To appear as a chapter in :
"Spatial Vision in Humans and Robots"
eds. L. Harris and M. Jenkin, Cambridge University Press
Preprint date: May 1992

Hand-printed digit recognition using deformable models*

Christopher K. I. Williams, Michael D. Revow and Geoffrey E. Hinton
Department of Computer Science, University of Toronto
Toronto, Ontario, Canada M5S 1A4

Abstract

Hand-printed digits can be modeled as splines that are governed by a small number of control points. For each known digit, the control points have preferred home locations, and deformations of the digit are generated by moving the control points away from their home locations. Images of digits can be produced by placing Gaussian ink generators uniformly along the spline. Real images can be classified by separately fitting each digit model to the image, and picking the model that fits best. We use an elastic matching algorithm to minimise an energy function that includes both the deformation energy of the digit model and the log probability that the model would generate the inked pixels in the image. The model with the lowest total energy wins. If a uniform noise process is included in the model of image generation, some of the inked pixels can be rejected as noise as a digit model is fitting a poorly segmented image. The digit models learn by modifying the home locations of the control points.

1 Introduction

Hand-printed characters can take on a great variety of shapes, especially when they are produced by a diverse population of writers. This variability makes the use of rigid templates impractical for hand-printed character recognition. It has long been realized (e.g. Ullmann 1972, Burr 1981b) that this limitation can be overcome by using elastically deformable models, and recent work (e.g. Yuille 1990, Grenander, Chow and Keenan 1991) has provided a general framework for the problem. For images of single digits this framework implies that the best interpretation of an image is the model that minimizes an energy function that includes both the deformation energy of the digit model and the data misfit between the model and the image.

An alternative approach to digit recognition is based on statistical pattern recognition techniques; an example is the use of a feedforward neural network for ZIP code digit recog-

*This chapter is a revised and expanded version of Hinton, Williams and Revow (1992)

nition by le Cun *et al* (1990). This method requires the segmentation and normalization of the digit image before the recognition stage. However, in some cases, it is not possible to correctly segment and normalize the digits without using knowledge of their shapes, so to achieve close to human performance on images of whole ZIP codes we believe it will be necessary to use models of shapes to influence the segmentation and normalization of the digits. One way of doing this is to use a large cooperative network that simultaneously segments, normalizes and recognizes all of the digits in a ZIP code. A first step in this direction is to take a poorly segmented image of a single digit and explain the image properly in terms of an appropriately normalized, deformed digit model plus noise. The ability of the model to reject some parts of the image as noise is the first step towards model-driven segmentation.

2 Elastic models

Ideas on the use of deformable models for pattern recognition go back at least as far as the early 1970's (Ullmann 1972, Widrow 1973, Fischler and Elschlager 1973). Burr (1981a, 1981b) investigated several types of elastic matching procedures for character recognition problems, using dot and line models, and showed how a model could be progressively deformed to fit data. However, his methods were not designed to optimise an objective function trading off data-fit and model deformation energies. More recent work (e.g. Terzopoulos and Fleischer 1988, Yuille 1990, Mjolsness 1990, Grenander, Chow and Keenan 1991) has emphasized the concept of a model deformation energy and a probability distribution over shapes. The deformation energy has a very physical interpretation as being the amount of energy required to elastically deform the fundamental shape into the distorted shape. By using the Gibbs distribution from statistical physics it is possible to relate the probability of a configuration to the deformation energy by $P_{def} \propto e^{-E_{def}}$. This means that shapes that are distorted further away from the fundamental shape will have a lower probability. By using this probability distribution a single deformable template can model a great variety of instances of an object.

Our elastic models are based on splines. Each model contains parameters that define an ideal shape and also define a deformation energy for departures from this ideal. These parameters are initially set by hand but can be improved by learning. They are an efficient way to represent the many possible instances of a given digit.

Each digit is modelled by a deformable spline whose shape is determined by the positions of about 8 control points. Every point on the spline is a weighted average of four control points, with the weighting coefficients changing smoothly as we move along the spline.[1] To generate an ideal example of a digit we put the 8 control points at their home locations for that model. To deform the digit we move the control points away from their home locations. With a spline model it is easy to model topological variants of a digit by small changes in

[1]In computing the weighting coefficients we use a cubic B-spline and treat the first and last control points as if they were doubled. Bartels, Beatty and Barsky (1987) is a useful reference on the use of splines for geometric modelling.

the locations of the control points. For example, small changes in control point locations can deform the loop of a 2 into a cusp and then an open bend (see Figure 1). This advantage of spline models is pointed out by Edelman, Ullman and Flash (1990) who use a different kind of spline that they fit to character data by directly locating candidate control points in the image.

Currently we assume that, for each model, the control points have independent, radial Gaussian distributions about their home locations. Thus the probability of finding the control points within a small hypervolume δV of control point space is

$$P_{def} = \delta V \prod_c \frac{1}{2\pi\sigma_c^2} \exp\{-\frac{(\mathbf{x}_c - \mathbf{x}_c^h)^2}{2\sigma_c^2}\} \quad (1)$$

$$-\log P_{def} = -\log \delta V + N_c \log 2\pi\sigma_c^2 + \sum_c \frac{(\mathbf{x}_c - \mathbf{x}_c^h)^2}{2\sigma_c^2} \quad (2)$$

where $\mathbf{x}_c$ is the position of a control point, $\mathbf{x}_c^h$ is the *home* position of a control point, N_c is the number of control points and σ_c^2 is the variance of a control point. The last term in equation 2, which is proportional to the sum of the squares of the departures of the control points from their home locations, is the deformation energy (E_{def}).

The deformation energy function only penalizes shape *deformations*. Translation, rotation, dilation, elongation, and shear do not change the shape of an object so we want the deformation energy to be invariant under these affine transformations.[2] We achieve this by computing the deformation energy of each model its own "object-based frame". When we fit the model to data, we repeatedly recompute the best affine transformation between the object-based frame and the image (see section 4). The repeated recomputation of the affine transform during the model fit means that the shape of the digit is influencing the normalization.

Although we will use our digit models for recognizing images, it helps to start by considering how we would use them for generating images. The generative model is an elaboration of the probabilistic interpretation of the elastic net given by Durbin, Szeliski and Yuille (1989). Given a particular spline curve in the image, we space a number of "beads" uniformly along the spline. Each bead defines the centre of a Gaussian ink generator. The number of beads on the spline and the variance of the ink generators can easily be changed without changing the spline itself.

[2] Currently we do not impose any penalty on extremely sheared or elongated affine transformations, though this might improve performance (see section 7).

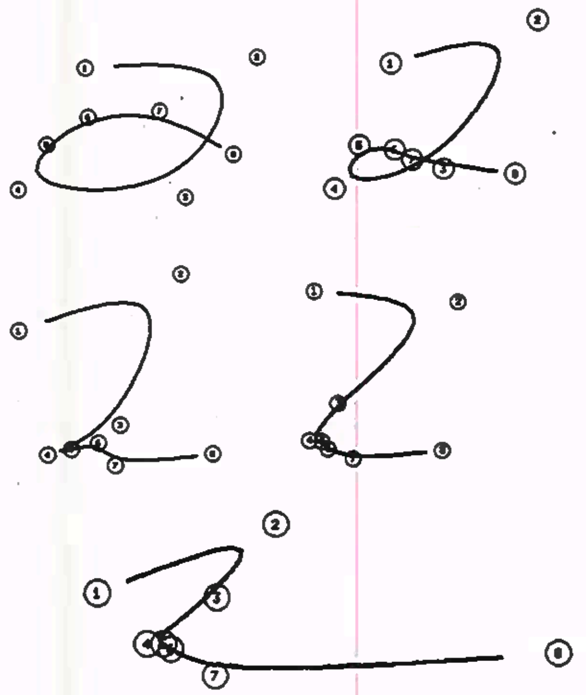

Figure 1: A cubic B-spline model of a 2 using eight control points. By varying the positions of the control points it is possible to change the loop of a 2 into a cusp and then an open bend. In the bottom example, control point number 8 has been moved to the right creating a long "tail".

The generative model

To generate a noisy image of a digit we would run the following procedure:

- Pick one of the digit models
- Pick a deformation of the model, in object-based coordinates, by choosing a location for each control point from under the Gaussian distribution defined by the model. The probability of picking a deformation is proportional to $e^{-E_{def}}$
- Pick an affine transformation from the model's intrinsic reference frame to the image frame (i.e. pick a size, position, orientation, slant and elongation for the digit). Map the object-based locations selected for the control points into image-based locations
- Compute the spline curve in image coordinates and space Gaussian generators uniformly along the spline
- Repeat many times:
  Either (with probability π_{noise}) add a randomly positioned noise pixel
  Or pick a bead at random and generate a pixel from the Gaussian
  distribution defined by the bead.

3 Recognizing isolated digits

We recognize an image by finding which model is most likely to have generated it. Each possible model is fitted to the image and the one that has the highest posterior probability $P(D|X)$ is the winner.

$$P(D|X) = \frac{P(D)}{P(X)} \int P(X|D,m)P(m|D)dm \qquad (3)$$

where X is the data, m is a vector of instantiation parameters which includes the affine transform variables and the control point locations, and D indexes a model. $P(D)$, the prior probability on model D and $P(X)$ are assumed to be the same for all models. We can approximate this integral by just considering the best fitting model instance, which has parameters m^*.[3] Letting E denote the negative logarithm of this approximate probability, we have

$$E_D = -\log P(m^*|D) - \log P(X|m^*, D) \qquad (4)$$

[3] In effect, we are assuming that the integral in equation 3 can be approximated by the height of the highest peak, and so we are ignoring variations between models in the width of the peak or the number of peaks. Using this approximation means that we are in fact finding the maximum a posteriori (MAP) estimator.

The first term factorizes into the sum of the deformation energies for each separate control point, because they have independent probability distributions (see equation 2). The second term can also be factorized into a sum of separate log probabilities for each inked pixel provided we assume that each inked pixel is generated by an independent sample from a probability density function defined by the Gaussian beads and a uniform noise field. The probability of a point sample landing within a particular pixel is the product of the probability density at the centre of the pixel and the area of the pixel.[4] The probability is the sum of the probabilities of all the possible ways of generating the point sample from the mixture of Gaussian beads or the uniform noise field.

$$P(i) = \frac{\pi_n}{N} + \frac{1-\pi_n}{B} \sum_{beads} P_b(i) \qquad (5)$$

where N is the total number of pixels, B is the number of beads, π_n is the mixing proportion of the uniform noise field, and $P_b(i)$ is the probability of the pixel i under Gaussian bead b.

4 The search procedure for fitting a model to an image

The objective for each model is to find a set of instantiation parameters m^* which minimize the total energy E. We search for this maximum by starting from an initial guess and then using a continuation method to home in on a good solution. The continuation method solves a sequence of progressively harder tasks with the solution to one task providing a good starting point for the solution to the next task in the sequence (Blake and Zisserman, 1987).

The initial position for a digit model is determined by calculating a minimal vertical rectangle around the data, and adjusting the offset and the x and y scale factors of the affine transformation so that the character model just fills that box. We start by finding a near optimal fit of the model to the data when the Gaussian beads all have a large variance. This involves only a few iterations of moving the control points and recomputing the affine transformation. At high variance, only a few beads are required to form a smooth ridge of higher probability along the spline, so the fit is very fast. After fitting at the highest variance, we slightly increase the number and reduce the variance of the beads according to a predetermined schedule and adjust the fit of the model to the data. The final fit of the model to the data is achieved after increasing the number of beads and reducing their variance four times, with several iterations at each variance. This fitting technique resembles the elastic net algorithm of Durbin and Willshaw (1987), except that our elastic energy function is much more complex and we are also fitting an affine transformation.

We have used conjugate gradient methods for optimizing the objective function E, but our preferred technique for the elastic matching is based on the Expectation and Maximiza-

[4] We assume that the probability density function varies linearly across the pixel.

tion (EM) algorithm of Dempster, Laird and Rubin (1977). The algorithm is guaranteed to not increase E as it adjusts the instantiation parameters.[5]

The Expectation step

Given the current locations of the Gaussian beads, compute the responsibility that each Gaussian has for each inked pixel. This is just the posterior probability of generating the pixel from that Gaussian given that it must be generated from one of the Gaussians or from the uniform noise field. The responsibility of bead b for pixel i is given by

$$r_b(i) = \frac{P_b(i)}{\sum_g P_g(i) + \frac{B\pi_n}{N(1-\pi_n)}} \tag{6}$$

We can think of each inked pixel as attached to each Gaussian bead by a spring whose stiffness is proportional to the responsibility of the bead for the pixel.[6] This mechanical analogy results from the fact that the log probability under a Gaussian is proportional to the squared distance from the mean, so it acts just like the energy of a spring.

The Maximization step

The M step of the EM algorithm requires the adjustment of the instantiation parameters to maximize the posterior probability (or equivalently minimise E), assuming that the responsibilities remain fixed. For our problem this would require solving a set of simultaneous cubic equations for the affine transformation and deformation parameters. This is too difficult, so we use a two stage procedure which decreases (or at least does not increase) the objective function at each stage. First, holding the affine transformation constant, we invert a 16×16 matrix to find the image locations for the 8 control points at which the forces pulling the control points towards their home locations are balanced by the forces exerted on the control points by the inked pixels.[7] These forces come via the forces that the inked pixels exert on the Gaussian beads. The 16 linear equations for the control point positions are derived by taking $\partial E/\partial \mathbf{x}_c = 0$ for each control point.

The second stage involves adjusting the affine transformation and deformations by holding the new *image* locations of the control points fixed. This is also a non-linear problem,

[5]When the variance of the Gaussians is decreased, it is quite possible that the objective function will be increased. However, if the variance is adaptively adjusted by the EM algorithm rather than being externally imposed, the objective function will not increase.

[6]The stiffness of the spring is also inversely proportional to the variance of the Gaussian.

[7]In the generative model presented above, the deformation occurs in object-based coordinates. For fitting the model to data it is more convenient to work entirely in image coordinates. We can achieve this by mapping the radial Gaussian prior distribution for each control point through the affine transformation to get an appropriately scaled elliptical Gaussian distribution in the image. This elliptical Gaussian defines the cost of deformations in image coordinates.

but we have found that choosing the affine transformation that minimizes the sum of the squared distances, in image-based coordinates, between the control points and their home locations gives satisfactory results. The second step is guaranteed not to alter the data-fit term because the data-fit depends only on the image positions and variances of the Gaussian beads, and these are uniquely determined by the image positions of the control points.

Some stages in the fitting of a model to data are shown in Figure 2. The search technique usually avoids local minima when fitting models to isolated digits, but if we detect an unsatisfactory fit, we try alternative starting configurations for the models. We use four other positions, translated right, above, left and below the original one. These are tried in turn, and the retry process stops when a fit is found that is not rejected. If all five possible starting positions are rejected, then the case is rejected.

There are three situations in which we reject a fit and try the search from a different initial position.

- We reject a fit when the posterior probabilities of both models are similar, i.e. if the absolute difference between the energies E_D (equation 4) of the models is small.[8]
- A fit is rejected when the "winning" model accounts for the data by becoming excessively deformed, i.e. if the deformation energy is large. See Figure 3(a) and (b).
- A model can find an incorrect fit while remaining relatively undeformed (Figure 3(c) and (d)). In many cases this problem can be related to an inadequacy in our definition of the data-fit energy. There is no explicit penalty for beads that generate pixels where there are none in the image. For example, the obviously incorrect fit in figure 3(c) should be rejected. One simple solution is to penalise beads that are far from any inked pixels. We used the penalty term

$$C = -\sum_{b \in \text{beads}} \log \sum_{\substack{i \in \text{inked} \\ \text{pixels}}} P_b(i) \tag{7}$$

A bead only makes a large contribution to this cost when all the inked pixels are far from the bead.[9] We reject cases where C is large for the winning model.

5 Learning the digit models

We have investigated two different learning techniques for adjusting the home positions and variances of the control points. In maximum likelihood (ML) learning each digit model is

[8]We can do this if we make energies insensitive to scale, see section 7.

[9]A more correct approach is discussed in section 7 under the heading "Explaining the white pixels".

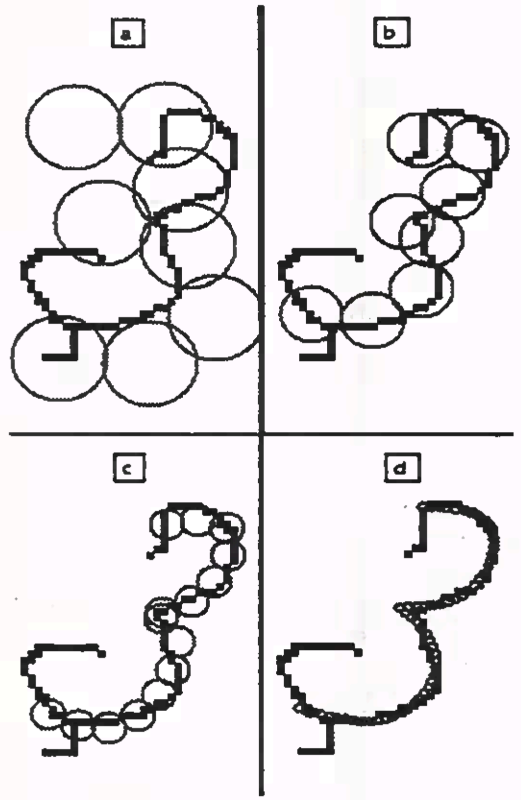

Figure 2: The sequence (a) to (d) shows some stages of fitting a model 3 to data. The grey circles represent the beads on the spline, and the radius of the circle represents the standard deviation of the Gaussian. (a) shows the initial configuration, with eight beads equally spaced along the spline. In (b) and (c) the variance is progressively decreased and the number of beads is increased. The final fit using 60 beads is shown in (d). The initial variance was about 65 times the final variance. We use about three iterations at each of five variances on our "annealing schedule". In this example, we used $\pi_{noise} = 0.3$ which makes it cheaper to explain the extraneous noise pixels and the flourishes on the ends of the 3 as noise rather than deforming the model to bring Gaussian beads close to these pixels.

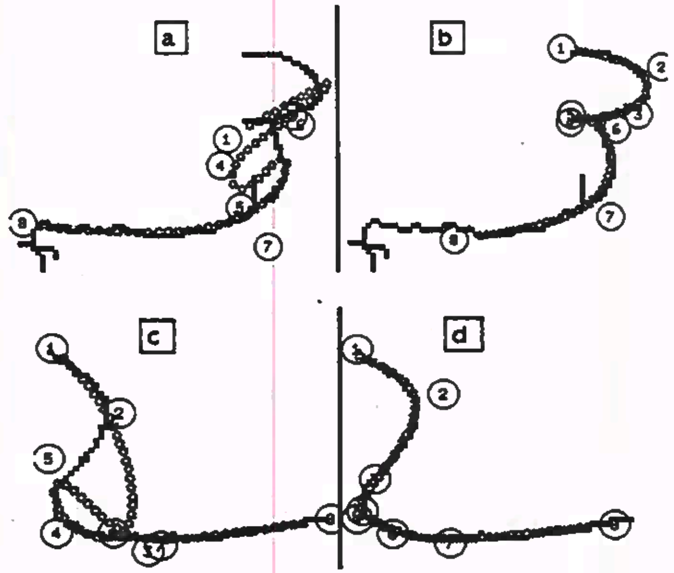

Figure 3: The final model configurations shown in (a) and (c) are rejected. In (a) the deformation energy exceeds the threshold, and the search is restarted from a different initial position, leading to the fit shown in (b). (c) is rejected because the the value of C is large, but an alternative starting position leads to the configuration shown in (d).

trained only on the images of that digit in the training set.[10] The objective is to adjust the model parameters (θ^D) so as to maximize the likelihood of generating those images from the model.

$$L_{ML} = \sum_{\substack{\text{training images} \\ \text{of class } D}} [\log P(X^D|\theta^D, m^*) + \log(m^*|\theta^D)] \tag{8}$$

where X^D is a training data image containing a digit of class D. This maximization is done iteratively, using EM updates given by $\partial L/\partial\theta_i^D = 0$. For example, the updated home location of each control point (in the object-based frame) is the average location of the control point in the final fits of the model of the digit to the instances of the digit, as illustrated in Figure 4.

[10] Actually, learning only occurs on those training cases where the correct model wins, because the model will tend to be excessively distorted when it has fitted wrongly.

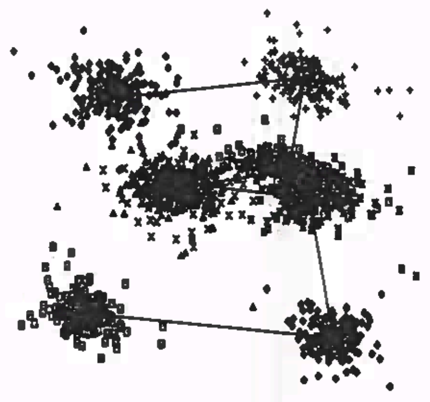

Figure 4: The figure illustrates the maximum likelihood learning procedure for the control point home locations of a 3 model. Each symbol cluster shows the final positions of a control point (in the object based frame) on each of the training cases. The solid lines connect the new control point home locations.

There is a potential difficulty with the calculation of $\partial L_{ML}/\partial\theta_j^D$. Changing a parameter of an elastic model causes a simple change in the energy of the configuration that the model previously settled to, but the model no longer settles to that configuration. So it appears that we need to consider how the energy is affected by the change in the configuration. Fortunately, derivatives are simple at an energy minimum because small changes in the configuration make no change in the energy (to first order). Thus the inner loop settling leads to simple derivatives for the outer loop learning, as in the Boltzmann machine (Hinton, 1989).

An alternative to maximizing the likelihood of the image given the digit is to maximize the mutual information between the correct digit class and the probabilities assigned to the various classes by the digit models. The maximum mutual information (MMI) criterion emphasizes correct discrimination rather than correct modeling of the image data, and it generally leads to better discriminative performance. We do discriminative learning by

maximising

$$L_{MMI} = \sum_{\substack{all\\images}} \log P(C|X) \tag{9}$$

where

$$P(C|X) = \frac{e^{-E_C}}{\sum_D e^{-E_D}} \tag{10}$$

is the probability of the correct digit C.

We have tried both of these learning techniques, and have obtained similar results with each one. Discriminative learning requires that we fit each model to each image; in contrast, with maximum likelihood training we only have to fit the correct digit model to each image, so it is much faster. If the distribution of images really can be modelled by choosing the appropriate parameters for our set of digit models, and if the fitting and learning processes do not get trapped at local optima, then maximum likelihood learning does just as well as discriminative learning at discrimination (Brown, 1987). If, however, the distribution of images is not well modelled by any set of parameters for our digit models, discriminative learning will generally yield better discrimination, since it will not waste parameters trying to model aspects of the images that are irrelevant to discrimination. The fact that MMI was no better than ML is weak evidence that the type of generative model we are using may be fairly good.

6 Results on the hand-filtered dataset

We are trying out the scheme out on a relatively simple task—we have a model of a two and a model of a three, and we want the two model to win on "two" images, and the three model to win on "three" images.

We have tried many variations of the character models, the preprocessing, the initial affine transformations of the models, the annealing schedule for the variances, the mixing proportion of the noise and the control point variances.

We took images of five digit ZIP codes from the training portion of the United States Postal Service Handwritten ZIP Code Database (1987).[11] The images were preprocessed to eliminate variations due to stroke-width and paper and ink intensities. First, a standard local thresholding algorithm (White and Rohrer, 1983) is used to make a binary decision for each pixel. We then segment the image by cutting out five boxes, each of which contains one of the five largest connected components. The connected components are thinned, leaving an image with a spine one pixel thick.[12] We manually checked all images produced, removing those in which this procedure failed (e.g. touching digits).

[11]Made available by the USPS Office of Advanced Technology

[12]Thinning allows us to reduce the variability in the data-fit term due to thin or thick pen strokes.

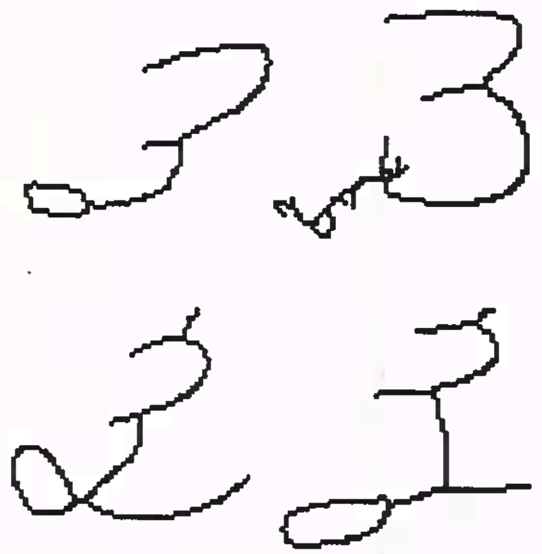

Figure 5: The four cases in the test set which were wrongly classified. The bottom two cases have "flourishes" (see section 7)

Clearly, the dataset is not comparable to sets used by other researchers. Since our initial goal was to find a way of accurately characterizing the shapes of digits, we decided to restrict our initial experiments to two's and three's that are clearly discriminable by humans. Our system now achieves very good discrimination on these examples. Of course, our results give very little indication of how well our method would perform on data that includes examples of all of the digits and examples that people cannot reliably discriminate.[13]

The performance of the system using the hand-crafted models was 18 errors and 1 reject on a test set of 304 two's and 304 three's. This improved to 4 errors and 2 rejects after learning. The four errors are shown in Figure 5. The training set has 418 cases, and we have a validation set of 200 cases to tell us when to stop learning. Most of the improvement in performance occurred after the first pass of learning. Figure 6 shows the effect of maximum likelihood learning on the home positions of the control points. Similar results were obtained

[13]The US Postal Service will soon be releasing a segmented database which will allow us to make better comparisons with other methods.

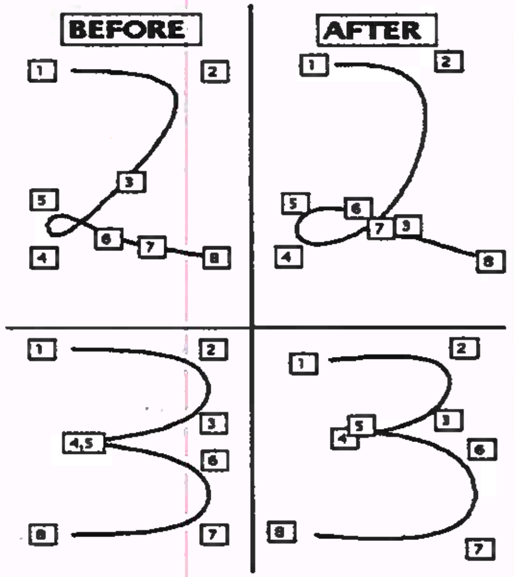

Figure 6: The two and three models before and after learning. The control points are labelled 1 through 8. We used maximum likelihood learning in which each digit model is trained only on instances of that digit.

with discriminative training. It would also be possible to adapt the variances of each control point, although we did not do this for the simulations reported.

7 Discussion

Spacing the beads

In determining where on the spline to place the Gaussian beads, we initially used a fixed set of blending coefficients for each bead. These coefficients are the weights used to specify the bead location as a weighted centre of gravity of the locations of 4 control points. Unfortunately this yields too few beads in portions of a deformed digit that have been greatly stretched but are still governed by just a few control points (like the elongated tail of the bottom 2 in Figure 1). So these portions are expensive to explain even though the spline fits them well. Performance was much improved by spacing the beads uniformly along the spline. However, using this uniform spacing of beads does mean that the search procedure (see section 4) could conceivably increase the objective function, as the repositioning of the beads is not taken into account when the new control point positions are calculated.

Penalizing Affine transformations

In the research reported here we do not penalize extreme elongations, shears, or rotations. In effect we have assumed that all affine transformations of a digit are equiprobable. It would be better to penalize unlikely affine transformations by associating an additional deformation energy with the affine transformation itself. This energy, which could be learned, would represent the negative log of the prior probability of the affine transformation.

Having an explicit representation of the affine transformation of each digit should, in the future, prove very helpful for recognizing multiple digits, since it will allow us to impose a penalty on differences in the affine transformations of neighbouring digits.

Invariance of recognition under affine transformations

There is an obvious way to ensure that the classification of a digit is invariant under affine transformations of the original image. We simply define the deformation energy and the data-fit energy in such a way that they are invariant under affine transformations. This ensures that the equilibrium configuration at which the deformation forces balance the data-fit forces will remain invariant, and also the energy differences between the fits of the different digit models will remain the same.[14]

It is easy to make the deformation energy invariant: We simply define the deformation in the object-based coordinate system. Affine transformations of the original image are then balanced by the affine transformation from the object-based to the image coordinate system, thus leaving the positions of the control points unchanged in the object-based system. It is also fairly easy to make the data-fit energy invariant if we assume that when an affine transformation is applied to the original image, it is also applied to the grid that is used to turn a continuous intensity field into discrete pixels. So when the image of the digit gets bigger, each pixel gets bigger and the number of pixels remains the same. We must also apply the same affine transformation to the Gaussian beads so that they scale up appropriately.[15]

[14]Actually, since the fitting process may get trapped at local energy minima, it is not sufficient to make the energy function invariant. It is also necessary to ensure that the whole fitting process, including the starting configuration of the model, is appropriately transformed.

[15]The obvious way to do this is by defining the Gaussian beads in object-based coordinates and then mapping them into the image through the affine transformation. One problem with doing this is that different digit models may use very different affine transformations leading to very different variances for their Gaussian beads. We therefore scale the variances of the circular beads in the image by the determinant of

Unfortunately, the digitization process has its own scale, orientation, shear and elongation and these do not change when affine transformations are applied to the original intensities. Given that the digitization process defines a natural scale, we could simply give up on the idea of achieving invariant recognition. Consider, for example, a digit that is so small that it only occupies a few pixels. There is not enough data to justify a highly improbable deformation so it is not clear that we want the same balance between data-fit and deformation terms when a digit is very small. Alternatively, we can try to compensate for the fact that the digitization process does not undergo the appropriate affine transformation. The major difficulty is that the number of pixels in a digit increases as it gets larger and this increases the number of data-fit forces. We can compensate for this by simply treating each inked pixel as some fraction of an "ideal" pixel. We could try to calculate this fraction by using the determinant of the affine transform from object-based to image coordinates, but then models with very different determinants would weight the inked pixels very differently. We have found that it is better to simply down-weight each pixel in proportion to the number of pixels in the image.[16]

One more correction is necessary in order to make the data-fit energy independent of scale. The cost of explaining a pixel as noise must remain constant, and this cost involves the total number of pixels in the image (see equation 5). Scale invariance requires that we use the number of ideal pixels, so in addition to down-weighting each actual inked pixel in proportion to the number of inked pixels, we should also treat N in equation 5 as a constant that does not change as the size of the image increases.

More elaborate spline models

By using spline models, we build in a lot of prior knowledge about what characters look like, so we can describe the shape of a character using only a small number of parameters (16 coordinates and 8 variances). This means that the learning is exploring a much smaller space than a conventional feed-forward network. Also, because the parameters are easy to interpret, we can start with fairly good initial models of the characters. So learning only requires a few updates of the parameters. We can probably afford to use more parameters in the digit models. Obvious extensions of the deformation energy function include using elliptical Gaussians for the distributions of the control points (5 parameters instead of the current 3), or using full covariance matrices for neighbouring pairs of control points.

the initial affine transformation (found by fitting a rectangular box around the data). Using the mechanical view, it may seem strange that the energies can remain unchanged even when distances, and hence the extensions of the springs, increase by a factor of s. Spring stiffnesses decrease by a factor of s^2 and forces therefore *decrease* by a factor of s. The work done in getting from one configuration to another remains invariant because it is the product of the force and the distance.

[16]This has the desired effect when an image is scaled up. However, if an image only contains a few inked pixels, it can deform a model just a strongly as an image of the same size containing many more pixels. Also, if there are a lot of noise pixels in the image they can reduce the ability of the other pixels to deform a model.

More elaborate ways of generating ink from splines

One obvious generalization is to use elliptical rather than circular Gaussians for the beads. If strokes curve gently relative to their thickness, the distribution of ink can be modelled much better using elliptical Gaussians. However, an ellipse takes about twice as many operations to fit and is not helpful in regions of sharp curvature. Our simulations suggest that, on average, two circular beads are more flexible than one elliptical bead.

At high variance, our generative model is obviously inadequate because the inked pixels that it produces will be very scattered and will have no tendency to cohere into sharp strokes. Our only way of achieving coherence is to lower the variance. At the beginning of the fitting process, however, we know that our estimates of the position of the pieces of the stroke are unreliable but we also know that the inked pixels should cohere into a sharp stroke. No single variance can capture both these kinds of knowledge.

There is an interesting way of allowing the generative model to assign much higher probability to more coherent images, even when the Gaussians have high variance. Instead of generating ink directly from the Gaussians, we use the Gaussians to generate probabilities that individual pixels are inked. Then, we treat these probabilities as bias terms in a Markov Random Field (MRF) that stochastically assigns one of the two labels "figure" or "noise" to each inked pixel. The MRF uses a locally defined energy function that penalizes label assignments in which nearby inked pixels have different labels. A mean field approximation of this MRF can easily be combined with the elastic fitting of a digit model. On each iteration of the elastic fit, the digit model provides new bias terms for each inked pixel in the MRF, and a few iterations of the mean field MRF are then used to update the label probabilities on the inked pixels. The data-fit term in the cost function is modified to reflect the probability of generating the inked pixels with a particular figure/noise labelling.

As a consequence of its preference for coherent images, the MRF method makes it much cheaper to explain noise that forms continuous lines or blobs than it is to explain the same number of randomly scattered noise pixels. Also, it assigns heavier penalties to places where the labeling switches within a continuous line of inked pixels than it does to places where the figure and the noise are not connected. We are currently investigating this technique.

Throughout this paper we have assumed that the data that must be fitted consists of inked pixels. But there is nothing to prevent a similar approach being applied to preprocessed images that contain short oriented stroke segments or stroke-edge segments. Each Gaussian bead can represent a probability distribution over oriented features. We simply add an orientation distribution whose mean is defined by the orientation of the spline at that bead and whose variance is influenced by the curvature of the spline. Unfortunately, our current implementation of this approach is prone to get trapped at local optima when a section of the spline (typically in the loop of a 2) gets fitted to the bar segment data with an orientation error of 180 degrees.

Structured noise and flourishes

Presegmented images of single digits contain many different kinds of noise that cannot be eliminated by simple bottom-up operations. These include descenders from the line above, deliberate underlines, bits of other digits, corrections, dirt in recycled paper, smudges, and misplaced postal franks. To really understand the image we probably need to model a wide variety of structured noise. We are currently experimenting with one simple way of incorporating noise models. We first fit a mixture of a digit model and a uniform noise field to a poorly segmented image of a digit. Then we try to fit more complicated noise models to the residual noise. A good fit greatly decreases the cost of that noise and hence improves this interpretation of the image.

The idea of trying to find better explanations of the residual noise image can also be used to deal with the flourishes that frequently appear on the ends of digits. The basic model of a digit does not include these flourishes and at least half the errors in our test set are caused by the excessive cost of explaining large flourishes using the uniform independent noise model. Figure 7 shows an example of this type of error which occurred in the training set. The 3 model made a reasonably good fit to the data, but was unable to account for the flourish on the bottom end of the data and was forced to explain each of those pixels as an independent piece of noise. The 2 model used the flexibility of its loop to model the flourish, leaving only the cusp region of the data to be explained as noise.

We have briefly experimented with a way of handling flourishes. Figure 7 shows the residual image obtained by subtracting out the portion of the image accounted for by the digit model. Top and bottom flourish models are then initialized. Each flourish is allowed three control points (so it can easily model a curved stroke). One control point is strongly tied to its "home location" which is coincident with the end control point of the digit model. The other two control points can easily be moved by the data-fit forces. Hence flourishes have much weaker priors on their shapes as compared to digit models. This makes good initialization even more critical than for digit models. A separate affine transformation for the flourish models is unnecessary as they only have deformation energy associated with a single control point.

Ultimately, of course, we want to use a less greedy algorithm in which the flourish and structured noise models are fitted *during* the fitting of each digit model, but we currently have no efficient way of doing this because it is hard to choose sensible initial configurations for the flourish and noise models before we know which bits of the image are noise and which bits are the digit. This objection does not apply to the very primitive uniform noise model in which noise pixels are generated independently, which is why we *can* use this noise model during the fitting of the digit. Nor does it apply to the Markov Random Field model of noise described above.

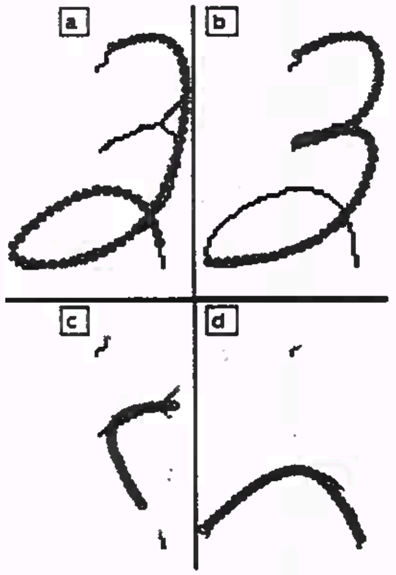

Figure 7: The top two panels show the final fit of the models to an image of a three which has a flourish at the bottom; (a) 2 model (b) 3 model. In (a), the 2 model accounts for most of the flourish data and hence has a lower data-fit energy. The lower panels illustrate the residual images obtained by subtracting off the portion of the image accounted for by the basic model. The fit of flourish models is also shown.

Explaining the white pixels

A significant drawback of our generative 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. Some of the classification errors seem to be caused by this weakness of the generative model. 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

beads and the uniform noise process (see equation 5). We imagine taking N_B samples from this distribution, where N_B is the number of inked pixels in the image[17] 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 inked, and its complement $\hat{P}(i)$ is the predicted probability that the pixel is inked.

$$\hat{P}(i) = 1 - (1 - P(i))^{N_B} \tag{11}$$

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

$$\log P(X|m, D) = \sum_{\substack{i \,\in\, inked \\ pixels}} \log \hat{P}(i) + \sum_{\substack{j \,\in\, uninked \\ pixels}} \log(1 - \hat{P}(j)) \tag{12}$$

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

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:

$$\log P(X|m, D) \simeq \sum_{\substack{i \,\in\, inked \\ pixels}} \log P(i) + N_B \log N_B \tag{13}$$

Note that the term involving N_B will be the same for all models, and can therefore be ignored. So at high variance, this generative model is almost the same as the one we are already using. At low variance, the two generative models are very different in the way they penalize different fits. In particular, the second generative model makes it more expensive for parts of an instantiated digit model to lie in white space. We are currently experimenting to see whether this improves discrimination.

Because there are so many more white pixels, the second generative model appears to require much more computation, but this is an illusion. At high variance, where many white pixels fall under each Gaussian bead, we 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. So we never have to deal with the product of all the beads with all the pixels.

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

A completely different elastic model

Before we tried using splines to model digits, we used models that consisted of a fixed number of Gaussian beads with elastic energy constraints operating between neighbouring beads. The deformation energy was described by three kinds of terms—displacements of the beads away from their home positions, changes in the distance between pairs of beads and changes in the curvature of triples of beads. With this type of energy function, we had great difficulty using a single model to capture topologically different instances of a digit. For example, in Figure 1 the sign of the curvature reverses as the loop of the 2 changes to a cusp and then to an open bend.

Spline models also make it easy to increase the number of Gaussian beads as their variance is decreased. This coarse-to-fine strategy is much more efficient than using a large number of beads at all variances, but it is much harder to implement if the deformation energy explicitly depends on particular bead locations, since changing the number of beads then requires a new function for the deformation energy.

8 Conclusion

One of our main motivations in developing elastic models is the belief that a strong prior model should make learning easier, should reduce confident errors, and should allow top-down segmentation. Although we have shown that elastic spline models can be quite effective, we have not yet demonstrated that they are superior to feedforward nets and there is a serious weakness of our approach: Elastic matching is slow. Fitting the models to the data takes *much* more computation than a feedforward net. So in the same number of cycles, a feedforward net can try many alternative bottom-up segmentations and normalizations and select the overall segmentation that leads to the most recognizable digit string. In the long run, however, we expect that it will be more efficient to use shape knowledge to guide segmentation, rather than just using it to filter out poor segmentations.

Acknowledgements
This research was funded by Apple and by the Ontario Information Technology Research Centre. We thank Peter Brown for advice on Gaussian mixture models, Allan Jepson and Richard Durbin for suggesting splines, David MacKay for insights on Bayesian methods and the members of the Toronto Connectionist Research Group for many helpful discussions. Geoffrey Hinton is the Noranda fellow of the Canadian Institute for Advanced Research.

References

Bartels, R. H., Beatty, J. C., and Barsky, B. A. (1987). *An Introduction to Splines for use in Computer Graphics and Geometric Modelling*. Morgan Kaufmann.

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

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.

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.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Proc. Roy. Stat. Soc.*, B-39:1–38.

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. *Internat. Journal of Comput. Vision*, 5(3):303–331.

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

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

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

Hinton, G. E., Williams, C. K. I., and Revow, M. D. (1992). Adaptive elastic models for hand-printed character recognition. To appear in *Advances in Neural Information Processing Systems 4*, J. E. Moody, S. J. Hanson and R. P. Lippmann eds, San Mateo, CA: Morgan Kaufmann.

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 *Advances in Neural Information Processing Systems 2*, pages 396–404. Morgan Kaufmann.

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.

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

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

White, J. M. and Rohrer, G. D. (1983). Image thresholding for optical character recognition and other applications requiring character image extraction. *IBM J. Res. Develop.*, 27(4):400–410.

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

Yuille, A. L. (1990). Generalized Deformable Models, Statistical Physics, and Matching Problems. *Neural Computation*, 2(1):1–24.