, that will, in general, differ in length, orientation and position. Mjolsness (1990) essentially considers the match of every point on a line segment with every point on the other segment. Lowe (1985) compares line segments by measuring the perpendicular distance of each endpoint of the model line from the corresponding line in the image. He favours this definition because "in practice low-level vision routines are relatively good at finding the transverse location of lines, but are much less certain about exactly where the line terminates." Other examples of metrics for comparing line segments are given by Burr (1981b) and Harris (1991). The mixture model approach which was described above for point features can be easily extended to evaluate the match between a model and data which both consist of line segments, or indeed of any other features for which we have a probabilistic model.

The use of an explanation-type datafit measure sometimes seems unwieldy; consider a cluttered scene where there are a large number of tools lying on top of one another. If the objective is to check whether one particular tool is present in the scene, modelling the entire image seems like a large amount of trouble, although attending to a candidate region would reduce this burden. To perform a complete interpretation requires having (deformable) models of all of the objects in the scene and fitting them to the data. If some of the models are not subsets of others, then the support measure will be a useful way of evaluating if there is sufficient evidence for an instantiated model at a particular location (e.g. Yuille *et al*, 1989). The support measure has also be used with the tracking of a moving target (Kass *et al*, 1987; Metaxas and Terzopoulos, 1991; Cipolla and Blake 1990).

2.2.3 The iterative improvement of hypotheses

For some kinds of deformable model, it is possible to efficiently find the best match of the model to the data. For example, the Viterbi algorithm (see Rabiner *et al*, 1983) finds, in polynomial time, the best sequence of states through a Hidden Markov

Model for matching serial data such as an acoustic signal. However, for image data, no similar dynamic programming algorithms have been found, so search strategies are based on iteratively improving an initial hypothesis for the instantiation parameters by optimizing an objective function, for example increasing the *a posteriori* probability. These techniques are based on the hope that there are quite large "basins of attraction" (i.e. there are relatively few local maxima) in parameter space so that the initial hypothesis does not have to be very accurate.

Burr (1981a, 1981b) did some interesting work on elastically matching a model to an image. The earlier paper deals with deforming a model composed of points onto an image. The idea is that each point on the model looks for the closest point in the image, and the vector between those two points is called a Feature Displacement Vector. A similar calculation is done by all image points looking for nearby model points. Then these vectors are smoothed with a Gaussian kernel, and the model points are moved some fraction along the smoothed displacement vector field. This process is iterated, while shrinking the variance of the Gaussians, until it converges. Notice that any model can be warped onto any image; there is no notion of a deformation energy in the model which resists the data forces. The data-fit term in Durbin and Willshaw's (1987) elastic net provides a another way of obtaining similar behaviour. An important difference is that Durbin and Willshaw obtain the forces acting on model points as the gradient of an energy function.

Burr (1981b) extended these ideas to the matching of figures composed of line segments rather than dots. He first described a distance function between line segments that took into account both orientation and displacement mismatch, and then used it to define Feature Displacement Vectors, which were smoothed in a similar manner to the dot case. In this case there is some residual mismatch between the two line drawings when the deformation process stops and Burr defined a function that quantified the mismatch between the final model position and the image.

Scale-space continuation methods are commonly used with iterative fitting. Kass, Witkin and Terzopoulos (1987) subjected their active contour models ("snakes") to forces derived from smoothing an intensity image with a Gaussian, and allowing the snake to follow the gradient of the resulting image; depending on the choice of sign, the snake is either attracted to light or dark lines. They initially use large Gaussian blurrings of the intensity and edge images, and gradually reduce the blur. A model that starts off far away from the data can be attracted by using a large blurring scale, but once it is in the right neighbourhood the blurring can be reduced to improve localization.

Lowe (1985, 1991) has worked on the idea of iterative improvement in parameter space. Given a set of correspondences between model and data line segments he uses an iterative method to minimize an objective function quantifying the mismatch between model and data line segments. After each iteration the new projection of the model into the image is calculated and the error vector updated. Ayache and Faugeras (1986) use a more sophisticated version of this scheme using a Kalman filter in their HYPER system. As the models used were either rigid or articulated, the model deformation term is irrelevant and the aim is to minimize the data-fit mismatch.

With deformable models we want to include the effects of the data-fit and model probabilities. Grenander *et al* (1991) sample from the posterior distribution of their hand models. Yuille *et al* (1989) do gradient-descent on the objective function in order to fit their deformable templates to facial features. Cootes and Taylor (1992) perform similar optimizations to fit face models to images.

2.3 Previous work on Optical Character Recognition

There is a vast literature concerning optical character recognition (OCR) which spans at least three decades. Useful reviews can be found in the papers by Suen (1980), Mantas (1986) and Suen *et al* (1992). It is also important to note that the OCR problem is distinct from the on-line character recognition problem, in which the data is captured from an electronic tablet, so that the time-ordering of the data points is also known. On-line character recognition is reviewed in Tappert *et al* (1990).

There are a number of classes of OCR problem. The easiest is fixed-font machine printed character recognition; the most difficult is unconstrained cursive script recognition. The data which was used for the experiments in this thesis and most of the examples described below consists of unconnected handwritten digits. In this case it is assumed that the "segmentation problem" has been at least approximately solved, so that each image will contain only one character (plus possibly some noise)[11]. The usual approach to segmentation for handwritten characters is to use connected-components analysis on a binary image, followed by a clean-up stage, e.g. using a vertical cut to separate touching or overlapping characters.

As with the general object-recognition problem, approaches to character recognition can be divided into statistical pattern recognition and model-based techniques. The statistical techniques can be further divided into template matching (in the input space), or feature-based methods where features are extracted from the image and then fed to a classifier. A great variety of features has been suggested for the character recognition problem. These can broadly be classed into global or local features. Global features, such as the number of holes in a binary image, require the whole image for their computation. In contrast, local features are extracted from a spatially localized area of the image (e.g. the response of a filter applied to the bottom left-hand corner). Correlations between local features that lie in a particular spatial arrangement can be detected by higher-order feature detectors.

Examples of recognizers using (i) global features (Cash and Hatamian, 1987) (ii) local features (Le Cun *et al*, 1990) (iii) template matching (Simard *et al*, 1993) (iv) model-based structural matching (Lam and Suen, 1988) are described in more detail below.

Cash and Hatamian (1987) developed a classifier for isolated characters based on the *moments* of the test image. For a greyscale image with intensity $f(x,y)$ at pixel location (x,y), the moment of order (p,q) is defined by

$$m_{pq} = \sum_x \sum_y x^p y^q f(x,y) \qquad p,q = 0,1,2\ldots \tag{2.10}$$

The definition in equation 2.10 is for "raw moments". Central moments are defined similarly, but the origin of the coordinate system is located at the centre of gravity of

[11] Some interesting work has also been done on integrating segmentation and recognition for character strings such as ZIP codes (Keeler, Rumelhart and Leow, 1991; Burges *et al*, 1992; Matan *et al*, 1992).

the image. They also used "normalized central moments", which are central moments divisively normalized so that the measure is invariant under a scaling of the image. A nearest-neighbour classifier was used. The best classification performance was obtained using the normalized central moments up to order three, plus the horizontal and vertical centres of gravity. Cash and Hatamian experimented with distance metrics and found that a weighted Euclidean distance[12] gave the best performance.

One serious problem with this kind of approach is that the features to be used are selected by the experimenter; there is no capacity to "learn" good features from the data. Neural networks containing hidden layers offer a method to overcome this problem; the lower levels can be viewed as feature detectors, and the higher levels as a classifier. By back-propagating the error through all layers during the training phase, the feature detectors are adapted to the data. The work of Le Cun *et al* (1990) is a sophisticated example of the use of neural networks for character recognition. An image, normalized to 16×16 pixels, is fed through four hidden layers of processing and is finally classified at the output layer. Layers H1 and H3 are comprised of feature extractors, whilst layers H2 and H4 compute local averages of the feature maps from the layer below. H4 is fully connected to the ten output units. Layer H1 computes the convolution of four different, trainable masks/filters with the image. Similarly, layer H3 computes the convolution of twelve different masks with the outputs of H2. H3 learns to combine the features in H2 in order to build higher-order features that are appropriate for the task. Overall, the network has 98,442 connections, but only 2578 independent parameters, because of extensive weight sharing among the feature extractors. The latter statistic is important because reducing the number of free parameters in a network is a way to obtain good generalization performance (on new test images) and avoid "overfitting" the training data[13].

In general, feature-based classifiers can be fooled by jumbled images that contain the right features but in the wrong spatial relationships. For example, Lee (1991) showed that a neural network could produce confident classifications of images that do not look anything like a character. This problem suggests that it would be useful to model the probability density of each class in the image space. When using template-based methods, choosing a distance metric which reflects the properties of the classes will help to reduce the number of templates needed to obtain good performance. Simard *et al* (1993) have used a "tangent-distance" measure, which exploits the fact that we have knowledge of valid transformations of the pixel array which leave the identity of the character unchanged (e.g. translations, rotations of the image), to define a distance measure which makes these kinds of transformations very cheap. The tangent distance between a test pattern and an exemplar is the minimum (Euclidean) distance between the two linear subspaces defined by the tangent vectors at each point.

Lam and Suen (1988) have described a model-based approach to character recognition, which also used ideas from structural pattern recognition. Structural pattern recognition techniques are based on the idea that "a complex pattern [can] be described recursively in terms of simpler patterns" (Pavlidis, 1977). For character recognition a

[12]The weighted Euclidean distance $d(x, y)$ between points x and y is given by $d^2(x, y) = (x - y)^T W(x - y)$, where W is a diagonal positive definite matrix.

[13]Other techniques to avoid overfitting in neural networks include the use of Bayesian methods, e.g. MacKay (1992), Neal (1993a).

natural decomposition is into a set of strokes (this is even more evident for some character sets, such as Chinese). Lam and Suen preprocessed their images to obtain a polygonal approximation to the contour, and then decomposed this representation to describe the contour in terms of curve features (convex polygons) and line segments. They were then able to use a relaxation matching method to determine the quality of the match between the test image and each of a set of templates. Contrasting this approach to template matching schemes in pixel space, we see that the description of the data in terms of strokes will reduce the number of primitives to be matched, but induces some ambiguity in the matching of features in the template and test image that necessitates a search process.

Although the hierarchical methodology of structural pattern recognition is appealing, there are some limitations. One serious problem is the difficulty in robustly extracting the higher-level features bottom-up from noisy input data. Another difficulty is that some shapes, such as smooth curves, do not have a particularly obvious decomposition into parts. For these reasons the recognition system described in this thesis operates at the pixel rather than the stroke level.

The preprocessing used by Lam and Suen to describe the data in terms of line and curve segments recognizes the fact that handwritten characters are produced by a pen or pencil making a one-dimensional curve on the two-dimensional paper. If it is possible to extract the ordering information along the stroke, then techniques for classifying contours (e.g. Turney, Mudge and Volz, 1985; Mokhtarian and Mackworth 1986; Dudek and Tsotsos, 1991) can be used. However, obtaining ordering information bottom-up from a noisy image is difficult, although Doermann and Rosenfeld (1992) have shown that some progress in this direction is possible by automatically detecting some clues that are used in forensic handwriting analysis.

2.3.1 Discussion

Current character recognition systems can achieve high performance. For example, on a standard database of lightly constrained pre-segmented handwritten digits the best systems achieve error rates of about 3% with no rejections (Wilkinson *et al*, 1992). However, more work is required to match human performance. A generative model-based approach should be able to address the following weaknesses of statistical pattern recognizers:

- Most recognizers use normalized images as input. However, if the image is only roughly segmented, this normalization step can be dangerous. A model-based system has the potential to iteratively refine the match so that normalization and segmentation errors can be mitigated, and parts of the image which are not close to the fitted model can be labelled as noise.

- Systems that do not incorporate any prior knowledge about the shapes of characters must learn all of their knowledge from the training examples. As we already know that digits are composed of one-dimensional strokes it seems wasteful to use up training data to learn this.

- A recognizer that "understands" an image should be able to not only label it with the correct class, but should also be able to return the instantiation parameters such as the position, size, orientation, shear and elongation. For handwritten digits we may also want information on the writing style since this is occasionally crucial in disambiguating other digits in the same string.

Chapter 3

The Basic System

In chapters 3 and 4 I describe a system for the recognition of isolated handwritten digits using deformable models. Each of the ten digits has its own model. A digit-image is recognized by finding out which of the ten deformable models matches it best. During the matching process, a model is deformed in an attempt to ensure that every piece of ink in the image is close to some part of the model. The quality of the final match depends on the amount of deformation of the model, and the log likelihood of the ink data under the model. Much of the work in this chapter and the next has been described in Hinton, Williams and Revow (1992a), Williams, Revow and Hinton (1993) and Revow, Williams and Hinton (1993).

Section 3.1 describes the deformable spline models and the affine transformation used to map the models from their object-based frame into the image. Section 3.2 discusses the imaging model used for the experiments. The process of fitting a model to an image is outlined in section 3.3. Details of improving the models using training data are given in section 3.4. Section 3.5 describes the use of a postprocessing neural network, and discusses the performance of the system on a database of handwritten digits.

3.1 Deformable Spline Models

Each digit is modelled by a deformable spline whose shape is determined by the positions of the control points in an object based frame. All models have eight control points, except for the one model which has three, and the seven model which has five. A cubic B-spline is used to generate a smooth curve from the control points, treating the first and last control points as if they were doubled. To generate an ideal example of a digit the control points are positioned at their "home" locations. Deformed characters are produced by perturbing the control points away from their home locations. Thus a single deformable model defines an entire probability distribution across shapes. Assuming a Gaussian distribution for these deformations, the probability density of the n control points is given by:

$$P(\mathbf{x}) = \frac{1}{(2\pi)^n |\Sigma|^{1/2}} \exp[-\frac{1}{2}(\mathbf{x}-\mathbf{h})^T \Sigma^{-1} (\mathbf{x}-\mathbf{h})] \tag{3.1}$$

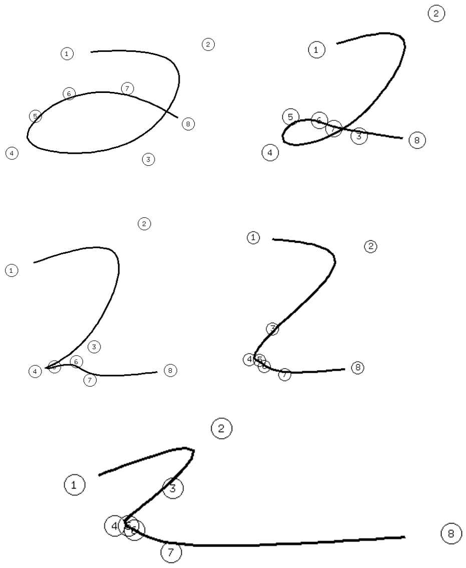


Figure 3.1: Illustrating the ability of spline models to capture topological variations. This figure shows the variety of shapes that can be produced by the 2 model. The large loop of the 2 can smoothly decrease in size and eventually become a cusp.

where $\mathbf{x}$ is the vector of the control point locations (a concatenation of the (x, y) positions of each control point), $\mathbf{h}$ describes the home locations of the control points and Σ is the covariance matrix. The centre of gravity of the home locations is the origin of the object-based frame, so that translations and rotations are not confounded when applying an affine transformation (see below).

The deformation energy E_{def} is defined to be the negative log probability of the deformation.

$$E_{def}(\mathbf{x}) = \frac{1}{2}(\mathbf{x}-\mathbf{h})^T \Sigma^{-1}(\mathbf{x}-\mathbf{h}) + \frac{1}{2}\log|\Sigma| + \text{const} \tag{3.2}$$

Splines are a convenient method for modelling handwritten digits as it is easy to incorporate topological variations. For example, small changes in the relative locations of the control points can turn the loop of a 2 into a cusp or an open bend (see figure 3.1). This advantage of spline models has also been pointed out by Edelman *et al* (1990) where a different kind of spline is used to fit on-line character data by directly locating candidate control points on strokes in the image.

The shape models described above exist in an object-based frame. Unlike the approach taken in many OCR systems, we do not preprocess images in order to remove the effects of translation, scale, rotations, shear, etc. Instead these effects are handled by mapping the models from the object-based frame through an affine transformation into the image-frame[1]. The affine transformation is characterized by a 2×2 matrix and a 2-vector giving the x and y translations. The affine transformation is refined during the

[1] For the 1 model we use a 2D similarity transformation that permits translation, scaling and rotation of the model instead of the full affine transformation. This is because a model that consists of only three (non collinear) control points can become very distorted under a general affine transformation.

matching process so that knowledge about the shape of the digit can influence the choice of affine transformation. This is not possible if normalization precedes recognition, as it does for most OCR systems. The tangent-distance method (Simard *et al*, 1993) recognizes the problem, but can only deal with it as a local approximation in pixel space. Affine transformations are not penalized during the matching process, i.e. they do not contribute to the deformation energy, so deformations are only used to handle true variations in shape that cannot be accommodated by global affine transformations.

The theory outlined in chapter 2 requires that we choose a prior distribution for the affine parameters. In practice we do not impose any restrictions on the affine parameters. This can be viewed as using a very broad and flat prior (e.g. a large variance multidimensional Gaussian) for the affine parameters, which is then effectively constant over the area of interest (however, see section 3.5.1 concerning deformed or rotated affine transformations).

3.2 Imaging Models

We next need to specify the imaging model $P(I|\mathbf{z}, M)$. The general flavour of a (binary) imaging model for this problem is that there should be a high probability of inked pixels close to the spline, and lower probabilities further away. We have investigated two different imaging models; the mixture of Gaussians model which is described in this section, and the "black and white" model which appears in section 4.2.

In general, given a set of normalized probability densities $\{f_i\}$ and a set of mixing proportions $\{\pi_i\}$ which are constrained to be non-negative and sum to one, a mixture probability density f is defined by $f = \sum_i \pi_i f_i$. Note that f is also properly normalized. By spacing out a number of Gaussian "ink generators" along the spline, we can approximate a ridge of high probability density running along it with a mixture model. The quality of the fit to an image can be found by evaluating the likelihood of the inked pixels under the mixture density. Each 2D Gaussian will have the form

$$f_i(\mathbf{y}) = \frac{1}{2\pi\sigma^2} \exp\{-\frac{\|\mathbf{m}_i - \mathbf{y}\|^2}{2\sigma^2}\} \tag{3.3}$$

where $\mathbf{m}_i$ is the location of bead i, $\mathbf{y}$ is a general point in the image plane and σ^2 is the variance of the Gaussian. The generators are spaced approximately two standard deviations apart. A plot of the mixture density will look rather like a spinal cord, with the Gaussians' centres corresponding to the position of vertebrae. We have found that it is also useful to have a background noise process that is able to account for pixels that occur far away from the generators. This is a uniform process over the area of the image, leading to the mixture density

$$f(\mathbf{y}) = \frac{\pi_n}{A} + \frac{1-\pi_n}{B} \sum_i f_i(\mathbf{y}) \tag{3.4}$$

where A is the area of the image, B is the number of generators, π_n is the mixing proportion of the uniform noise field, and $f_i(\mathbf{y})$ is the probability density of the data point at $\mathbf{y}$ under Gaussian bead i. The mixing proportions for the Gaussians are all equal, as ink has an equal probability of being generated anywhere along the spline.

Even though the uniform noise process is a poor model of the highly correlated, structured noise typically found in digit images, it improves the ability of the digit models to correctly segment out the data from the noise in the image (see section 3.3).

Although the digit models are used for recognition, it is helpful to consider how we would use them for generating images. The generative model is an elaboration of the probabilistic interpretation of the elastic net given in Durbin *et al* (1989). To generate a noisy image of a particular digit class, we run the following procedure:

(1) Pick a deformation of the model (i.e. move the control points away from their home locations) to give a particular instantiation $\mathbf{x}$. This defines the spline in object-based coordinates. The log probability of picking a deformation is proportional to the quadratic term in equation 3.2. It is important that the deformation is measured in object-based coordinates.

(2) Pick an affine transformation from the model's intrinsic reference frame to the image frame (i.e. pick a size, position, orientation, slant and elongation for the digit).

(3) Map the spline into image coordinates and place beads uniformly along its length. Each bead is a circular Gaussian ink generator. The number of beads and their variance can easily be changed without changing the spline itself. Typically the variance is chosen so that the bead centres are two standard deviations apart.

(4) Repeat many times:
Either (with probability π_n) add a randomly positioned noise pixel to the image
Or pick a bead at random and generate an inked pixel from the normal distribution defined by the bead.

This is not a very good generative model of the way in which handwritten digits are actually produced. If, for example, the beads have large variances the inked pixels in the image will have the correct overall shape but will be disconnected and much too scattered. However, the generative model is useful for recognizing digits, as explained in the following sections.

The imaging model described above has a nicer generative interpretation than some other models that have been proposed to measure the datafit for deformable models. For example, Yuille *et al* (1990) compute the misfit of a datapoint from a curve by finding the minimum distance between the curve and the point, and paying a cost proportional to the square of this distance. If the standard deviation of the noise is small relative to the curvature of the contour, then data can be generated by randomly choosing a point on the contour and placing a sample from the Gaussian distribution along the perpendicular to the tangent. However, if the contour has significant curvature the generated point may turn out to be closer to some other point on the contour. Our model is "softer", in that all points on the contour could theoretically generate any datapoint, although it is highly likely that nearby points will actually be responsible.

3.3 Fitting a model to an image

To classify an image, we fit each model to the data and choose the model that best "explains" the image. For each digit model, the aim is to search for a minimum of $E_{tot} = E_{fit} + E_{def}$ by adjusting $\mathbf{z}$, the vector of affine and deformation parameters. For a model with eight control points this is a 22-dimensional search involving the six affine parameters and the (x, y) positions for each of control points.

The default starting $\mathbf{z}$ for a model is found by setting the deformations to zero and fitting an upright rectangular box around the black pixels in the binary image. The affine parameters are set so that the transform takes the bounding box of the control points in the object based frame into the image box. However, if the system has low confidence in the classification that is obtained based on the default starting positions, other starting points will be explored, as described in section 3.5.3. In chapter 5 I discuss results obtained by using a neural network to predict the starting $\mathbf{z}$ for each model.

The search algorithm consists of an inner loop which adjusts the instantiation parameters to improve E_{tot}, and an outer loop which adjusts the number and variance of the ink generators in n_{var} stages. The algorithm is outlined in figure 3.2. The reasons for changing the variance of the ink generators during the search are discussed in section 3.3.1. The annealing schedule used for these changes is described in section 3.3.3. Details of the procedure used to update $\mathbf{z}$ are given in section 3.3.2.

3.3.1 Annealing the variance

The search procedure starts with a small number of ink generators with the same large variance placed uniformly along the spline, forming a broad, smooth ridge of high ink-probability[2]. Because of the high variance, the generators are attracted to inked pixels even if they are fairly far away, so the spline is quickly pulled towards the data. The number of beads is gradually increased while their variance decreases according to a predetermined "annealing" schedule. The annealing implements a coarse-to-fine search strategy; with a few large beads the data is viewed at a very coarse scale, so the models can concentrate on adjusting the affine parameters. Only later in the search do they begin to model the finer details of the data. Another way of looking at the annealing process is that it is a continuation method which ultimately assigns correspondences between pixels and beads. At a large bead variance the responsibilities of the beads for an inked pixel are all roughly equal; the model is non-committal about the correspondences. But as the variance shrinks, the responsibilities tend to zero or one, indicating a firm assignment. Notice how this avoids the combinatorics of considering all possible bead-pixel assignments (but at the expense of possibly missing the global optimum solution). This fitting technique resembles the elastic net algorithm of Durbin and Willshaw (1987) except that our elastic energy function is more complex and we are also fitting an affine transformation.

[2] We continue the terminology of Durbin *et al* (1989) by also referring to the generators as beads.

Inputs: a rectangular binary image
a deformable model with parameter vector $\mathbf{z}$

initialize $\mathbf{z}$

for $i = 1 \ldots n_{var}$

set the number and variance of the ink generators from the annealing schedule (table 3.1) and map them into the image-based frame (see section 3.3.4)

evaluate E_{def} (equation 3.2) and E_{fit} (equation 3.7)

$E_{tot} := E_{def} + E_{fit}$

for $j = 1 \ldots max\_iter\_n_{var}$ (see table 3.3.4)

update $\mathbf{z}$ to reduce E_{tot} as detailed in section 3.3.2, using the 3 steps

1. E-step: Evaluate the responsibility of each bead for each pixel using equation 3.5
2. M-step part 1: Calculate the matrix M and vector $\mathbf{b}$ (see equations B.3 and B.4 in Appendix B) and solve $M\mathbf{x}^i = \mathbf{b}$ for $\mathbf{x}^i$, the vector of positions of the control points in the image-based frame
3. M-step part 2: Adjust the affine transformation and deformation parameters while keeping the positions of the control points in the image-based frame fixed, so as to absorb as much deformation energy as possible into the affine transformation (see Appendix B)

evaluate E_{def} and E_{fit}

$E'_{tot} := E_{def} + E_{fit}$; $\Delta E_{tot} := E_{tot} - E'_{tot}$

if $\Delta E_{tot}/E_{tot} < \delta$ **break** (δ is specified in section 3.3.3)

$E_{tot} := E'_{tot}$

end **for**

end **for**

Output: final E_{def}, E_{fit} and other measures defined in section 3.5.1

Figure 3.2: An overview of the search algorithm.

.

If the bead variances are set to sufficiently large values, the behaviour of the model is to shrink to zero scale at the centre of gravity of the inked pixels. There is a critical variance below which this does not happen, which depends on the covariance matrix of the data. I present an analysis of this process in Appendix A.

3.3.2 Updating the instantiation parameters

In some early experiments, a conjugate gradient method was used to adjust the instantiation parameters. This method is slow because each conjugate gradient step may require a few evaluations of E_{tot} in the line-search, where each evaluation involves a computational burden proportional to the product of number of beads and the number of pixels. Our preferred method is based upon the Expectation Maximization (EM) algorithm (Dempster, Laird and Rubin, 1977); the use of the EM algorithm for penalized maximum likelihood estimation has been discussed by Green (1990).

It is useful to first review the EM method for fitting an unconstrained mixture of Gaussians to data, and then go on to discuss its application to the deformable models. The EM algorithm is a way of finding a local maximum of the log likelihood; for the unconstrained mixture this means finding appropriate means for the Gaussians[3]. The basic idea underlying the EM algorithm is that the data supplied is incomplete, and that the problem could be solved more easily if this data were available. For the mixture of Gaussians case, the missing data is the assignment of each datapoint to a particular Gaussian. EM involves repeated applications of the expectation (E) and maximization (M) steps. The assignments of the datapoints to Gaussians are estimated in the E step. Given this information it is easy to adjust the means of Gaussians to maximize the likelihood of the data. This is the M step. Dempster *et al* showed that the EM algorithm is guaranteed not to decrease the log likelihood (or equivalently not to increase E_{fit}) on each iteration.

During the E step, the Gaussians are frozen at their current locations and the "responsibility" that each bead has for each inked pixel is computed. This responsibility is just the probability of generating the pixel under the Gaussian distribution for the bead normalized by the total probability of generating the pixel (including the background noise model). The responsibility of generator i for the pixel at $\mathbf{y}$ is given by

$$r_i(\mathbf{y}) = \frac{f_i(\mathbf{y})}{\sum_{b=1}^{B} f_b(\mathbf{y}) + \frac{\pi_n B}{(1-\pi_n)A}} \tag{3.5}$$

In the M step the mean of each Gaussian is set to a weighted mean of the data points, where the weight factors are simply the responsibility of the Gaussian for the relevant data point. Because the negative log likelihood (energy) under a Gaussian distribution is quadratic in the distance from the mean, the M step is analogous to finding the minimum energy configuration of a system of zero rest-length springs which attach each mean to every pixel by a spring whose stiffness is proportional to the responsibility of the Gaussian for the pixel.

When fitting a digit model to data, the beads are constrained to lie on the spline; the underlying variables are the deformations and affine transformation parameters. If we attempt to apply the EM method directly, the E step is identical to the unconstrained case. However, the M step yields a set of non-linear simultaneous equations for the instantiation parameters, so we have developed a two-stage method to approximate it, as outlined in the next two paragraphs. More details of the search procedure are given in Appendix B.

In the first stage of the M-step, the affine transformation is held fixed and we solve a set of linear simultaneous equations to find the posterior deformations of the control points[4]. Because the affine transformation is fixed, the deformation energy can be translated into the image-based frame and the computations carried out there. Let $\mathbf{x}^i$ be the vector of the position of the control points in the image-based frame. The linear system defined by $\partial E_{tot}/\partial \mathbf{x}^i = M\mathbf{x}^i - \mathbf{b} = \mathbf{0}$ is solved to obtain the new $\mathbf{x}^i$ vector. This process is guaranteed not to increase E_{tot}[5]. The resulting $\mathbf{x}^i$ corresponds

[3] It is also possible to adjust the mixing proportions and variances of the Gaussians, but to strengthen the correspondence to the model fitting process these are kept fixed in this discussion.

[4] There will be $2n$ equations if there are n control points.

[5] The matrix M defined above actually contains terms involving responsibilities which vary with $\mathbf{x}^i$.

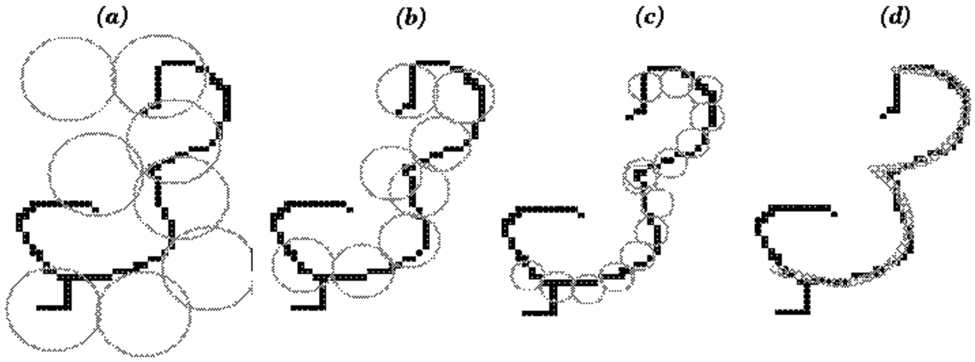


Figure 3.3: The sequence (a) to (d) shows some stages of fitting a model 3 to some data. The circles represent the beads, with the radius representing the standard deviation of its normal distribution. **(a)** shows the initial configuration, with eight beads equally spaced along the spline. In **(b)** and **(c)** the variance is progressively decreased and the number of beads is increased. The final fit using 60 beads is shown in **(d)**.

to the minimum energy equilibrium point where the forces pulling the beads towards the nearby pixels are balanced by the forces pulling the beads towards their home locations.

The second stage of the M-step involves adjusting the affine transformation and deformations to minimize E_{def} whilst holding the new *image* locations of the control points fixed. In effect, we are absorbing as much of the deformation as possible into the global affine transformation. This can be achieved by minimizing the deformation energy with respect to the parameters of the affine transformation matrix and the translation vector. This step cannot increase E_{fit} because this energy depends only upon the image locations of the beads and their variances and these are unchanged during this stage.

Some stages in the fitting of a model to data are shown in figure 3.3. The search technique almost always avoids local minima when fitting models to isolated digits[6]. In the few cases where local minima are encountered they can usually be overcome by starting with a different guess for the initial affine transformation. It is easy for a human to detect a local minimum by simply inspecting the fit, but it is more difficult to automate this process. A restart procedure for the search is discussed in section 3.5.3, and some ideas on detecting poor fits are outlined in section 6.3.

Figure 3.4 illustrates how the addition of the noise model improves the ability of the digit models to correctly segment out the data from the noise in the image; in effect the system is performing model-driven image segmentation.

However, using the EM method we can freeze these responsibilities at their current values, solve for $\mathbf{x}^i$ and guarantee that E_{tot} does not increase. In Appendix B I show that M is a symmetric positive definite matrix.

[6]The search procedure also works if more than one model is used. However, if the data consists of close-together or overlapping digits, the search needs to be started near to the correct solution to find the global minimum successfully.

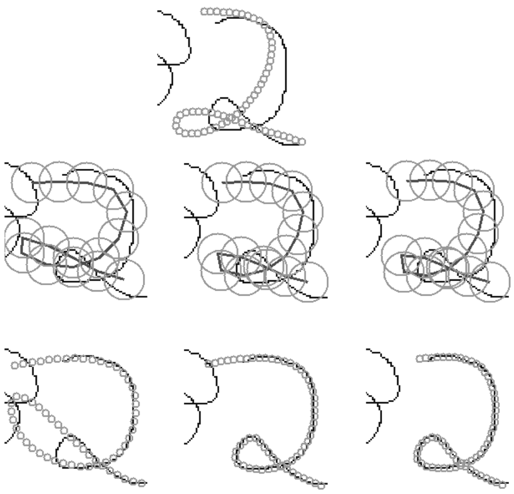


Figure 3.4: An illustration of how the noise model helps in model driven segmentation. The top row shows an image of a two with a portion of left neighbour. The initial configuration of the two-model is also shown. The remaining rows show the configurations of the model mid way in the settling (second row) and the final settled configuration. The three columns show different mixing proportions of the noise; column (a) has the least and column (c) the most. Notice in (a) how the model has been unable to ignore the extra data, but that in (c) it has done so.

3.3.3 The annealing schedule and speeding up the search

In order to classify an image of a single digit, ten models must be allowed to settle on the image[7]. The annealing schedule for each model is described in table 3.1. The performance of the system does not depend critically on the exact details of the schedule.

Settling every model using this annealing schedule is a very large computational task; we have used two shortcuts to reduce the amount of computation required. Firstly, it is wasteful to repeatedly calculate updates if the algorithm has converged at a particular variance. The fractional change in the total energy $\Delta E_{tot}/E_{tot}$ is monitored, and if it falls below a threshold (we used 0.001) the set of iterations is terminated and we move on to the next variance step. Of course, this strategy runs the risk of terminating prematurely if there is a plateau in the settling curve, but we have found that it works well in practice. The maximum number of iterations permitted at low variances is set quite high in table 3.1. On average only about one quarter of these iterations are required, but we have observed that occasionally considerably more iterations are required in order to get a good fit.

Secondly, at low variance, each pixel will have significant probability of having been generated by only a few beads. One way to dramatically speed up the search is to eliminate all of the computations that are used to update the responsibilities of

[7] More than ten if multiple models are used for each digit. For example we may choose to have a separate model for the "crossed seven" or the "two circles eight".

Number of beads	Maximum number of iterations $max\_iter\_n_{var}$	Variance
8	5	0.04
15	3	0.02
	3	0.01
	35*	0.008
30	35*	0.0025
60	45*	0.0006
	2	0.0006

Table 3.1: The Annealing schedule. Iterations marked with a * used the freeze-EM algorithm. The scale is set so that in the object-based frame each model is one unit high.

beads that are many standard deviations from a pixel. This is particularly important at low variance, where the number of beads, and hence the number of floating point operations per iteration, is large. During the E-step of the settling we can simply freeze these responsibilities at their current low values, and only update the *relative* responsibilities of those beads that have a significant probability of generating the pixel. After performing a number of EM iterations with these frozen responsibilities, they are unfrozen and a few full EM iterations are performed. In this way we would hope to achieve the same minimization of E_{tot} with much less computation. We call this procedure freeze-EM, and used it on the iterations at variances 0.008, 0.0025 and 0.0006 (except for the final two iterations, which were always done without freezing). This method is validated by recent work of Neal and Hinton (1993) which shows that incremental schemes can be accommodated in the EM framework.

3.3.4 Setting the bead variances in the image-based frame

In the annealing schedule, the bead variances are set in the object based frame, where the scale is fixed so that each model is one unit high. It would be possible to map the circular Gaussians from the object-based frame through the affine transformation to yield elliptical Gaussians in the image-based frame. However, this would increase the amount of computation, and also does not really correspond to the desired generative model. Another problem is that different digit models may use very different affine transformations, producing very different variances for the beads, making model comparison more difficult. We therefore scale the standard deviations of the circular beads in the image by the scaling factor on the y-axis in the initial affine transformation (found by fitting a rectangular box around the data). As this factor is very similar for all of the models there is little variability in the final bead variances.

Instead of annealing the bead variances according to a pre-determined schedule, it is also possible to estimate bead variances from the data at each iteration. For the case where all beads have the same radial distribution the maximum likelihood estimator for the variance is simply:

$$\hat{\sigma}^2 = \frac{1}{2N_B} \sum_i^{N_B} \sum_b^{B} r_{kb} \, \|\mathbf{m}_b - \mathbf{y}_i\|^2 \tag{3.6}$$

where r_{kb} is the responsibility of bead b at location $\mathbf{m}_b$ for pixel k at location $\mathbf{y}_k$. Experiments using the update in equation 3.6 show that in most situations the variance will anneal similarly to the hand-crafted schedule, but usually at the expense of more iterations.

3.3.5 Scale and the imaging model

It would be attractive if the recognition process was invariant under scalings of the image. Hence we aim to define the deformation and datafit energies in such a way that they are invariant under a scaling transformation. As the deformation energy is measured in the object-based frame it is already invariant.

Under the imaging model described in section 3.2, the datafit energy would be given by $E_{fit} = -\sum_i^{N_B} \log f(\mathbf{y}_i)$, where i indexes the inked pixels. This is clearly not invariant under scale changes; if the linear dimension of the image changes by a factor s, there will be s times as many inked pixels (or s^2 times more if the image is not thinned), and therefore more pixels will enter the sum. There are two factors that influence how the datafit changes with scale (i) the likelihood of each pixel and (ii) the number of pixels. Because of the scaling of the bead variances described in section 3.3.4, the Mahalanobis distance from each bead to each pixel will be unchanged. However the normalizing factor (the $(2\pi\sigma^2)^{-1}$ in equation 3.3) will vary with the scale. The effect of the change in the number of pixels can be dealt with by simply down-weighting the contribution from each pixel. The datafit energy that we actually use for the settling is:

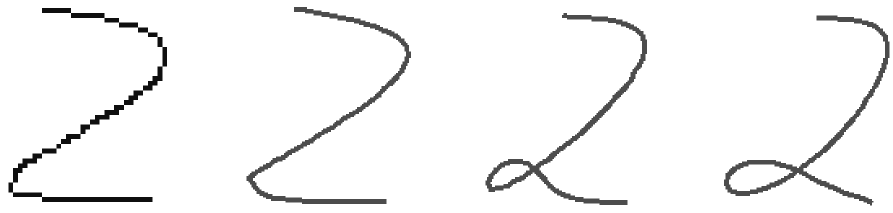
$$E_{fit} = -\frac{N_0}{N_B} \sum_{\substack{i \,\in\, inked \\ pixels}}^{N_B} \log f(\mathbf{y}_i) \tag{3.7}$$

where N_0 is a standard number of pixels[8]. Notice that the normalizing factor $(2\pi\sigma^2)^{-1}$ is just an additive constant in the datafit energy, so the dynamics of the settling, which depend on the *forces* exerted by the deformation and datafit terms (and thus to the *gradients* of the two energies), are unchanged by the scaling.

In fact we do not simply use the deformation and datafit energies to compare models, but send them (plus other measures of the fit) to a neural network (see section 3.5.2). The datafit measures that are sent to the net are the differences between each model's datafit energy and the model with the lowest (best) datafit. As the offset in the datafit energy due to scaling will be roughly the same for each model, the preprocessed datafit measures should not vary very much with a scaling of the image.

The effect of varying N_0 is illustrated in figure 3.5.

[8]The results reported below were obtained with N_0 set to 160. This value was obtained by experiment.



(a) (b) (c) (d)

Figure 3.5: The effect of varying the weighting parameter N_0 in the objective function $E_{tot} = E_{def} + E_{fit}(N_0)$. Panel (a) shows the image to which the models have been fitted. Panel (b) shows the fit obtained with the usual weighting between the two terms. Panel (c) illustrates the fit when the N_0 is reduced to one fifteenth of its usual value; this weighting increases the relative cost of model deformations, so the model deforms less than in panel (b). Panel (d) shows the undeformed 2 model.

3.4 Learning the models

Each deformable model is described by a vector of mean or *home* locations ($\mathbf{h}$) of the control points and a covariance matrix, Σ (see equation 3.2). We initially start from hand constructed models, and then adapt them using a training set. We have investigated two training schemes—maximum likelihood (ML) and maximum mutual information (MMI) learning. In maximum likelihood training the fit of a model on the correct data is taken into account in order to improve the model. This is to be contrasted with maximum mutual information training, which aims to get good performance on the correct data, but also explicitly wants to do poorly on the wrong data. For example under MMI training, a five model would be trained to win on instances of fives, but also to do poorly on other digits. I will first explain maximum likelihood learning and then go on to discuss maximum mutual information learning.

The penalized log likelihood of a set of images is given by

$$L_{ML} = -\sum_{i \in images} E^i_{tot} = \sum_i [\log P(I_i|\theta, \mathbf{z}^*_i) + \log P(\mathbf{z}^*_i|\theta)] \tag{3.8}$$

where θ denotes the model parameters ($\mathbf{h}$ and Σ), and $\mathbf{z}^*_i$ is the vector of best-fitting instantiation parameters for image I_i (see equation 2.6). The objective of ML learning is to maximize L_{ML} by adjusting the model parameters θ; by evaluating $\partial L/\partial\theta_j$ we can use a gradient-based method to carry out the search. However, there is a potential difficulty with the calculation of the gradient. Changing a parameter of a deformable model causes a relatively simple change in the energy of the configuration that the model previously settled to, but the model no longer settles to that configuration (i.e. $\mathbf{z}^*_i$ changes). Fortunately, because the model had settled to a stationary point of the inner loop objective function, these effects do not contribute to the gradient calculation, analogously to the deterministic Boltzmann machine (Hinton, 1989). This argument is also explained in equation 3.9, bearing in mind that $\left.\frac{\partial E^i_{tot}}{\partial z_k}\right|_{\mathbf{z}=\mathbf{z}^*} = 0$ after settling.

$$\frac{\partial E^i_{tot}}{\partial \theta_j} = \left(\frac{\partial E^i_{tot}}{\partial \theta_j}\right)_{explicit} + \sum_k \frac{\partial z^*_k}{\partial \theta_j} \left.\frac{\partial E^i_{tot}}{\partial z_k}\right|_{\mathbf{z}=\mathbf{z}^*} \tag{3.9}$$

As $P(I_i|\theta, \mathbf{z}^*)$ does not depend explicitly on θ the derivative depends only on the deformation energy term. Under the iteration defined by $\partial L_{ML}/\partial \theta_j = 0$, we find that the updated home location of each control point is the average location of that control point in the final fits. Adaptation of the covariance matrix is discussed in section 4.1.1.

The training procedure requires repeatedly settling the models onto the dataset. Learning proceeds rapidly with models learning their final configurations after only a few passes through the training set (figure 3.6), probably because we start off with good models. In practice, training was only done on those cases where the correct model was the winner, because the model will often be excessively distorted when it has fitted wrongly. As an example, the initial and final configurations of the two and three models are shown in figure 3.7.

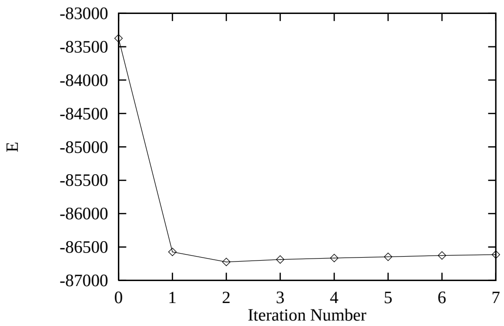


Figure 3.6: A training run for the two model. The ordinate shows the sum of E_{tot} over all training images. The model has essentially completed its learning after the second pass.

An alternative to maximizing the likelihood of the image given the digit is to maximize the mutual information between the correct digit class and the probabilities assigned to the various classes by the digit models. The maximum mutual information criterion emphasizes correct discrimination rather than correct modeling of the image data, and it generally leads to better discriminative performance (Hampshire and Waibel, 1989), although this advantage of discriminative learning vanishes if the generative model is correct and the fitting process produces the true probability of the data given the model (Brown, 1987). Early experiments showed that, for our generative models, maximum likelihood learning was as effective as discriminative learning, perhaps because the deformable models provide a reasonable approximation to the shape variability observed in the data. Maximum likelihood learning is much faster because there is no need to fit the incorrect digit models to each of the training digit images.

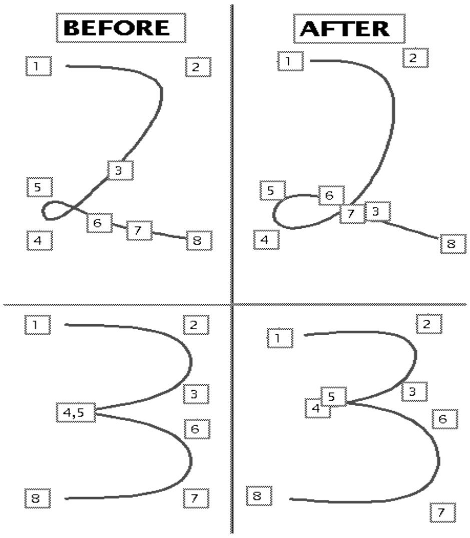


Figure 3.7: The two and three models before and after learning. The control points are labelled 1 through 8.

3.5 Recognizing isolated digits

3.5.1 Additional quantities used to assess a fit

After fitting all the models to a particular image, we wish to evaluate which of the models best "explains" the data. The natural measure is the sum of E_{fit} and E_{def} and the volume factor. However, because our generative models are not exactly correct, we have found that performance can be improved by including four additional terms which are easily obtained from the final fit of a model to the image, and then training a simple neural network to optimize the discrimination based on these inputs.

One way in which our generative model is deficient is that it does not explicitly penalize matches in which there are beads far from any inked pixels. We call this the "beads in white space" problem[9]. In terms of the distinction brought out in section 2.2.2, the model *explains* the black pixels, but is not well *supported.* To quantify this we defined another energy term E_w which penalizes beads in white space

$$E_w = -\sum_{b=1}^{B} \log \sum_{i=1}^{N_B} f_b(\mathbf{y}_i) \tag{3.10}$$

where i indexes inked pixels. This equation is similar to equation 3.7, except that the rôles of the beads and the black pixels are reversed. A bead only makes a large contribution to this cost when all inked pixels are far from the bead. This energy term could be incorporated into the fitting procedure, but in the present system it is simply used as an additional term when evaluating the final fits.

While fitting to an image, a model can adopt any affine transformation without penalty. However, we do have prior beliefs that certain affine transformations that

[9]There is a small implicit penalty in that beads far from inked pixels are not available for accounting for inked pixels.

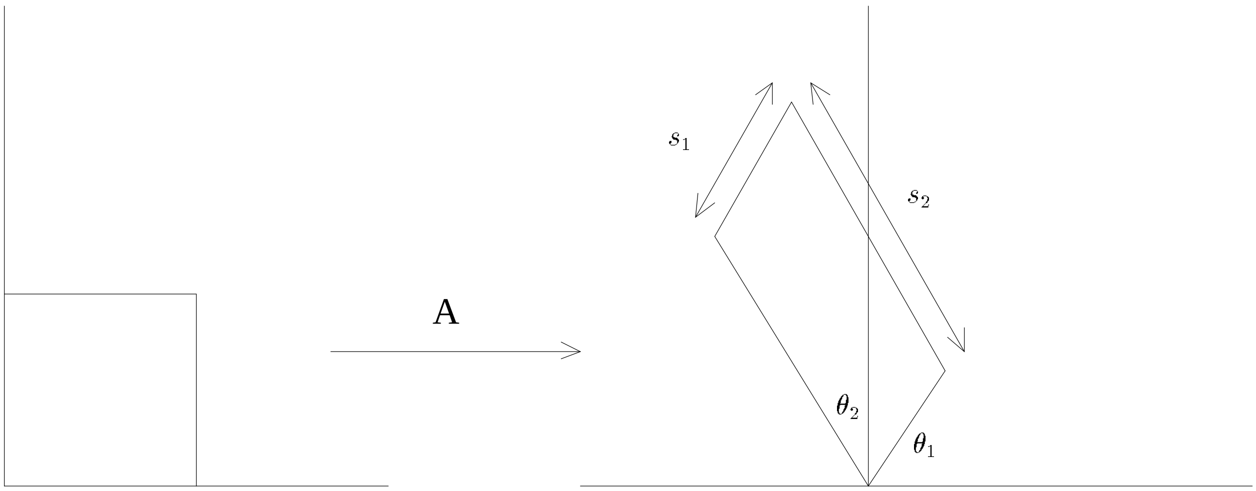


Figure 3.8: An affine transformation acting on a unit square to yield a parallelogram.

require a model to be highly rotated, sheared or elongated are unlikely. Hence the rotation, shear and elongation of the affine transformation for each model are computed and supplied to the classifier. These quantities are defined in terms of the $(s_1, \theta_1, s_2, \theta_2)$ representation of the transformation matrix

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} s_1 \cos\theta_1 & -s_2 \sin\theta_2 \\ s_1 \sin\theta_1 & s_2 \cos\theta_2 \end{pmatrix} \tag{3.11}$$

$$\begin{aligned} shear(A) &= \sin^2(\theta_1 - \theta_2) && (3.12) \\ orientation(A) &= \sin^2\theta_2 && (3.13) \\ elongation(A) &= s_2/s_1 && (3.14) \end{aligned}$$

A geometrical interpretation of the effect of A is given in figure 3.8. Note that the orientation and shear measures are normalized to the range $[0, 1]$.

3.5.2 Using a neural network

In the case where a model fit is evaluated on the basis of only two quantities, E_{def} and E_{fit}, there is an unknown regularization parameter (usually denoted by λ) which determines the tradeoff of these two terms. Under a strict interpretation of our framework, this regularization parameter is related to the scale of the covariance matrix in the deformation energy. However, it is possible to relax this somewhat, so that there is both a tradeoff parameter and a scale for the covariance matrix. In this case the λ for each model can be adapted independently to maximize the classification performance of the system. If more measures of the fit are available, a neural network can be trained to optimize the performance.

The deformation energy, datafit energy, approximate volume factor[10], white-space penalty, shear, orientation and elongation of each model are used as inputs to a neural

[10] The framework set out in section 2.1 requires the evaluation of the determinant of the information matrix $\det H$. This quantity was approximated by calculating the determinant of the matrix M defined in section 3.3.2. Both quantities give an indication of how the value of E_{tot} changes for small changes in the instantiation parameters. It can be shown that as the responsibilities of the beads for each datapoint tend to 0 or 1, M tends to information matrix with respect to $\mathbf{x}^i$. The quantity provided to the neural network is $\log\det M$. In future work I would like to study how $\det H$ and $\det M$ covary.

net. These inputs are connected directly to the ten output units. Each output unit computes a weighted sum of the inputs belonging to its model. The output units compete using the "softmax" function (Bridle, 1990) which guarantees that the 10 output values will sum to one, forming a probability distribution over the choices. Neural networks are much easier to train if all inputs are approximately of the same magnitude, i.e. it is good practice to subtract out any constant offsets and scale the input data. The E_{fit} term has a large offset, so we used the difference $\tilde{E}^k_{fit} = E^k_{fit} - E^{min}_{fit}$ as the data fit term for the k^{th} model, where E^{min}_{fit} is the minimum data fit obtained by any of the models for the current image. Including biases, the network has 80 weights and is trained using conjugate gradient to minimize a cross-entropy error function. After training an image is classified according to which of the output units has the largest activation. Cases for which the maximum output activation is below some threshold T are rejected.

We tried a variety of architectures for this "post-processing" network. For example, a digit recognition system developed by Hastie and Tibshirani (1992) suggests that discrimination would be much better if the net was totally connected, so that the output unit that represents a particular digit receives detailed information about the way in which other digit models fit the data. However, we found that discrimination was just as good if each output unit only received connections from the seven inputs representing terms describing the fit of that digit model. Including a hidden layer also did not improve performance.

3.5.3 Procedure for recognizing isolated digits

Our data consists of binary images, which were thinned using a standard algorithm (Gonzalez and Wintz, 1987). Thinning is used because it reduces the number of black pixels in an image, thus speeding up the search. It is also useful because it removes some extra detail like the thickness of the stroke, and reduces the image to its essence, the shape. Of course, thinning is not perfect and occasionally artifacts are created.

The recognition process for a single image involves settling each of the ten models onto the data, and then computing the variables that are fed into the neural network. If the winning output is less than a threshold[11], it is assumed that the models have not fitted properly to the data and a restart procedure is invoked. This involves settling each model from four other starting positions translated right, above, left and below the original one. The fit with the lowest E_{tot} is then chosen for each model and the data fed to the neural network.

3.5.4 Results on isolated digits

The performance of the deformable models in recognizing isolated digits has been tested on data from the CEDAR CDROM 1 database of Cities, States, ZIP Codes, Digits, and Alphabetic Characters[12]. The *br* training set of binary segmented digits was subdivided into 3 training sets of size 2000, 7000 and 2000 respectively (TR1, TR2, and TR3), and a validation set of 2000 examples (VAL). The sets were constructed by drawing images in the order presented on the CDROM, and have equal numbers of all digits in

[11] A threshold of 0.9 was chosen empirically.

[12] Made available by the United States Postal Service Office of Advanced Technology.

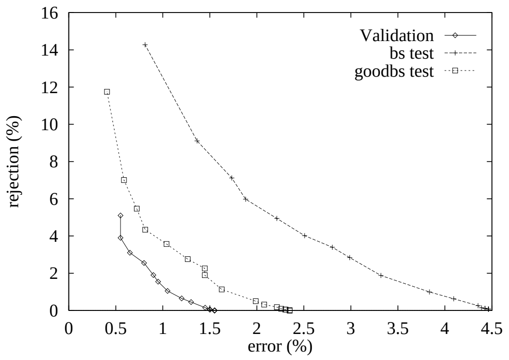


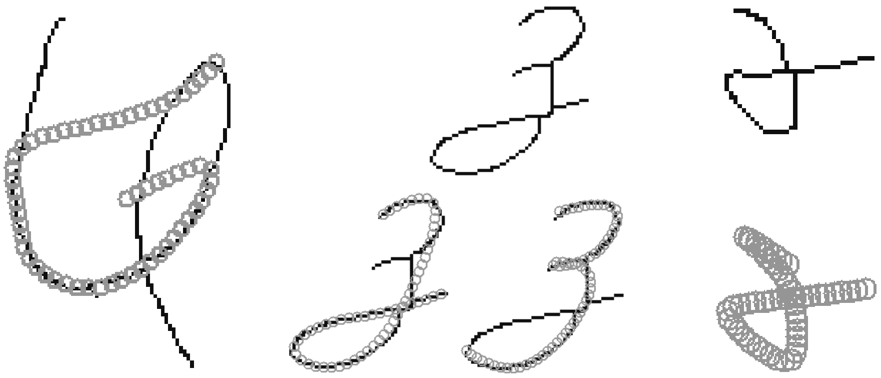
Figure 3.9: Error-rejection relationships for the validation set, *goodbs* test set and *bs* test set

each set. The CEDAR database also includes two test sets. The *goodbs* (2213 images) set is a subset of the *bs* (2711 images) set containing only well segmented digits. The deformable models were trained on TR1 and the post-processing net on TR3. TR2 was used for training the mixture of local models (see section 4.1.2). The validation set was used to compare different versions of the system, and a run on the test set was only done when a final version had been chosen, to avoid tuning the system to the test data.

The results obtained were 1.55% errors on the validation set, 2.34% on the *goodbs* and 4.57% on the *bs* test sets with no rejections. Varying the rejection threshold in the post processing neural network allows us to trade off errors against rejects. Figure 3.9 shows error-rejection curves obtained on the validation and test sets. For these experiments all of the models had the same diagonal covariance matrix $\frac{1}{\sigma^2}\mathbf{I}$ with $\sigma^2 = 0.01$. Other experiments are reported in chapter 4. So far, there have not been many other published results on this database. Lee and Srihari (1993) have reported a result of 2.95% errors on the *bs* test set for their best classifier, but they used the *bs* data for validation purposes, and so tuned their system to this data.

We studied the errors made by the system on the validation set, in order to understand how to improve performance. Although it is difficult to condense the information contained in the fit of all ten models to a particular image into a few bits, the errors can be put roughly into the following classes:

- S Search: The correct model appears to have got stuck in a local minimum. By adjusting the instantiation parameters the right model can be made to win. An example of a poor fit is shown in figure 3.10(a).
- Pe Preprocessing: Important features are lost in the thinning operation, making it difficult for the correct model to win.
- D Model deficiency: There are extra strokes (e.g. ligatures leading to/from a neighbouring character or embellishments of the basic shape such as the cross-bar on European style 7's) which the correct model does not account for; this problem is illustrated in figure 3.10(b).



(a) (b) (c)

Figure 3.10: Some typical errors. (a) A search error. By starting the search with the model rotated anti-clockwise, the correct fit can be obtained. (b) illustrates the problem with ligatures. The bottom left panel shows the fit of the two model to an image of a 3 with a large ligature; the fit of the three model is shown bottom right. The two model has been able to "explain" the ligature using its loop, and has only paid a small datafit cost for ignoring the middle (cusp) section of the 3. (c) illustrates a creative fit by the wrong model. The 8 model has rotated and deformed its upper left part in order to fit the 2 data shown above.

W One of the wrong models is creative in its fit, i.e. it was able to deform cheaply in a way which we know is inappropriate for that digit. See figure 3.10(c) for an example.

Po Postprocessing: The fit of the correct model is good, but for no obvious reason it does not win after processing by the neural net.

For any particular example there may be a combination of factors that conspire to produce an error. However, an exercise to evaluate the source of errors on the validation set indicates that modelling problems (the D and W categories) and search errors are the most prevalent, with pre and postprocessing being relatively unimportant. The work in chapters 4 and 5 deals with improvements aimed at obtaining better performance. In chapter 4 improvements to the deformation modelling and the datafit terms are discussed. Better deformation modelling should help by reducing the flexibility of the models to be creative in the wrong ways, and should also help the search by restricting the freedom of the model. Better datafit measures would improve the search and the evaluation of the final fits. Chapter 5 deals with ways of using neural networks to help instantiate the models, with the aim of improving the speed and search capability of the system.

Chapter 4

Extensions to the basic system

In section 4.1 I discuss improvements to the modelling of the deformation energy, using both a full covariance matrix Gaussian and a mixture model, and show results obtained with these new models on the data used in chapter 3. Section 4.2 describes a different generative model which accounts for the white as well as the black pixels in the image. In section 4.3 I show quantitatively that the style of writing for one digit of a ZIP code has some predictive power regarding the style for other digits in the same code; this kind of information is not available from conventional classifiers.

4.1 Improving the deformable models

The system tested in chapter 3 used a simple spherical Gaussian distribution for the deformations of the control points away from their home locations in the object-based frame. By using the posterior locations of the control points from a training set it is possible to better characterize the distribution; the two approaches considered below are (i) fitting a general covariance matrix Gaussian and (ii) using a mixture distribution. Better characterization of the distribution helps to penalize configurations that arise when the model fits to "wrong" data, i.e. images of digits which do not correspond to the model's identity.

4.1.1 Fitting a full covariance matrix

In section 3.4 we discussed how to adjust the home locations of the models to perform maximum likelihood learning. A similar procedure can also be followed to estimate the covariance matrix, although the derivation is a little more complex.

Let Σ^{-1} be the inverse covariance matrix for the prior distribution of the instantiation parameters, $\mathbf{z}$. As we can assume that the deformations and affine transformations are independent, Σ^{-1} can be written in block-diagonal form:

$$\Sigma^{-1} = \begin{pmatrix} \Sigma_d^{-1} & 0 \\ 0 & \Sigma_a^{-1} \end{pmatrix} \tag{4.1}$$

where Σ_d^{-1} is the inverse covariance matrix for the control points and Σ_a^{-1} is an inverse covariance matrix with small eigenvalues, corresponding to a broad prior on the affine parameters. The mean vector $\tilde{\mathbf{h}}$ of the prior distribution is the concatenation of the

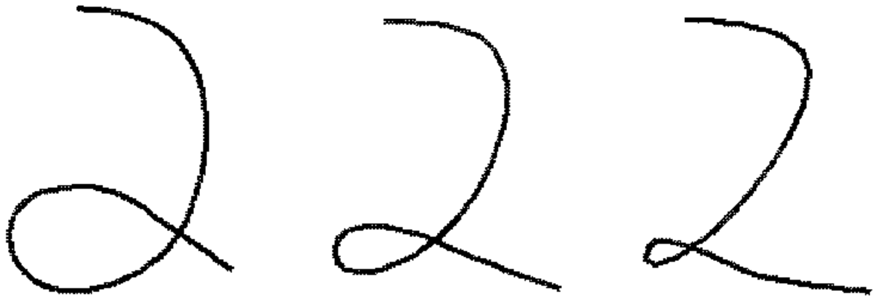


Figure 4.1: The principal mode of variation for the two model at ± 2 standard deviations from the mean. The undeformed model is shown in the centre.

home locations of the control points (in the OBF), and the mean values of the affine parameters. For image i (which has best-fit parameters $\mathbf{z}_i^*$), the log likelihood L_i is given by:

$$L_i = \log P(I_i|\mathbf{z}_i^*) + \frac{1}{2}\log\det\Sigma^{-1} - \frac{1}{2}(\mathbf{z}_i^* - \tilde{\mathbf{h}})^T\Sigma^{-1}(\mathbf{z}_i^* - \tilde{\mathbf{h}}) \tag{4.2}$$

Using the same argument that applied in section 3.4, the derivative of L_i with respect to Σ^{-1} depends only on the terms that explicitly involve Σ^{-1}. Making use of the matrix identity $\frac{d}{dM}\log\det M = M^{-1}$ for a symmetric matrix M, we can differentiate equation 4.2 with respect to Σ^{-1} to obtain

$$\frac{\partial L_i}{\partial(\Sigma^{-1})} = \frac{1}{2}[-(\mathbf{z}_i^* - \tilde{\mathbf{h}})(\mathbf{z}_i^* - \tilde{\mathbf{h}})^T + \Sigma] \tag{4.3}$$

We can re-estimate Σ by solving $\partial L_{ML}/\partial\Sigma^{-1} = 0$ (where $L_{ML} = \sum_i L_i$ is the total log likelihood of a set of images) to obtain:

$$\Sigma = \frac{1}{N_t}\sum_i^{N_t}(\mathbf{z}_i^* - \tilde{\mathbf{h}})(\mathbf{z}_i^* - \tilde{\mathbf{h}})^T \tag{4.4}$$

where N_t is the number of training images. In fact only Σ_d was updated, leaving the prior distribution for the affine parameters unchanged.

The covariance matrices for each of the ten models were re-estimated using this approach. Interestingly, a principal component analysis of the covariance matrices shows a large range in the eigenvalue spectrum, which indicates that certain "modes" of deformation are much more important than others. For example, the principal mode of the two model is to open and close the loop (see figure 4.1). The eigenvalues ranged from around 10^{-2} down to 10^{-10} for the smallest ones, and for most digits 95% of the variance was explained by about one half of the components. Each covariance matrix was estimated using 700 examples, which may not be a great deal of data for a 16×16 matrix. Because of this, we regularized the solution by setting the smaller eigenvalues to be at least one hundredth of the largest eigenvalue on each model[1]. This also produces

[1] This was arrived at after some experimentation with different values of the max/min eigenvalue ratio and the weighting between the data-fit and deformation energies.

better conditioning in the settling process. Results obtained on the CEDAR CDROM data for models with these covariance matrices are given in section 4.1.3. It would also be interesting to investigate maximum mutual information training for the covariance matrices.

The use of a full covariance matrix is similar to the work of Cootes and Taylor (1992), except that they fitted their models *by hand* to the data, rather than using an automatic procedure. They explicitly use only a small number of the principal components, chosen so that they capture most of the variance.

4.1.2 A mixture model for deformations

A particular instantiation of a deformable model in the object-based frame can be specified by the (x, y) locations of n control points. Therefore an instantiated model can be thought of as a point in $\mathcal{R}^{2n}$ and the population of models forms a distribution in $\mathcal{R}^{2n}$. In equation 3.1 this distribution has been described as a Gaussian hyper-ellipsoid. This characterization of the distribution by a single *generic* model can result in poor approximations to the true distribution; figure 4.2 illustrates such a case in two dimensions. For example, under the single Gaussian approximation to the distribution, point **A** would have higher probability than point **B**, which is clearly incorrect.

An alternative way to approximate the distribution, is to use a mixture of L *local* models:

$$P_{mix}(\mathbf{x}) = \sum_{l=1}^{L} \frac{\beta_l}{(2\pi\sigma_l^2)^n} \exp\{-\frac{\|\mathbf{x} - \mathbf{h}_l\|^2}{2\sigma_l^2}\} \tag{4.5}$$

where β_l is the mixing proportion, σ_l^2 is the variance and $\mathbf{h}_l$ is the centre for the l^{th} local model in the mixture. In equation 4.5 we have chosen to model each component in the mixture with a spherical covariance matrix. In practice ten components were used for each model. Experiments with other variations such as centering a spherical Gaussian on each data point[2] (rather than learning the centres and variances) found no improvement in performance over the simpler scheme.

The centres and the variances were computed by using the EM algorithm to maximize the log likelihood of training set TR2 under the mixture distribution. For each digit we did 20 runs of fitting the mixture to the data, using different random starting points, and chose the run which gave the best log likelihood of the data. Figure 4.3 shows the 10 local models in the mixture for the two-model. The mixture has been able to capture dominant styles. For example, variations in the presence and size of the loop have been well represented.

The obvious way to use this mixture is to fit each of the local models to an image. This has the disadvantage of increasing the recognition time by a factor of L. Fortunately we found that the generic model nearly always fits correctly to the image, so an alternative strategy is to fit the generic model, but then evaluate its deformation energy under the mixture distribution. This strategy is much more efficient since the most computation intensive portion, the fitting of the model to the data, is done only once. Evaluating the distance of the final fit from each of the local models in the distribution involves computing only $2n$ squared distances and so is negligible compared to the amount of computation incurred in evaluating E_{fit}.

[2] This technique is known as kernel density estimation.

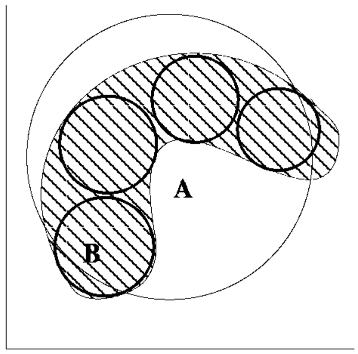


Figure 4.2: Approximating a distribution with a mixture of Gaussians. The high density area of the distribution is contained within the shaded area. The large circle represents the poor approximation that would be made by a single Gaussian of large variance. A better approximation is obtained by using a mixture of Gaussians each with a smaller variance. Under the single Gaussian approximation point **A** would be incorrectly considered to be more likely than point B.

Figure 4.4 illustrates the added classification power obtained using a mixture of local models. There is considerable overlap between the distributions of E_{def} for correct and incorrect classification when only a single generic model is used. The overlap has been diminished and increased separation achieved with the mixture distribution. The effect of using the mixture of local models is examined in more detail in figure 4.5, where the deformation energies under the generic and mixture models are plotted for all ten digit models on the 4 data from VAL. Similar results can be obtained from data corresponding to other digits.

There is an interesting interpretation to these local models. One may think of each local model as capturing dominant writing styles in the population. A particular instantiation can then be classified in terms of style. This is explored further in section 4.3.

4.1.3 Results obtained using the improved models

Table 4.1 shows the performance of the deformable models when the rejection threshold was set to zero. It compares the performance obtained using the diagonal covariance matrix discussed in section 3.5, and the two alternatives discussed in this section. For each option a different neural network was trained to give the best performance. A restart threshold of 0.9 was used—if the probability of the winning model (as evaluated by the neural net) was below this threshold, the four alternative starting points (see section 3.5.3) were also evaluated. Restarts took place on approximately 5 to 6% of cases for the validation set, and between 7 to 10% of cases on the *bs* test set. The results on the test sets show a small increase in performance for the more elaborate methods.

Varying the rejection threshold in the post processing neural network allows us to trade off errors against rejects. Figure 4.6 shows error-rejection curves obtained on the validation and test sets.

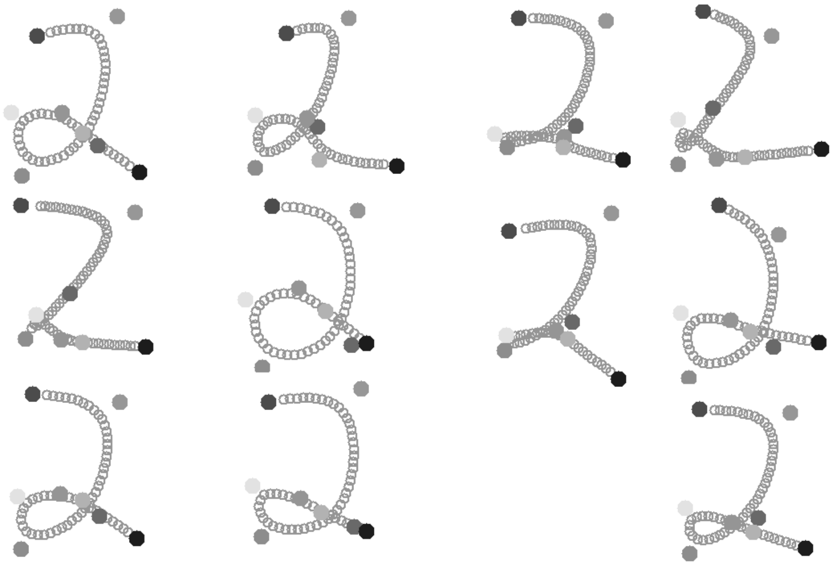


Figure 4.3: Components in the mixture for the 2 model. The generic model is shown in the bottom right corner.

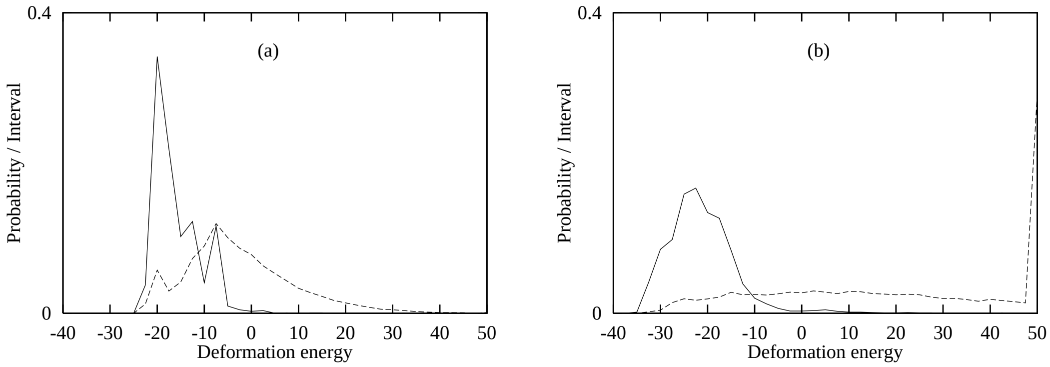


Figure 4.4: The distribution of the deformation energy under the generic model and mixture of local models. The solid line shows the E_{def} distribution of models when fitted to examples images containing the correct digit (e.g. a three model fitted to an image of a 3). The dotted curve shows the E_{def} distribution of the models when fitted to incorrect data. (a) Using a single generic model. (b) Using a mixture of local models. Many instances of fitting to incorrect data in panel (b) had very large E_{def}. For display purposes all of these were assigned the value 50, accounting for the "spike" at the right edge of panel (b)

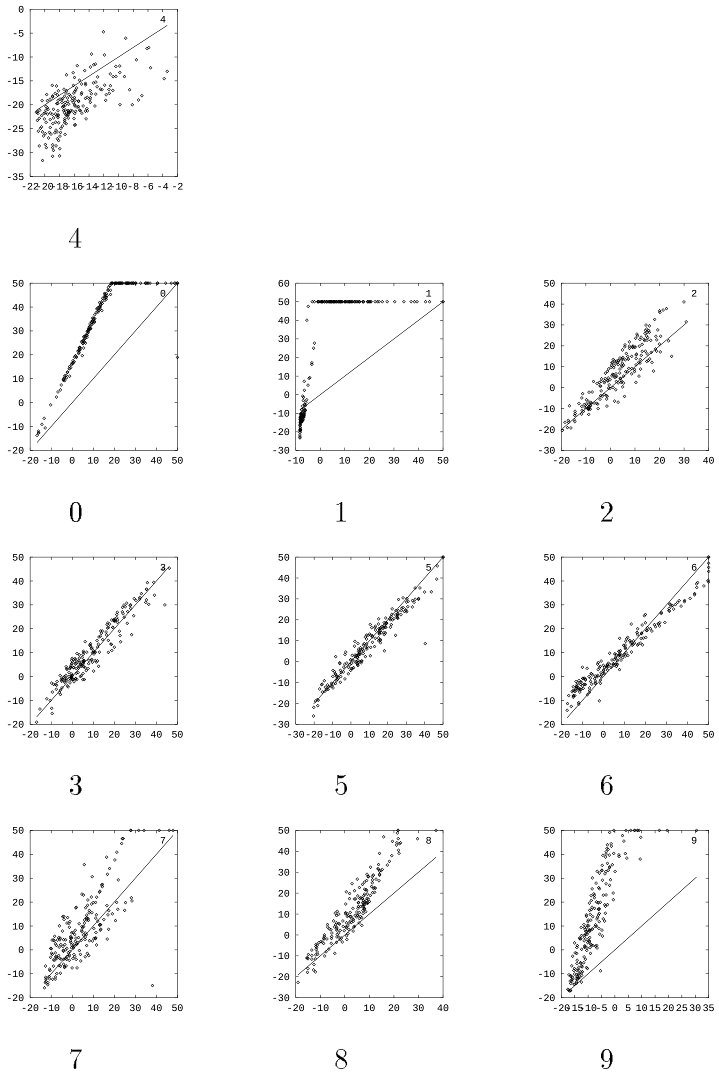


Figure 4.5: A comparison of the deformation energy under the generic model and mixture models for all ten digit models on the 4 data in VAL. In each cell the deformation energy under the generic model is plotted on the abscissa, and the deformation energy under the mixture of local models is plotted on the ordinate. The straight line in each cell is the function $y = x$. Deformation energies greater than 50 have been assigned a value of 50 for display purposes. Notice that for the 4 model most of the points lie below the line, showing that the deformation energy under the mixture model is less than for the generic model. Conversely, for many of the other models the majority of the points lie above the line, indicating that the deformations are cheaper under the generic model. The straight line behaviour of the data for the 0 cell is due to a lack of variability in the mixture centres for the zero model, so that the main effect is due to the smaller variance of the local models.

	Validation Set	*goodbs* test set	*bs* test set
Diagonal Covar. Matrix	1.55	2.34	4.57
Full Covar. Matrix	1.60	1.81	3.58
Mixture of local models	1.20	1.90	4.10

Table 4.1: Percentage of images incorrectly classified by the system with no rejections.

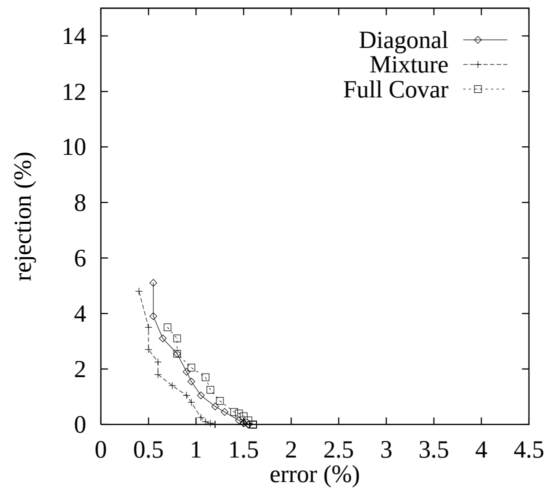
It is also interesting to compare the errors made using the three different deformation energy measures. On the *bs* test set, there were 124 errors when using the diagonal covariance matrix. 67 of these errors were the same as the ones made by the system with the full covariance matrix, and 97 were the same as the mixture model. This is not too surprising, as the mixture model is only a different way of evaluating the same final position as the diagonal covariance matrix, whereas with the full covariance matrix the whole settling trajectory is different. Classifiers that tend to make different kinds of error can be combined to give a superior overall performance if extra training data is available to determine how the individual classifiers should be weighted in the combination. Some methods for combining classifiers are discussed in Xu *et al* (1992) and Lee and Srihari (1993).

4.2 The black and white generative model

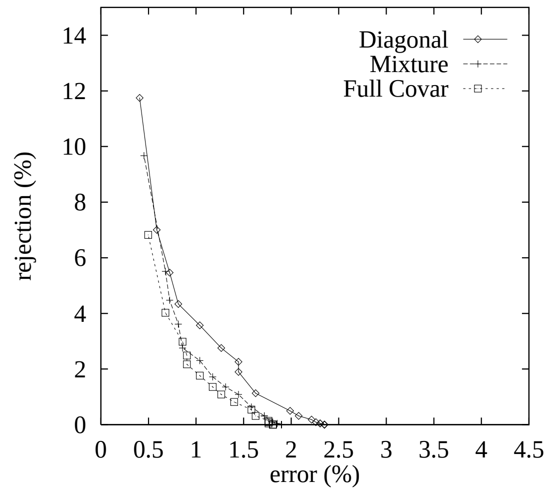
A significant drawback of the mixture of Gaussians imaging model is that it does not treat the white pixels as evidence. It maximizes the likelihood of generating the inked pixels, but it does not pay a sufficiently severe penalty for assigning high probabilities of ink to white pixels. As a result, a model can fit the data well even if some of its Gaussian beads are a long way from the nearest inked pixel. We have therefore formulated an alternative generative model in which the white pixels must also be generated.

We assume that the image is generated from the spline by a two-stage stochastic process. The first stage computes the probability $\hat{P}(i)$ that each pixel in the image would be inked if *multiple* samples were taken from the probability distribution defined by the Gaussian generators and the uniform noise process (see equation 3.4). We imagine taking N_B samples from this distribution, where N_B is the number of inked pixels in the image[3] and we compute the probability that none of these samples landed within a particular pixel. This gives the predicted probability that the pixel is not

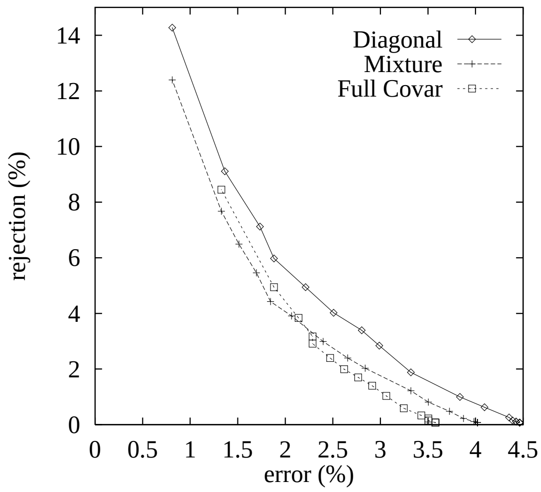
[3] To maximize the likelihood of generating the image we should, ideally, take more samples than there are inked pixels because several samples may fall on the same pixel. However, the penalty incurred by using the wrong number of samples is unlikely to affect the relative goodness of fit of different models.



(a)



(b)



(c)

Figure 4.6: Error-rejection relationships for (a) the validation set (b) *goodbs* test set (c) *bs* test set.

inked, and its complement $\hat{P}(i)$ is the predicted probability that the pixel is inked.

$$\hat{P}(i) = 1 - (1 - P(i))^{N_B} \tag{4.6}$$

where $P(i)$ is the probability that pixel i is inked under a single sample, obtained by integrating the probability density over the pixel. The second stage of the image generation process then uses these predicted probabilities to independently decide whether to ink each pixel. Given the predicted probabilities, $\hat{P}(i)$, the log probability of generating exactly the correct image is

$$\log P(I|\mathbf{z}, M) = \sum_{\substack{i \,\in\, inked \\ pixels}} \log \hat{P}(i) + \sum_{\substack{j \,\in\, uninked \\ pixels}} \log(1 - \hat{P}(j)) \tag{4.7}$$

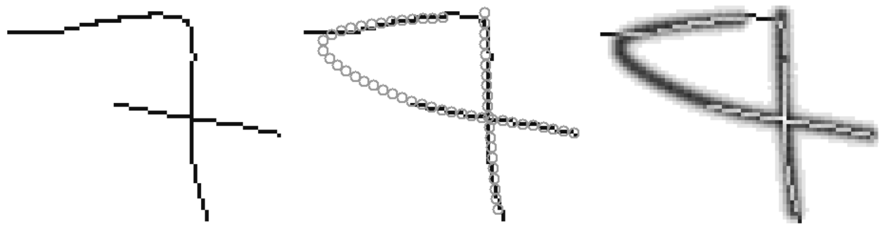


Figure 4.7: The black and white imaging model in action. The left panel shows an image of a 7. The fit of a four model using the mixture of Gaussians generative model is shown in the centre panel. Under this formulation the centre bar of the four model would not be penalized for lying in white space. The right most panel shows the image weighted by the probabilities of the full generative model with areas of low probability shaded dark. For display purposes, the noise model was set to zero when evaluating the full generative model so to prevent the background from turning grey.

This can also be viewed as the cost of encoding the image data using the predicted probabilities to do the encoding.

Run generatively, the model would tend to have some gaps in the contour, because the decision to ink every pixel is independent. Better coherence could be achieved by using a Markov Random Field (MRF) model. The inking probabilities for each pixel are then treated as biases to the MRF, but there is also a locally defined energy function that penalizes label assignments in which nearby pixels have different labels. It is relatively easy to generate images from the MRF using Monte Carlo Markov chain methods, but it is more difficult to determine the likelihood of the data under such a model, because of the difficulty in evaluating the normalizing factor.

When the Gaussians have high variance, each sample only has a very small probability of landing on a particular pixel, and the chance of two samples landing on the same pixel is negligible. So there is a simple approximation for the probability of inking a pixel; $\hat{P}(i) \simeq N_B P(i)$. Also, when the Gaussians are large, all of the pixels have a low predicted probability of being inked and in this case the log probability of generating the image given the predicted pixel probabilities is dominated by the cost of generating the inked pixels:

$$\log P(I|\mathbf{z}, M) \simeq \sum_{\substack{i \in inked \\ pixels}} \log P(i) + N_B \log N_B \tag{4.8}$$

Note that the term involving N_B will be the same for all models, and can therefore be ignored. At high variance the black and white generative model is almost the same as the mixture of Gaussians model. At low variance, the two generative models are very different in the way they penalize different fits. In particular, the black and white generative model makes it more expensive for parts of an instantiated digit model to lie in white space, as shown in figure 4.7.

An experiment was carried out to compare the performance of the two generative models, using the final configurations found with the regular settling procedure. All inputs to the neural net were the same for the two runs, except for the datafit energies which were determined by equation 4.7. The nets were trained on TR3 and tested on

VAL. The results show no improvement in performance from using the black and white image model, which is not very surprising because we are only evaluating the model on the final fit, rather than settling using the alternative image model. It is much more computationally expensive to settle using the second generative model because there are so many more white pixels than black in an image[4].

4.3 Style consistency

Conventional ZIP code recognition systems almost always segment the digit string into isolated digits, and then try to recognize each digit separately, ignoring the contextual information that may be present from the neighbouring digits. For example, we usually expect the size and slant of handwriting to be fairly consistent, and rounded or spidery styles persist whatever is being written. Specific examples of a digit that we have successfully recognized elsewhere in the ZIP code may help the decision regarding an ambiguous character. The instantiation parameters that are extracted when we fit the models to data allow us to go some way towards quantifying this style information.

The mutual information between two random variables measures the degree to which knowledge of one of the variables predicts (or reduces the uncertainty about) the other variable. For two variables X and Y the mutual information is defined to be

$$I(X;Y) = H(X) + H(Y) - H(X,Y) = H(X) - H(X|Y) = \sum_{x,y} P(X,Y) \log \frac{P(X,Y)}{P(X)P(Y)} \tag{4.9}$$

where $H(X)$ denotes the entropy of X. It would be possible to analyze correlations in both the deformation parameters of the models and variables like the size and slant parameters, although we have concentrated on the former. For example, we will be able to quantify how knowing that one of the digits in the ZIP code is a two with a large loop will change the predictions for the styles of other possible digits, compared to our default predictions based on statistics collected from a large set of images. It would be possible to estimate the mutual information in a digit pair under a Gaussian assumption by computing the covariance matrix of the deformation parameters (typically this would be a 32×32 matrix). However, with only a few hundred examples per pair this would be very poorly determined. Instead we have vector quantized this space using the mixture models presented in section 4.1.2, so that the style of each digit is selected out of ten possible alternatives, and have then collected statistics for the co-occurrences in a ten by ten contingency table for each digit pair. The mutual information under this representation can then be calculated directly from equation 4.9 for each digit pair.

The statistics were collected from over 3000 ZIP code images in the training and validation sets. Each five digit ZIP code yields ten digit pairings. For each digit pair we calculated the probability of the first digit under each of its ten local models, and multiplied these values by the corresponding probabilities for the paired digit, to obtain a set of one hundred numbers that sum to one in the appropriate table. Usually these

[4] This holds even though it is possible to economize on the calculations needed at high and low variances. At high variance, we can simply use the other generative model since it is almost equivalent. At low variance, each Gaussian bead only has a significant effect on a small number of white pixels, because the estimated probabilities for pixels far from any bead are dominated by the noise term. However, at intermediate variance the full computation is necessary.

	0	1	2	3	4	5	6	7	8	9
0	122*	44	57	71	85*	88*	69	66	77*	64
1		102*	69	74	68	95	64	78	59	85
2			522*	80*	128*	119*	115*	89	105*	100*
3				222*	131*	138*	78	93*	78	78
4					484*	128*	86*	106*	101*	132*
5						275*	89	85*	77	119*
6							332*	100*	88	107*
7								329*	100*	81*
8									379*	103*
9										225*

Table 4.2: The mutual information between digit pairs (measured in thousandths of a nat). The entries that are significant at the 99% confidence level are marked with a *.

probabilities are strongly peaked in one particular cell. These statistics are accumulated over all examples of co-occurring digits. The mutual information for each pairing is shown in table 4.2.

Having obtained these results, we have to ask if the computed value of the mutual information statistic is statistically significant, for each digit pair. The null hypothesis is that the mutual information in each distribution is really zero, but that sampling noise yields observations greater than zero[5]. A simple way to test for significance is to compare the observed values against the situation where the same number of digit pairs are drawn randomly from different ZIP codes, so that it is very unlikely that paired digits came from the same writer. By repeating this 100 times for each digit pair, a mean and standard deviation for the mutual information under the null hypothesis was obtained. The observed mutual information and null hypothesis (with error bars) for each pairing are shown in figure 4.8.

An alternative way to test the null hypothesis is to use the chi-squared test. Under the null hypothesis the probability p_{ij} in cell (i,j) is equal to the product of the marginal probabilities $p_{i.}p_{.j}$. Hence we calculate the statistic

$$\chi^2 = \sum_{i,j} \frac{(n_{ij} - n_{i.}n_{.j}/N)^2}{n_{i.}n_{.j}/N} \tag{4.10}$$

where n_{ij} is the count in cell (i,j), $n_{i.} = \sum_j n_{ij}$, $n_{.j} = \sum_i n_{ij}$ and N is the total number of observations. For a contingency table of size $r \times c$, the appropriate number of degrees of freedom for the chi-squared test equals $(r-1)(c-1)$, i.e. 81 d.o.f for our 10×10 tables. The results obtained with this method are similar to the randomized approach. It is also interesting to note that for small deviations from independence the mutual information and chi-squared statistic are approximately related; $I(X;Y) \simeq \chi^2/2N$ (Kullback, 1959).

[5]It is well known that the mutual information cannot be less than zero. Li (1990) has discussed how finite sample sizes affect the estimation of mutual information in contingency tables, and shows that the error is inversely proportional to the number of observations.

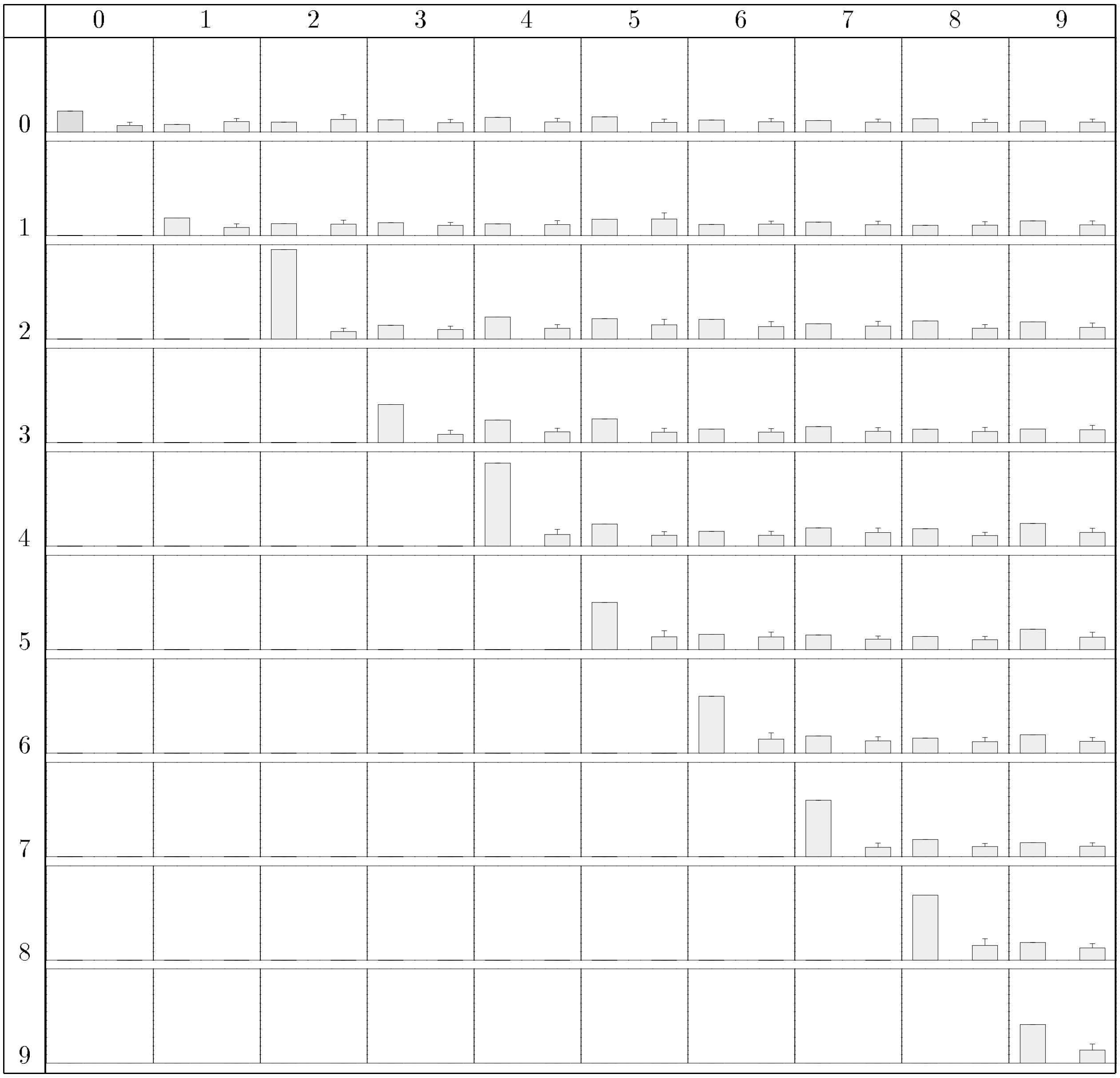


Figure 4.8: The observed mutual information and the null hypothesis result for each digit pair. In each cell the bar on the left shows the observed mutual information, and the bar on the right the mutual information under the null hypothesis. A 99% confidence error bar for the null hypothesis is also shown. The scale is the same for all cells.

The results in figure 4.8 are not too surprising; they indicate that generally the most reliable correlations are for pairings of a digit with itself. This makes sense, because style consistency implies that writers should tend to use the same local model for repeated digits in a ZIP code. For mixed pairings, some writers may group a particular style for digit A with another style for digit B, but the effect might be "washed out" by averaging over writers. It is interesting that for digits which have obvious style attributes (such as the size of loops in twos and sixes) the mutual information is larger than for digits which do not have much style variability, for example the one.

We have shown that significant style consistency information can be extracted from models that have been fitted to data. This could be used in a recognition system in a number of ways. For isolated digit recognition, the digits in a ZIP code would be recognized separately, and then the joint probability of the styles of the winning models could be checked. A very low joint probability would suggest inconsistent styles and a low confidence in the classification. Alternatively, if many models were being fitted simultaneously, there could be scale and style interactions between the different models as they settled.

Chapter 5

Predicting the instantiation parameters with neural networks

Two major problems with using deformable models for object recognition are that the search procedure is very computationally intensive, and that it can get stuck in local minima of the objective function. In the system as described in chapters 3 and 4, very little information from the image is used to choose the initial model instantiation parameters. In this chapter we discuss ways of "compiling down" some of the knowledge gained from fitting models to data to obtain better starting points for the search.

Section 5.1 reviews previous work on instantiating models. Section 5.2 reports results on using a neural network to predict the instantiation parameters of the digit models when using the whole image as input, and section 5.3 discusses experiments to predict corresponding points in the image and on the spline models using local patches of the image.

5.1 Previous work

Methods for predicting instantiation parameters can be broken down into two broad classes, correspondence based search and parameter space search. These are reviewed in the next two subsections.

5.1.1 Parameter space search

For rigid objects the parameters are the pose variables that transform the model into the image space; with deformable objects there are additional parameters corresponding to the deformational degrees of freedom.

For rigid 2D objects in binary images it is possible to use image moments to solve for the instantiation parameters. Consider the case where the image is a translated, scaled and rotated version of the reference object. Horn (1986) shows that the translation can be obtained from the centre of gravity, the scale from the ratio of the areas and that the rotation can be computed from the three second moments. Of course the moments are sensitive to noise features, which limits their use in realistic problems. It is not obvious how to extend this approach to deal with all of the parameters in our deformable models, although high-order moments would probably be required.

An alternative parameter space approach is the Hough transform, which was discussed in section 2.1.3. This has the advantage that it is robust to spurious image features. An interesting example of the use of Hough transform techniques to instantiate a deformable model is found in Yuille *et al* (1990). They were concerned with the detection of tracks in bubble chamber images. The tracks are circles, or (degenerately) straight lines. A Hough transform was used to obtain rough estimates of the number and approximate location of the tracks, and a gradient based search was then used to optimize the instantiation parameters. However, Hough transform techniques are impractical in high dimensional spaces, such as the 22-d instantiation parameter space which most of the deformable digit models inhabit. In section 5.2 we demonstrate that a feedforward network which takes an image as input and outputs predictions for the instantiation parameters produces good results on the CEDAR CDROM data.

5.1.2 Correspondence based search

The alternative to parameter space search is correspondence based search. The idea is to extract features from the image and consider all possible correspondences between them and similar features in the model. For example, with polyhedral objects the edges between the faces will give rise to line segment features in the image. Correspondence space matching techniques can be set up as a search problem in an interpretation tree (Grimson, 1990), which enumerates all possible matchings of the data to the model features. Given a number of features in the model and the image, we search for correspondences between the two sets that are consistent with the overall constraints of the model. At each level in the tree the match of an image feature to all possible model features is considered. Because there may be features from other objects in the image, it is necessary to add a "wild-card" feature to the set of model features at each level in the tree. The tree is traversed in a depth first search manner, using unary and binary constraints on the features to prune away subtrees. Finally, a pose is calculated based on the matched features.

Clearly, there is much to be gained by structuring the search; an example of this is seen in the Local Feature Focus method of Bolles and Cain (1982). A particular type of (focus) feature is searched for at first, and once it has been found the search then looks for features in the image that should be close to the focus feature if the correct model has been selected.

It is possible to combine correspondence based and parameter space search. We first search for correspondences until enough potential matches have been found to determine the pose of the object, and then project the model into the image space for verification. This is the basis of Ullman's (1989) technique, "aligning pictorial descriptions" (also known as object recognition by alignment). He uses three point correspondences to solve for the (orthographic) projection parameters of rigid objects.

In Grimson's example of matching polyhedral objects using edge features, every model feature can potentially match every image feature. For our character recognition problem, every inked pixel in the image could correspond to any point on the spline. The key idea that can improve the search is the *grouping* of low-level image elements into higher level features. By finding more complex features in the image the number of possible matches to model features in the interpretation tree is reduced, speeding up the search. The difficulty is, of course, that reliably detecting higher level features is usually more difficult and computationally intensive. The Gestalt psychologists emphasized the importance of grouping principles such as proximity, good continuation and closure (Wertheimer, 1923). There are two sub-problems associated with creating higher level features; the selection of which lower level features to group, and the construction of an index vector (or descriptor) from a given grouping in order to identify the corresponding part of the appropriate model. Useful groupings will have *non-accidental* structure (Witkin and Tenenbaum, 1983; see also Jepson and Richards, 1993), i.e. they should not often arise by chance when several objects are juxtaposed, and are thus indicative of a single object.

An example which illustrates both of these aspects is to be found in the work of Beis and Lowe (1993). In their system, image edges of manufactured objects are grouped (using the proximity principle) to create four-segment chains. The three angles and the ratio of the interior edge lengths are then used to create a four-dimensional index vector, which is invariant to translations, scalings and image-plane rotations. Two degrees of freedom remain for each chain, corresponding to the position on the viewing sphere. In a training phase, a large number of index vectors are created for each possible chain in the database of objects, and used to train a radial basis function classifier. At run time, the network takes an index vector as input, and outputs a probability distribution over the possible chains.

Other examples of systems that have used grouping of low level features are due to Lowe (1985), Lamdan and Wolfson (1988) and Zemel *et al* (1990). Lowe exploited collinearity and parallelism in images of objects with rectangular faces to construct higher level features, which were then used to index into corresponding features in the model(s). Lamdan and Wolfson's geometric hashing technique constructs an invariant index vector using groupings of point features (from a planar object), which is then decoded using a hash table. Zemel *et al* built a connectionist model (TRAFFIC) which is based on the idea that object recognition should be hierarchical; first of all, local groupings are created that correspond to parts of an object, and then these parts are themselves grouped into larger structures, until the object level is reached. The implementation acts on data made up of (noisy) patterns of constellations of stars which can be arbitrarily scaled, rotated or translated. The key idea is that the predictions of the lower level parts must agree in order for the higher level object to be instantiated.

Most of the work on correspondence based search has been applied to manufactured objects which have corners and edges. For other kinds of object such as handwritten characters, the correspondence based method runs into difficulty when trying to find (by hand) the relevant groupings of pixel level features needed to produce higher level features; even if the trajectory of the pen could be obtained from an image, breaking it up into useful parts is a difficult problem. In section 5.3 we discuss an approach where a local window of the image surrounding a black pixel is used as a higher level feature; a neural network is used to map from this input to a position (or set of positions) on the spline.

5.2 Hypothesizing in parameter space

The limitations of existing techniques for parameter space prediction for deformable models have been discussed in section 5.1.1. Our approach is to feed the image through a neural network and output a prediction of the instantiation parameters. This can be regarded as a minimal way of providing some information on the posterior probability density $P(\mathbf{z}|I, M)$ by just specifying the strongest peak in the distribution. Zemel and Hinton (1991) used this idea with simple 2-d objects that undergo rotations, scalings and translations (in an unsupervised learning framework). More recently, Beymer *et al* (1993) have demonstrated a similar scheme, mapping from a face image to a 2-d parameter space spanning head rotations and a smile/no-smile dimension. However, their method does not directly map from images to instantiation parameters; they use a computer vision correspondence algorithm to determine the displacement field of pixels in a novel image relative to a reference image, and then use this field as the input to the network. This step limits the use of the approach to images that are sufficiently similar so that the correspondence algorithm functions well.

We can obtain a set of training pairs (image, instantiation parameters) from the usual (slow) search procedure. A number of issues immediately arise with this scheme, to do with the representation of the input and the output. The images of digits typically vary in size (i.e the number of pixels) quite considerably, so they cannot be directly fitted into a fixed width input vector. The standard solutions are either (i) to normalize the image, or (ii) to extract a fixed number of features from the image. In general, both of these options are dangerous, unless we are certain that the image contains only one object which fills the frame well, i.e. that the image is well-segmented. On the whole, the ZIP code digit data that we are using does meet these requirements, which means that normalized images can be used.

There are also many issues concerning the representation of the outputs. Minimally we need to be able to represent one set of instantiation parameters for each model. For our models, this might mean outputting control point deformations plus affine parameters, or alternatively the control point locations in the normalized image; the latter alternative was used.

The network which was used to predict the model instantiation parameters is shown in figure 5.1. The binary images were normalized to give 16×16 8-bit greyscale images which were fed into the neural network. As most of the original images were larger than 16×16, this provided some smoothing of the inputs to the net, which was probably beneficial to the performance of the net. The network uses a standard three layer architecture; each hidden unit i computes a weighted sum $y_i = \mathbf{w}_i.\mathbf{x} + b_i$ of its inputs $\mathbf{x}$ and its weight vector $\mathbf{w}_i$ plus a bias b_i, and then feeds this value through a sigmoidal nonlinearity $a_i = \sigma(y_i) = 1/(1 + e^{-y_i})$. The output values are a weighted linear combination of the hidden unit activities plus (output) biases.

The network was trained with backpropagation to minimize the squared prediction error, using 900 training images (datasets TR1 and TR2) and 200 validation images (dataset TR3) of each digit. There are 144 output units as eight of the models have eight control points and hence 16 outputs each, the one model has three control points and the seven model has five. Using twenty hidden units means that there are 2880 hidden to output weights, 5120 input to hidden weights and 164 biases. With this large number of weights it is important to regularize the solution obtained by the network by using a complexity penalty; we used a weight decay penalty $\lambda \sum_j w_j^2$ with λ set to 1.0 . Targets were only set for the correct digit at the output layer; nothing was backpropagated from the other output units. Networks with ten and twenty hidden

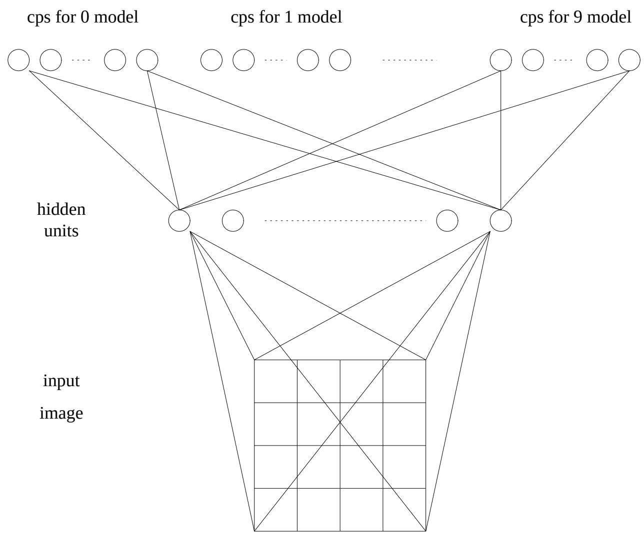


Figure 5.1: The prediction network architecture. “cps” stands for control points.

units were tried, and gave similar performance. It would be possible to construct ten separate networks to carry out the same task as the net described above, but this would increase the amount of computation required when the network is used and intensify the danger of overfitting. We protect against this by giving the network a common pool of hidden units which it can use as it decides appropriate. The results reported below are for the twenty hidden unit network, which we will call the prediction net. It took 440 epochs to train using the default conjugate gradient search method in the Xerion neural network simulator[1]. Training stopped when the line search failed; the validation error decreased consistently during training.

For comparison, a trivial network which just consisted of output biases was trained; this network simply learns the average value of the instantiation parameters. On the validation set the squared error of the prediction net was over three times smaller than the trivial net. Although this is encouraging, the acid test is to evaluate the performance of deformable models settled from the predicted positions using a shortened annealing schedule; if the predictions are good, then only a short amount of settling will be required.

The feedforward net predicts the position of the control points in the normalized image. By inverting the normalization process, the positions of the control points in the un-normalized image are determined. The deformation and affine transformation corresponding to these image control point locations is determined by running the second stage of the M step (see section 3.3). Experiments were then conducted with a number of shortened annealing schedules; for each one, data obtained from settling on TR3 was used to train the postprocessing net. The performance was then evaluated on the VAL set, using the mixture of local models to calculate the deformation energies.

The full annealing schedule has seven stages; the initial bead variance is 0.04, and

[1]Xerion was designed and implemented by Drew van Camp, Tony Plate and Geoffrey Hinton at the University of Toronto. In one training epoch all of the training patterns are presented once to the network, and the weights are then updated.

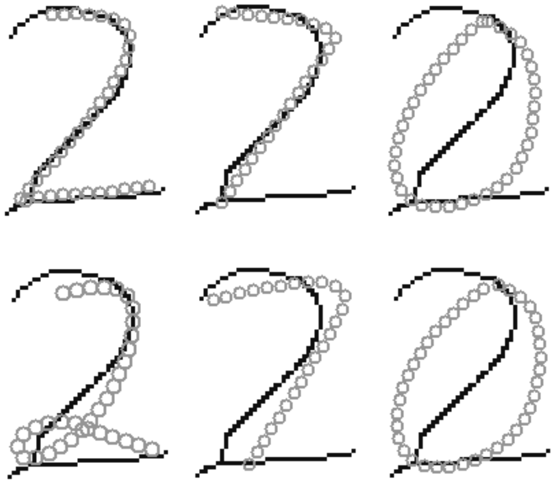


Figure 5.2: A comparison of the initial instantiations for the 2, 7 and 0 models due to the prediction net (top row) and the trivial net (bottom row) on an image of a 2. Notice that for the two model the prediction net is much closer to the data. The other digit models may or may not be helped by looking at the input data; for example, the predictions of both nets are essentially the same for the zero, but for the seven the prediction net puts the model nearer to the data.

at the end of the schedule it is reduced to 0.0006 (see section 3.3.3 for details). The shortened annealing schedules are:

1. No settling at all
2. Two iterations at the final variance of 0.0006
3. One iteration at 0.0025 and two at 0.0006
4. The full annealing schedule (for comparison)

The results on the VAL test set are shown in table 5.1. The general trends are that the performance obtained using the prediction net is consistently better than the trivial net, and that longer annealing schedules lead to better performance. Figure 5.2 compares the outputs of the prediction and trivial nets on a particular example.

A run on the *bs* test set using schedule 3 gave an error rate of 4.76 % (129 errors), which is very similar to the 122 errors obtained using the full annealing schedule[2]. The settling process is also significantly shorter when using the prediction net; a comparison of the timings for schedules 3 and 4 in table 5.1 shows that schedule 3 is more than a factor of two faster than the full annealing schedule. Using the *bs* test set a comparison of the errors made by the prediction net with the short annealing schedule as compared to the full settling shows that only 72 out of the 129 errors were common to the two sets. This suggests that it would be possible to reject cases where the two methods do not agree.

[2]The experiments reported in this section used data from a slightly earlier version of the system than the one that is reported in chapters 3 and 4.

Schedule number	Trivial net	Prediction net	Average time required to settle one model (s)
1	427	200	0.12
2	329	58	0.25
3	160	32	0.49
4	40	36	1.11

Table 5.1: Errors on the VAL test set of 2000 examples for different annealing schedules. The timing trials were carried out on a R-4400 machine.

It is not easy to discover how the net is able to obtain the improved performance. Judging from the weight vectors of the hidden units, it does not seem that some units are specialized for some particular digit class. If the net approximately memorized the appropriate set of patterns (and their corresponding outputs) for a particular class, then it might be expected to give the average (trivial net) output when the input pattern does not match. However, observations comparing the average sum-squared distance between the control point locations predicted by the trivial and prediction nets for inputs of the correct and incorrect classes showed that this was not the case.

5.2.1 Discussion

The scheme used above can be viewed as an interpolation scheme in the control point position (CPP) space of each digit,

$$\mathbf{z}(I) = \mathbf{z}_0 + \sum_i a_i(I)\mathbf{z}_i \tag{5.1}$$

where $\mathbf{z}(I)$ is the predicted position in the control point space, $\mathbf{z}_0$ is the contribution due to the biases, a_i is the activity of hidden unit i and $\mathbf{z}_i$ is its location in the control point position space (learned from the data). Exactly the same equation would apply to a Radial Basis Function (RBF) network (as introduced by Broomhead and Lowe (1988) and further extended by Poggio and Girosi (1990)), except in this case the hidden unit activities are computed as a Gaussian function of the distance between the input pattern and a centre *in the input space*. For both cases it is important to note that the $\mathbf{z}_i$'s are *learned* from the training data. In some other function approximation schemes (e.g. kernel regression, see Härdle, 1990), each $\mathbf{z}_k$ is taken to be the instantiation parameter vector for example k. In kernel regression the (non-negative) activities are normalized to sum to one, so the output prediction $\mathbf{z}(I)$ cannot lie outside the convex hull of the $\mathbf{z}_k$'s. With reasonably small training datasets (in the order of 1000 examples), this may be quite a serious limitation.[3] It may be possible to overcome these convex

[3] I did some experiments using an explicit convex-hull construction algorithm, the beneath-beyond method (Edelsbrunner, 1987). This algorithm scales as $O(N^{\lfloor d/2 \rfloor + 1})$, where N is the number of points and d is the dimension of the space. A convex hull was constructed using 900 training examples of data from the two model. Because of the way the complexity of the algorithm scales, it was only possible to construct the hull in 6-d, using the first six values of each 16 dimensional vector. However, as any point that lies outside of the convex hull in the lower dimensional space will also lie outside the hull in

hull limitations by using activation functions that permit negative activities (such as a difference of Gaussians model). A kernel regression scheme could be used to predict the instantiation parameters, using a separate net for each model. If all 900 training cases were used for each class, this net would run about 450 times slower than the prediction net.

If there are more hidden units than output dimensions, then for any particular image I there are an infinite number of ways to make equation 5.1 hold exactly. However, the network will tend to find solutions so that the $a_i(I)$'s will vary smoothly as the image is perturbed.

The method described above gives a one-step calculation for $\mathbf{z}(I)$ based on the activities and the $\mathbf{z}_k$'s. However, this is not the only possible scheme; for example Zemel (1993) has used an iterative scheme which (in our terms) involves matching a Gaussian bump in the CPP space. Under this model an instantiated object should produce a Gaussian bump of activity in the CPP space. Given a pattern of activity, a Gaussian bump is fitted by minimizing the discrepancy between the actual and predicted activities at each unit, using a gradient-based minimization routine. Experiments have shown this scheme can successfully represent points that lie outside of the convex hull of the basis points. However, from a network training view, this scheme is a lot more complex than the simple one-step procedure, as it is more difficult to back-propagate through.

The scheme described above for the digit data produces just one set of instantiation parameters for a given model. However, it may be preferable to be able to represent a number of guesses about model instantiation parameters; one way of doing this is to train a network that has multiple sets of output parameters, as in the "mixture of experts" architecture of Jacobs *et al* (1991) (see also Bishop, 1994). The outputs can be interpreted as a mixture distribution in the CPP space, conditioned on the input image. A disadvantage of this technique is that the number of mixture components must be pre-specified, or estimated from the data; this is a difficult problem. However, as each output also has an adaptive importance associated with it (its mixing proportion), in theory providing sufficient components and allowing the network to learn which ones to use should be an effective strategy. In practice, limited amounts of training data and local minima in the search may make this less attractive.

The mixture of experts approach models the posterior distribution with a fixed number of Gaussians whose means, variances and mixing proportions are adaptive. An alternative approach is to use a fixed set of basis functions $\{\phi_i\}$, and to approximate the distribution as $f(\mathbf{z}|I) = \sum_i a_i(I)\phi_i(\mathbf{z})$ (Hinton, Williams and Revow, 1992b). For example, the Hough transform uses indicator functions of the bins as basis functions[4], although it would clearly be possible to use other classes of functions, e.g. Gaussians. When using fixed basis functions, a further step is necessary in order to decode the pattern of activities; we have to go "bump hunting" in the instantiation parameter space to find the (local) maxima of the posterior density. Note that in this scheme there is no pre-imposed limit on the number of peaks (components) that can be detected; this

the higher dimensional space, this procedure will give a lower bound on the number of test points that lie outside of the convex hull. 62 out of 200 test points lay outside of the 6-d convex hull, indicating that a technique like kernel regression could have quite serious problems.

[4]In the one dimensional case this leads to a histogram as an approximation of the density function.

is determined by the data and the density of basis functions. As the prediction net hypothesizes just one point, it can combine the responses from units from across the whole of CPP space to produce its answer. However, with a "bump hunting" scheme, there must be more locality, so that clumps of units will "light up" to represent various possible instantiations. There are many issues involved with the details of this kind of scheme, concerning the density of units in the space (which in turn affects the minimum resolvable separation of two instantiations), and the method used to decode the activity patterns.

There is also an issue of whether there should be some representation of the degree of belief in a particular model given the input image, i.e. something like a probability of the model given the data. As the aim of the prediction network is to produce an hypothesis for each model which can be tested by the generative models, this information is not essential, but may be useful to reduce the amount of computation performed in the testing stage, particularly if there is a large database of models. The degree of belief might be represented by a special unit associated with each model (as in TRAFFIC, Zemel *et al*, 1990), or be encoded in the overall level of activity over a set of population code units.

5.3 Hypothesizing correspondences

This section describes some preliminary experiments which were designed to find out how easy it is to predict an inked pixel's position on the spline using a neural net, given a local window of context around it[5]. This would facilitate correspondence based search techniques, and would be especially useful if well-segmented images (as used in section 5.2) are not available. In essence, when poor segmentation prevents us from doing recognition on the whole image, we turn to pattern recognition "in the small", and look for parts of the overall pattern. Given sufficient matches, it is then possible to find the vector of instantiation parameters that minimizes the deformation energy of the model.

An obvious difficulty for the approach is that the digit data has large variations in overall scale, and to a lesser extent rotations, which confound the prediction task. This was partially addressed by rescaling all images to a standard height 32 pixels on the y axis[6]. We did not build in any invariance to rotation—in fact, for ZIP code recognition, there is a strong expectation that the characters will be roughly upright (otherwise it would be difficult to distinguish 6's and 9's), so complete rotation invariance is not appropriate. However, approximate invariance to small rotations would be desirable.

The experiments were conducted with isolated digit data from a single digit class. The experiments were run (separately) on data from twos and threes. A more general task is to also predict the model to which the data belongs. Training and validation data was obtained by taking thinned binary images to which the deformable models had been fitted. The input data was created by centering the window on each pixel in

[5] This task should be easier for on-line data, where the temporal ordering means that the context is basically one-dimensional, instead of two-dimensional.

[6] Height normalization is a plausible operation on a raw ZIP code that contains all five digits. Using a different scaling factor on the x dimension is not really possible without segmentation, which we did not want to assume.

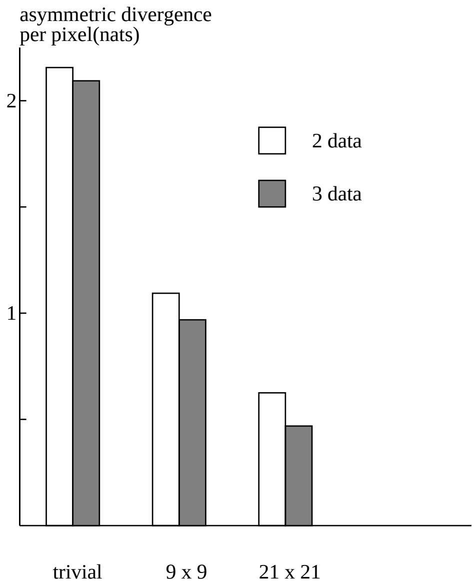


Figure 5.3: A comparison of the prediction performance of the trivial, 9×9 and 21×21 nets.

turn. If the window overlapped the edge of the image it was assumed that the data beyond the edge consisted of uninked pixels. Windows of size 9×9 and 21×21 were used in the experiments. The target for each example is the corresponding position on the spline. I used a position representation consisting of twenty "bins" spaced equally along the spline, plus one bin to represent an assignment to the noise process. The target values were calculated from the responsibility of each bead for the pixel in question, evaluated at the final bead variance of 0.0006. Hence the target is a 21-d vector which sums to one. This representation is useful as it allows the net to predict correspondences at more than just one point on the spline, and avoids the averaging problem, where the average of two good predictions is a bad prediction. Each data set contained approximately 7000 patterns drawn from around 100 images.

To learn the input/output mapping, one hidden layer neural networks with a "soft-max" output layer were trained, using a variety of different numbers of hidden units and weight decay factors. The error measure used was the Kullback-Liebler asymmetric divergence between the predictions of the network and the target values[7]. Results on the validation data from the best prediction networks are shown in figure 5.3, for networks using 0×0 (trivial), 9×9 and 21×21 windows (for the trivial case the net simply learns to set the bias weights to predict the average probabilities of each output class). The best non-trivial nets that were found had 10 hidden units, and were trained with a weight penalty of 1.0 .

The results obtained with the local windows of context are encouraging; the 21×21 net shows roughly a factor of four improvement over the trivial net. It is not particularly surprising that the best results were obtained with the larger windows; however, it is desirable to use as small a window as possible as this will give greater robustness

[7] As the targets have non-zero entropy, the asymmetric divergence differs from the cross entropy by the entropy of the targets, but it has the same derivatives with respect to the weights.

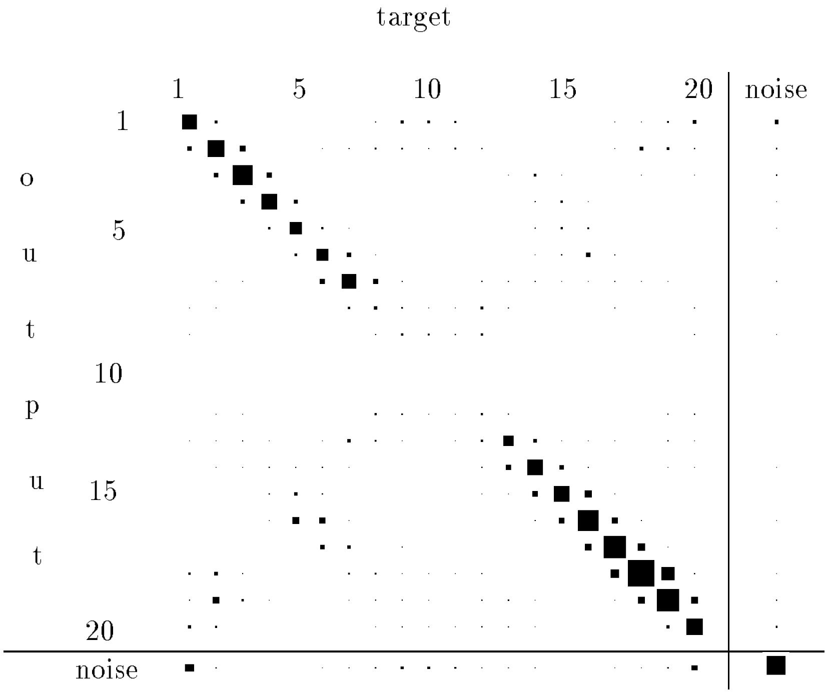


Figure 5.4: A two-dimensional histogram illustrating the performance of the 9×9 net on three data. The target bins run from left to right, the output bins from top to bottom. Bins 1 through 20 correspond to points on the spline; bin 21 is for the noise process. The linear size of each square is proportional to the number of votes for that cell. See the text for more discussion of the plot.

to extraneous information in the image. When using a large window, the input also contains some information on the position of the pixel; for example, the data for pixels near to the top of the image will have a broad uninked strip at the top. The network can pick up on this regularity to help with its predictions.

Although the overall level of performance of the networks is encouraging, the real test of the system comes by evaluating how good the predictions are on a case-by-case basis. The targets usually consist of a sharp single spike centered on the correct location, although it is possible to obtain a more complex pattern, e.g. when the spline crosses itself, as in a "loopy" two. The patterns output from the network tend to be rather fuzzier, and sometimes multimodal. In order to illustrate the performance of the system, each target or output vector has been labelled according to which of the 21 bins was the most active. Figure 5.4 is a two-dimensional histogram showing the 21 target bins running from left to right, and the output bins running from top to bottom; it shows the performance of the network using a 9×9 input window on the validation set of 3 data. The linear size of the square in bin (t, o) represents the number of validation examples for which an example from class t was classified as class o. Perfect performance would concentrate all of the counts on the diagonal. The structure of the off-diagonal entries is interesting—there is a confusion between the vertical sections in

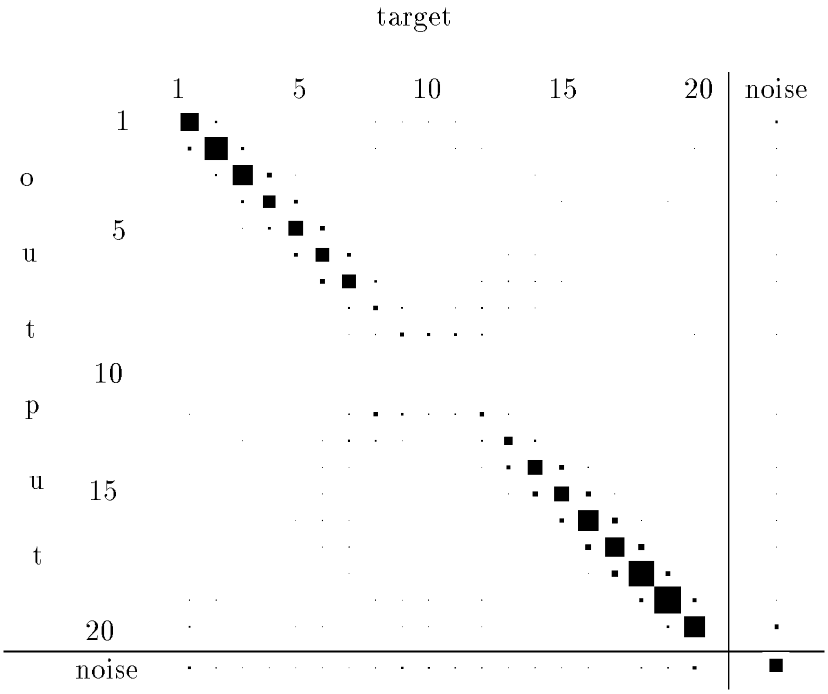


Figure 5.5: A two-dimensional histogram illustrating the performance of the 21×21 net on three data. The target bins run from left to right, the output bins from top to bottom. Bins 1 through 20 correspond to points on the spline; bin 21 is for the noise process. The linear size of each square is proportional to the number of votes for that cell.

the top and bottom loops of the 3, and also between the horizontal strokes that appear at the top, middle or bottom of the 3. The network which used the 21×21 window produces a figure with more weight on the diagonal, as can be seen from figure 5.5. Similar results were obtained for the two data.

5.3.1 Discussion

The two-dimensional histograms show that most of the predictions fall close to the targets. However, sometimes the errors are much larger, which will lead to large errors when solving for the instantiation parameters. In future work I would like to investigate the use of a threshold to create a reject criterion, and find out how this increases the reliability of the predictions. As some parts of a character are more predictable than others, it may make sense to have different thresholds at different bins.

The decoding method used above assumes that each input should correspond to only one point on the spline, but we do know that there are cases where the spline crosses itself. So another possible classifier would use templates in the output space, some of which could be multimodal.

As can be seen from figure 5.4, the neural network will sometimes output multiple hypotheses because the input is insufficiently discriminating. However, information of this nature can be used during the settling process. In the system described in chapters 3 and 4 the probability of pixel i given generator j depends only on their separation, i.e. $P(p_i|g_j) = P_d(|\mathbf{y}_i - \mathbf{y}_j|)$. We can augment this by including information from a local window or neighbourhood of context $N(p_i)$, and write[8]

$$P(p_i|g_j) = P_d(|\mathbf{y}_i - \mathbf{y}_j|)P(N(p_i)|g_j) \tag{5.2}$$

The additional factor "gates" the interaction of pixel i and generator j. The neural network estimates $P(g_j|N(p_i))$, and we can then use Bayes' theorem to obtain

$$P(N(p_i)|g_j) = \frac{P(g_j|N(p_i))P(N(p_i))}{P(g_j)} \tag{5.3}$$

$P(g_j)$ is easily estimated from a training set of models settled to data (it is just the relative frequency of each bead), but $P(N(p_i))$ presents more difficulty. However, for the settling process we require only the "responsibility" $r_j^i = P(p_i|g_j)/\sum_k P(p_i|g_k)$ and the $P(N(p_i))$ factor cancels out. Preliminary experiments that show that the additional gating factors can be useful, for instance when settling a model from a poor starting position which is far from the data.

There is some concern that the predictions made by the network will just reflect the orientation of the stroke at each point of the spline, based on the statistics in the training dataset. This would mean that the predictions for a considerably rotated digit would have a systematic error. One way to reduce this problem is to create synthetic training data, by manipulating the existing training images. For instance, images could be rotated through a range of angles, to increase the robustness of the system. Another related approach would be to incorporate "tangent-prop" (Simard *et al*, 1992) into the training of the networks. Under this scheme we calculate the derivative of the output vector with respect to a rotation of the image, and demand that this change is small.

Instead of trying to predict position on the spline from local context, a network could be trained to predict the orientation of the spline. The gating term could then be something like $P_\theta(\theta_i - \theta_j) = \frac{1}{(2\pi\sigma_\theta^2)^{1/2}} \exp(-\frac{sin^2(\theta_i - \theta_j)}{2\sigma_\theta^2})$, where θ_i is the orientation predicted at pixel i and θ_j is the orientation of the spline at g_j. This is simply a Gaussian based on the sine of the difference of the angles. Run generatively, this describes a model which produces an image consisting of points with an orientation attribute, given an instantiated spline. The prediction of orientation could also be carried out by more standard image-processing techniques. For example, we could convolve the image with Gabor filters at many different orientations and locations, and look for intensity-edge or bar signatures in the phase map (Wang, 1991). Preliminary results indicate that this is effective and can be applied to greyscale images, although it is necessary to roughly match the filter period to the stroke thickness[9].

As we have used isolated digit data, there is also some concern that the network may be able to use the bordering white space to help it make predictions. It would be interesting to use noiser data to explore this possibility. Again, the data could be

[8]The incorporation of the window of context means that the generative model is less "clean", as there are interactions between pixels. We would need to use Markov chain Monte Carlo methods to sample from the distribution over images. However, it may be a useful way of overcoming some of the deficiencies of the mixture of Gaussians model.

[9]I thank Prof. Allan Jepson for assistance with this work.

produced synthetically; in this case images of digits could be overlapped to produce touching or nearly-touching data[10]. A further extension of this idea is to synthetically produce images from our generative models to use as training data; this avoids the slow step of fitting models to the images, and could therefore provide much more training data. However, it may lead to poor performance on real data if the generative model is deficient.

[10] Keeler *et al* (1991) have also discussed creating synthetic images in this fashion.

Chapter 6

General Discussion

In section 6.1 I contrast the approach to object recognition developed in this thesis with related work. Section 6.2 summarizes the content of the thesis and outlines the main contributions. Section 6.3 describes some possible directions for future work.

6.1 Related work

Most of the other work on deformable models has focussed on localizing a given object in an image, i.e. calculating the instantiation parameters, rather than addressing classification problems. For example, Grenander *et al* (1991) have located hand shapes in noisy images and Cootes and Taylor (1992) have located resistors on images of printed-circuit boards. However, more recently there has been some work done using deformable models for object recognition, for example by Lades *et al* (1993) and Prof. Taylor's group at the University of Manchester (Lanitis, Taylor and Cootes, 1993). Lades *et al* match a template image to a test image by allowing the template to undergo "rubber sheet" deformations. However, the work of Lanitis *et al* is closer to our own, as it aims to characterize the underlying variability of the objects, rather than using generic assumptions. They have applied their methods to the recognition of plant seeds, handwritten characters and human faces. In the rest of this section I describe their work and compare it to that described in this thesis.

The Manchester group use Point Distribution Models to characterize the shape variability of objects. Contours are defined by linear splines running between the control points. The models are trained by hand fitting each one to appropriate data, after the images have been normalized to remove variability due to scale, rotation and translation. The mean and covariance matrix of the vector of points is then calculated, and a principal components analysis is used to determine the significant modes of variation. Typically three or four modes are used. Lanitis *et al* report using relatively small training sets of up to twenty-five examples per model. The number of control points per model ranged from twenty (for character recognition) up to 144 (for faces).

Their imaging models do not have a generative interpretation, i.e. they do not yield $P(I|\mathbf{z}, M)$, but use more heuristic measures of agreement. However, for the character recognition task they did specifically have to use an explanation-type measure to avoid favouring models that only explain a subset of the data. The plant seed and character recognition examples use binary data, but the face classifier work was done on grey-level images. In this case the datafit measure was defined as the Mahalanobis distance between a set of points on the model and the corresponding points in the image.

The search procedure used by the Manchester group is based on "profiles", which are perpendicular line segments constructed at each of the points on the contour. For each profile a new target position for the point is selected by assessing the quality of the match at a number of locations along the profile and picking the best one. Changes to the global transformation and deformations are then computed to move all of the points as near as possible to their targets. Notice that this procedure does not compute the gradient of the datafit and then use steepest descent (or some more sophisticated technique), but takes finite sized steps. However, unlike with the EM algorithm, there is no guarantee that the datafit has to be improved by the step taken.

Their system has the same overall feel as ours, in that it builds models to characterize the variability of objects and fits these models to novel images. However, the framework we have used is well founded in signal-detection theory, while theirs is more heuristic, particularly with respect to imaging models. On the other hand, they have been able to work with grey-level images, while our work has dealt only with binary images. The lack of an objective function means that their search procedure is heuristic, although it seems to perform well in practice.

The hand fitting of the models will tend to reduce the amount of training data that can be processed, which may mean that some modes of variation are missed. However, using some manually-fitted data to initialize an estimate of the mean and covariance matrix would be a sensible idea for a system that then uses an automatic procedure to refine them. An advantage of hand fitting is that it should eliminate cases where bad training data is obtained because of a local minimum in the search[1].

6.2 Summary

I have described a generative model approach to handwritten digit recognition. The major motivation for building this system was to investigate the use of stochastic generative models for object recognition in a realistic domain. We have achieved recognition rates comparable to the state of the art recognizers, and have demonstrated some of the advantages of a model based approach such as the quantification of style consistency and model-driven segmentation. These results have been obtained on images that vary in size from many hundreds to tens of thousands of pixels, in contrast to the normalized images which most character recognition systems use. The ability to scale the model rather than the image is crucial for tasks where multiple, overlapping objects are to be recognized.

The generative spline models are compact, being specified by relatively few parameters, even when using a full covariance matrix Gaussian or a mixture model for the control points. I have shown that they can capture most common kinds of variation, although they cannot easily model large embellishments to the basic shape (some ideas concerning this problem are discussed in section 6.3). I have also shown that the spline

[1]When using automatic fitting, we can avoid the problem of local minima by inspecting the data after fitting and rejecting or correcting bad fits. As the models usually fit well, this will be much less work than hand fitting all of the examples.

models can be refined automatically, based on the fit of a model to the data in its class.

A good search procedure is crucial for the use of deformable models for object recognition. I have shown how to adapt the efficient EM algorithm for this problem, thereby making the search much faster than simple gradient-following methods. Nevertheless, the full search procedure still takes about one second per digit on a fast computer, which is too slow to be commercially applicable for a task like ZIP code recognition. I have investigated the use of a neural network to predict the instantiation parameters for the models given an image, and have demonstrated that a significant speed-up can be obtained without sacrificing performance. Using machine learning techniques like neural networks to help reduce the amount of search required to fit complex models to data may be useful for many other problems. The search procedure usually manages to avoid local minima in the parameter space, but occasionally (on about 1% of cases) it becomes trapped; it may be possible to improve this figure by using neural networks to predict correspondences, as discussed in section 5.3.

6.2.1 Main contributions

In this thesis I have constructed a system that can be used to recognize handwritten digits. In doing so I have had to deal both with specific problems relating to the domain and more general issues around the use of stochastic generative models for object recognition, and have:

- Designed spline models of the digits and an imaging model which are appropriate and give good performance on the task.
- Demonstrated an efficient search algorithm which generalizes the usual EM algorithm for fitting mixtures of Gaussians, by dealing with the affine transformation from object-based to image-based frame and constraints on the Gaussians due to the spline.
- Analyzed the behaviour of the search algorithm at high variance.
- Shown that the spline models can be improved by a training procedure which does not require hand-fitting the models to data.
- Shown how deficiencies in the generative models can be mitigated by using extra measures of the quality of fit which are then fed to a post-processing neural network.
- Used mixture models to capture writing style variations and demonstrated that style consistency can be detected in ZIP codes.
- Shown how a neural network can be used to speed up the search process by finding better starting points.
- Demonstrated that neural networks can be used to predict correspondences between image patches and points on a model.

6.3 Possible directions for future work

Some suggestions for future work have already been given in previous chapters. Some other extensions are discussed below.

Modelling embellishments

The spline models developed in this thesis are able to capture most of the common kinds of variability of digits, but they are not good at modelling "optional extra" parts of digits, such as ligatures or the cross-bar on European style 7 's. It would be possible to create additional models incorporating the extra parts, but this would slow down the recognition process even further[2]. The aim of the ideas discussed below is to construct models which make the explanation of data corresponding to the extra parts cheaper than it is under the uniform noise model.

One possible solution (mentioned in Williams *et al*, 1993) is to use short spline models for the extra parts, which can be fitted to the residual image, i.e. the image obtained by subtracting out the portion of the image which is accounted for by the digit model. A problem with this idea is that these models have to be very flexible, so they need to be fitted from a good starting point. This could be addressed by extending the neural net approach of section 5.2 to predict not only the instantiation parameters of the main model, but also the existence and instantiation parameters of the extra parts.

Another direction for investigation is to add extra Gaussians into the generative model. For example, the 3 model could have large-variance Gaussians at either end to model ligatures, and the 7 model could have an extra Gaussian on the down-stroke to deal with cross-bars. The mixing proportions of these additional Gaussians could be adapted during the search.

Rejecting non-digits

So far we have only used the models to answer the question "which of these generative models best explains the image ?". However, we could go further and ask if the explanation that the best model provides is reasonable, and if not, reject the classification. For example, if we fit all the digits models to an image of a letter "X", one of the models will fit best, but the explanation will probably be poor. One way to judge the quality of a fit is to compare the values of the measures used to evaluate the fit (i.e. the deformation and datafit energies, the volume factor, the white-space penalty, affine parameters) against the distribution of these quantities on training examples. A density estimator could be constructed in this low-dimensional space, for example using a kernel density estimator or a Gaussian mixture. Each digit model would probably need its own density estimator. If the probability of the measures is below some threshold, the classification would be rejected. This method could be used not only to reject non-digit images, but also to reject cases in which a local minimum fit was found or when the models do not have sufficient flexibility to generate the image.

[2]It would also be inefficient, as the models would not exploit the compositionality of the parts, e.g. to model possible 3 digits we would need an unembellished 3 model, a 3 with a ligature at the top, another 3 model with a ligature at the bottom, and a model with ligatures at the top and the bottom.

Unsupervised learning of the models

The training procedure described in chapters 3 and 4 is a supervised method, i.e. each model was only trained on data which was known to come from the appropriate class. However, it would also be interesting to carry out unsupervised training, where the objective is to maximize the likelihood of the data under a mixture of deformable models[3]. The training would start with a pool of spline models, each of which would have randomly chosen means and a spherical covariance matrix. The number of control points per spline could also be chosen randomly, although it would probably be sensible to ensure that the mean would be around eight. On each iteration of the training all of the models would be fitted to all of the images. For each image we would find out which models wins, i.e. ascribes the highest likelihood out of all of the models. The means and covariance matrix of each model would then be updated to the sample mean and covariance matrix for the fits on the images for which that particular model wins[4]. The idea of incorporating an object-based to image-based frame transformation into a clustering algorithm was described in Williams, Zemel and Mozer (1993), but in that work it was assumed that the correspondences between features was known. This means that the best transformation could be determined analytically, whereas for the spline models it has to be determined iteratively by the search process.

[3] Related work on clustering non-deformable models is described in Gold *et al* (1994) and Pope and Lowe (1993).

[4] This is a "hard" clustering algorithm similar to the k-means algorithm used in vector spaces.

Appendix A

Analyzing the settling dynamics at high variance

In this appendix some results are derived concerning the settling behaviour of a model which can (only) undergo affine transformations. I show that:

- At a sufficiently large bead variance there is a stable equilibrium when the translation components of the affine transformation are equal to the centre of gravity of the inked pixels, and the scale parameters are set to zero.
- The critical variance at which this ceases to be a stable equilibrium is determined by the covariance matrix of the inked pixels.
- The initial growth mode at this critical variance can be determined from the covariance matrices of the data and the model.

The discussion is based on the paper by Durbin, Szeliski and Yuille (1989) which performs a similar analysis for the elastic net, as used in their approach to the Traveling Salesman problem.

For the purposes of this analysis we consider a rigid template with a fixed variance for all beads in the image based frame. Thus the relevant energy function is the data-fit energy

$$E = -\sum_{i}^{N} \log(\sum_{j}^{M} \phi_{ij}) \tag{A.1}$$

where $\phi_{ij} = e^{-d_{ij}^2/2\sigma^2}$, d_{ij}^2 is the distance between the ith pixel and the jth bead and σ^2 is the variance of the beads in the image based frame. There are N inked pixels and M beads. The affine transformation taking bead j from position (x_j^1, x_j^2) in the object based frame to position (z_j^1, z_j^2) in the image based frame is given by

$$z_j^1 = ax_j^1 + bx_j^2 + t^1 \tag{A.2}$$

$$z_j^2 = cx_j^1 + bx_j^2 + t^2 \tag{A.3}$$

where t^1 and t^2 are translations and a, b, c, d are the entries in a 2×2 matrix. Hence $d_{ij}^2 = (z_j^1 - y_i^1)^2 + (z_j^2 - y_i^2)^2$, where (y_i^1, y_i^2) are the coordinates of datapoint i. We also assume that the template is centered so that $\sum_j x_j^\mu = 0$.

The first task is to show that the position where $t^\mu = \frac{1}{N}\sum_i y_i^\mu$ for $\mu = 1, 2$ and $a = b = c = d = 0$ is a turning point of the energy function. Let this position be denoted by p^*. For example,

$$\left.\frac{\partial E}{\partial t^\mu}\right|_{p^*} = -\sum_i \left(\frac{\sum_j \frac{\partial \phi_{ij}}{\partial t^\mu}}{\sum_k \phi_{ik}} \right) \tag{A.4}$$

$$= \sigma^{-2} \sum_i \left(\frac{\sum_j \phi_{ij}(z_j^\mu - y_i^\mu)}{\sum_k \phi_{ik}} \right) \tag{A.5}$$

$$= \sigma^{-2} \sum_i (t^\mu - y_i^\mu) \tag{A.6}$$

$$= 0 \tag{A.7}$$

where we have used the fact that at p^* $z_j^\mu = t^\mu$ and the ϕ_{ij}'s are equal for all j. Similarly,

$$\left.\frac{\partial E}{\partial a}\right|_{p^*} = \sigma^{-2} \sum_i \left(\frac{\sum_j \phi_{ij}(z_j^1 - y_i^1)x_j^1}{\sum_k \phi_{ik}} \right) \tag{A.8}$$

$$= \sigma^{-2} \sum_i [(t^1 - y_i^1) \sum_j x_j^1] \tag{A.9}$$

$$= 0 \tag{A.10}$$

because of the normalization of the template. Likewise it can be shown that the other partial derivatives are equal to zero at p^*.

Our task is now to examine the Hessian of E, i.e. the matrix of second derivatives, to determine the stability of the solution. This is achieved by calculating the eigenvalues of the Hessian; a stable solution will have a positive definite Hessian, with all eigenvalues positive.

The covariance matrices for the beads (in the object based frame) and pixel data will enter into these calculations so they are defined by

$$B = \begin{pmatrix} b_{11} & b_{12} \\ b_{12} & b_{22} \end{pmatrix} = \frac{1}{M} \begin{pmatrix} \sum_j (x_j^1)^2 & \sum_j x_j^1 x_j^2 \\ \sum_j x_j^1 x_j^2 & \sum_j (x_j^2)^2 \end{pmatrix} \tag{A.11}$$

$$D = \begin{pmatrix} d_{11} & d_{12} \\ d_{12} & d_{22} \end{pmatrix} = \frac{1}{N} \begin{pmatrix} \sum_i (y_i^1 - t^1)^2 & \sum_i (y_i^1 - t^1)(y_i^2 - t^2) \\ \sum_i (y_i^1 - t^1)(y_i^2 - t^2) & \sum_i (y_i^2 - t^2)^2 \end{pmatrix} \tag{A.12}$$

The calculation of the Hessian is somewhat tedious, and only a few examples of the elements are illustrated below.

$$\left.\frac{\partial^2 E}{\partial a^2}\right|_{p^*} = \sigma^{-2} \sum_i \frac{\partial}{\partial a} \left\{ \frac{\sum_j \phi_{ij}(z_j^1 - y_i^1)x_j^1}{\sum_k \phi_{ik}} \right\} \tag{A.13}$$

$$= \sigma^{-2} \sum_i \left\{ \frac{\sum_j [-\sigma^{-2}\phi_{ij}(z_j^1 - y_i^1)^2 (x_j^1)^2 + \phi_{ij}(x_j^1)^2]}{\sum_k \phi_{ik}} + \right.$$

$$\left. \frac{[\sum_j \phi_{ij}(z_j^1 - y_i^1)x_j^1][\sum_k \sigma^{-2}\phi_{ik}(z_k^1 - y_i^1)x_k^1]}{(\sum_k \phi_{ik})^2} \right\} \tag{A.14}$$

$$= \sigma^{-2}\{N b_{11} - \sum_i [\sigma^{-2}(t^1 - y_i^1)^2 \sum_j (x_j^1)^2]\} \tag{A.15}$$

$$= \sigma^{-2} N b_{11}[1 - \sigma^{-2} d_{11}] \tag{A.16}$$

The step from equation A.14 to equation A.15 is derived by realizing that the term on the second line of equation A.14 is equal to zero because $\sum_j x_j^1 = 0$.

$$\begin{aligned}
\left.\frac{\partial^2 E}{\partial a \partial b}\right|_{p^*} &= \sigma^{-2} \sum_i \frac{\partial}{\partial b} \left\{ \frac{\sum_j \phi_{ij}(z_j^1 - y_i^1) x_j^1}{\sum_k \phi_{ik}} \right\} && \text{(A.17)} \\
&= \sigma^{-2} \sum_i \left\{ \frac{\sum_j [-\sigma^{-2}\phi_{ij}(z_j^1 - y_i^1)^2 x_j^1 x_j^2 + \phi_{ij} x_j^1 x_j^2]}{\sum_k \phi_{ik}} + \right. \\
&\qquad \left. \frac{[\sum_j \phi_{ij}(z_j^1 - y_i^1) x_j^1][\sum_k \sigma^{-2}\phi_{ik}(z_k^1 - y_i^1) x_k^2]}{(\sum_k \phi_{ik})^2} \right\} && \text{(A.18)} \\
&= \sigma^{-2}\{N b_{12} - \sum_i [\sigma^{-2}(t^1 - y_i^1)^2 \sum_j x_j^1 x_j^2]\} && \text{(A.19)} \\
&= \sigma^{-2} N b_{12} [1 - \sigma^{-2} d_{11}] && \text{(A.20)}
\end{aligned}$$

The full Hessian matrix evaluated at p^* is

$$H = \frac{N}{\sigma^2} \begin{pmatrix} b_{11}(1-\sigma^{-2}d_{11}) & b_{12}(1-\sigma^{-2}d_{11}) & -b_{11}\sigma^{-2}d_{12} & -b_{12}\sigma^{-2}d_{12} & 0 & 0 \\ b_{12}(1-\sigma^{-2}d_{11}) & b_{22}(1-\sigma^{-2}d_{11}) & -b_{12}\sigma^{-2}d_{12} & -b_{22}\sigma^{-2}d_{12} & 0 & 0 \\ -b_{11}\sigma^{-2}d_{12} & -b_{12}\sigma^{-2}d_{12} & b_{11}(1-\sigma^{-2}d_{22}) & b_{12}(1-\sigma^{-2}d_{22}) & 0 & 0 \\ -b_{12}\sigma^{-2}d_{12} & -b_{22}\sigma^{-2}d_{12} & b_{12}(1-\sigma^{-2}d_{22}) & b_{22}(1-\sigma^{-2}d_{22}) & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.21}$$

where the ordering of the variables is a, b, c, d, t^1, t^2.

This matrix has an interesting structure. The first observation is that it is block diagonal, with a 4×4 block H_1 and a 2×2 block H_2. This means that the eigenvalues of H can be calculated by separately finding the eigenvalues of H_1 and H_2 (and the eigenvalues of H_2 are 1 and 1 by inspection). This leaves H_1, which has a Kronecker product structure.

If A is a $n \times n$ matrix and B is a $m \times m$ matrix then the Kronecker (or direct or tensor) product of the two matrices $A \otimes B$ is a $nm \times nm$ matrix defined by

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1}B & a_{n2}B & \cdots & a_{nn}B \end{pmatrix} \tag{A.22}$$

For our purposes, the most important properties of Kronecker products are to do with the eigenvalue/vector structure. If A has eigenvectors $\{\mathbf{x}_1 \ldots \mathbf{x}_n\}$ and eigenvalues $\{\lambda_1 \ldots \lambda_n\}$, and B has eigenvectors $\{\mathbf{y}_1 \ldots \mathbf{y}_m\}$ and eigenvalues $\{\mu_1 \ldots \mu_m\}$, then $A \otimes B$ has eigenvectors $\{\mathbf{x}_i \otimes \mathbf{y}_j\}$ and corresponding eigenvalues $\{\lambda_i \mu_j\}$. For a proof of these results see (for example) Steeb (1991).

H_1 is the Kronecker product of the matrices $D' = I - \sigma^{-2}D$ and B. As B is a covariance matrix its eigenvalues must be greater than or equal to zero (strictly greater than if the data consists of three or more non-collinear points, as we shall assume here). Hence zero eigenvalues of the product matrix must be due to zero eigenvalues of D'. For very large σ^2 D' is effectively the identity matrix, so we would expect zero eigenvalues to appear as σ^2 is reduced. D' has a zero eigenvalue when $\det(D') = 0$ at critical variance σ_*^2. This translates into a quadratic equation in σ_*^2

$$(\sigma_*^2 - d_{11})(\sigma_*^2 - d_{22}) - d_{12}^2 = 0 \tag{A.23}$$

The larger of the two roots to this equation (which is the eigenvalue equation for the covariance matrix D) is the prediction for the critical variance at which p^* becomes unstable. This has been confirmed experimentally.

This analysis does not only predict the critical variance, but also the initial growth mode, using the eigenvector(s) of H which have zero eigenvalue. If D' has one zero eigenvalue, then H_1 (and therefore H) will have two zero eigenvalues, using the properties of Kronecker product matrices. However, at infinitesimally lower variance, the more negative eigenvalue of H_1 will be produced by the product of the larger of the two eigenvalues of B (corresponding to the principal component of the bead distribution) with the negative eigenvalue of D'. Hence the initial growth mode will correspond to the vector

$$(\cos\theta_d, \sin\theta_d) \otimes (\cos\theta_b, \sin\theta_b) = \begin{pmatrix} \cos\theta_d \cos\theta_b \\ \cos\theta_d \sin\theta_b \\ \sin\theta_d \cos\theta_b \\ \sin\theta_d \sin\theta_b \end{pmatrix} \tag{A.24}$$

where $(\cos\theta_d, \sin\theta_d)$ is the principal eigenvector of the data, and $(\cos\theta_b, \sin\theta_b)$ is the principal eigenvector of the beads. The full six dimensional eigenvector is obtained by concatenating the vector given in equation A.24 with $(0, 0)$.

This vector can be further interpreted by looking at an alternative representation of the affine transformation matrix

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} r_1 \cos\theta_1 & -r_2 \sin\theta_2 \\ r_1 \sin\theta_1 & r_2 \cos\theta_2 \end{pmatrix} \tag{A.25}$$

Equating the a, b, c and d elements in equations A.24 and A.25, we can show that $\tan\theta_1 = \tan\theta_d$, $\tan\theta_2 = \tan(\theta_d - \pi/2)$, $r_1^2 = \cos^2\theta_b$ and $r_2^2 = \sin^2\theta_b$. Furthermore, the equation $\tan\phi = \tan\psi$ for $\phi,\ \psi \in [0, 2\pi]$ has solutions $\phi = \psi$ or $\phi = \psi \pm \pi$, using whichever sign is necessary to keep ϕ within range. Hence θ_1 will be aligned with the principal component of the data and θ_2 will be at rightangles to this direction. These predictions have also been verified experimentally. Similar results can be obtained for the case where the affine transformation is restricted to a similarity transformation.

Appendix B

Updating the instantiation parameters

This appendix provides more details on the EM-based algorithm discussed in section 3.3.2, and shows that the update can be performed so that E_{tot} does not increase on each iteration of the search procedure.

When fitting a digit model to data, the beads are constrained to lie on the spline; the underlying variables are the deformations and affine transformation parameters. If we attempt to apply the EM method directly, the M step yields a set of non-linear simultaneous equations for the instantiation parameters. This is due to the non-linear operation, the *multiplication* of the control point positions in the object-based frame (OBF) by the affine transformation matrix that is used to obtain the control point positions in the image-based frame (IBF). We have therefore developed a two-stage method, each stage of which is guaranteed not to increase E_{tot}[1].

In the first stage, the affine transformation is held fixed, and we find an update of the position of the control points in the IBF so that E_{tot} does not increase. The key observation is that with the affine transformation fixed, the deformation energy can be re-expressed in IBF coordinates. Let $\mathbf{x}^o$ denote the position of the control points in the object-based frame, and $\mathbf{x}^i$ the corresponding points in the IBF. The matrix A denotes the block diagonal matrix based on the 2×2 affine transformation matrix, and the translation vector $\mathbf{t}$ is the concatenation of the translation vector for each control point $\mathbf{t} = (t_x, t_y, t_x, t_y, \ldots, t_x, t_y)^T$. Using $\mathbf{x}^i = A\mathbf{x}^o + \mathbf{t}$, we obtain

$$\begin{aligned} E_{def} &= \frac{1}{2}(\mathbf{x}^o - \mathbf{h}^o)^T(\Sigma^o)^{-1}(\mathbf{x}^o - \mathbf{h}^o) + \frac{1}{2}\log|\Sigma^o| + \text{const} && \text{(B.1)} \\ &= \frac{1}{2}(\mathbf{x}^i - A\mathbf{h}^o - \mathbf{t})^T(\Sigma^i)^{-1}(\mathbf{x}^i - A\mathbf{h}^o - \mathbf{t}) + \frac{1}{2}\log|\Sigma^i| + \text{const} && \text{(B.2)} \end{aligned}$$

where $(\Sigma^i)^{-1} = A^{-T}\ (\Sigma^o)^{-1}\ A^{-1}$. The extra factor of A^{-1} in the determinant term in equation B.2 arises from the Jacobian of the transformation between the two frames. Using E_{def} as defined in equation B.2, we can solve the linear system defined by $\partial E_{tot}/\partial \mathbf{x}^i = M\mathbf{x}^i - \mathbf{b} = \mathbf{0}$ to obtain the new $\mathbf{x}^i$ vector. This is simply one iteration of the EM algorithm for a penalized maximum likelihood problem (see Green,

[1]For rigid models, where only the affine parameters are variable, the M step of the EM algorithm can be obtained exactly by solving a set of simultaneous linear equations.

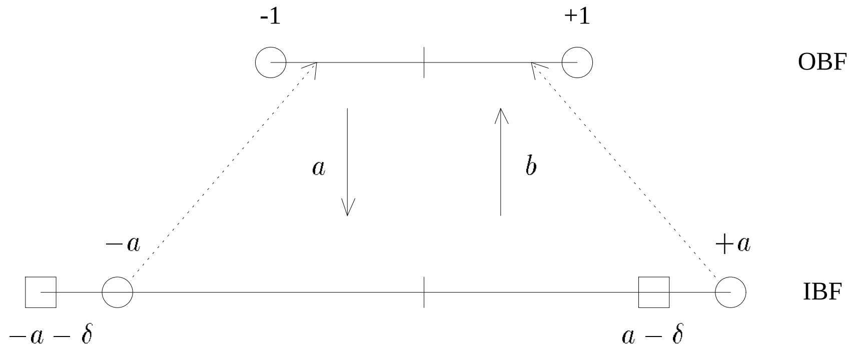


Figure B.1: Illustrating the effects of "shrinkage". The model consists of two points located at ± 1. An ideal image is obtained by scaling the model by a factor a. A noisy image has points at $-a-\delta$ and $a-\delta$. The diagram shows that the optimal inverse transformation b of the noisy data maps the points $\pm a$ in the IBF to the locations shown by the dotted arrows. Note that these points lie inside ± 1.

1990), and is therefore guaranteed not to increase E_{tot}. The resulting $\mathbf{x}^i$ corresponds to the minimum energy equilibrium point where the forces pulling the beads towards the nearby pixels are balanced by the forces pulling the beads towards their (image-based) home locations.

Setting $\partial E_{fit}/\partial \mathbf{x}^i = M^{fit}\mathbf{x}^i - \mathbf{b}^{fit}$ and $\partial E_{def}/\partial \mathbf{x}^i = M^{def}\mathbf{x}^i - \mathbf{b}^{def}$, we have $M = M^{fit} + M^{def}$ and $\mathbf{b} = \mathbf{b}^{fit} + \mathbf{b}^{def}$. Let $\mathbf{m}^g = B^g\mathbf{x}^i$, where B^g is a $2 \times 2n$ matrix mapping from the positions of the n control points to $\mathbf{m}^g$, the mean of generator g. Using the definition of E_{fit} given in equation 3.7, we obtain

$$M^{fit}_{\alpha\beta} = \frac{N_0}{\sigma^2 N_B}\sum_g (\sum_i r_g(\mathbf{y}_i))(B^g_{1\alpha}B^g_{1\beta} + B^g_{2\alpha}B^g_{2\beta}) \tag{B.3}$$

$$b^{fit}_{\alpha} = \frac{N_0}{\sigma^2 N_B}\sum_{i,g} r_g(\mathbf{y}_i)(B^g_{1\alpha}y^1_i + B^g_{2\alpha}y^2_i) \tag{B.4}$$

where i indexes inked pixels and g indexes ink generators (beads). The responsibilities $r_g(\mathbf{y}_i)$ are evaluated in the E step of the EM algorithm. Using equation B.2 we find that $M^{def} = (\Sigma^i)^{-1}$ and $\mathbf{b}^{def} = (\Sigma^i)^{-1}(A\mathbf{h}^o + \mathbf{t})$. M^{fit} is symmetric positive definite since it is a weighted sum of outer products (with all of the weights positive). M is symmetric positive definite since both M^{fit} and M^{def} are.

The second stage of the M-step involves adjusting the affine transformation and deformations to minimize E_{def} whilst holding the new *image* locations of the control points fixed. In effect, we are absorbing as much of the deformation as possible into the global affine transformation. Setting $\mathbf{x}^o = A^{-1}\mathbf{x}^i + A^{-1}\mathbf{t} = \mathbf{x}^o = B\mathbf{x}^i + \mathbf{u}$, it is possible to minimize E_{def} as given in equation B.1 with respect to B and $\mathbf{u}$ to obtain the optimal (inverse) affine transformation. This is possible because the deformation energy (in the object-based frame) is simply a quadratic form in terms of these parameters. This step cannot increase E_{fit} because this energy depends only upon the image locations of the beads and their variances and these are unchanged during this stage. Hence E_{tot} is not increased in this stage either.

It should be noted that solving for the inverse affine transformation displays a slight "shrinkage" effect, which can be illustrated by the following one-dimensional example. Consider the situation shown in figure B.1. The model consists of two points centered at $+1$ and -1 on the real line. Data is produced by adding noise to these points and then scaling them by a factor a. In figure B.1 the generated points lie at $(-a - \delta)$ and $(a - \delta)$. Solving for the inverse transformation b which minimizes the residual sum-squared distance in the OBF, we find that the points $\pm a$ in the IBF correspond to the points $\pm a^2/(a^2 + \delta^2)$ in the OBF, which lie inside the points ± 1. In contrast, the value of the forward scaling factor that minimizes the sum-squared residual in the IBF is simply a. Because of this shrinkage effect (and for historical reasons), we actually solve for the best A and $\mathbf{t}$ in equation B.2, while keeping $(\Sigma^i)^{-1}$ fixed, thereby ignoring the fact that the contour lines of equal deformation energy change as the affine transformation varies. We have found that this produces satisfactory results, although E_{def} is not guaranteed to decrease when using this update.

References

Ayache, N. and Faugeras, O. D. (1986). HYPER: A new approach for the recognition and positioning of two-dimensional objects. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(1):44–54.

Bajcsy, R., Lieberson, R., and Reivich, M. (1983). A computerized system for the elastic matching of deformed radiographic images to idealized atlas images. *Journal of Computer Assisted Tomography*, 7(4):618–625.

Ballard, D. H. (1981). Generalizing the Hough transfrom to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122.

Beis, J. S. and Lowe, D. G. (1993). Learning Indexing Functions for 3-D Model-Based Object Recognition. In *AAAI Fall 1993 Symposium on Machine Learning in Computer Vision*, pages 50–54. Proceedings available as AAAI Tech Report FSS-93-04.

Beymer, D., Shashua, A., and Poggio, T. (1993). Example Based Image Analysis and Synthesis. AI Memo 1431, AI Laboratory, MIT.

Binford, T. O. (1971). Visual perception by computer. In *Proceedings of the IEEE Conference on Systems and Control*. Miami.

Bishop, C. M. (1994). Mixture Density Networks. Technical Report NCRG/4288, Neural Computing Research Group, Aston University, UK.

Blake, A. and Zisserman, A. (1987). *Visual Reconstruction*. MIT Press.

Bolles, R. C. and Cain, R. C. (1982). Recognizing and locating partially visible objects: The local-feature-focus method. *International Journal of Robotics Research*, 1(3):57–82.

Bookstein, F. L. (1978). The measurement of biological shape and shape change. In Levin, S., editor, *Lecture Notes in Biomathematics 24*. Springer-Verlag.

Bridle, J. S. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In Fougelman-Soulie, F. and Hérault, J., editors, *Neuro-computing: algorithms, architectures and applications*. NATO ASI series on systems and computer science. Springer-Verlag.

Brooks, R. A. (1981). Symbolic reasoning among 3-d models and 2-d images. *Artificial Intelligence*, 17:285–348.

Broomhead, D. and Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355.

Brown, P. (1987). *The Acoustic-Modeling Problem in Automatic Speech Recognition*. PhD thesis, Carnegie-Mellon University. Also published as IBM Research Division Technical Report RC 12750.

Burges, C. J. C., Matan, O., Le Cun, Y., Denker, J. S., Jackel, L. D., Stenard, C. E., Nohl, C. R., and Ben, J. I. (1992). Shortest path segmentation: A method for training a neural network to recognize character strings. *IJCNN*, 3:III–165.

Burr, D. J. (1981a). A dynamic model for image registration. *Comput. Graphics Image Process.*, 15:102–112.

Burr, D. J. (1981b). Elastic matching of line drawings. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 3(6):708–713.

Cash, G. L. and Hatamian, M. (1987). Optical character recognition by the method of moments. *Computer Vision, Graphics and Image Processing*, 39:291–310.

Cipolla, R. and Blake, A. (1990). The Dynamic Analysis of Apparent Contours. In *Proc. International Conference on Computer Vision*. Osaka.

Cootes, T. F. and Taylor, C. J. (1992). Active shape models—smart snakes. In Hogg, D. and Boyle, R., editors, *Proceedings of the British Machine Vision Conference*, pages 266–275. Springer-Verlag.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Proceedings of the Royal Statistical Society*, B-39:1–38.

Doermann, D. S. and Rosenfeld, A. (1992). Recovery of temporal information from static images of handwriting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Dudek, G. and Tsotsos, J. K. (1991). Shape representation and recognition from curvature. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 35–41.

Durbin, R., Szeliski, R., and Yuille, A. L. (1989). An analysis of the elastic net approach to the travelling salesman problem. *Neural Computation*, 1:348–358.

Durbin, R. and Willshaw, D. (1987). An analogue approach to the travelling salesman problem. *Nature*, 326:689–691.

Edelman, S., Ullman, S., and Flash, T. (1990). Reading cursive handwriting by alignment of letter prototypes. *International Journal of Computer Vision*, 5(3):303–331.

Edelsbrunner, H. (1987). *Algorithms in combinatorial geometry*. Springer-Verlag.

Fischler, M. A. and Elschlager, R. A. (1973). The representation and matching of pictorial structures. *IEEE Trans. Computers*, C-22(1):67–92.

Freeman, W. T. (1994). The generic viewpoint assumption in a framework for visual perception. *Nature*, 368:542–545.

Fua, P. and Hanson, A. J. (1989). Objective Functions for Feature Discrimination: Theory. In *Proc. DARPA Image Understanding Workshop*.

Gardiner, M. (1965). The Superellipse: A curve between the ellipse and the rectangle. *Scientific American*, 213:222–234.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

Gold, S., Mjolsness, E., and Rangarajan, A. (1994). Clustering with a Domain-Specific Distance Measure. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Neural Information Processing Systems, Vol. 6.* Morgan Kaufmann, San Mateo, CA.

Gonzalez, R. C. and Wintz, P. (1987). *Digital Image Processing.* Addison-Wesley, Reading, Massachusetts, second edition.

Green, P. J. (1990). On the use of the EM Algorithm for Penalized Likelihood Estimation. *J. Roy. Stat. Soc. B*, 52(3):443–452.

Grenander, U., Chow, Y., and Keenan, D. M. (1991). *Hands: A pattern theoretic study of biological shapes.* Springer-Verlag.

Grimson, W. E. L. (1990). *Object recognition by computer.* MIT Press, Cambridge, MA.

Gull, S. F. (1988). Bayesian inductive inference and maximum entropy. In Erickson, G. J. and Smith, C. R., editors, *Maximum Entropy and Bayesian Methods in science and engineering, vol1: Foundations.* Kluwer.

Hampshire, J. and Waibel, A. (1989). A novel objective function for improved phoneme recognition using time-delay neural networks. Technical Report CMU-CS-89-118, Carnegie-Mellon, Pittsburgh, PA.

Härdle, W. (1990). *Applied Nonparametric Regression.* Cambridge University Press, Cambridge, England.

Harris, M. R. (1991). Attracting similar shapes towards each other. In Meyer, J. A. and Wilson, S. W., editors, *From Animals to Animats.* MIT Press.

Hastie, T. and Tibshirani, R. (1992). *Handwritten digit recognition via deformable prototypes.* Technical Report, Department of Statistics, University of Toronto.

Hinton, G. E. (1989). Deterministic Boltzmann learning performs steepest descent in weight-space. *Neural Computation, 1*:143–150.

Hinton, G. E., Revow, M., and Dayan, P. (1994). Recognizing Handwritten Digits Using Mixtures of Linear Models. To appear in NIPS-7.

Hinton, G. E., Williams, C. K. I., and Revow, M. D. (1992a). Adaptive elastic models for hand-printed character recognition. In Moody, J. E., Hanson, S. J., and Lippmann, R. P., editors, *Advances in Neural Information Processing Systems 4.* Morgan Kauffmann.

Hinton, G. E., Williams, C. K. I., and Revow, M. D. (1992b). Combinining two methods of recognizing hand-printed digits. In Aleksander, I. and Taylor, J., editors, *Artificial Neural Networks 2*. Elsevier Science Publishers.

Horn, B. K. P. (1986). *Robot Vision*. MIT Press, Cambridge, MA.

Hough, P. V. C. (1962). Methods and Means for Recognizing Complex Patterns. U.S. Patent 3069654.

Hunt, D. J., Nolte, L. W., and Ruedger, W. H. (1988). Performance of the Hough Transform and its Relationship to Statistical Signal Detection Theory. *Computer Vision, Graphics and Image Processing*, 43:221–238.

Illingworth, J. and Kittler, J. (1988). A Survey of the Hough Transform. *Computer Vision, Graphics and Image Processing*, 44:87–116.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3(1).

Jepson, A. D. and Richards, W. (1993). What makes a good feature ? In Harris, L. and Jenkin, M., editors, *Spatial Vision in Humans and Robots*. Cambridge University Press.

Kass, M., Witkin, A., and Terzopoulos, D. (1987). Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–332.

Keeler, J. D., Rumelhart, D. E., and Leow, W. K. (1991). Integrated segmentation and recognition of hand-printed numerals. In Lippmann, R. P., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 3*, pages 557–563, San Mateo, CA. Morgan Kaufmann Publishers.

Keren, D., Subrahmonia, J., and Cooper, D. B. (1992). Robust Object Recognition Based on Implicit Algebraic Curves and Surfaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Knoerr, A. (1988). Global models and natural boundaries: Theory and applications. Reports in Pattern Analysis 148, Division of Applied Mathematics, Brown University, Providence, RI.

Kullback, S. (1959). *Information Theory and Statistics*. Wiley, New York.

Lades, M., Vorbrüggen, J. C., Buhmann, J., Lange, J., von der Malsburg, C., Würtz, R. P., and Konen, W. (1993). Distortion Invariant Object Recognition in the Dynamic Link Architecture. *IEEE Transactions on Computers*, 42(3):300–311.

Lam, L. and Suen, C. Y. (1988). Structural classification and relaxation matching of totally unconstrained handwritten zip-code numbers. *Pattern Recognition*, 21(1):19–31.

Lamdan, Y. and Wolfson, H. J. (1988). Geometric Hashing: A General and Efficient Model-Based Recognition Scheme. In *Proceedings of the Second International Conference on Computer Vision*. IEEE.

Lanitis, A., Taylor, C. J., and Cootes, T. F. (1993). A generic system for classifying variable objects using flexible template matching. In Illingworth, J., editor, *Proceedings of the British Machine Vision Conference*, volume 1, pages 329–338. BMVA Press.

Le Cun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L. (1990). Handwritten digit recognition with a back-propagation network. In Touretzky, D., editor, *Advances in Neural Information Processing Systems*, volume 2, pages 396–404, Denver. Morgan Kaufmann, San Mateo.

Leclerc, Y. G. (1989). Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision*, 3(1):73–102.

Lee, D. and Srihari, S. N. (1993). Handprinted digit recognition: A comparison of algorithms. In *Third international workshop on frontiers in handwriting recognition*, pages 153–162, Buffalo, NY.

Lee, Y. (1991). Handwritten digit recognition using k- nearest-neighbor, radial-basis function, and backpropogation neural networks. *Neural Computation*, 3:440–449.

Li, W. (1990). Mutual Information Functions versus Correlation Functions. *Journal of Statistical Physics*, 60(5/6):823–837.

Lowe, D. E. (1985). *Perceptual Organization and Visual Recognition.* Kluwer Academic Publishers, Hingham, MA.

Lowe, D. G. (1991). Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-13:441–450.

MacKay, D. J. C. (1992). Bayesian Interpolation. *Neural Computation*, 4(3):415–447.

Mantas, J. (1986). An overview of character recogntion methodologies. *Pattern Recognition*, 19(6):425–430.

Marr, D. and Nishihara, H. K. (1978). Representation and recognition of the spatial organization of three-dimensional images. *Proceedings of the Royal Society of London, Series B*, 200:269–294.

Matan, O., Burges, C. J., LeCun, Y., and Denker, J. S. (1992). Multi-digit recognition using a space displacement neural network. In Moody, J. E., Hanson, S. J., and Lippmann, R. P., editors, *Advances in Neural Information Processing Systems 4*. Morgan Kauffmann.

Metaxas, D. and Terzopoulos, D. (1991). Recursive estimation of shape and non-rigid motion. In *Proceedings of IEEE Workshop on Visual Motion*, Princeton, New Jersey.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21 or 6:1087–1091.

Mjolsness, E. (1990). Bayesian inference on visual grammars by neural nets that optimize. Technical Report YALEU-DCS-TR-854, Dept. of Computer Science, Yale University.

Mjolsness, E. (1992). Visual Grammars and their Neural Nets. In Moody, J. E., Hanson, S. J., and Lippmann, R. E., editors, *Neural Information Processing Systems, Vol. 4*. Morgan Kaufmann, San Mateo, CA.

Mokhtarian, F. and Mackworth, A. (1986). Scale-space description and recognition of planar curves and two-dimensional shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8:34–43.

Neal, R. M. (1993a). Bayesian Learning via Stochastic Dynamics. In Hanson, S. J., Cowan, J. D., and Giles, C. L., editors, *Neural Information Processing Systems, Vol. 5*, pages 475–482. Morgan Kaufmann, San Mateo, CA.

Neal, R. M. (1993b). Probabilistic Inference Using Markov Chain Monte Carlo Methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto.

Neal, R. M. and Hinton, G. E. (1993). A new view of the EM algorithm that justifies incremental and other variants. *submitted to Biometrika.*

Pavlidis, T. (1977). *Structural Pattern Recognition.* Springer Verlag, New York.

Pentland, A. (1989). Part segmentation for object recognition. *Neural Computation*, 1:82–91.

Pentland, A. and Sclaroff, S. (1991). Closed-Form Solutions for Physically Based Shape Modelling and Recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(7):715–729.

Poggio, T. and Girosi, F. (1990). Networks for approximation and learning. *Proceedings of IEEE*, 78:1481–1497.

Poggio, T., Torre, V., and Koch, C. (1985). Computational vision and regularization theory. *Nature*, 317:6035:314–319.

Pope, A. R. and Lowe, D. G. (1993). Learning 3d object recognition models from 2d images. In *AAAI Fall 1993 Symposium on Machine Learning in Computer Vision*, pages 35–39. Proceedings available as AAAI Tech Report FSS-93-04.

Rabiner, L. R., Levinson, S. E., and Sondhi, M. M. (1983). On the application of Vector Quantization and Hidden Markov Models to speaker independent isolated word recognition. *AT&T Technical Journal*, 62(4).

Revow, M. D., Williams, C. K. I., and Hinton, G. E. (1993). Using mixtures of deformable models to capture variations in hand printed digits. In Srihari, S., editor, *Proceedings of the Third International Workshop on Frontiers in Handwriting Recognition*, pages 142–152, Buffalo, New York, USA.

Rissanen, J. (1985). Minimum description length principle. In Kotz, S. and Johnson, N. L., editors, *Encyclopedia of Statistical Sciences vol. 5*. Wiley and Sons.

Rissanen, J. (1989). *Stochastic Complexity in Statistical Inquiry*. World Scientific Publ. Co.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423,623–656.

Simard, P., Le Cun, Y., and Denker, J. (1993). Efficient pattern recognition using a new transformation distance. *Advances in Neural Information Processing Systems 4*, pages 50–58.

Simard, P., Victorri, B., LeCun, Y., and Denker, J. (1992). Tangent prop - a formalism for specifying selected invariances in an adaptive network. In Moody, J. E., Hanson, S. J., and Lippmann, R. P., editors, *Advances in Neural Information Processing Systems 4*. Morgan Kaufmann, San Mateo CA.

Steeb, W.-H. (1991). *Kronecker product of matrices and applications*. Mannheim : B.I. Wissenschaftsverlag.

Stephens, R. S. (1991). Probabilistic approach to the Hough transfrom. *Image and vision computing*, 9(1):66–71.

Suen, C. . Y., Nadal, C., Legault, R., Mai, T. A., and Lam, L. (1992). Computer recognition of unconstrained handwritten numerals. *Proceedings of the IEEE*, 80(7):1162–1180.

Suen, C. Y. (1980). Automatic recognition of handprinted characters - the state of the art. *Proceedings of the IEEE*, 68(4):469–487.

Szeliski, R. (1989). *Bayesian modeling of uncertainty in low-level vision*. Kluwer.

Tappert, C. C., Suen, C. Y., and Wakahara, T. (1990). The State of the Art in On-Line Handwriting Recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(8):787–808.

Taubin, G., Cukierman, F., Sullivan, S., Ponce, J., and Kriegman, D. J. (1992). Parameterizing and fitting bounded algebraic curves and surfaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Terzopoulos, D. (1986). Regularization of inverse problems involving discontinuities. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8:413–424.

Terzopoulos, D. and Fleischer, K. (1988). Deformable models. *The Visual Computer*, 4:306–331.

Terzopoulos, D. and Metaxas, D. (1990). Dynamic 3d models with local and global deformations: Deformable superquadrics. In *Proceedings of the Third International Conference on Computer Vision (ICCV '90)*. Osaka, Japan.

Turney, J. L., Mudge, T. N., and Volz, R. A. (1985). Recognizing partially occluded parts. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 7(4):410–421.

Ullman, S. (1989). Aligning pictorial descriptions: An approach to object recognition. *Cognition*, 32(3):193–254.

Ullmann, J. R. (1972). Correspondence in character recognition. In *Machine Perception of Patterns and Pictures*. Institute of Physics, London, U.K.

Waldram, J. R. (1985). *The theory of thermodynamics*. Cambridge University Press.

Wang, Z. (1991). Using Complex Gabor Filters to Detect and Localize Edges and Bars. Technical Report CS-91-04, Department of Computer Science, York University, Ontario.

Wertheimer, M. (1923). Laws of organization in perceptual forms. *Psychologische Forschung*, 4:301–350. Translated in W. D. Ellis, *A Source Book of Gestalt Psychology*. New York, Humanities Press, 1967.

Widrow, B. (1973). The 'rubber-mask' technique–I. Pattern Measurement and Analysis. *Pattern Recognition*, 5:175–197.

Wilkinson, R. A., Geist, J., Janet, S., Burges, C. J. C., Creecy, R., Hammond, B., Hull, J. J., Larsen, N. J., Vogl, T. P., and Wilson, C. L. (1992). NISTIR 4912. The first census optical character recognition systems conference. Technical report, U.S. National Institute of Standards and Technology.

Williams, C. K., Zemel, R. S., and Mozer, M. C. (1993). Unsupervised learning of object models. In *AAAI Fall 1993 Symposium on Machine Learning in Computer Vision*, pages 20–24. Proceedings available as AAAI Tech Report FSS-93-04.

Williams, C. K. I., Revow, M. D., and Hinton, G. E. (1993). Hand-printed digit recogntion using deformable models. In Harris, L. and Jenkin, M., editors, *Spatial Vision in Humans and Robots*. Cambridge University Press.

Witkin, A. P. and Tenenbaum, J. M. (1983). On the role of structure in vision. In Rosenfeld, A., editor, *Human and Machine Vision*. Academic Press.

Witkin, A. P., Terzopoulos, D., and Kass, M. (1986). Signal matching through scale space. In *Proceedings of the National Conference on Artificial Intelligence*, pages 714–719, Philadelphia, PA.

Xu, L., Krzyzak, A., and Suen, C. Y. (1992). Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition. *IEEE Trans. Systems Man and Cybernetics*, 22(3):418–435.

Xu, L. and Oja, E. (1993). Randomized Hough Transform (RHT): Basic Mechanisms, Algorithms, and Computational Complexities. *CVGIP: Image Understanding*, 57(2):131–154.

Yuille, A., Honda, K., and Peterson, C. (1990). Deformable Templates for Particle Tracking. Technical Report 90-8, Harvard Robotics Laboratory.

Yuille, A. L. (1991). Deformable templates for face recognition. *Journal of Cognitive Neuroscience*, 3(1):59–70.

Yuille, A. L., Cohen, D. S., and Hallinan, P. W. (1989). Feature extraction from faces using deformable templates. In *Proceedings of Computer Vision and Pattern Recognition*. IEEE Computer Society Press, Washington, D.C.

Zemel, R. S. (1993). *A Minimum Description Length Framework for Unsupervised Learning*. PhD thesis, Dept. of Computer Science, University of Toronto.

Zemel, R. S. and Hinton, G. E. (1991). Discovering viewpoint-invariant relationships that characterize objects. In Lippmann, R. P., Moody, J. E., and Touretzky, D. S., editors, *Advances In Neural Information Processing Systems 3*, pages 299–305. Morgan Kaufmann Publishers.

Zemel, R. S., Mozer, M. C., and Hinton, G. E. (1990). TRAFFIC: Recognizing objects using hierarchical reference frame transformations. In Touretzky, D. S., editor, *Neural Information Processing Systems, Vol. 2*, pages 266–273. Morgan Kaufmann, San Mateo, CA.