# Using generative models for handwritten digit recognition

Michael Revow (revow@cs.toronto.edu)

Christopher K. I. Williams (ckiw@cs.toronto.edu)

Geoffrey E. Hinton (hinton@cs.toronto.edu)

Department of Computer Science

University of Toronto

6 Kings College Road

Toronto, Ontario Canada M5S 1A4

Submission to PAMI as a regular paper

## 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.

# I. 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 neighbouring 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 neighbours.

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 3-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 III, 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 IV, we show how the underlying parameters of the models may be learned. Section V discusses refinements of the basic ideas. Section VI 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.

## II. 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], [4]). 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, [5] extracts 8 central and 2 raw moments as features. On the other hand the recognizer used by Lam and Suen [6] 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 [7] 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 [8], [9], [10] of successful attempts to automatically learn appropriate local features using feedforward neural networks. Some researchers [11], [4], [12] 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 [13]. 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 [14], 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] These are broad terms applied to the object recognizer as a complete entity. Obviously feature selection and classifier design may be independently described.

1. Conventionally, a recognizer does not help to guide segmentation by dividing the image into significant and irrelevant parts. So a system typically [15] 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 [16]. This can be ameliorated by training the system to reject junk images [17], 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 [18], we have investigated the use of deformable elastic models for handwritten digit recognition [19]. Models of this general type have been used in computer vision since the early 1970's. Ullmann [20] 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 [21], 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 [22] and line drawings [23]. 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 [24], global size and rotation adjustments were included. The method has been adapted to match tomographic [25] and thermographic images [26].

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 [27], [28] 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 [29] 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 [30]. The facial feature model example they used has been extended by Yuille [30], 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 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" [31], 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 [32], 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. We describe a matching process with a simple statistical interpretation under which the probability of the model generating the image is maximized.

# III. Matching Elastic Spline Models to Images

## A. Overview

Each of the ten digits has its own elastic model. 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.

## B. Elastic Spline Models

We model each digit with a deformable spline whose shape is determined by the positions of at most 8 control points[2]. A particular realization, $\mathbf{X}$, of a model[3] can be characterized by giving the positions of its $n$ 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. In computing the weighting coefficients we use a cubic B-spline and treat the

[2]The model of a one needs only 3 control points while the seven-model uses 5.

[3]We use upper case to denote a particular realization of the model. Lower case is used to designate the $(x, y)$ location of an individual control point.

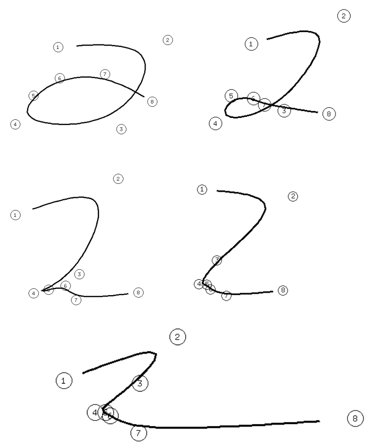


Figure 1: Illustrates the flexibility of spline models to capture topological variations. The large loop of the 2 can smoothly decrease in size and eventually become a cusp. The control points are labelled 1 through 8.

first and last control points as if they were doubled. 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 hypersphere $\delta V$ is approximately:

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

where $\mathbf{H}$ is the *home* locations of the control points and $\Sigma$ the covariance matrix. Thus a single deformable model defines an entire probability distribution across shape instances.

Following [1] and [33] we define the *deformation energy*, $E_{def}$, to be the negative log probability of the deformation.

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

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 (figure 1). This advantage of spline models is pointed out in [34] 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.

## C. 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 $\mathbf{X}$. This defines the spline in object-based coordinates. The log probability of picking a deformation is proportional to the quadratic term in (2). It is important that the deformation is measured in object-based coordinates.

(2) Pick an affine transformation[4] from the model's intrinsic reference frame to the image frame (i.e. pick a size, position, orientation, slant and elongation for the digit).

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

(4) Repeat many times:

Either (with probability $\pi_n$) 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.

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

## D. Fitting a model to an image

To classify an image ($I$), each of the models ($m$) is fitted to the data and the model that best "explains" the image is chosen. If we assume a uniform prior over all digits, then the posterior probability, $P(m|I)$, for each model is proportional to the *evidence* [35], $P(I|m)$:

$$P(I|m) = \int P(I|\boldsymbol{\alpha}, m) P(\boldsymbol{\alpha}|m) d\boldsymbol{\alpha} \tag{3}$$

where $\boldsymbol{\alpha}$ is the vector of instantiation parameters for the model. To avoid confusion, we clarify that $\boldsymbol{\alpha}$ is the instantiation vector of the model in the *image* frame. In other words, $\boldsymbol{\alpha}$ is the concatenation:

$$\boldsymbol{\alpha} = [\mathbf{X} \;\; \mathbf{Y}]^T \tag{4}$$

where $\mathbf{Y}$ is the vector of affine instantiation parameters and $\mathbf{X}$ are the deformations or deviations of the control points from their desired home locations in the object frame. Performing the integration over instantiation parameter space is infeasible, so we make the assumption that the integrand in (3) has a strong peak around the most probable parameter values ($\boldsymbol{\alpha}^*$). The evidence can then approximated by the height ($P(I|\boldsymbol{\alpha}^*, m)P(\boldsymbol{\alpha}^*|m)$) of the peak multiplied by the volume of the parameter space under the peak. The negative logarithm of the evidence can then be approximated by:

$$-\log P(I|m) \approx -\log P(I|\boldsymbol{\alpha}^*, m) - \log \;\; P(\boldsymbol{\alpha}^*|m) - H \tag{5}$$

where $H$ is the logarithm of the volume term. When the posterior is well modelled by a Gaussian, then $H = \frac{k}{2}\log 2\pi - \frac{1}{2}\log|\mathcal{H}|$, where $\mathcal{H} = -\nabla\nabla \log P(\boldsymbol{\alpha}|I, m)$ is the Hessian evaluated at $\boldsymbol{\alpha}^*$. In the sequel we treat $H$ as a constant, but we allow it to be a different constant for each model (see section III.E). The second term is just $E_{def}$. 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:

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

Assuming that each inked pixel in the image is generated independently from a distribution defined by the Gaussian distributed beads and a uniform noise field, the data fit term is the sum of log probabilities of each inked pixel.

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

where $N_i$ is the the number of inked pixels in the image and $P_k$ is the probability of inking pixel $k$:

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

where $N$ is the number of pixels which the uniform noise field is distributed over (normally the whole image), $B$ is the number of beads, $\pi_n$ the mixing proportion of a uniform noise field and $P_{kb}$ is the probability of inking pixel $k$ under bead $b$, which is assumed to be Gaussian. 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:

$$E_{fit} = -\sum_{k=1}^{N_i} W_k \log P_k \tag{9}$$

Normally we set $W_k = \lambda/N_i$ where $\lambda$ 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.

It is reasonable to assume that model-deformations are independent of the affine parameters and 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 III.E 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 set of parameters 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 [36] 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 [37]. This involves the repeated application of a two step procedure which will not increase $E_{tot}$ as $\boldsymbol{\alpha}$ 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 $\boldsymbol{\alpha}$ computed to minimize $E_{tot}$. In the conventional application of EM, the beads would be unconstrained, and hence a bead would move to the centre of gravity of the data (pixels), weighted by the responsibilities that the bead has for each pixel. 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 $\boldsymbol{Y}$ is held fixed and we solve a set of *linear* simultaneous equations ($\partial E_{tot}/\partial \mathbf{X} = 0$) to find the posterior deformations of the control points. These computations are carried out in the image based frame. Although the deformations are defined in the object based frame, they can be mapped into the image frame because the affine is held constant at this stage. The control point covariance matrix, $\Sigma$, must also be mapped through the same affine transformation. 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[5].

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 the deformation energy in the *image based frame*:

$$E^i_{def} = \frac{1}{2}(\mathbf{X}^i - A\mathbf{H}^o - \mathbf{T})^T(\Sigma^i)^{-1}(\mathbf{X}^i - A\mathbf{H}^o - \mathbf{T}) + \frac{1}{2}\log|\Sigma^i| + \text{const} \tag{10}$$

where we have distinguished the image and the object-based frames of reference by superscripts ${}^i, {}^o$ and $(A\mathbf{H}^o + \mathbf{T})$ is the vector of control point home locations when transformed into the image frame through the affine transform, represented in (10) 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 $2 \times 2$ 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. $\Sigma^i$ is the covariance matrix referred to the image reference frame ($(\Sigma^i)^{-1} = A^{-T}\ (\Sigma^o)^{-1}\ A^{-1}$). The minimization of (10) with respect to the affine parameters is a non-linear problem, but we have found that satisfactory results are obtained when we treat $\Sigma^i$ 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

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

a quadratic form of the affine parameters and the minimization is straightforward[6].

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 $\mathbf{x}_b$ using the update:

$$\hat{\sigma}^2 = \frac{1}{2N_i} \sum_{k=1}^{N_i} \sum_{b=1}^{B} r_{kb} \left\| \mathbf{x}_b - \mathbf{z}_k \right\|^2 \tag{11}$$

where $\mathbf{z}_k$ is the location of pixel $k$. In order to ensure that bead centres remain approximately two standard deviations apart, the number of beads along the spline is periodically adjusted.

Some stages in fitting models to data are shown in Figure 2. 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 $E_{tot}$ (equation 5) is chosen, then the conclusion is that the data is most likely to have been generated by a 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 III.E, 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 $E_{tot}$.

Our generative models of digits include a noise model. Each inked pixel may be generated either by the model or by 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. Figure 3 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.

[6] If we had performed the non linear minimization then $E_{tot}$ would be guaranteed not to increase during the second stage of the M-step.

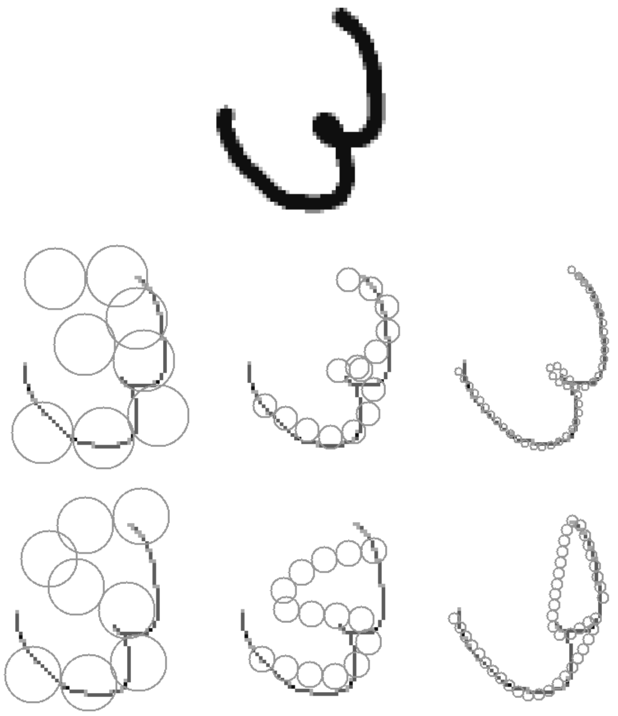


Figure 2: Some stages of fitting models to an image of a 3. The image is displayed in the top row. In subsequent rows, the circles represent the beads. The radius has been set to one standard deviation of the circular Gaussian distribution. Because the bead variance shrinks to approximate the stroke 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 centre 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.

## E. Recognizing isolated digits

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.

One way in which our generative model is deficient is that it does not explicitly penalize matches in which there are beads far from any inked pixels. For example in figure 2, the five

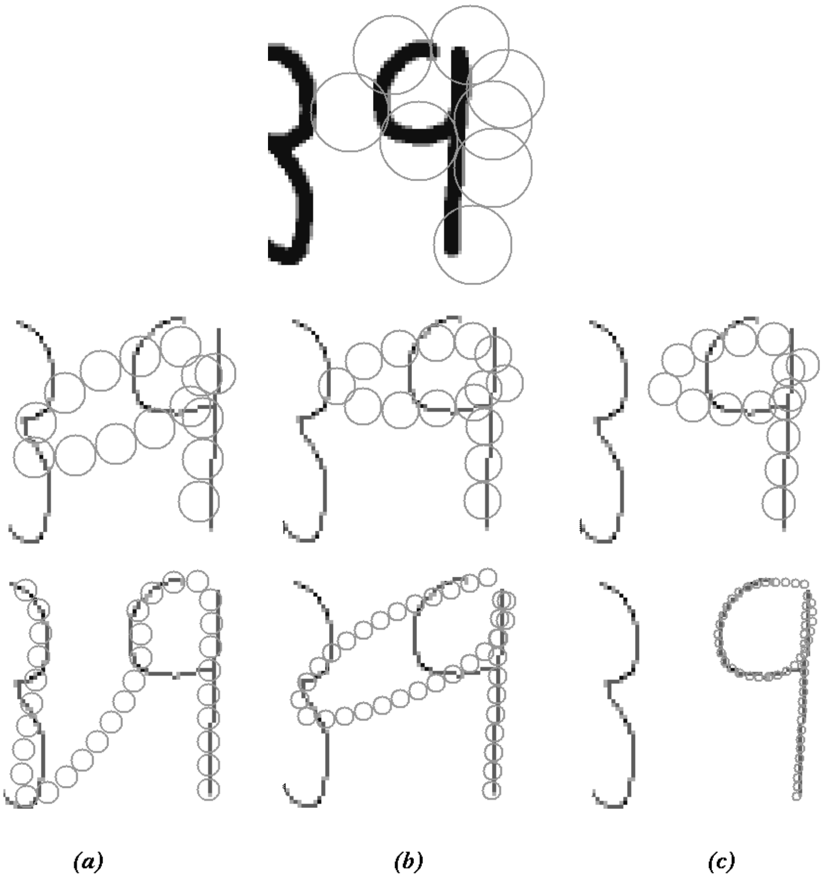



Figure 3: Illustration of how the noise model helps in model driven segmentation. The top row shows an image of a nine with a portion of its left neighbour. The initial configuration of the nine-model is also shown. The remaining rows show the configurations of the model part way in the settling (second 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 figure 3.

model has accounted for all the inked pixels, but the final fit has left its centre bar far from any inked pixels. We call this the "beads in white space" problem[7]. Motivated by research on "snakes" [31], another energy term, $E_w$, was defined to penalize beads in white space. This term is similar to the "support measure" [32] or the symmetric matching used in [22] and [23].

[7]There is a small implicit penalty in that beads far from inked pixels are not available for accounting for inked pixels. A more elaborate generative model for both inked and non-inked pixels is suggested in section V.

$$E_w = -\sum_{b=1}^{B} \log \sum_{k=1}^{N_i} P_{kb} \tag{12}$$

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.

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. So we take into account terms which reflect rotation, shear and elongation of the affine.

The last term used is the final variance of the beads in each model. This term is included because it affects the normalizer for $P_{kb}$ and hence the data fit energy (equation (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 figures 3 and 10).

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 postprocessing neural network. 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 $\tilde{E}^k_{fit} = E^k_{fit} - E^{min}_{fit}$ as the data fit term for the $k^{th}$ model, where $E^{min}_{fit}$ is the minimum data fit obtained by any of the models for the current image. 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 [38] which guarantees that the 10 output values form a probability distribution. Including biases on the output units[8], the network has 80 weights and is trained using conjugate gradient to minimize a cross-

[8]The bias on each output unit can be thought of as the volume factor for each model in equation 5.

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 [39] 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 (equation (5)) also did not improve performance.

# IV. Learning the Models

Each elastic model is described by a vector of mean or *home* locations ($\mathbf{H}$) and a covariance matrix, $\Sigma$ (see equation 2). 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 (figure 4), probably because we start off with good models.

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 [40], 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 [41]. Early experiments showed that, for our generative models, maximum likelihood learning was just as effective as discriminative learning, per-

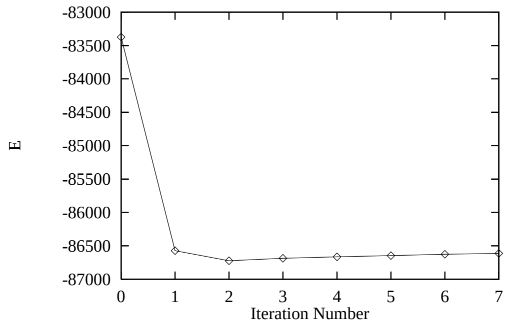



Figure 4: Training the two-model. The ordinate shows the sum $E = \sum_I E_{tot}(I)$ over all two-images in the training set. The model has essentially completed its learning after the second pass through the training set.

haps 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 equation (5), the sample covariance matrix is estimated from $N_T$ training examples using:

$$\Sigma = \frac{1}{N_T}(\mathbf{X} - \mathbf{H})(\mathbf{X} - \mathbf{H})^T. \tag{13}$$

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 $\Sigma$. 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 $\Sigma$, we regularized $\Sigma$ 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.

# V. Refining the model

## A. 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 equation (1) we have chosen to describe this distribution as a Gaussian hyper-ellipsoid. For a typical model with 8 control points this characterization requires specification of a $16 \times 16$ covariance matrix. It is interesting to investigate different simplifications.

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

$$\Sigma = \sigma^2 \mathbf{I} \tag{14}$$

where $\mathbf{I}$ is the identity matrix. The control points then all have identical, independent radial Gaussian distributions. The deformation energy simplifies to:

$$E_{def} = \frac{\|\mathbf{X} - \mathbf{H}\|^2}{2\sigma^2} + n \log 2\pi\sigma^2 \tag{15}$$

This characterization of the distribution by a single *generic* model can result in poor approximations to the true distribution. Figure 5 illustrates a situation in 2 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 (15). Under this approximation the distribution of models is given by:[9]

$$P(\mathbf{X}) = \sum_{l=1}^{L} \frac{\beta_l}{(2\pi\sigma_l^2)^n} \exp\{-\frac{\|\mathbf{X} - \mathbf{H}_l\|^2}{2\sigma_l^2}\} \tag{16}$$

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

[9] 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.

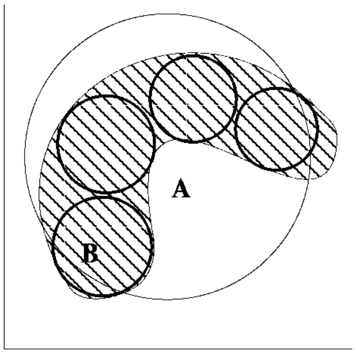



Figure 5: An arbitrary 2-dimensional distribution represented by the shaded region could be modeled by a a single Gaussian with large variance as illustrated by the large circle. A better approximation would be to use a mixture of Gaussians each with a smaller variance. Under the single Gaussian approximation point **A** would be incorrectly considered to be more likely than point B.

The centres $\mathbf{H}_l$ and variances $(\sigma_l^2)$ are computed using EM to maximize the log likelihood of a training set under the mixture distribution (16). Figure 6 shows the 10 local models in the mixture for the two-model. The mixture has been able to capture dominant styles. For example, variations in the presence and size of the loop have been well represented.

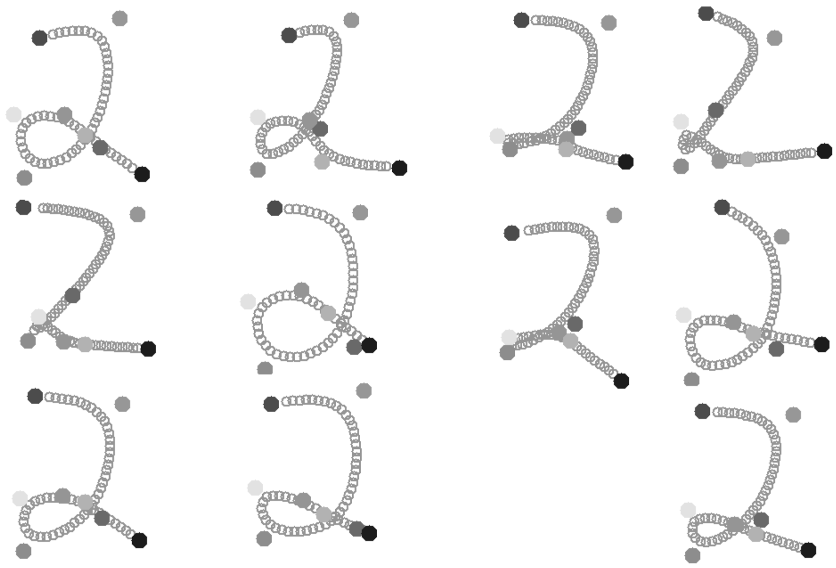


Figure 6: Local models in the mixture for the 2 model. The generic model is shown in the bottom right corner.

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 the fitting is finished the deformation energy is evaluated using the log probability under the mixture distribution (16) instead of using $E_{def}$ as in (15). 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.

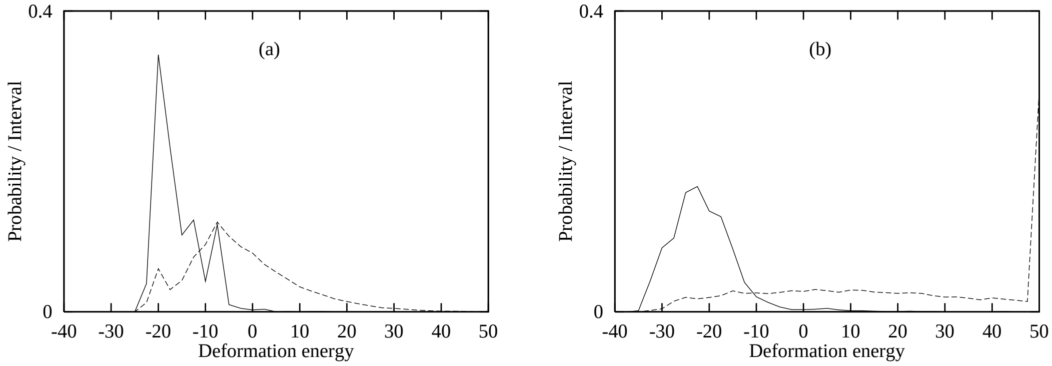



Figure 7: The solid line is the $E_{def}$ distribution of models when fitted to example images containing the correct digit (e.g.. a three model fitted to an image of a 3). The dotted curve is the $E_{def}$ distribution of the models when fitted to incorrect data. (a) Using a single generic model. (b) 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))

Figure 7 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.

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 VII). There may even be mutual information between the pairings of local models for different digits (e.g. between a 4 and a 6) [42].

## B. Generating both the black and 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 black 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 beads are a long way from the nearest inked pixel. In section III.E we compensated for this deficiency by introducing another term, $E_w$, which is evaluated *after* the model has fitted the data. Unfortunately, this approach does not stop a model from settling into a configuration with beads "in white space". A more principled way of penalizing beads in white space is to formulate a generative model that generates both the white and the black pixels.

We assume that the image is generated from the spline by a two-stage stochastic process. The first stage computes the probability $P_k(w)$ that each pixel in the image would be remain white if *multiple* samples were taken from the probability distribution defined by the beads and the uniform noise process (see equation (8)). We take $N_m$[10] samples from this distribution. The probability that none of these samples landed within a particular pixel is:

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

The predicted probability of a pixel being inked, $\hat{P}_k(b)$, is simply the complement.

$$\hat{P}_k(b) = 1 - \hat{P}_k(w) \tag{17}$$

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

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

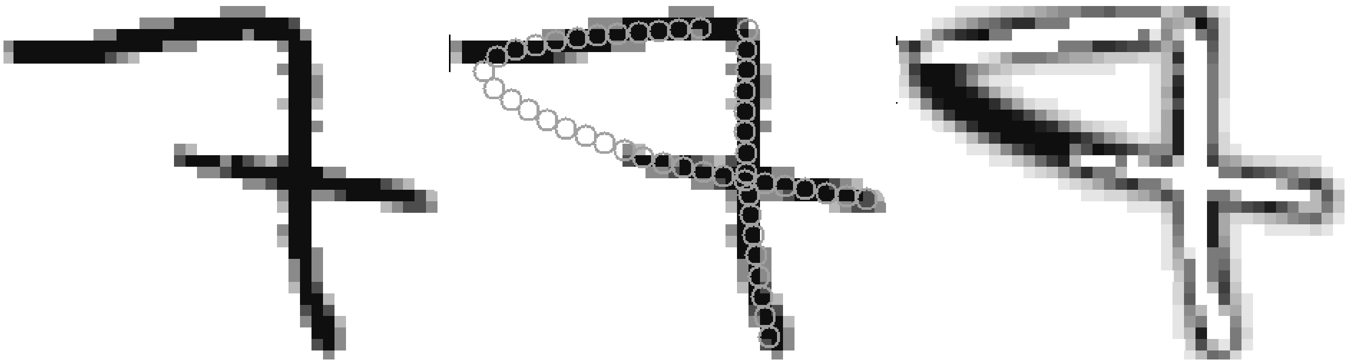


Figure 8: The left panel shows an image of a 7. The fit of a four using the generative model which produces only black pixels is shown in the centre panel. Under this formulation the centre bar of the four model would not be penalized for lying in white space. The right most panel shows the image weighted by the probabilities of the full generative model with areas of low probability shaded dark. For display purposes, the noise model was set to zero when evaluating the full generative model so to prevent the background from turning grey.

$$\log P(I|m) = \sum_{\substack{k \,\in\, black \\ pixels}} \log \hat{P}_k(b) + \sum_{\substack{j \,\in\, white \\ pixels}} \log \hat{P}_j(w) \tag{18}$$

In figure 8 we show a model of a four settled on the image of a seven using the generative model of section III. The right most panel figure shows the probabilities generated by (18). Areas of low probability are shaded dark. The portion of the centre 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.)

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 (18) will be dominated by the the cost of generating black pixels. Using $(1 + \delta x)^n \;\simeq\; 1 + n\delta x$, we see that at high variance:

$$\log P(I|m) \;\;\simeq\;\; \sum_{\substack{k \,\in\, black \\ pixels}} \log \hat{P}_k(b)$$

$$
\begin{aligned}
&= \sum_{\substack{k \in \, black \\ pixels}} \log(1 - (1 - P_k)^{N_m}) \\
&\simeq \sum_{\substack{k \in \, black \\ pixels}} \log(P_k) + N_m \log N_m
\end{aligned}
$$

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 III (see equation 9).

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 III. 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. The main disadvantage of the method is that it is much more computationally intensive because it considers both inked and non-inked pixels. We have used (18) to *evaluate* the data-fit of the settled configurations obtained by running the simpler model of section III and found no improvement over using the combination of equations (9) and (12).

## C. Speeding up the search

In order to classify an image of a single digit, 10 models[11] 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 of pixels. As the model settles onto the image the bead variance generally decreases with a concomitant rise in the number of beads (see section III.D) 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

[11] 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".

freeze these responsibilities at their current low values. In other words, during the E-step of the settling algorithm described in section III.D, 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 (5) with much less computation.

Conventionally the EM algorithm is seen as a way of maximizing the log likelihood, $L(\boldsymbol{\alpha}) = \log P(I|\boldsymbol{\alpha})$, of a model parameterized by $\boldsymbol{\alpha}$, 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 $\boldsymbol{\gamma}$ and maximization is achieved with the help of $\boldsymbol{\gamma}$. With this view of the EM algorithm, it is not immediately obvious that partial implementation of the expectation step is justified. However in a recent alternative interpretation of the EM algorithm [43], Neal and Hinton present the EM algorithm as maximizing a joint function, $F(P(\boldsymbol{\gamma}), \boldsymbol{\alpha})$ of the distribution of the unobserved data and model parameters. If $\boldsymbol{\gamma}$ is chosen to be the the optimal distribution, $\gamma^*_\alpha$, that maximizes $F$ for the current value of $\boldsymbol{\alpha}$, then $F = L(\boldsymbol{\alpha})$. At the $t^{th}$ iteration, the first step of the standard EM algorithm chooses $P^t(\boldsymbol{\gamma})$ to maximize $F(P, \boldsymbol{\alpha})$ and the second sets $\boldsymbol{\alpha}$ to maximize $F(P^t, \boldsymbol{\alpha})$. However, it is not necessary to compute the optimal distribution of $\boldsymbol{\gamma}$ in the E step. Any change in $\boldsymbol{\gamma}$ that reduces the Kullback-Liebler distance between $\boldsymbol{\gamma}$ and $\gamma^*_\alpha$ is guaranteed to improve $F$. In the elastic model the unobserved variables, $\boldsymbol{\gamma}$, 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 $F$.

# VI. Results on isolated digits

The performance of the elastic net in recognizing isolated digits has been tested on data from the CEDAR CDROM 1 database of Cities, States, ZIP Codes, Digits, and Alphabetic Characters [44]. The *br* training set of binary segmented digits was subdivided into 3 training sets of size 2000, 7000 and 2000 respectively. A validation set of 2000 examples was also generated from the *br* training set to allow as 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 IV) 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 2 test sets. The *goodbs* (2213 images) set is a subset of the *bs* (2711 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 [44].

| | 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 |

Table 1: Percentage of images incorrectly classified by the elastic net with no rejections. 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.

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 ten models, an initial affine is computed so as to position the model over the enclosing rectangle of all the black pixels in the image.

(3) The models are allowed to settle with the iterative algorithm described in section III so

as decrease $E_{tot}$ in equation (5). The fractional change in $E_{tot}$ is monitored and when it falls below 0.001, the number of beads are adjusted (section III.D). This is repeated about 6 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.

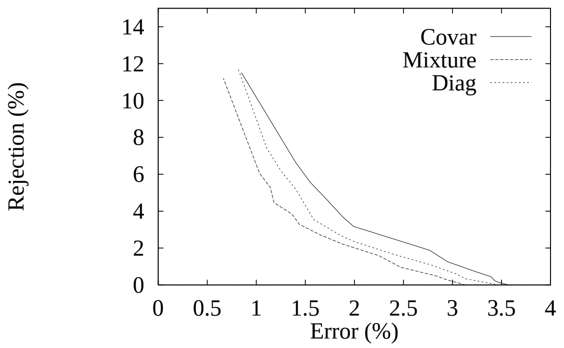
Table 1 shows the performance of the elastic net when the rejection threshold was set to zero. The first line in the table shows results when $\Sigma^{-1}$ (see equation 2) was estimated from the data. The second row shows results obtained when all models had the same diagonal covariance matrix $\frac{1}{\sigma^2}\mathbf{I}$ 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 V.

Varying the rejection threshold in the post-processing neural network allows us to trade off errors against rejects. Figure 9 shows error-rejection curves obtained on the validation and test sets.

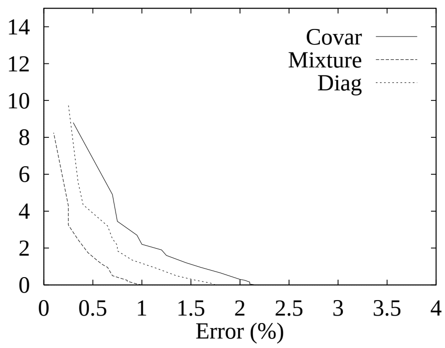
# VII. DISCUSSION

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. An examination of the errors made on the validation set reveals that almost all mis-classifications can be attributed to two problems: local minima in the search space and modeling difficulties.

During the fitting, models navigate a search space with many local minima. 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




*bs* Test Set




Validation Set

Figure 9: Error-rejection relationships. Full covariance matrix - solid curve. Diagonal covariance matrix - dotted curve. Mixture model - dashed curve.

the current system, very little information from the image is used to pick initial model instantiation parameters. The deformations, $\mathbf{X}$ (see equation (4)) 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 III 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 could train a conventional supervised multi-layer neural network to predict model instantiation parameters from the image [42]. Given an input image[12], 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 overfitting than conventional neural network recognizers [8], [9], [45]. In conventional networks, each training example only provides $\log_2 10$ bits of constraint on 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

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

present in handwritten digits. Although spline models are good at capturing most common variations (see figure 1) they cannot easily model large embellishments to the basic shape. Figure 10 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 (equation (8)). An improvement would be to use a more structured noise model. Another possibility is to model the residual images using "flourish models" [46]. 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.

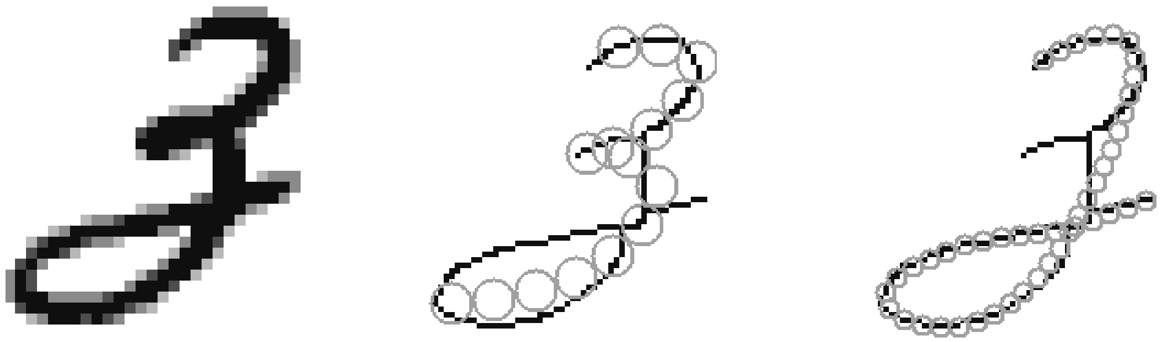


Figure 10: The difficulty of flourishes. The middle panel show the fit of the three model while the right panel shows how the model of a two captured the entire flourish. Notice how the three model did not shrink its variance to the same degree as the two-model because it used the large variance to compensate for its inability to model the flourish portion. The data has been thinned in the last two panels for the same reasons explained in figure 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 figure 2). 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 figure 2. A related benefit of the approach is the model driven segmentation illustrated in figure 3.

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 figure 4). This is in contrast to the thousands of free parameters and multiple passes over the training data required by a typical neural net recognizer [8].

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 V 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 figure 11.

It is immediately clear that for all digits, except ones, there is a significant amount of information 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[13], which does not allow for much style variation.

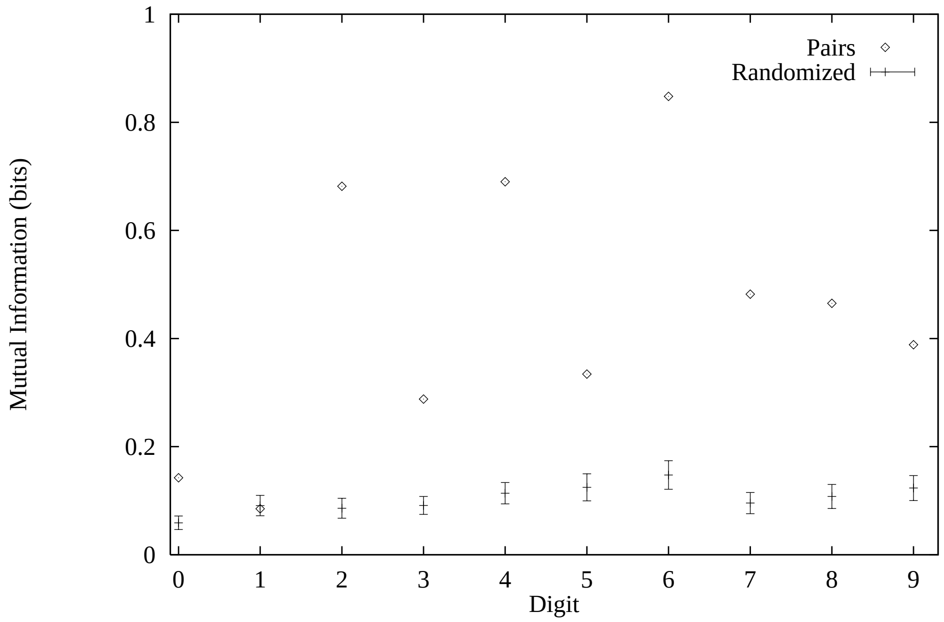



Figure 11: Mutual information computed for pairs of digits from the same zip code. Also shown are means and error bars of mutual information computed when images are randomly paired.

Thus far we have only used the models to answer the question *"which of these generative models best explains the image?"*. However, we could go further and ask if the explanation that the best model provides is reasonable. So for example if we fit all the digits models to an image of a cup, one of the models will fit best, but the explanation will probably be poor. This suggests another method for performing rejection which could not only be used to reject non-digit images, but could also be used to reject cases in which a local minimum fit was found or when the models do not have sufficient flexibility to generate the image.

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

One simple way of deciding whether to reject the fit of a winning model to an image is to compare the quantities used in the post-processing network against the distribution of these quantities on training examples. In some preliminary experiments we chose to use only $E_{fit}$, $E_w$ and $E_{def}$. Because $E_{fit}$ depends on the number of inked pixels in the image we used $\tilde{E}_{fit} = E_{fit} - \hat{E}_{fit}$ where it turns out that the predictor $\hat{E}_{fit}$ is adequately modelled as a linear function of $\log N_i$. There is a different linear function for each model. Assuming independence of the three quantities we compute[14]:

$$-\log P(\tilde{E}_{fit}) - \log P(E_{def}) - \log P(E_w) \tag{19}$$

and reject fits which exceed a threshold.

A data set of 1554 upper and lower case letters was extracted from data on the CEDAR CDROM 1. We attempted to have all letters equally represented, but some letters such as z, Z, q and Q were under represented. The threshold was chosen to reject about 10% of the training digits. Using this threshold, 78% of the letters were rejected. While this may appear to be rather low, it should be noted that many of the accepted letters are easily confused in isolation with digits. For example, people have difficulty distinguishing a zero from a upper or lower case "o" when no context is supplied. In an informal test with people unfamiliar with the digit recognizer we found that they confused about half of the 22% images mis-classified as digits.

It is also possible to construct error-rejection curves using a range of thresholds with (19). The results with the mixture of local models are shown in figure 12. Because the distributions were created from histograms it is possible that a probability term in (19) could become zero and so the method will insist on rejecting those images. This method gives similar error-rejection curves to varying the threshold in the post-processing neural network (see figure 9). However, on the *bs* test set, the error-rejection tradeoff is slightly better using the simpler method of choosing a threshold in the post-processing network.

[14]The individual distributions were approximated using histograms from the training examples

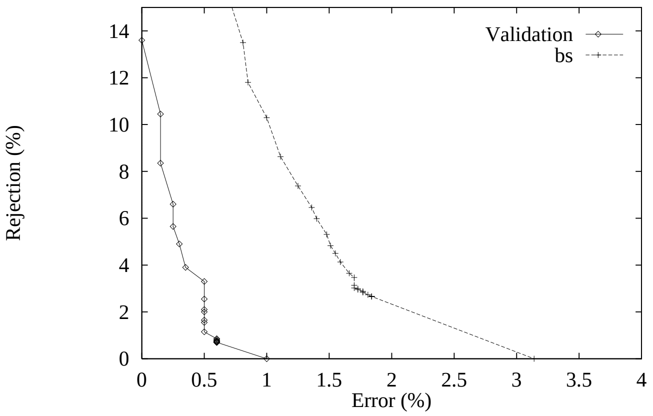



Figure 12: Rejection-error curves on the validation and test sets when the fit of the winning model is examined in light of (19). $E_{def}$ was computed using the mixture of local models. As explained in the text, the method always rejects some images, but in the figure we have included the zero rejection point for reference purposes.

# VIII. 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 Centre.

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] C. Y. Suen, "Automatic recognition of handprinted characters - the state of the art", *Proceedings of the IEEE*, vol. 68, no. 4, pp. 469–487, 1980.

[3] J. Mantas, "An overview of character recogntion methodologies", *Pattern Recognition*, vol. 19, no. 6, pp. 425–430, 1986.

[4] C . Y. Suen, C. Nadal, R. Legault, T. A. Mai, and L. Lam, "Computer recognition of unconstrained handwritten numerals.", *Proceedings of the IEEE*, vol. 80, no. 7, pp. 1162–1180, July 1992.

[5] 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.

[6] 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.

[7] M. Shridhar and A. Badreldin, "Recognition of isolated and simply connected handwritten numerals", *Pattern Recognition*, vol. 19, no. 1, pp. 1–12, 1986.

[8] Y. Le Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel, "Handwritten digit recognition with a back-propagation network", in *Advances in Neural Information Processing Systems*, D.S. Touretzky, Ed., Denver, 1990, vol. 2, pp. 396–404, Morgan Kaufmann, San Mateo.

[9] J. D. Keeler, D. E. Rumelhart, and W. K. Leow, "Integrated segmentation and recognition of hand-printed numerals", in *Advances in Neural Information Processing Systems 3*, R. P. Lippmann, J. E. Moody, and D. S. Touretzky, Eds., San Mateo, CA, 1991, pp. 557–563, Morgan Kaufmann Publishers.

[10] K. Fukushima and N. Wake, "Handwritten alphanumeric character recognition by the neocognitron", *IEEE Trans. Neural Networks*, vol. 2, no. 3, pp. 355–365, 1991.

[11] D. Lee and S. N. Srihari, "Handprinted digit recognition: A comparison of algorithms", in *Third international workshop on frontiers in handwriting recognition*, Buffalo, NY, May 1993, pp. 153–162.

[12] F. Kimura and M. Shridhar, "Handwritten numerical recognition based on multiple algorithms", *Pattern Recognition*, vol. 24, no. 10, pp. 969–983, 1991.

[13] J. Geist, R. A. Wilkinson, S. Janet, P. J. Grother, B. Hammond, N. W. Larsen, R. M. Klear, M. J. Matsko, C. J. C. Burges, R. Creecy, J. J. Hull, T. P. Vogl, and C. L. Wilson, "NISTIR 5452. The second census optical character recognition systems conference", Tech. Rep., U.S. National Institute of Standards and Technology, 1994.

[14] P. Simard, Y. Le Cun, and J. Denker, "Efficient pattern recognition using a new transformation distance.", *Advances in Neural Information Processing Systems 4*, pp. 50–58, 1993.

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

[16] Y. Lee, "Handwritten digit recognition using k- nearest-neighbor, radial-basis function, and backpropogation neural networks.", *Neural Computation*, vol. 3, pp. 440–449, 1991.

[17] 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.

[18] D. E. Lowe, *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, Hingham, MA, 1985.

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

[20] J. R. Ullmann, "Correspondence in character recognition", in *Machine Perception of Patterns and Pictures*. Institute of Physics, London, U.K., 1972.

[21] B. Widrow, "The 'rubber-mask' technique–I. Pattern Measurement and Analysis", *Pattern Recognition*, vol. 5, pp. 175–197, 1973.

[22] D. J. Burr, "A dynamic model for image registration", *Comput. Graphics Image Process.*, vol. 15, pp. 102–112, 1981.

[23] D. J. Burr, "Elastic matching of line drawings", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 3, no. 6, pp. 708–713, 1981.

[24] D. J. Burr, "Matching elastic templates", in *Physical and biological processing of images: proceedings of an international symposium organized by the Rank Prize Funds*, O. J. Braddick and A. C. Sleigh, Eds. Springer-Verlag, 1983.

[25] 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.

[26] M. Varga and R. Hanka, "Dynamic elastic image stretching technique applied to thermographic images", *IEE Proceedings*, vol. 137, Pt. 1, no. 3, pp. 146–156, 1990.

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

[28] R. Bajcsy, R. Lieberson, and M. Reivich, "A computerized system for the elastic matching of deformed radiographic images to idealized atlas images", *Journal of Computer Assisted Tomography*, vol. 7, no. 4, pp. 618–625, 1983.

[29] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures", *IEEE Trans. Computers*, vol. C-22, no. 1, pp. 67–92, 1973.

[30] A. L. Yuille, "Deformable templates for face recognition", *Journal of Cognitive Neuroscience*, vol. 3(1), pp. 59–70, 1991.

[31] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models", in *Proceedings of the First International Conference on Computer Vision*, Washington, D. C., 1987, IEEE Computer Society Press.

[32] A. Lanitis, C. J. Taylor, and T. F. Cootes, "A generic system for classifying variable objects using flexible template matching", in *Proceedings of the British Machine Vision Conference*, J. Illingworth, Ed., vol. 1, pp. 329–338. BMVA Press, 1993.

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

[34] S. Edelman, S. Ullman, and T. Flash, "Reading cursive handwriting by alignment of letter prototypes", *International Journal of Computer Vision*, vol. 5, no. 3, pp. 303–331, 1990.

[35] D. J. C MacKay, "Bayesian methods for backpropagation networks", in *Models of Neural Networks II*, J. L. van Hemmen, E. Domany, and K. Schulten, Eds. 1993, Springer.

[36] R. Durbin and D. Willshaw, "An analogue approach to the travelling salesman problem", *Nature*, vol. 326, pp. 689–691, 1987.

[37] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", *Proceedings of the Royal Statistical Society*, vol. B-39, pp. 1–38, 1977.

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

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

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

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

[42] C. K. I. Williams, PhD Thesis, Department of Computer Science, University of Toronto, 1994.

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

[44] J. J Hull, "A database for handwritten text recognition research", *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 16, no. 5, pp. 550–554, 1994.

[45] A. Gupta, M. V. Nagendraprasad, A. Liu, P. S. P. Wang, and S. Ayyadurai, "An integrated architecture for recognition of totally unconstrained handwritten numerals", *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 4, pp. 757–773, 1993.

[46] C. K. I. Williams, M. D. Revow, and G. E. Hinton, "Hand-printed digit recogntion using deformable models", in *Spatial Vision in Humans and Robots*, L. Harris and M. Jenkin, Eds. Cambridge University Press, 1993.