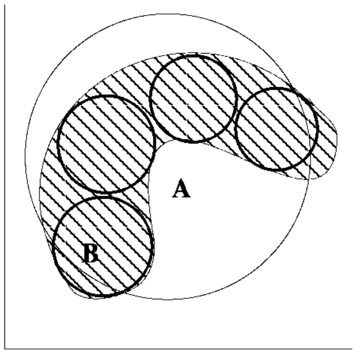

Figure 4.2: Approximating a distribution with a mixture of Gaussians. The high density area of the distribution is contained within the shaded area. The large circle represents the poor approximation that would be made by a single Gaussian of large variance. A better approximation is obtained by using 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.

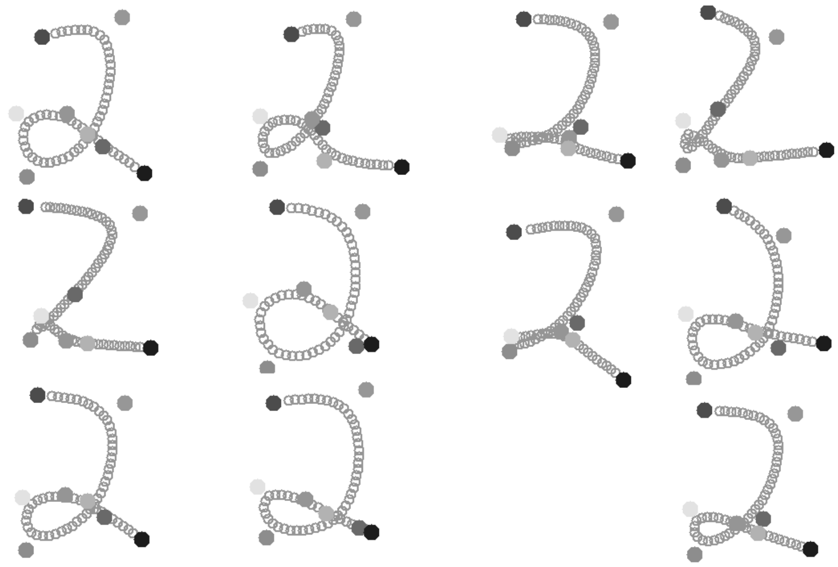

Figure 4.3: Components in the mixture for the 2 model. The generic model is shown in the bottom right corner.

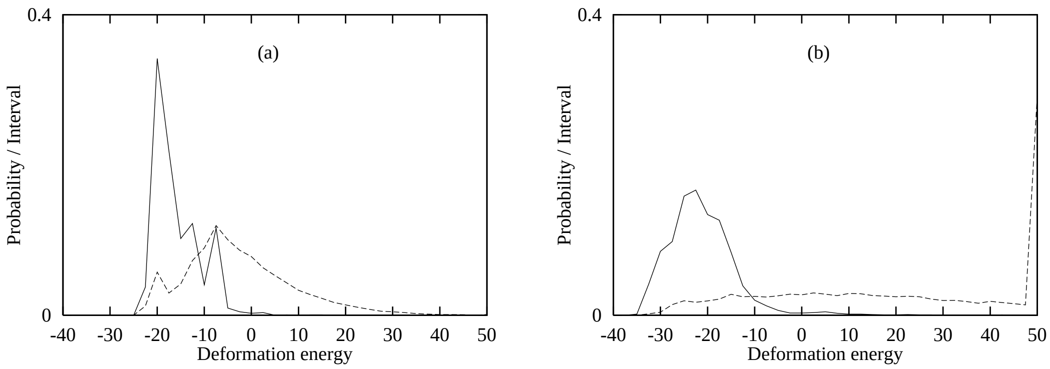

Figure 4.4: The distribution of the deformation energy under the generic model and mixture of local models. The solid line shows the E_{def} distribution of models when fitted to examples images containing the correct digit (e.g. a three model fitted to an image of a 3). The dotted curve shows the E_{def} 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)

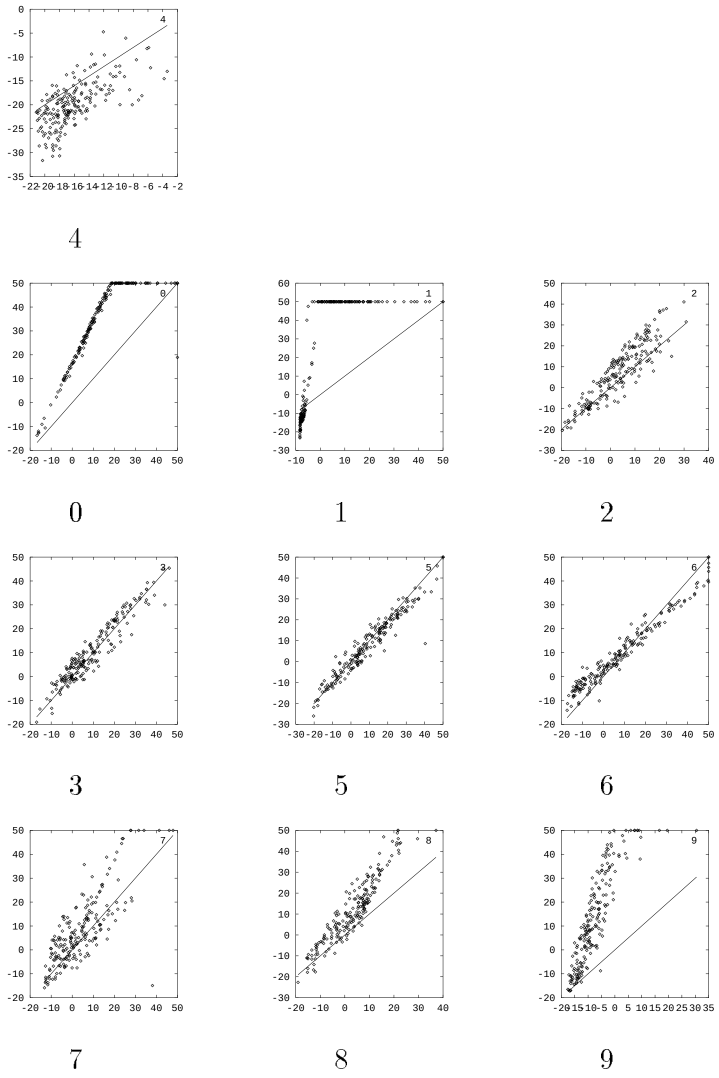

Figure 4.5: A comparison of the deformation energy under the generic model and mixture models for all ten digit models on the 4 data in VAL. In each cell the deformation energy under the generic model is plotted on the abscissa, and the deformation energy under the mixture of local models is plotted 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.

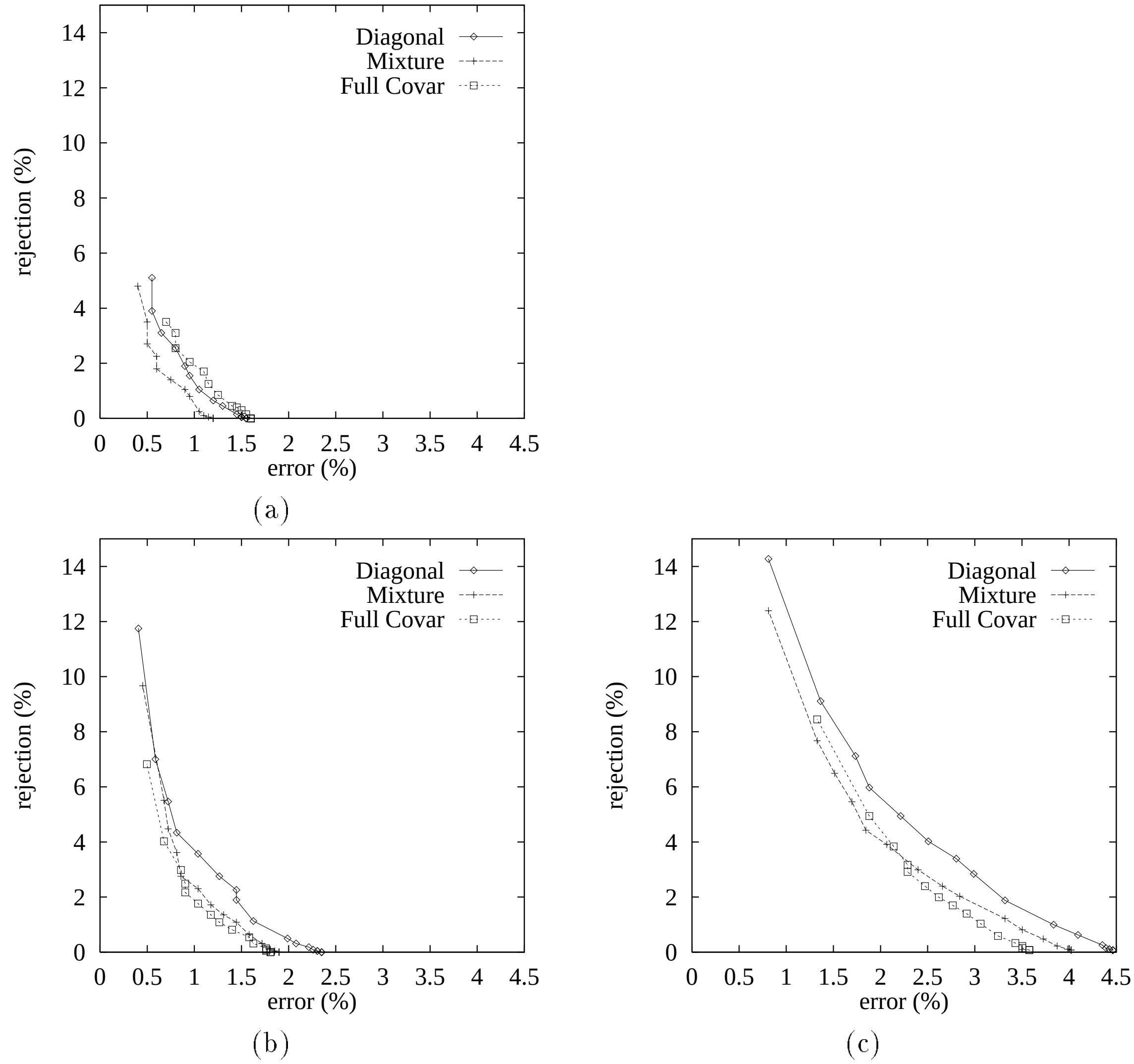

Figure 4.6: Error-rejection relationships for (a) the validation set (b) *goodbs* test set (c) *bs* test set.

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

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

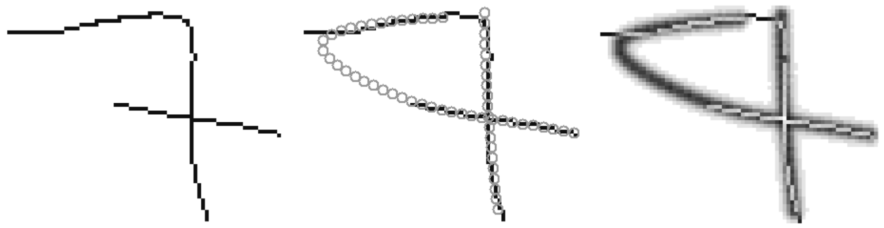

Figure 4.7: The black and white imaging model in action. The left panel shows an image of a 7. The fit of a four model using the mixture of Gaussians generative model is shown in the centre panel. Under this formulation the centre bar of the four model would not be penalized for lying in white space. 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.

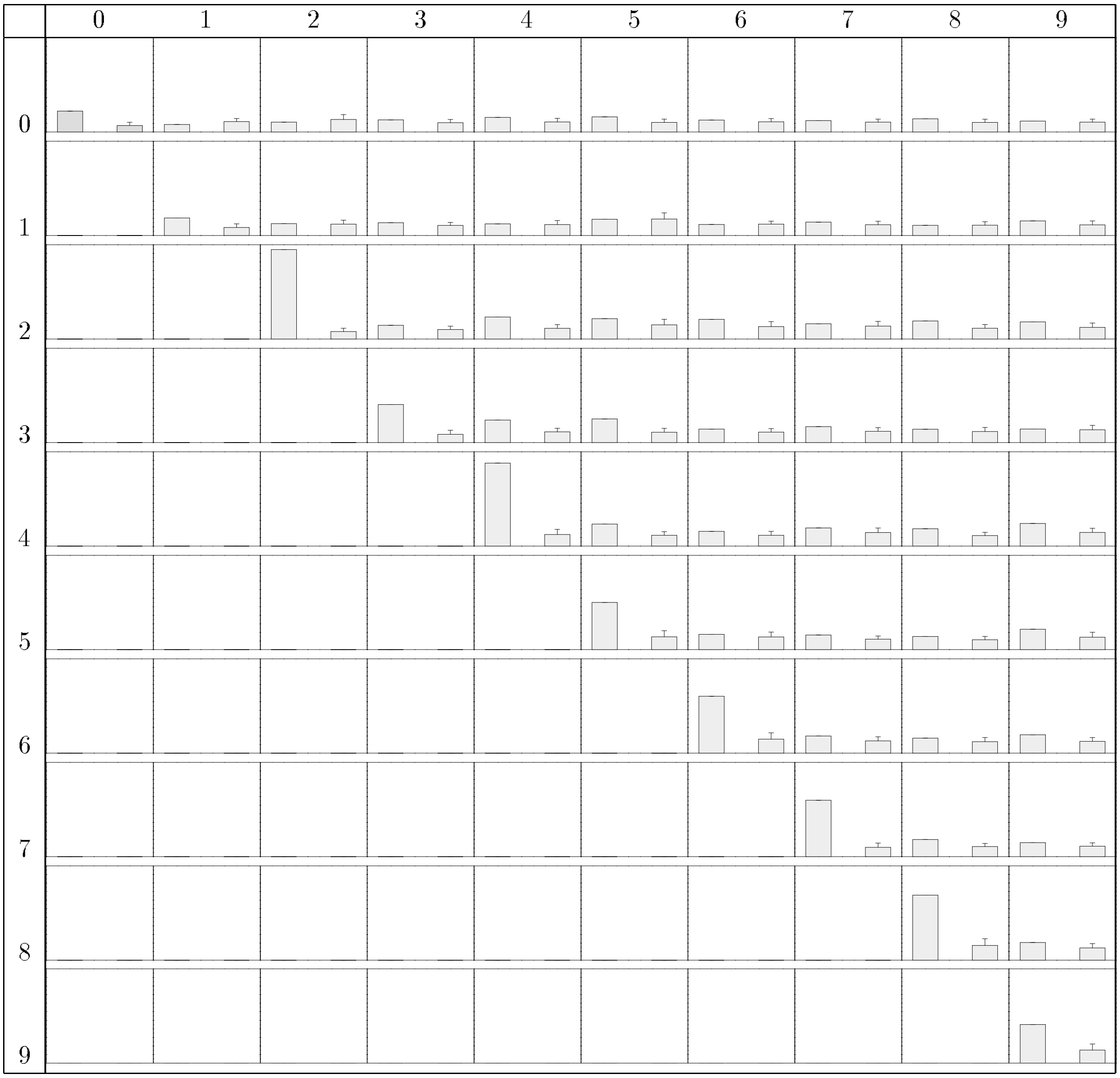

Figure 4.8: The observed mutual information and the null hypothesis result for each digit pair. In each cell the bar on the left shows the observed mutual information, and the bar on the right the mutual information under the null hypothesis. A 99% confidence error bar for the null hypothesis is also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

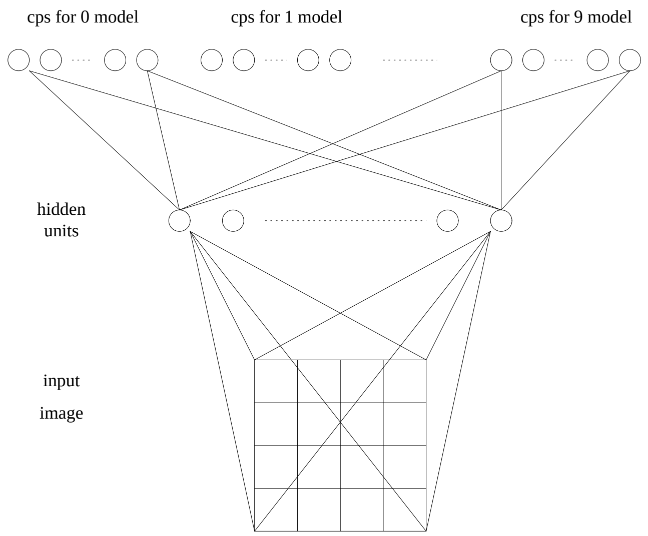

Figure 5.1: The prediction network architecture. "cps" stands for control points.

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.

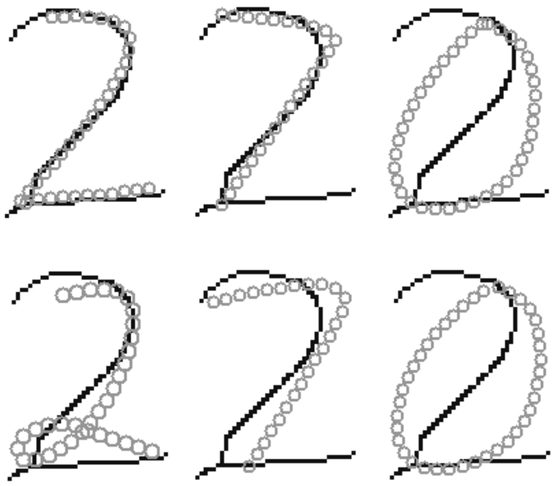

Figure 5.2: A comparison of the initial instantiations for the 2, 7 and 0 models due to the prediction net (top row) and the trivial net (bottom row) on an image of a 2. Notice that for the two model the prediction net is much closer to the data. The other digit models may or may not be helped by looking 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.

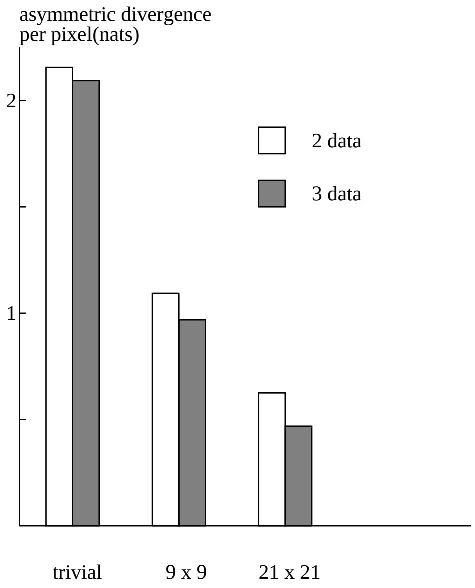

Figure 5.3: A comparison of the prediction performance of the trivial, 9×9 and 21×21 nets.

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.

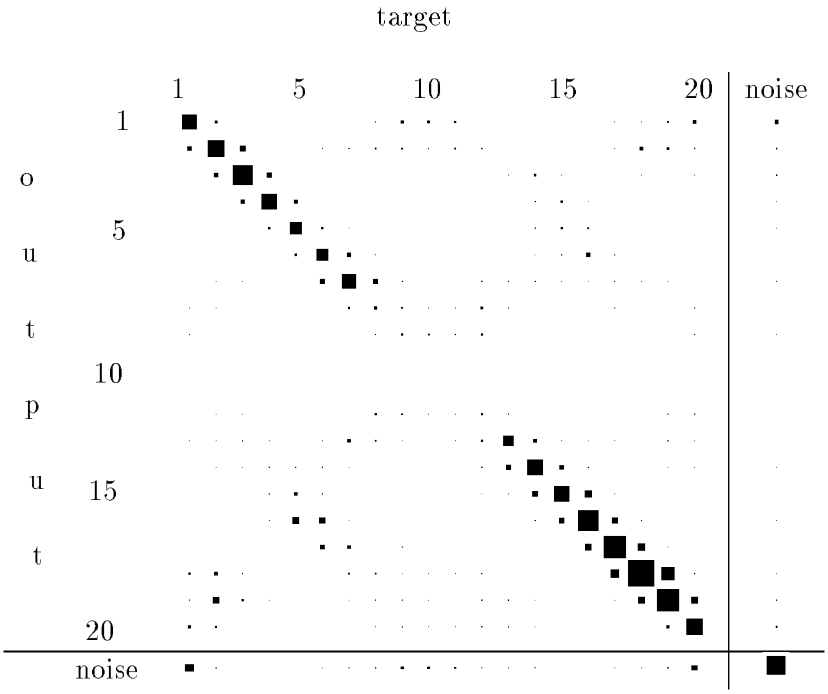

Figure 5.4: A two-dimensional histogram illustrating the performance of the 9×9 net on three data. The target bins run from left to right, the output bins from top to bottom. Bins 1 through 20 correspond to points on the spline; bin 21 is for the noise process. The linear size of each square is proportional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

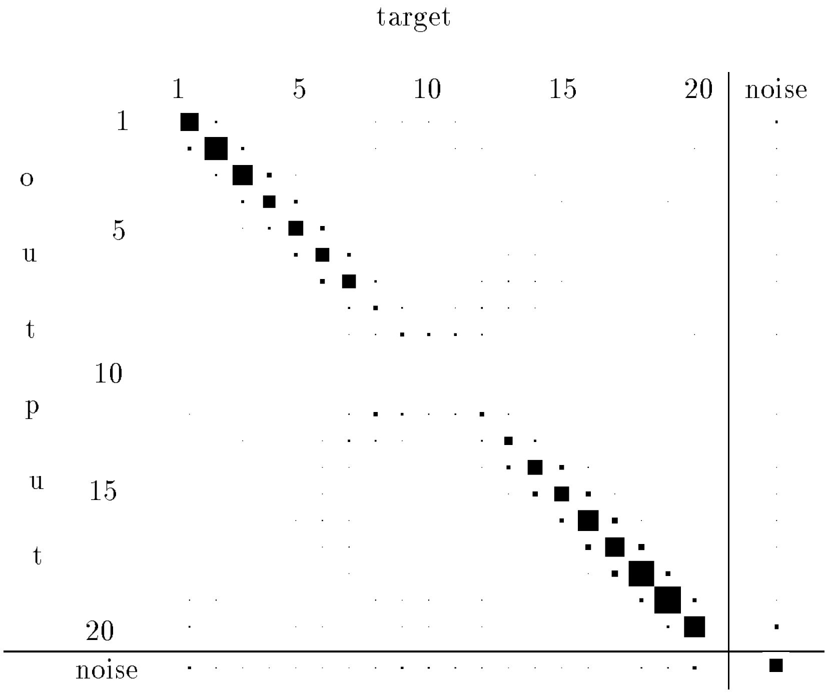

Figure 5.5: A two-dimensional histogram illustrating the performance of the 21×21 net on three data. The target bins run from left to right, the output bins from top to bottom. Bins 1 through 20 correspond to points on the spline; bin 21 is for the noise process. The linear size of each square is proportional 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 $\sum_j x_j^1 = 0$.

$$\left.\frac{\partial^2 E}{\partial a \partial b}\right|_{p^*} = \sigma^{-2} \sum_i \frac{\partial}{\partial b} \left\{ \frac{\sum_j \phi_{ij}(z_j^1 - y_i^1)x_j^1}{\sum_k \phi_{ik}} \right\} \tag{A.17}$$

$$\begin{aligned} &= \sigma^{-2} \sum_i \left\{ \frac{\sum_j [-\sigma^{-2}\phi_{ij}(z_j^1 - y_i^1)^2 x_j^1 x_j^2 + \phi_{ij} x_j^1 x_j^2]}{\sum_k \phi_{ik}} + \right. \\ &\qquad \left. \frac{[\sum_j \phi_{ij}(z_j^1 - y_i^1)x_j^1][\sum_k \sigma^{-2}\phi_{ik}(z_k^1 - y_i^1)x_k^2]}{(\sum_k \phi_{ik})^2} \right\} \end{aligned} \tag{A.18}$$

$$= \sigma^{-2}\{N b_{12} - \sum_i [\sigma^{-2}(t^1 - y_i^1)^2 \sum_j x_j^1 x_j^2]\} \tag{A.19}$$

$$= \sigma^{-2} N b_{12}[1 - \sigma^{-2} d_{11}] \tag{A.20}$$

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 $\mathbf{t} = (t_x, t_y, t_x, t_y, \dots, t_x, t_y)^T$. 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.

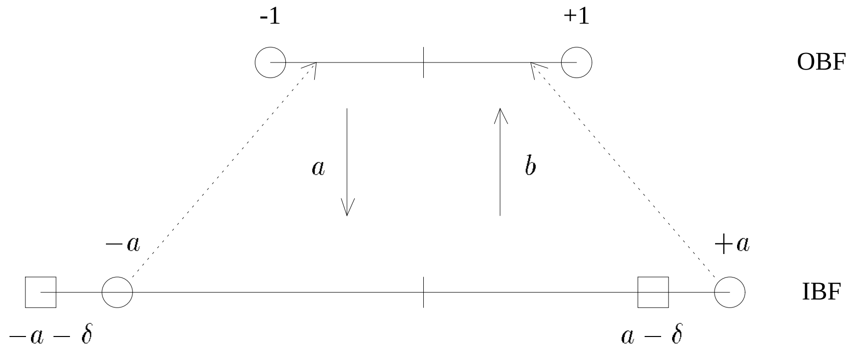

Figure B.1: Illustrating the effects of "shrinkage". The model consists of two points located at ± 1. An ideal image is obtained by scaling the model by a factor a. A noisy image has points at $-a-\delta$ and $a-\delta$. The diagram shows that the optimal inverse transformation b of the noisy data maps 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.