Using Generative Models for Handwritten Digit Recognition

Michael Revow, Christopher K.I. Williams, and Geoffrey E. Hinton

Abstract—We describe a method of recognizing handwritten digits by fitting generative models that are built from deformable B-splines with Gaussian "ink generators" spaced along the length of the spline. The splines are adjusted using a novel elastic matching procedure based on the Expectation Maximization (EM) algorithm that maximizes the likelihood of the model generating the data. This approach has many advantages. 1) After identifying the model most likely to have generated the data, the system not only produces a classification of the digit but also a rich description of the instantiation parameters which can yield information such as the writing style. 2) During the process of explaining the image, generative models can perform recognition driven segmentation. 3) The method involves a relatively small number of parameters and hence training is relatively easy and fast. 4) Unlike many other recognition schemes, it does not rely on some form of pre-normalization of input images, but can handle arbitrary scalings, translations and a limited degree of image rotation. We have demonstrated our method of fitting models to images does not get trapped in poor local minima. The main disadvantage of the method is it requires much more computation than more standard OCR techniques.

Index Terms—Deformable model, elastic net, optical character recognition, generative model, probabilistic model, mixture model.

1 Introduction

The conventional statistical approach to performing classification is to use a discriminant classifier that constructs boundaries which discriminate between objects of different categories. An alternative approach is to use generative models. This paper explores the use of generative models for recognizing handwritten digits. In the simplest version there is one model for each digit. Given an image of an unidentified digit the idea is to search for the model that is most likely to have generated that image. This approach has the attractive property that, in addition to providing a label, it can also say something about the particular way in which the digit is instantiated. So, in some sense, it explains the image rather than just labeling it. This is important when the recognizer forms part of a larger computer vision system since there may be interest in more than just the labels. For example, given a roughly segmented image of a single digit we may want to know which parts of the image represent the digit and which parts are caused by noise or by some incorrectly segmented neighboring digit. We may also want to know the pose of the digit (i.e., its position, size, orientation, shear, and elongation) so that we can check for consistency with its neighbors.

- *M. Revow and G.E. Hinton are with the Department of Computer Science, University of Toronto, 6 Kings College Road, Toronto, Ont., M5S 3H5. Canada. E-mail: {revow; hinton}@cs.toronto.edu.*
- *C.K.I. Williams is with the Department of Computer Science and Applied Mathematics, Aston University, Birmingham B4 7ET, UK. E-mail:c.k.i.williams@aston.ac.edu.*

Manuscript received Aug. 31, 1994; revised Apr. 22, 1996. Recommended for acceptance by R. Kasturi.
For information on obtaining reprints of this article, please send e-mail to: transpami@computer.org, and reference IEEECS Log Number P96047.

We chose unconstrained handwritten digit recognition because it is a task of great practical importance for which there are standard databases that allow different approaches to be compared. It also has the attractive property that there are only ten different classes so it is feasible to explore all ten different ways of generating each unidentified digit image. Although handwritten digit recognition is an easier task than general three-dimensional object recognition, it retains, albeit in reduced form, many of the problems associated with general computer vision such as variability in shape and pose, overlapping objects and both structured and unstructured noise.

The paper is organized as follows: Following a brief review of some past approaches to optical character recognition, we discuss elastic models which have been used for at least two decades to deal with signal and image variability. In Section 3, we introduce our basic elastic model for handwritten digits. We use the probabilistic interpretation of elastic models introduced in an analysis of the elastic net algorithm [1]. In Section 4, we show how the underlying parameters of the models may be learned. Section 5 discusses refinements of the basic ideas. Section 6 describes the performance of our system on a realistic database of handwritten digits. The final two sections discuss some implications of the approach and present conclusions.

2 Review of Past Work

We will not attempt to review here the voluminous work on optical character recognition that has spanned more than three decades (useful reviews can be found in [2], [3]). However, it is helpful to summarize the trends. Most researchers have adopted the classical pattern recognition approach in which image pre-processing is followed by

feature extraction and classification. There have been many variations, but these may be roughly described using two dimensions: statistical/structural and global/local.[1] As an example of a global, statistical approach, [4] extracts eight central and two raw moments as features. On the other hand the recognizer used by Lam and Suen [5] uses local features. They extract local geometric primitives consisting of line segments and convex polygons and use these as input to a structural classifier. Others extract topological features which depend on the global properties of the data. For example, Shridhar and Badreldin [6] use features derived from the character profiles in the image. They then feed these features into a tree classifier. More recently there have been a number [7], [8], [9] of successful attempts to automatically learn appropriate local features using feed forward neural networks. Some researchers [10], [3], [11] have boosted performance using combinations of classifiers.

Significant progress has been made in OCR. On a standard database of lightly constrained pre-segmented handwritten digits the very best systems achieve error rates of about 1.5% with no rejections [12]. But more work is required to match human performance, especially on unsegmented strings of digits. We hypothesize that in order to achieve human performance without astronomically large training sets, recognizers must embed some form of prior knowledge about the objects they expect to find in images. This is common in structural systems but rarer in statistical systems. There have been some statistical systems that allow for typical digit transformations [13], but discriminant classifiers generally do not address the issue of explicitly "explaining the data". This leads to a number of weaknesses that may limit the achievable performance:

1) Conventionally, a recognizer does not help to guide segmentation by dividing the image into significant and irrelevant parts. So a system typically [14] tries many candidate segmentations and all the recognizer can indicate is whether a particular segmentation leads to confident recognition. In general, this type of hypothesize-and-test search procedure is much less efficient than a procedure that can use information from the recognition to refine the segmentation hypothesis.
2) Statistical recognizers can occasionally confidently classify images that do not look anything like a character [15]. This can be ameliorated by training the system to reject junk images [16], but it is hard to get a good sample of rare types of junk.
3) Systems that do not incorporate any prior knowledge about the shapes of characters must learn all their knowledge from the training examples. We already know that digits are composed of one-dimensional strokes and so it seems wasteful to use up training data to learn this.
4) A recognizer that "understands" an image should be able to not only label it with the correct class, but should also be able to return the instantiation parameters such as the position, size, orientation, shear and elongation. For handwritten digits we may also want information on the writing style since this is occasionally crucial in disambiguating other digits in the same string.

Motivated by the success of model-based shape recognition in overcoming some of these shortcomings [17], we have investigated the use of deformable elastic models for handwritten digit recognition [18]. Models of this general type have been used in computer vision since the early 1970s. Ullmann [19] discusses the idea of finding a distortion mapping from a test image to a stored template such that there is *correspondence* between like features rather than exact matches. Widrow [20] also suggests the idea of using rubber templates to achieve fuzzy matches to a variety of natural objects and waveforms.

Burr presents an iterative framework for computing elastic matches in dot and grey-scale images [21] and line drawings [22]. Using a coarse-to-fine matching strategy he shows how an image can be progressively deformed under the influence of misalignment force fields to fit another image. In a later version [23], global size and rotation adjustments were included. The method has been adapted to match tomographic [24] and thermographic images [25]. One weakness with the approach is that it does not allow the amount of deformation to be traded off against the fidelity of the data match. It also has no principled way of handling noise or missing data.

Bajcsy and co-workers [26], [27] integrate the notion of a trade-off between data fit and deformation in their multiresolution elastic matching scheme for registering an image with respect to a template. They consider a test image to be drawn on an elastic membrane. The membrane is subjected to external forces which are proportional to the gradient of the similarity measure. The system iterates until an equilibrium exists between the forces trying to increase the similarity measure (a measure of cross-correlation between the two images) and the restraining forces arising from the elastic properties of the membrane. The multiresolution approach is attractive as it initially concentrates on achieving large-scale registration between the images with fine-scale matching coming later in the process.

Early work by Fischler and Elschlager [28] described a model with local (data fit) and global (model deformation) energy terms. Their model is composed of (rigid) features whose spatial arrangement is constrained by springs and hence the deformation is related to the energy required to stretch or compress these springs. Their matching procedure works on a coarse scale, but it is scale dependent and degrades in the presence of noise [29]. The facial feature model example they used has been extended by Yuille [29], who constructs a more detailed descriptions of the feature shapes and global matching criteria in terms of peak, valley and edge intensities. In addition, the original dynamic programming search was replaced with a gradient method. From an image explanation point of view this type of matching scheme is deficient as it does not account for the entire image. Instead of ensuring that every part of the *image* is explained by the model (or explicitly attributed to some additional noise process) the matching process tries to

1. These are broad terms applied to the object recognizer as a complete entity. Obviously, feature selection and classifier design may be independently described.

ensure that every part of the *model* is supported by some part of the image and a match may be good even though it leaves large parts of the image unaccounted for. "Snakes" [30] use different shape constraints, but also attempt to match each part of the model to some part of the image rather than vice versa. Point distribution models [31], recognize the importance of doing both types of matching, i.e., the model must be *supported* by the data and the model should *explain* the data.

The digit models we propose have a sound generative probabilistic basis and explicitly incorporate much prior knowledge of handwritten digits, for example, that they are made up of strokes and that they are globally invariant to affine transformations, unlike other implementations [13] which attempt to achieve only local invariance.

3 Matching Elastic Spline Models to Images

3.1 Overview

Each of the ten digits has its own elastic model.[2] A digit-image is recognized by choosing the elastic model which best matches the image. During the matching process, the model is deformed in an attempt to ensure that every piece of ink in the image is close to some part of the model. The fidelity of the final match depends on the amount of deformation of the model, the amount of ink that is attributed to noise, and the distance of the remaining ink from the deformed model.

Unlike the approach taken in many OCR systems, we do not pre-process images in order to remove the effects of translation, scale, rotations, shear, etc. Instead we handle arbitrary global affine transformations of the image by defining the model in an "object-based" frame which is mapped through an affine transformation into the "image-frame." The affine transformation is refined during the matching process so that knowledge about the shape of the digit can influence the choice of affine transformation. This is not possible if normalization precedes recognition. Affine transformations are not penalized during the matching process, so deformations are only used to handle true variations in shape that cannot be accommodated by global affine transformations.

Similarly, we do not assume that the image has been perfectly segmented. The matching process decides which pieces of the model correspond to which pieces of the image and it can explicitly reject some parts of the image as noise. Thus knowledge about the shape can be used to refine the segmentation.

What we have just described is an instantiation of the general framework of *generative* or *latent variable* models [32]. The key idea is that the manifest variables are attributable to a smaller number of underlying *hidden* or *latent* variables. In our case, the manifest variables are the pixels and the hidden variables are the positions of the elastic model's control points in the image-frame. Section 3.2 describes the elastic model. Section 3.3 gives the underlying probabilistic interpretation of how the model generates an image from the hidden variables (required for computing the log-likelihood of the image given a model instantiation). An algorithm for the more difficult problem of inferring the hidden variables from the manifest variables is presented in Section 3.4.

3.2 Elastic Spline Models

We model each digit with a uniform, cubic B-spline [33]. Each model has at most 8 control points.[3] Let $\mathbf{X} \equiv \{x_1, x_2, \ldots, x_n\} = \{x_1, x_2, \ldots, x_{2n-1}, x_{2n}\}$ denote an instantiation of the model in terms of its n control points. The ith control point is located at (x_{2i-1}, x_{2i}). Similarly, $\mathbf{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_n\}$ indicates the *home* or undeformed control point locations and $\mathbf{Y}$ the affine transformation with its six degrees of freedom. The location of any point,[4] $s(b)$, on the spline can be written as a linear function [33] of the control points locations.[5]

$$s(b) = \sum_{l=1}^{n} \gamma_l(b) x_l \tag{1}$$

Because of the local control feature of B-splines some of the coefficients, $\gamma_l(b)$, will be zero. For future convenience, we also write (1) as:

$$s(b) = \begin{pmatrix} \sum_{j=1}^{2n} \gamma_j^x(b) x_j \\ \sum_{j=1}^{2n} \gamma_j^y(b) x_j \end{pmatrix}.$$

To generate an ideal example of a digit we put the control points at their home locations. To deform the digit we move the control points away from their home locations. Assuming a Gaussian distribution for these deformations, the probability of the n control points lying within a small hypervolume δV is approximately:

$$P(\mathbf{X}) = \delta V \frac{1}{(2\pi)^n |\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{X}-\mathbf{H})^T \Sigma^{-1}(\mathbf{X}-\mathbf{H})\right] \tag{2}$$

where Σ is the covariance matrix of the distribution. Thus, a single deformable model defines an entire probability distribution across shape instances.

Following [1] and [34] we define the *deformation energy*, E_{def}, to be the negative log probability of the deformation.

$$E_{def}(\mathbf{X}) = \frac{1}{2}(\mathbf{X}-\mathbf{H})^T \Sigma^{-1}(\mathbf{X}-\mathbf{H}) + \frac{1}{2}\log|\Sigma| + \text{const} \tag{3}$$

Splines are a convenient method for modeling handwritten digits as it is easy to incorporate topological variations. For example, small changes in the relative locations of the control points can turn the loop of a 2 into a cusp or an open bend (Fig. 1). This advantage of spline models is pointed out in [35] where a different kind of spline is used to fit on-line character data by directly locating candidate control points on strokes in the image. It is lost (as pointed out in [36]) when models based more directly on Durbin and Willshaw's elastic net are employed [37].

2. C-code implementing the model is available from http://www.cs.toronto.edu/~revow.

3. The model of a one needs only three control points while the seven-model uses five.

4. The spline is a one-dimensional continuous curve parameterized by b. In the development, we consider a discrete version.

5. We treat the first and last control points as if they are doubled.

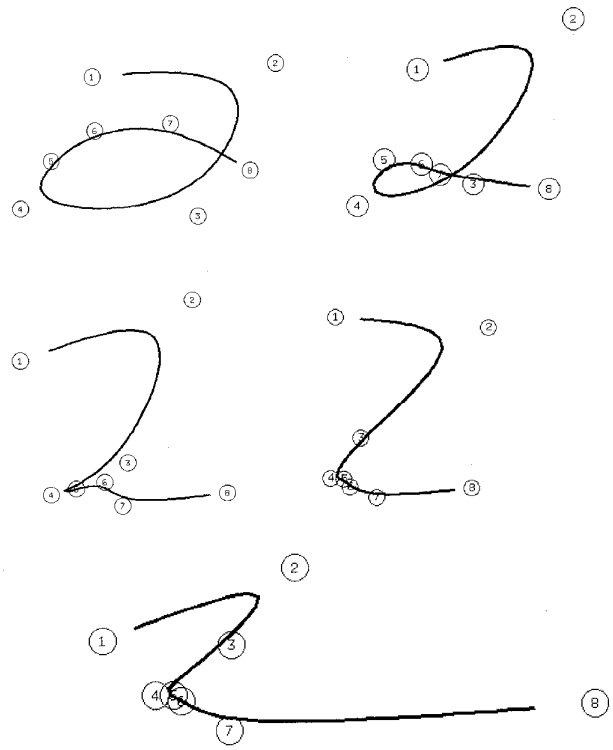


Fig. 1. Illustrates the flexibility of spline models to capture topological variations. The large loop of the 2 can smoothly decrease in size and eventually become a cusp. The control points are labelled 1 through 8.

3.3 Generative Models

Although we use our digit models for recognition, it is helpful to consider how we would use them for generating images. The generative model is an elaboration of the probabilistic interpretation of the elastic net given in [1]. To generate a noisy image of a particular digit class, run the following procedure:

1) Pick a deformation of the model (i.e., move the control points away from their home locations) to give a particular realization **X**. This defines the spline in object-based coordinates. The log probability of picking a deformation is proportional to the quadratic term in (3). It is important that the deformation is measured in object-based coordinates.
2) Pick an affine transformation[6] from the model's intrinsic reference frame to the image frame (i.e., pick a size, position, orientation, slant and elongation for the digit).
3) Map the spline into image coordinates and place beads uniformly along its length. Each bead is a circular Gaussian ink generator. The number of beads and their variance can easily be changed without changing the spline itself. Typically the variance is chosen so that the bead centers are two standard deviations apart.
4) Repeat many times:

   Either (with probability π_n) add a randomly positioned noise pixel to the image—**Or** pick a bead at random and generate an inked pixel from the Gaussian distribution defined by the bead.

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

3.4 Fitting a Model to an Image

In this section, a Bayesian interpretation of the fitting process is adopted and we demonstrate how using a maximum posterior framework (see, for example [38]) yields a practical algorithm. First we need to refine the notation; A superscript ^{O} or ^{I} is used to qualify if a quantity is in the object or image frame respectively. So $\mathbf{X}^O$ represents control points locations in the *object* frame. We can parameterize a model instantiation in the image frame by $\alpha \equiv \{\mathbf{X}\}^O, \mathbf{Y}$.

To classify an image, $\mathbf{I}$, specified as a vector of locations of its N_I inked pixels $\{\mathbf{z}_1, \ldots, \mathbf{z}_{N_I}\}$, each of m models is fitted to the data and the model that best "explains" the image is chosen. Using a uniform prior over all digits, the posterior probability, $P(m \mid \mathbf{I})$, for each model is proportional to the *evidence*, $P(\mathbf{I} \mid m)$:

$$P(\mathbf{I}|m) = \int P(\mathbf{I}|\alpha, m)P(\alpha|m)d\alpha \tag{4}$$

Performing the integration over instantiation parameter space is infeasible, so instead we compute the most probable parameter values (α^*).[7] The evidence is approximated by the height of the posterior peak ($P(\mathbf{I} \mid \alpha^*, m)\ P(\alpha^* \mid m)$), multiplied by the volume of the parameter space under the peak. The negative logarithm of the evidence is then:

$$-\log P(\mathbf{I} \mid m) \approx -\log P(\mathbf{I} \mid \alpha^*, m) - \log P(\alpha^* \mid m) - K \tag{5}$$

where K is the logarithm of the volume term. When the posterior is well modelled by a Gaussian, then $K = \frac{k}{2}\log 2\pi - \frac{1}{2}\log|\mathcal{H}|$, with $\mathcal{H} = -\nabla\nabla \log P(\alpha \mid \mathbf{I}, m)$ the Hessian evaluated at α^*. In the sequel we treat K as a constant, but we allow it to be a different constant for each model (see Section 6). The second term is just E_{def} (3). The first term is the log-likelihood of the image given a particular instantiated model. We refer to this as the *data fit* (E_{fit}). This leads to a convenient objective function consisting of just two energy terms:

$$E_{tot} = E_{fit} + E_{def} \tag{6}$$

If each inked pixel $\mathbf{z}_k$ in the image is generated independently from a distribution defined by B circularly symmetric Gaussian beads, each with a mean $\mathbf{s}(b)$ and variance σ_b^2, and a uniform noise field, then the data fit term is the sum of log probabilities of each inked pixel.

6. Using our prior knowledge that ones tend to be stroke-like, we used a similarity transformation for the one-model.

7. This is reasonable for this problem because there will usually only be one setting of the control points and affine transformation that will provide a good fit.

$$-\sum_{k=1}^{N_I} \log P_k \tag{7}$$

where P_k is the probability of inking pixel k:

$$P_k = \frac{\pi_n}{N} + \frac{1-\pi_n}{B}\sum_{b-1}^{B} P_{kb} \tag{8}$$

$$P_{kb} = \frac{1}{2\pi\sigma_b^2}\exp - \left|\mathbf{s}(b) - \mathbf{z}_k\right|^2 / 2\sigma_b^2 \tag{9}$$

with N the number of pixels which the uniform noise field is distributed over (normally the whole image) and π_n the mixing proportion of a uniform noise field. Using (7) to compute E_{fit} has the undesirable property that it depends on the number of inked pixels in the image. For example, a simple resizing of the image, will change E_{fit} whereas E_{def}, being defined in object based frame, is invariant to scale changes. In order to mitigate this, we allow each pixel to have its own weight W_k. Thus, we compute E_{fit} using:

$$E_{fit} = -\sum_{k=1}^{N_I} W_k \log P_k \tag{10}$$

Normally we set $W_k = \lambda / N_I$ where λ is a constant. This has the desired effect of ensuring that all images have the same total weight of ink and therefore about the same tradeoff between E_{fit} and E_{def} regardless of the number of inked pixels. However, it is also possible that a bottom up processor could assign different weights to pixels based on other knowledge.

We assume that the deformations and affine are independent (for example, the size of a digit or its location in the image is unlikely to be correlated with its style), so the second term in (5) can be factorized into the sum of E_{def} and a term involving the affine parameters. During the fitting procedure, we treat the affine parameters as if they have a uniform prior. However, in Section 6, we show how solutions with unusual affines may be penalized after the fitting procedure is complete.

The objective in fitting a model to an image is to find the α which minimizes E_{tot}. We start with zero deformations and an initial guess for the affine parameters which ensure that the control points are mapped within an upright rectangular box around the inked pixels in the image. A small number of beads with equal, large variance are placed along the spline. These large variance beads form a broad, smooth ridge of high ink-probability along the spline. Because of the high variance, the beads are attracted to inked pixels even if they are fairly far away so the spline is quickly pulled towards the data. During the fitting process the variance of the beads will generally decrease and the number of beads increase as the model gets closer to the data and begins to explain its finer structure. The fitting technique resembles the elastic net algorithm of Durbin and Willshaw [39] except that our elastic energy function is much more complex and we are also fitting an affine transformation.

In early experiments, we used a conjugate gradient method to optimize E_{tot}. Unfortunately this method is slow because each conjugate gradient step may require a few evaluations of E_{tot}, each of which is of the order of $B \times N$ operations. Our preferred method is based upon the Expectation Maximization (EM) algorithm [40]. This involves the repeated application of a two step procedure which will not increase E_{tot} as α is adjusted at each application.

During the expectation (E) step, the beads are frozen at their current locations and the responsibility that each bead has for each inked pixel is computed. This is just the probability of generating the pixel under the Gaussian distribution for the bead normalized by the total probability of generating the pixel ($r_{kb} = \frac{1-\pi_n}{B}\frac{P_{kb}}{P_k}$). Because the negative log likelihood (energy) under a Gaussian distribution is quadratic in the distance from the mean, it is sometimes convenient to think of minimizing E_{tot} as analogous to finding the minimum energy configuration of a system of springs. For a fixed bead variance, consider a system in which each fixed pixel is attached to mobile beads by springs whose stiffness is proportional to the responsibility of the bead for the pixel. As we show shortly the EM method finds the minimum energy configuration of the system of springs.

In the second (M) step, the responsibilities are fixed, and new values of α computed to minimize E_{tot}. In the conventional application of EM, the beads would be unconstrained, and hence a bead would move to the center of gravity of the data (pixels), weighted by the responsibilities that the bead has for each pixel:

$$\mathbf{s}'(b) = \frac{\Sigma_k \mathbf{z}_k r_{kb} W_k}{\Sigma_k r_{kb} W_k} \tag{11}$$

However in our system the beads are *constrained* to lie on the spline defined by the control points; the free variables are really the control point locations and the affine parameters. Directly minimizing E_{tot} results in a set of non-linear equations. We circumvent the expensive step of solving a set of non-linear equations using a two stage procedure. In the first stage, the affine transformation $\mathbf{Y}$ is held constant.[8] This means we can do the minimization exclusively in the *image* frame. To do this we first need to define the deformation energy there. This involves mapping the control point covariance matrix, Σ^O, through the affine (see below). Setting $\partial E_{tot}^I / \partial \mathbf{X}^I = 0$ and with the help of (6), (1), (3), (10), and (11) we update the control point locations by solving the set of linear equations:

$$B\mathbf{X}^I = \mathbf{d}$$

$$B_{mn} = \left(\Sigma^I\right)^{-1}_{mn} + \sum_b \frac{\gamma_m^m(b)\gamma_n^m(b)R_b}{\sigma_b^2}$$

$$d_m = \sum_k \left(\Sigma^I\right)^{-1}_{mk} h_k^I + \sum_b \frac{R_b \gamma_m^m(b)\mathrm{s}'(\mathrm{b})}{\sigma_b^2} \tag{12}$$

with $R_b = \Sigma_k W_k r_{kb}$. In (12), we have used the shorthand; γ_n^m to denote γ_n^x (γ_n^y) and s' the x (y) component of s' for m odd (even).

In the spring system analogy, this stage corresponds to finding the minimum energy equilibrium point where the forces pulling the beads towards the nearby pixels are balanced by the forces pulling the beads towards their home locations.[9]

8. This is an example of Expectation/Conditional Maximization [41].

9. More precisely, the pixel forces on the beads can be transferred onto the control points and at equilibrium there is a balance between these forces and those pulling the control points towards their home locations.

In the second stage of the M-step, the control point locations in the image are kept constant and the deformations (as measured in the object based frame) and affine parameters are adjusted so as minimize the deformation energy. In effect, we are absorbing as much of the deformation as possible into the global affine transformation. This 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. The minimization is achieved by considering the deformation energy in the *image based frame*:

$$E^I_{def} = \frac{1}{2}\left(\mathbf{X}^I - A\mathbf{H}^O - \mathbf{T}\right)^T \left(\Sigma^I\right)^{-1}\left(\mathbf{X}^I - A\mathbf{H}^O - \mathbf{T}\right) + \frac{1}{2}\log\left|\Sigma^I\right| + \text{const} \quad (13)$$

where $(A\mathbf{H}^O + \mathbf{T})$ is the vector of control point home locations when transformed into the image frame through the affine transform, represented in (13) by $(A, \mathbf{T})$. Since all control points undergo the same global affine transform, matrix A is a block diagonal matrix formed by repeating the 2×2 affine along the main diagonal and $\mathbf{T}$ is the concatenation of the (same) translations for each control point. Notice that A has only 4 degrees of freedom while $\mathbf{T}$ has 2. Σ^I is the covariance matrix referred to the image reference frame ($\left(\Sigma^I\right)^{-1} = A^{-T}\left(\Sigma^O\right)^{-1}A^{-1}$). The minimization of (13) with respect to the affine parameters is a non-linear problem, but we have found that satisfactory results are obtained when we treat Σ^I as a constant during the minimization, that is we ignore the fact that the contour lines of equal deformation energy change as the affine varies. The deformation energy is then a quadratic form of the affine parameters and the minimization is straightforward.[10]

After each complete iteration of the algorithm, the bead variances (all beads are constrained to have equal variance) are set to the variance that maximizes the log likelihood of the N_I inked pixels given the current positions of the beads using the update:

$$\sigma_b^2 = \sigma^2 = \frac{1}{2N_I}\sum_{k=1}^{N_I}\sum_{b=1}^{B} r_{kb}\left\|\mathbf{s}(b) - \mathbf{z}_k\right\|^2 \left(b = 1 \ldots B\right) \quad (14)$$

In order to ensure that bead centers remain approximately two standard deviations apart, the number of beads along the spline is periodically adjusted. The fitting algorithm is summarized in Fig. 2.

1. Map the the current control point locations from the object frame into the image frame $\mathbf{X}^I = A\mathbf{X}^O + \mathbf{T}$
2. Construct the spline (1) and place the Gaussian beads $2\sigma_b$ apart.
3. Evaluate probabilities of inking pixels (9)
4. Compute new control point and bead locations (12) and (1)
5. Given these new locations choose the new affine and X^O to minimize (13).
6. Update the bead variance (14) and the new E_{tot} using (6), (10) and (3).

Fig. 2. The fitting algorithm iterates over steps 1-6 until E_{tot} converges.

Some stages in fitting models to data are shown in Fig. 3. In this example the best data fit energy was achieved by the three model, but the five model managed to provide a creative explanation of the data. However, in doing so it had to pay a high deformation cost. For this image, none of the other eight models had better fits and so if the model with the lowest E_{tot} (6) is chosen, then the conclusion is that the data is most likely to have been generated by the three-model.

The search technique almost always avoids local minima when fitting models to isolated digits. In the few cases where local minima are encountered they can usually be overcome by starting with a different guess for the initial affine transformation. If the image is not recognized with sufficient confidence as explained in Section 6, we try four other initial guesses corresponding to positions translated right, above, left and below the original one and choose the fit with the lowest E_{tot}.

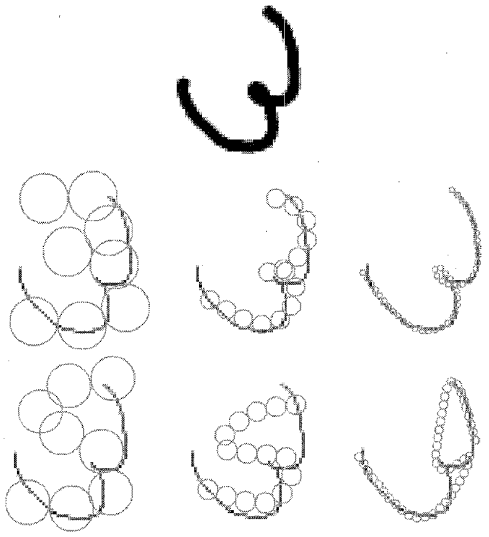


Fig. 3. Some stages of fitting models to an image of a 3. The image is displayed in the top row. In subsequent rows, the circles represent the beads. The radius has been set to one standard deviation of the circular Gaussian distribution. Because the bead variance shrinks to approximate the stroke thickness during the fitting process, the beads would become invisible towards the end of the search. Consequently in this and subsequent figures, we thin the data along its center line. We emphasize that this is done only for display purposes in order to make the beads visible. The middle row show the three-model being fitted while the bottom row illustrates the process for the five-model. The left column shows the initial configuration, with eight beads equally spaced along the spline. The second column is an intermediate fit as the model rotates and deforms in order to improve the log likelihood of the data. The final fit is shown in right column.

Our generative models also include a noise model. Each inked pixel may be generated either by the digit-model or a noise process. We have chosen the most simple type, a uniform noise process (see equation (8)). The addition of this noise model improves the performance of the system, even though a uniform distribution is a poor model of the highly correlated, structured noise typically found in digit images. Fig. 4 illustrates how the addition of the noise model improves the ability of the digit models to correctly segment out the data from the noise in the image; in effect the system is performing model-driven image segmentation. The fit without the noise model ($\pi_n = 0.0$) is totally unacceptable.

10. If we had performed the nonlinear minimization then E_{tot} would be guaranteed not to increase during the second stage of the M-step.

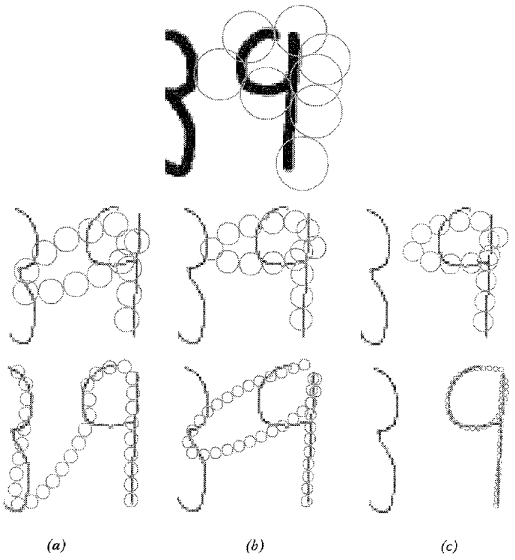


Fig. 4. Illustration of how the noise model helps in model driven segmentation. The top row shows an image of a nine with a portion of its left neighbor. The initial configuration of the nine-model is also shown. The remaining rows show the configurations of the model part way in the settling (second row) and the final settled configuration. The three columns show different mixing proportions of the noise. (a) $\pi_n = 0.0$, (b) $\pi_n = 0.2$, (c) $\pi_n = 0.45$. The final standard deviation of the beads from the correct fitting model better approximates the stroke thickness than the standard deviation from incorrect fitting models. The data have been thinned for the reasons mentioned in Fig. 3.

4 Learning the Models

Each elastic model is parameterized by a vector of mean or *home* locations and a covariance matrix (see (3)). These model parameters can be learned from training data. Starting with hand crafted digit models we adjust the home control point locations so that each model maximizes the likelihood of generating instances of that digit in a training set. Maximization is performed iteratively using EM updates. This yields a simple algorithm: the updated home location of each control point (in the object-based frame) is the average location of that control point in the final fits. Learning proceeds rapidly with models learning their final configurations after only a few passes through the training set (Fig. 5), probably because we start off with good models.

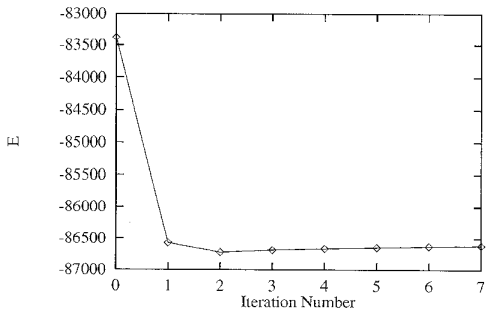


Fig. 5. Training the two-model. The ordinate shows the sum $E = \Sigma_I E_{tot}(I)$ over all two-images in the training set. The model has essentially completed its learning after the second pass through the training set.

An alternative to maximizing the likelihood of the image given the digit is to maximize the mutual information between the correct digit class and the probabilities assigned to the various classes by the digit models. The maximum mutual information criterion emphasizes correct discrimination rather than correct modeling of the image data, and it generally leads to better discriminative performance [42], although the advantage of discriminative learning vanishes if the generative model is correct and the fitting process produces the true probability of the data given the model [43]. Early experiments showed that, for our generative models, maximum likelihood learning was just as effective as discriminative learning, perhaps because the generative models are a reasonable approximation to the way in which the data is generated. Maximum likelihood learning is much quicker because there is no need to fit the incorrect digit models to each of the training digit images.

Neglecting the Hessian term in (5), the sample covariance matrix is estimated from N_T training examples using:

$$\Sigma = \frac{1}{N_T}(\mathbf{X} - \mathbf{H})(\mathbf{X} - \mathbf{H})^T. \tag{15}$$

In our experiments we used 700 training examples for each model. Having a limited amount of data requires that some precautions be taken to prevent those principal modes with small variance from "blowing up" when inverting Σ. It turns out that for all models the principal modes tend to group into a significant (large eigenvalues) and an insignificant (small eigenvalue) cluster. The modes corresponding to the large eigenvalues are generally intuitively obvious. For example, in the two-model the largest mode of variation corresponds to opening/closing of the loop. To prevent the insignificant modes from being problematic when inverting Σ, we regularized Σ by clamping all eigenvalues in the insignificant cluster to 10^{-2} of the largest eigenvalue. Generally we had to clamp about one third of the eigenvalues.

5 Refining the Model

5.1 Variants on the Deformation Energy

An instance of the elastic model in the *object* frame can be specified by giving only the (x, y) locations of n control points. Therefore any particular occurrence of the model can be thought of as a point in $\mathcal{R}^{2n}$ and the population of models would form a distribution in $\mathcal{R}^{2n}$. In (2) we have chosen to describe this distribution as a Gaussian hyperellipsoid. For a typical model with eight control points this characterization requires specification of a 16×16 covariance matrix. It is interesting to investigate different simplifications.

The obvious first approximation is a diagonal covariance matrix. We tried the most simple of these and set

$$\Sigma = \sigma^2 I \tag{16}$$

where I is the identity matrix. The control points then all have identical, independent radial Gaussian distributions.

The deformation energy simplifies to:

$$E_{def} = \frac{\|\mathbf{X} - \mathbf{H}\|^2}{2\sigma^2} + n \log 2\pi\sigma^2 \tag{17}$$

This characterization of the distribution by a single *generic* model can result in poor approximations to the true distribution. Fig. 6 illustrates a situation in two dimensions. For example, under a single Gaussian approximation to the distribution, point **A** would have higher probability than point **B**, which is clearly incorrect.

An alternate way to approximate a distribution, which also has an interesting interpretation for digit recognition, is to use a mixture of L *local* models, each of which is of the form (17). Under this approximation the distribution of models is given by:[11]

$$P(\mathbf{X}) = \sum_{l=1}^{L} \frac{\beta_l}{\left(2\pi\sigma_l^2\right)^n} \exp\left\{-\frac{\|\mathbf{X} - \mathbf{H}_l\|^2}{2\sigma_l^2}\right\} \tag{18}$$

where β_l is the mixing proportion for the lth local model in the mixture with $\sum_{l=1}^{L} \beta_l = 1$.

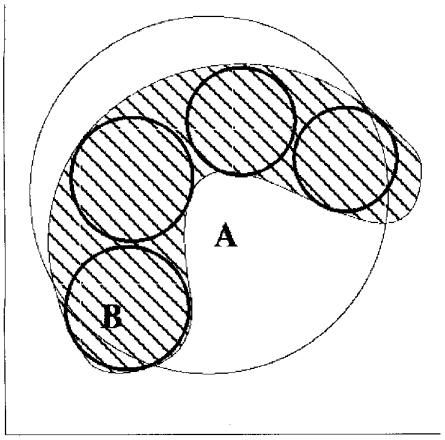


Fig. 6. An arbitrary two-dimensional distribution represented by the shaded region could be modeled by a a single Gaussian with large variance as illustrated by the large circle. A better approximation would be to use a mixture of Gaussians each with a smaller variance. Under the single Gaussian approximation point **A** would be incorrectly considered to be more likely than point **B**.

The centers $\mathbf{H}_l$ and variances (σ_l^2) are computed using EM to maximize the log likelihood of a training set under the mixture distribution (18). Fig. 7 shows the 10 local models that were automatically extracted from the training data of images of 2s. The mixture has been able to capture dominant styles. For example, variations in the presence and size of the loop have been well represented.

11. We have experimented with more complex variations such as allowing each mixture component to have its own adaptive variance, or kernel density estimation, but found no improvement in performance over the simpler characterization.

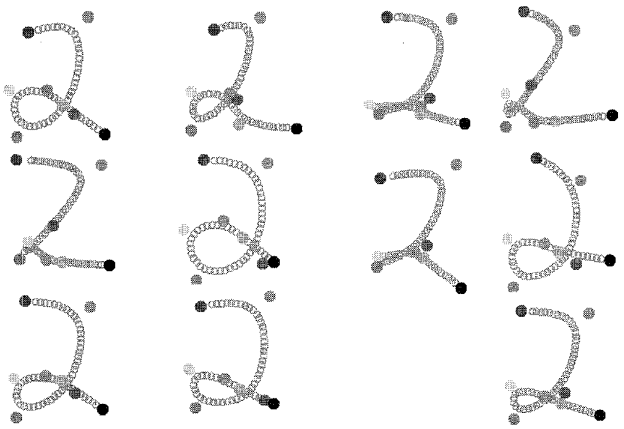


Fig. 7. Local models in the mixture for the 2 model. The generic model is shown in the bottom right corner.

The obvious way to use this mixture is to fit each of the local models to an image. This has the disadvantage of increasing the recognition time by a factor of L. Fortunately we found that a single generic model nearly always fits correctly to the image. So we fit the single generic model, but after fitting, the deformation energy is evaluated using the log probability under the mixture distribution (18) instead of using E_{def} as in (17). This strategy is much more efficient since the most computation intensive portion, the fitting of the model to the data, is done only once. Evaluating the distance of the final fit from each of the local models in the distribution only involves computing $2n$ squared distances and so is negligible compared to the amount of computation required for fitting.

Fig. 8 illustrates the added classification power obtained using the 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. Much better separation is achieved with a mixture distribution.

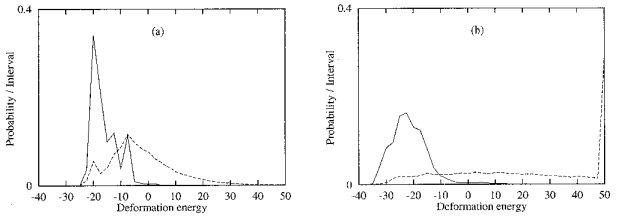


Fig. 8. The solid line is the E_{def} distribution of models when fitted to example images containing the correct digit (e.g., a three model fitted to an image of a 3). The dotted curve is the E_{def} distribution of the models when fitted to incorrect data. (a) Using a single generic model. (b) Using a mixture of local models. (Many instances of fitting to incorrect data in panel (b) had very large E_{def}. For display purposes, all these were assigned the value 50, accounting for the "spike" at the right edge of panel (b)).

There is an interesting interpretation to these local models. One may think of each local model as capturing a specific writing style in the population. A particular instantiation of the digit can then be classified in terms of style. This

may prove to be useful in strings of digits, where we would expect different instances of the same digit to have similar styles (see Section 7). There may even be mutual information between the pairings of local models for different digits (e.g., between a 4 and a 6) [44].

5.2 Generating Both the Inked and Uninked Pixels

A significant drawback of our generative model is that it does not treat the uninked 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 uninked pixels.[12] As a result, a model can fit the data well even if some of its beads are a long way from the nearest inked pixel. For example in Fig. 3, the five model has accounted for all the inked pixels, but the final fit has left its center bar far from any inked pixels. We call this the "beads in white space" problem. A computationally simpler approach to this problem is discussed in Section 6.

We assume that the image is generated from the spline by a two-stage stochastic process. The first stage computes the probability $P_k(p = 0)$ that each pixel in the image would not be inked if *multiple* samples were taken from the probability distribution defined by the beads and the uniform noise process (see (8)). We take N_m[13] samples from this distribution. The probability that none of these samples landed within a particular pixel is:

$$\begin{aligned}\hat{P}_k(p = 0) &= P(\textit{not inked by model}) \\ &= (1 - P_k)^{N_m}\end{aligned}$$

The predicted probability of a pixel being inked, $P_k(p = 1)$ is simply the complement.

$$\hat{P}_k(p = 1) = 1 - \hat{P}_k(p = 0) \tag{19}$$

Given these predicted probabilities, the second stage computes the probability of generating all the pixels in image $\mathbf{I}$. This can also be viewed as the cost of encoding the actual image data using the predicted probabilities to do the encoding.

$$\log P(\mathbf{I}|m) = \sum_{\substack{k \in \textit{inked} \\ \textit{pixels}}} \log \hat{P}_k(p = 1) + \sum_{\substack{j \in \textit{uninked} \\ \textit{pixels}}} \log \hat{P}_j(p = 0) \tag{20}$$

In Fig. 9 we show a model of a four settled on the image of a seven using the generative model of Section 3. The right most panel figure shows the probabilities generated by (20). Areas of low probability are shaded dark. The portion of the center bar of the four spanning white space becomes expensive under the full generative model. The dark fringes around the edges of the model arise because the beads have a standard deviation approximately equal to the stroke thickness and hence it predicts fuzzier edges than are present in the image. (The image was originally binary but has been shrunk to a quarter of it original area and so still retains its abrupt edges.)

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

13. 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.

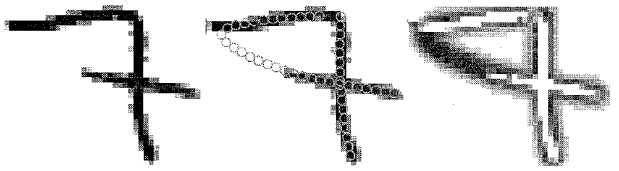


Fig. 9. The left panel shows an image of a 7. The fit of a four using the generative model which produces only inked pixels is shown in the center panel. Under this formulation the center bar of the four model would not be penalized for lying in white space. The right most panel shows the image weighted 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.

It is interesting to examine how this model behaves when the beads have high variance. Assuming a relatively low noise level, all pixels will have a low predicted probability of being inked and hence (20) will be dominated by the cost of generating inked pixels. Using $(1 + \delta x)^n \simeq 1 + n\,\delta x$, we see that at high variance:

$$\begin{aligned}\log P(\mathbf{I}|m) &\simeq \sum_{\substack{k \in \textit{inked} \\ \textit{pixels}}} \log \hat{P}_k(P = 1) \\ &= \sum_{\substack{k \in \textit{inked} \\ \textit{pixels}}} \log\left(1 - (1 - P_k)^{N_m}\right) \\ &\simeq \sum_{\substack{k \in \textit{inked} \\ \textit{pixels}}} \log(P_k) + N_m \log N_m\end{aligned}$$

Note that the second term involving N_m will be the same for all models, and hence at high variance this generative model is approximately the same as the one described in Section 3 (see 10).

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 much more expensive for parts of an instantiated digit model to lie in white space. One might therefore consider using this model to escape from a local minimum, in which parts of the model span white space, obtained using the simpler model of Section 3. However, this would fail because at low variance the model is unable to substantially change its configuration because the beads cannot "see" data more than a few standard deviations away. The model would therefore probably be most useful at intermediate bead variances. Unfortunately this is also the most computationally expensive situation and so we have not used it during the fitting process. We have used (20) to *evaluate* the data-fit of the settled configurations obtained by running the simpler model of Section 3 and found no improvement over the approach of Section 4.

5.3 Speeding Up the Search

In order to classify an image of a single digit, 10 models[14] must be allowed to settle on the image. Each iteration of the settling of a model involves a computational burden proportional to the product of the number of beads and number

14. If multiple models are contemplated per digit the computation burden becomes more acute. For example, we may choose to have a separate model for the "crossed seven" or the "two circles eight."

of pixels. As the model settles onto the image the bead variance generally decreases with a concomitant rise in the number of beads (Section 3.4) and so the number of floating point operations per iteration of the EM search increases towards the end of the search. However, at low variance, there will only be a few beads that have significant probability of generating each pixel. It is clear that one way to dramatically speed up the search is to eliminate all computations that are used to update the responsibilities of beads that are many standard deviations from a pixel. We could simply freeze these responsibilities at their current low values. In other words, during the E-step of the settling algorithm (step 3 of Fig. 2), we only update the *relative* responsibilities of those beads that have a significant probability of generating the pixel. After performing a number of EM iterations with the responsibilities frozen for distant bead-pixel pairs, they are unfrozen and a few full EM iterations are be performed. In this way we would hope to achieve the same maximization of (6) with much less computation.

Conventionally, the EM algorithm is seen as a way of maximizing the log likelihood, $L(\alpha) = \log P(\mathbf{I} \mid \alpha)$, of a model parameterized by α, for some observed data, **I**. Usually not all the data necessary to do the maximization is directly observed and so the first (E) step estimates a "unobserved" variable ρ and maximization is achieved with the help of ρ. With this view of the EM algorithm, it is not immediately obvious that partial implementation of the expectation step is justified. However in an alternative interpretation of the EM algorithm [45], the EM algorithm can be viewed as maximizing a joint function, $F(P(\rho), \alpha)$ of the distribution of the unobserved data and model parameters. If ρ is chosen to be the optimal distribution, ρ^*_α, that maximizes F for the current value of α, then $F = L(\alpha)$. At the tth iteration, the first step of the standard EM algorithm chooses $P^t(\rho)$ to maximize $F(P, \alpha)$ and the second sets α to maximize $F(P^t, \alpha)$. However, it is not necessary to compute the optimal distribution of ρ in the E step. Any change in ρ that reduces the Kullback-Liebler distance between ρ and ρ^*_α is guaranteed to improve F. In the elastic model the unobserved variables, ρ, are bead responsibilities. Our method of freezing a subset of the responsibilities and recomputing the optimal distribution of relative responsibilities for the remaining beads is guaranteed to improve F.

6 Results on Isolated Digits

The performance of the elastic net in recognizing isolated digits has been evaluated on data from the CEDAR CDROM 1 database of Cities, States, ZIP Codes, Digits, and Alphabetic Characters [46]. The *br* training set of binary segmented digits was subdivided into three training sets of size 2,000, 7,000, and 2,000, respectively. A validation set of 2,000 examples was also generated from the *br* training set to allow us to investigate different configurations of the post-processing neural network. The sets were constructed by drawing images in the order presented in the database so as to ensure equal representation of all digits in each set. The elastic models were trained (Section 6) on the first set, the mixture of local models on the second and the post-processing net on the third set. The CEDAR database also includes two test sets. The *goodbs* (2,213 images) set is a subset of the *bs* (2,711 images) set containing only well segmented digits. It is interesting to note that *br* training data were segmented with the same diligence as the *goodbs* test data [46].

After fitting all the models to a particular image, we wish to evaluate which of the models best "explains" the data. The natural measure is the sum of E_{fit} and E_{def} that is minimized during the fitting process. However, we found that performance is improved by including five additional terms which are easily obtained from the final fits of the model to the image.

Motivated by research on "snakes" [30], a simple approach to the beads in white space problem (Section 5.2), is to define another energy term, E_w to penalize beads spanning white space. This term is similar to the "support measure" [31] or the symmetric matching used in [21] and [22].

$$E_w = -\sum_{b=1}^{B} \log \sum_{k=1}^{N_I} P_{kb} \tag{21}$$

A bead only makes a large contribution to this cost when all inked pixels are far from the bead. This energy term could be easily incorporated into the fitting procedure, but in the present system we simply use it as an additional term when evaluating the final fits. The model presented in Section 5.2 is a more principled approach to the uninked pixels, but is computationally much more demanding.

While fitting to an image, a model can adopt any affine transformation without penalty. After fitting, the final affine transformation may contain some information which is relevant in evaluating the fit. For example we may want to reject an explanation which requires a model to be highly rotated, sheared or elongated. An arbitrary affine transformation matrix may be written in the form:

$$A = \begin{pmatrix} S_x cos\theta_x & -S_y sin\theta_y \\ S_x sin\theta_x & S_y cos\theta_y \end{pmatrix}$$

We included $\{\sin^2 \theta_y, \sin^2(\theta_x - \theta_y), S_y/S_x\}$ as measures of rotation, shear and elongation of the affine transformation as additional information into the final determination of class membership.

The last term used is the final variance of the beads in each model. This term is included because it has an effect on E_{fit} (see (9)). Also, when a model correctly explains the strokes in an image, the standard deviation of the beads is about half the stroke thickness, while for incorrect fit it tends to be larger (see Figs. 4 and 11).

It is hard to decide in a principled way on the correct weightings for all of these terms in the evaluation function, and it is not even clear that the relative weightings should be the same for the different digit models. So we estimate the weightings from the data by training a simple post-processing neural network. It should be emphasized that

this network is simply a convenient method for *linearly* weighting the different measures; it is unlike conventional neural net classifiers because it does not build internal representations of the input.

If 10 models are fitted to an image then there are 70 inputs to the net. These types of networks are much easier to train if all inputs are approximately of the same magnitude, i.e., it is good practice to subtract out any constant offsets and scale the input data. The E_{fit} term has a large offset, so we used the difference $\tilde{E}_{fit}^{k} = E_{fit}^{k} - E_{fit}^{min}$ as the data fit term for the kth model, where E_{fit}^{min} is the minimum data fit obtained by any of the models for the current image. A similar transformation is used for the final bead variance, while the remaining inputs are simply scaled.

Each of the seven input terms for a model is directly connected to the output unit for that model. The output units compete using the "softmax" function [47] which guarantees that the 10 output values form a probability distribution. Including biases on the output units,[15] the network has 80 weights and is trained using conjugate gradient to minimize a cross-entropy error function. After training we classify an image according to which of the output units has the largest activation. We reject classifications in which the maximum output activation is below some threshold T.

We tried a variety of architectures for this "post-processing" network. For example, a digit recognition system developed by Hastie and Tibshirani [48] suggested that discrimination would be much better if the net was totally connected so that the output unit that represents one digit receives detailed information about the way in which other digit models fit the data. However, we found that discrimination was just as good if each output unit only received connections from the six inputs representing terms describing the fit of that digit model. Including a hidden layer did not improve performance. Incorporating a local approximation to the Hessian (5) also did not improve performance.

To summarize, recognition of a single image consists of the following steps:

1) The image is down sampled to reduce the number of pixels to one quarter (i.e., the number of rows and columns were both halved). This was done primarily to reduce the number of operations required per image.
2) For each of the 10 models, an initial affine is computed so as to position the model over the enclosing rectangle of all the inked pixels in the image.
3) The models are allowed to settle with the iterative algorithm summarized in Fig. 2. The fractional change in E_{tot} is monitored and when it falls below 0.001, the number of beads are adjusted (Section 3.4). This is repeated about six times for each model.
4) Energies and affine transformation values are fed into the neural network to produce a classification of the image.
5) The winning output is tested against a threshold and if not sufficiently large all models are resettled from four other initial positions. After all positions have been tried, the best settled state for each model is selected as input to the post-processing network. About 6% of the validation set images and 8% of the *bs* test set images invoked this restart procedure.

Table 1 shows the performance of the elastic net when the rejection threshold was set to zero. The first line in the table shows results when Σ^{-1} (see (3)) was estimated from the data. The second row shows results obtained when all models had the same diagonal covariance matrix $\frac{1}{\sigma^2} I$ with $\sigma^2 = 0.01$. The final line is the performance when the models were settled with the same diagonal covariance matrix as in line 2, but E_{def} was evaluated using the mixture of local models discussed in Section 5. As a simple comparison, if all images are normalized to 16×16, then k-nearest neighbor[16] has a raw error rate of 4.7% on the validation set and 7.08 % on the *bs* test set. An in depth study by Lee and Srihari [10][17] evaluated eight algorithms and five combination schemes on the *bs* test set. The results in Table 1 are better than seven out the eight individual classifiers they used. Their best single algorithm has a raw error rate of 2.99% while the best combination scheme has a 2.51% error rate. Ha and Bunke [49] report error rates of 0.9 – 2.3 % on the *goodbs* data set using five schemes. Unfortunately it is not clear which data was used to finesse the many empirical constants involved.

TABLE 1
PERCENTAGE OF IMAGES INCORRECTLY CLASSIFIED BY THE ELASTIC NET WITH NO REJECTIONS

	Validation Set	*goodbs* test set	*bs* test set
Full Covariance Matrix	2.00	1.85	3.58
Diagonal Covariance Matrix	1.75	1.53	3.43
Mixture of local models	1.00	1.50	3.14

In comparing our error rates with others published for the same data, it is important to allow for the fact that some studies looked at the images in the test set or even used the test set to determine some parameters of their system. We have been very careful to avoid this and have never looked at the test set or at the specific errors we make on it. The poorly segmented digits in the bs *test set are not characteristic of the training data, so big improvements in performance should be obtained by tuning systems on it.*

The confusion matrix of errors for the *bs* set is shown in Table 2. The most striking feature is the confusion of a 7 as a 9—mainly cases of crossed 7s (Fig. 9). Varying the rejection threshold in the post-processing neural network allows us to trade off errors against rejects. Fig. 10 shows error-rejection curves obtained on the validation and test

15. The bias on each output unit can be thought of as the volume factor for each model in (5).

16. $K = 2$ chosen on the basis of the validation set.

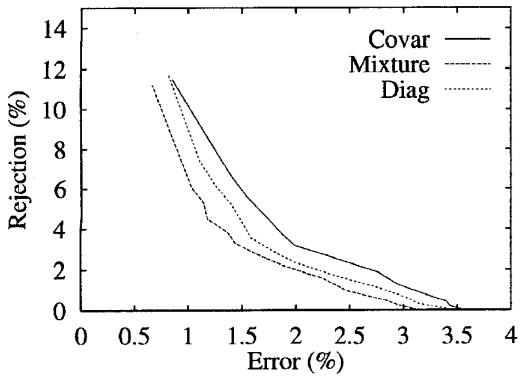
17. Also available from ftp://mirach.cs.buffalo.edu/pub/cdrom1/bs_digit_results/README.

sets. To achieve a 1% error we have to reject 6-10% on the *bs* test set. Lee and Srihari's [10] curves indicate that they have to reject 2.5-12% to achieve the same rate, but these curves are for a different test set.

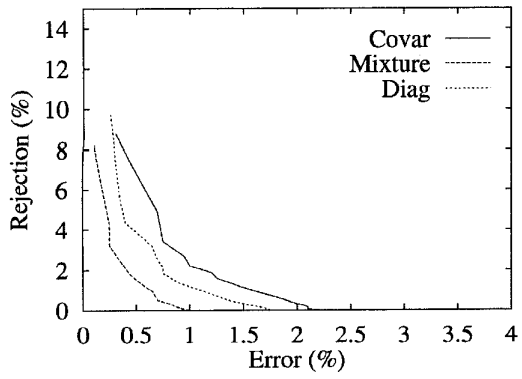
TABLE 2
CONFUSION MATRIX

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	1	2	2	3	1	1	1
1	1	0	1	0	0	0	2	2	0	0
2	2	0	0	1	0	2	1	1	2	3
3	2	0	3	0	0	1	0	2	0	0
4	0	1	1	0	0	3	1	2	0	1
5	1	0	0	5	0	0	1	0	0	0
6	0	0	1	2	0	2	0	1	1	0
7	0	1	3	2	4	0	0	0	0	5
8	1	0	1	0	0	0	2	0	0	1
9	0	0	0	0	2	1	0	2	0	0

Error counts on the bs *test set for the mixture of local models. The true identity is indicated by the row index.*



bs Test Set



Validation Set

Fig. 10. Error-rejection relationships. Full covariance matrix—solid curve. Diagonal covariance matrix—dotted curve. Mixture model—dashed curve.

7 Discussion

We have described a generative model approach to handwritten digit recognition. The major motivation for building this system was to investigate potential advantages of using stochastic generative models for object recognition in a realistic domain. While its classification performance is comparable to other state-of-the-art classifiers [10], [50] it is significantly more computationally intensive. With our simulation code running on a R4000 based workstation, a model settles on a typical image in about 1.1 seconds, resulting in a classification rate of about 5.5 images/minute. This is about two orders of magnitude slower than current practical classifiers. We have not investigated speedup methods in detail, but there are a number of simple approaches which could achieve dramatic improvements. Parallelizing the search across digit models is the simplest approach. Using total elapsed time to test the set of 2,000 validation digits on a six-processor workstation resulted in a speedup of 5.1. More elaborate schemes could involve computing the probabilities (9) in parallel. Also, we have observed that for most images it is apparent early in the search that some models do not explain the image well. For example a 9-model has to undergo unusual deformations to fit to an image of a 2. So this suggests another simple method; detect these poor performing models and terminate their searches after just a few iterations. Providing a good starting point for the search can also speedup the search. For example, a doubling of processing speed was achieved [51] using the multi-layered backpropagation network described below.

An examination of the errors made on the validation set reveals that almost all misclassifications can be attributed to two problems: local minima in the search space and modeling difficulties. When the beads have low variance, the search space has many local minima. By annealing the variance our search method manages to avoid nearly all of these, but occasionally (about 1% of cases) becomes trapped. The obvious solution is to start the search closer to the global minimum. In the current system, very little information from the image is used to pick initial model instantiation parameters. The deformations, are set to zero and the affine parameters are chosen to simply position an upright rectangle over the entire inked portion of the image. Any method which picks better initial instantiation parameters should improve the search. The restart procedure described in Section 6 is a very simple attempt to do this, but still does not use any more information from the image.

Using the rich set of instantiation parameters supplied by the correct elastic model after it has been fitted, we can train a conventional supervised multi-layer neural network to predict model instantiation parameters from the image [51]. Given an input image,[18] the network predicts the locations of the control points in image space for each of the 10 digit models. Running the second stage of the M-step of our fitting procedure gives the control point locations in the object frame. We would expect this type of network to be less susceptible to over-fitting than conventional neural network recognizers [7], [8], [52]. In conventional networks, each training example only provides $\log_2 10$ bits of constraint on

18. In this scheme, images would have to be normalized to fit into the fixed length input space of the network.

the weights of the network because that is the number of bits required to specify the largest output. A network trained to predict instantiation parameters provides much more constraint per training example.

The other difficulty that elastic models experience is caused by a special kind of variability present in handwritten digits. Although spline models are good at capturing most common variations (see Fig. 1) they cannot easily model large embellishments to the basic shape. Fig. 11 illustrates an extreme example. The 3-model (middle panel) has correctly modeled the main body of the image but does not have sufficient flexibility to explain the flourish portion. On the other hand, the 2-model has successfully modeled the flourish but has missed the perceptually important cusp portion. Increasing the flexibility of the models is not a solution since they are then able to model other digits. One possibility is to examine the residual image, i.e., the portion of the image left unexplained by the model. Currently the residual image is accounted for by a simple uniform noise process (8). An improvement would be to use a more structured noise model. Another possibility is to model the residual images using "flourish models" [36]. Before leaving this topic, it should be noted that some regional stylistic peculiarities, for example the middle bar on "crossed" sevens or the top and bottom of European style ones, may be modeled in this manner. We did not try this as in our study as we used a North American database in which the frequency of these styles was very low.

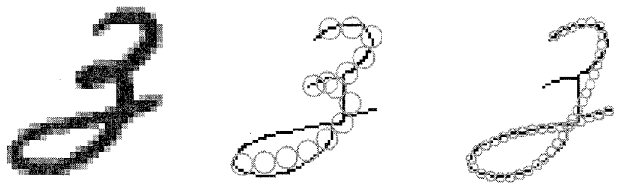


Fig. 11. The difficulty of flourishes. The middle panel show the fit of the three model while the right panel shows how the model of a two captured the entire flourish. Notice how the three model did not shrink its variance to the same degree as the two-model because it used the large variance to compensate for its inability to model the flourish portion. The data has been thinned in the last two panels for the same reasons explained in Fig. 3.

Our study of generative models as applied to handwritten digit recognition has highlighted a number of benefits. These relate to the search space and the type of information that can be extracted.

The images we have used vary in size from many hundred to tens of thousands of pixels. Thus the search space is high-dimensional and therefore requires an efficient search strategy. The search method we have developed has two advantages. Firstly, as an EM method it is faster than a gradient following technique. Secondly, it implements a coarse to fine search strategy (see Fig. 3). It starts with a few large beads which has the effect of viewing the data at a very coarse scale, and so the models can concentrate on adjusting the affine parameters, particularly translations and orientations. Only later in the search do they begin to model the details, for example the middle cusp in Fig. 3. A related benefit of the approach is the model driven segmentation illustrated in Fig. 4.

It is interesting to consider the intrinsic dimensionality of the manifold, in pixel space, that contains all the different instances of the same handwritten digit. Clearly the manifold has lower dimensionality than the number of pixels. For example, simply increasing the size of an image should not increase the dimensionality of the manifold. Almost all OCR approaches recognize the existence of a lower dimensional manifold by extracting a small feature vector from the high dimensional pixel space. The generative models are especially frugal, using at most 22 degrees of freedom (8 control points plus 6 degrees of freedom for the affine). While making no claim that this is the appropriate dimension, the models appear to make good use of the available degrees of freedom and because of the relatively low dimension and prior information available they are quickly and easily trained (see Fig. 5). This is in contrast to the thousands of free parameters and multiple passes over the training data required by a typical neural net recognizer [7].

We have claimed that one advantage of generative models for handwritten character recognition is that instantiation information from one character should be useful for other characters written by the same author. For example knowing that a writer draws a 2 with a large loop should assist in recognizing other examples from the same author. The mixture of local models presented in Section 5 can be thought of as quantizing the *style* space for digits.

One natural way of quantifying the amount of information from style is to use the mutual information measure. For two random variables x and y, the mutual information $I(x, y)$ conveys the uncertainty in one of the variables that can be accounted for by the other. Using repeated digits within a zip code, the mutual information present in say 2 sixes written by the same author can be computed. To check if this quantity is significant, we used the same set of images but randomly assigned pairs, i.e., so that it is very unlikely that paired digits came from the same author. This was repeated 100 times for each digit to give a mean and standard deviation. The results are shown in Fig. 12.

Except for ones, there is a significant amount of mutual information (0.15-0.85 bits[19]) in this simple style measure. It is interesting that for digits having obvious style attributes, for example 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 ones. The latter were modeled using only three control points and generally the images were a simple stroke,[20] which does not allow for much style variation. One simple way to incorporate this type of a style information in the task of recognizing strings of digits (e.g., zip codes) is to modify the mixing proportions, β_l, in (18) for each digit using the *styles* of previously recognized instances of that digit in the string.

19. The task of classifying a digit as one out of 10 takes about 3.3 bits.

20. The similarity transformation could account for slope variation and so this could not be considered a style under our present method of measurement.

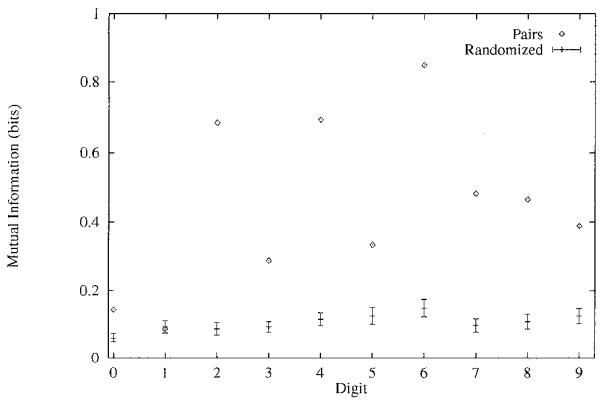


Fig. 12. Mutual information computed for pairs of digits from the same zip code. Also shown are means and error bars of mutual information computed when images are randomly paired.

8 Conclusion

We have explored generative models as a technique for object recognition. Unconstrained handwritten digit recognition was chosen as the test domain because it is an important two-dimensional problem which shares some of the complications found in three-dimensional object recognition. Although we achieved recognition rates comparable to current well tuned state of the art recognizers, we do not propose this method as a replacement for such methods, mainly because of its high computational demands. It may however be considered as a verification stage for faster recognizers. Because our method is so different from these other methods, we expect to have a low correlation between the errors made by the two types of recognizers and so it may be possible to obtain enhanced performance by combining it with other recognizers.

The study has shown that relatively complicated generative models can be fitted to real data using a method that almost always avoids poor local minima. We also demonstrated that generative models can extract extra information from images that can be useful for model-driven segmentation and for capturing the constraints between the styles of the digits within one zip code.

Acknowledgments

This research was funded by Apple and by the Ontario Information Technology Research Center. We thank Allan Jepson, Richard Durbin, Rich Zemel, Rob Tibshirani, and Yann Le Cun for helpful discussions. Geoffrey Hinton is the Noranda Fellow of the Canadian Institute for Advanced Research.

References

[1] R. Durbin, R. Szeliski, and A.L. Yuille, "An Analysis of the Elastic Net Approach to the Travelling Salesman Problem," *Neural Computation*, vol. 1, pp. 348–358, 1989.

[2] S. Impedovo, *Fundamentals in Handwriting Recognition*. Springer-Verlag, 1994.

[3] C.Y. Suen, C. Nadal, R. Legault, T.A. Mai, and L. Lam, "Computer Recognition of Unconstrained Handwritten Numerals.," *Proc IEEE*, vol. 80, no. 7, pp. 1,162–1,180, July 1992.

[4] G.L. Cash and M Hatamian, "Optical Character Recognition by the Method of Moments," *Computer Vision, Graphics, and Image Processing*, vol. 39, pp. 291–310, 1987.

[5] L. Lam and C.Y. Suen, "Structural Classification and Relaxation Matching of Totally Unconstrained Handwritten Zip-Code Numbers," *Pattern Recognition*, vol. 21, no. 1, pp. 19–31, 1988.

[6] M. Shridhar and A. Badreldin, "Recognition of Isolated and Simply Connected Handwritten Numerals," *Pattern Recognition*, vol. 19, no. 1, pp. 1–12, 1986.

[7] Y. Le Cun et al., "Handwritten Digit Recognition with a Back-Propagation Network," *Advances in Neural Information Processing Systems*, D.S. Touretzky, ed., Denver, vol. 2, pp. 396–404. San Mateo: Morgan Kaufmann, 1990.

[8] J.D. Keeler, D.E. Rumelhart, and W.K. Leow, "Integrated Segmentation and Recognition of Hand-Printed Numerals," *Advances in Neural Information Processing Systems* 3, R.P. Lippmann, J.E. Moody, and D.S. Touretzky, eds., pp. 557–563. San Mateo: Morgan Kaufmann, 1991.

[9] K. Fukushima and N. Wake, "Handwritten Alphanumeric Character Recognition by the Neocognitron," *IEEE Trans. Neural Networks*, vol. 2, no. 3, pp. 355–365, 1991.

[10] D. Lee and S.N. Srihari, "Handprinted Digit Recognition: A Comparison of Algorithms," *Third Int'l Workshop on Frontiers in Handwriting Recognition*, pp. 153–162, Buffalo, NY, May 1993.

[11] F. Kimura and M. Shridhar, "Handwritten Numerical Recognition Based on Multiple Algorithms," *Pattern Recognition*, vol. 24, no. 10, pp. 969–983, 1991.

[12] J. Geist et al., "NISTIR 5452. The Second Census Optical Character Recognition Systems Conference," Technical Report, U.S. National Institute of Standards and Technology, 1994.

[13] P. Simard, Y. Le Cun, and J. Denker, "Efficient Pattern Recognition Using a New Transformation Distance.," *Advances in Neural Information Processing Systems 5*, J.D. Cowan, S.J. Hanson, and C.L. Giles, eds., pp. 50–58. Morgan Kaufmann, 1993.

[14] C.J.C. Burges et al., "Shortest Path Segmentation: A Method for Training a Neural Network to Recognize Character Strings," *IJCNN*, vol. 3, pp. 165-171, 1992.

[15] Y. Lee, "Handwritten Digit Recognition Using k- Nearest-Neighbor, Radial-Basis Function, and Backpropogation Neural Networks," *Neural Computation*, vol. 3, pp. 440–449, 1991.

[16] J. Bromley and J. Denker, "Improving Rejection Performance on Handwritten Digits by Training With Rubbish," *Neural Computation*, vol. 5, no. 3, pp. 367–370, 1993.

[17] D.G. Lowe, *Perceptual Organization and Visual Recognition*. Hingham, Mass: Kluwer Academic Publishers, 1985.

[18] G.E. Hinton, C.K.I. Williams, and M.D. Revow, "Adaptive Elastic Models for Hand-Printed Character Recognition," *Advances in Neural Information Processing Systems 4*, J.E. Moody, S.J. Hanson, and R.P. Lippmann, eds., Morgan Kauffmann, 1992.

[19] J.R. Ullmann, "Correspondence in Character Recognition," *Machine Perception of Patterns and Pictures*. Institute of Physics, London, 1972.

[20] B. Widrow, "The 'Rubber-Mask' Technique–I. Pattern Measurement and Analysis," *Pattern Recognition*, vol. 5, pp. 175–197, 1973.

[21] D.J. Burr, "A Dynamic Model for Image Registration," *Computer Graphics Image Process.*, vol. 15, pp. 102–112, 1981.

[22] D.J. Burr, "Elastic Matching of Line Drawings," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 3, no. 6, pp. 708–713, 1981.

[23] D.J. Burr, "Matching Elastic Templates," *Physical and Biological Proc. Images: Proc. Int'l Symp. Organized By the Rank Prize Funds*, O.J. Braddick and A.C. Sleigh, eds., Springer-Verlag, 1983.

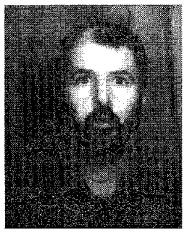
[24] M. Moshfeghi, "Elastic Matching of Multimodality Medical Images," *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing*, vol. 53, no. 3, pp. 271–282, 1991.

[25] M. Varga and R. Hanka, "Dynamic Elastic Image Stretching Technique Applied to Thermographic Images," *IEE Proc.*, vol. 137, pt. 1, no. 3, pp. 146–156, 1990.

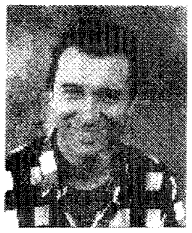
[26] R. Bajcsy and S. Kovacic, "Multiresolution Elastic Matching," *Computer Vision, Graphics, and Image Processing*, vol. 46, pp. 1–21, 1989.

[27] R. Bajcsy, R. Lieberson, and M. Reivich, "A Computerized System for the Elastic Matching of Deformed Radiographic Images to

Idealized Atlas Images," *J. Computer Assisted Tomography*, vol. 7, no. 4, pp. 618–625, 1983.
[28] M.A. Fischler and R.A. Elschlager, "The Representation and Matching of Pictorial Structures," *IEEE Trans. Computers*, vol. 22, no. 1, pp. 67–92, Jan. 1973.
[29] A.L. Yuille, "Deformable Templates for Face Recognition," *J. Cognitive Neuroscience*, vol. 3, no.1, pp. 59–70, 1991.
[30] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models," *Proc. First Int'l Conf. Computer Vision*, Washington, D.C., IEEE Computer Society Press, 1987.
[31] A. Lanitis, C.J. Taylor, and T.F. Cootes, "A Generic System for Classifying Variable Objects Using Flexible Template Matching," *Proc. British Machine Vision Conf.*, J. Illingworth, ed., vol. 1, pp. 329–338. BMVA Press, 1993.
[32] B.S. Everitt, *An Introduction to Latent Variable Models*. Chapman and Hall, 1984.
[33] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes, *Computer Graphics Principles and Practice*. Second edition, Addison-Wesley, 1990.
[34] U. Grenander, Y. Chow, and D.M. Keenan, *Hands: A Pattern Theoretic Study of Biological Shapes*, Springer-Verlag, 1991.
[35] S. Edelman, S. Ullman, and T. Flash, "Reading Cursive Hand-Writing by Alignment of Letter Prototypes," *Int'l J. Computer Vision*, vol. 5, no. 3, pp. 303–331, 1990.
[36] C.K.I. Williams, M.D. Revow, and G.E. Hinton, "Hand-Printed Digit Recognition Using Deformable Models," *Spatial Vision in Humans and Robots*, L. Harris and M. Jenkin, eds. Cambridge Univ. Press, 1993.
[37] J. Bertille, "An Elastic Matching Approach Applied to Digit Recognition," *Proc. Second Int'l Conf. Document Analysis and Recognition*, pp. 82–85, IEEE Computer Society Press, Los Alamitos, 1993.
[38] D.J.C MacKay, "Bayesian Interpolation," *Neural Computation*, vol. 4, pp. 415–447, 1992.
[39] R. Durbin and D. Willshaw, "An Analogue Approach to the Travelling Salesman Problem," *Nature*, vol. 326, pp. 689–691, 1987.
[40] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data Via the EM Aalgorithm," *Proc. Royal Statistical Society*, vol. 39, pp. 1–38, 1977.
[41] X. Meng and D.B. Rubin, "Recent Extensions to the EM Algorithm," *Bayesian Statistics 4*, J.M. Bernardo, J.O. Berger, A.P. Dawid, and A.F.M. Smith, eds., pp. 307–320. Oxford Univ. Press, 1992.
[42] J. Hampshire and A. Waibel, "A Novel Objective Function for Improved Phoneme Recognition Using Time-Delay Neural Networks," Technical Report CMU-CS-89-118, Pittsburgh: Carnegie Mellon Univ., 1989.
[43] P. Brown, *The Acoustic-Modeling Problem in Automatic Speech Recognition*, PhD thesis, Carnegie Mellon Univ., 1987. Also published as IBM Research Division Technical Report RC 12750.
[44] C.K.I. Williams, *Combining Deformable Models and Neural Networks for Handprinted Digit Recognition*, PhD thesis, Univ. of Toronto, 1994.
[45] P. Dayan, G.E. Hinton, R.M. Neal, and R.S. Zemel, "The Helmholtz Machine," *Neural Computation*, vol. 7, no. 7, pp. 889–904, 1995.
[46] J.J Hull, "A Database for Handwritten Text Recognition Research," *IEEE Trans. Pattern Analysis and Machine Intellegince*, vol. 16, no. 5, pp. 550–554, 1994.
[47] J.S. Bridle, "Probabilistic Interpretation of Feedforward Classification Network Outputs, With Relationships to Statistical Pattern Recognition.," *Neuro-Computing: Algorithms, Architectures and Applications*, F. Fougelman-Soulie and J. Hérault, eds., NATO ASI Series on Systems and Computer Science. Springer-Verlag, 1990.
[48] T. Hastie and R. Tibshirani, *Handwritten Digit Recognition Via Deformable Prototypes*, Technical Report, Dept. of Statistics, Univ. of Toronto, 1992.
[49] T.M. Ha and H. Bunke, "Handwritten Numeral Recognition By Perturbation Method," *Proc., Fourth Int'l Workshop on Handwriting Recognition*, pp. 97–106, 1994.
[50] G.E. Hinton, M. Revow, and P. Dayan, "Recognizing Handwritten Digits Using Mixtures of Linear Models," *Advances in Neural Information Processing Systems 7*, G. Tesauro, D.S. Touretzky, and T.K. Leen, eds., pp. 1,015–1,022. MIT Press, Cambridge Mass., 1995.
[51] C.K.I. Williams, M. Revow, and G.E. Hinton, "Using a Neural Net to Instantiate a Deformable Model," *Advances in Neural Information Processing Systems 7*, G. Tesauro, D.S. Touretzky, and T.K. Leen, eds., pp. 965–972. MIT Press, Cambridge Mass., 1995.
[52] A. Gupta, M.V. Nagendraprasad, A. Liu, P.S.P. Wang, and S. Ayyadurai, "An Integrated Architecture for Recognition of Totally Unconstrained Handwritten Numerals," *Int'l J. Pattern Recognition and Artificial Intelligence*, vol. 7, no. 4, pp. 757–773, 1993.



Michael Revow received the BSc and MSc degrees in electrical engineering from the University of the Witwatersrand, and the PhD degree from the University of Toronto in 1988. He is a research associate at the University of Toronto. His research interests include the application of neural networks and statistical pattern recognition to practical problems in high dimensions.



Christopher K.I. Williams received the BS degree in physics at Cambridge University in 1982 and continued on to do Part III maths (1983). He received a MSc degree in water resources at the University of Newcastle upon Tyne before going to work in Lesotho, South Africa, in low-cost sanitation. In 1988 he returned to academia, studying neural networks/AI at the University of Toronto, completing the MSc degree in 1990 and the PhD degree in 1994. He moved to Aston University in 1994 and is currently a lecturer in the Department of Computer Science and Applied Mathematics. His research interests cover a wide range of theoretical and practical issues in neural networks, statistical pattern recognition, computer vision, and artificial intelligence.

Geoffrey E. Hinton received his PhD in artificial intelligence from Edinburgh University in 1978. He is a fellow of the Canadian Institute for Advanced Research and professor of computer science and psychology at the University of Toronto He does research on ways of using neural networks for learning, memory, perception, and symbol processing, and has more than 100 publications in these areas including two *Scientific American* articles. His main contributions have been Boltzmann machines (with Sejnowski), Back-propagation (with Rumelhart and Williams), and Helmholtz machines (with Dayan, Neal, Zemel, and Frey).