# Efficient Deformable Template Detection and Localization without User Initialization

James Coughlan[*], Alan Yuille, Camper English and Dan Snow

Smith-Kettlewell Eye Research Institute

San Francisco, CA 94115

[*] Address correspondence c/o Smith-Kettlewell Eye Research Institute, 2232 Webster St., San Francisco, CA 94115. Phone: 415-561-0519, fax: 415-561-1610, email: coughlan@skivs.ski.org

# Abstract

*A novel deformable template is presented which detects the boundary of an open hand in a grayscale image* **without initialization by the user**. *A dynamic programming algorithm enhanced by pruning techniques finds the hand contour in the image in as little as 19 seconds on a Pentium 150. The template is translation- and rotation-invariant and accomodates shape deformation, significant occlusion and background clutter, and the presence of multiple hands.*

# Symbols

Boldface letters, e.g. $\mathbf{x}$, denote vectors.

$P(x|y)$ denotes conditional probability of $x$ given $y$.

$\sqrt{a}$ denotes the square root of a.

$\sum$ denotes summation.

$\prod$ denotes repeated product.

$\int$ denotes integration.

$\perp$ denotes "perpendicular to."

$<$, $>$ denote less than and greater than, respectively.

$\nabla I(\mathbf{x})$ denotes the gradient of $I$ with respect to $\mathbf{x}$.

$\propto$ denotes "proportional to."

$\approx$ denotes "approximately equal to."

$\underset{\mathbf{x}}{argmax}\ f(\mathbf{x})$ denotes the value of $\mathbf{x}$ that maximizes $f(\mathbf{x})$.

$f \star g$ denotes the convolution of $f$ with $g$.

$\pi$ denotes the constant 3.14159...

$|\mathbf{x}|$ denotes the modulus of $\mathbf{x}$.

$\infty$ denotes infinity.

Greek letters: $\alpha$, $\beta$, $\delta$, $\Delta$, $\epsilon$, $\lambda$, $\mu$, $\phi$, $\sigma$, $\theta$ denote alpha, beta, delta (lower and upper case), epsilon, lambda, mu, phi, sigma, theta.

# 1 Introduction

A promising approach to the detection and recognition of flexible objects involves representing them by deformable template models, for example, [14, 24, 25, 1, 22]. These models specify the shape and intensity properties of the objects. They are defined probabilistically so as to take into account the variability of the shapes and their intensity properties.

The flexibility of such models means that we have a formidable computational problem to determine if the object is present in the image and to find where it is located. In simple images, standard edge detection techniques may be sufficient to segment the objects from the background though we are still faced with the difficult task of determining how edge segments should be grouped to form an object. In more realistic images, however, large segments of object boundaries will *not* be detected by standard edge detectors. It often seems impossible to do segmentation without using high level models like deformable templates [14].

Many optimization strategies have been proposed for deformable templates. For a description of approaches and theoretical comparisions between certain methods see [26]. Broadly speaking, either the algorithms are too slow for real time processing or they require initialization by the user. If initialization is provided, then algorithms based on dynamic programming [4], such as [8, 9], or on gradient descent [5], can find the optimal solution. (Dynamic programming was also used in some previous work to find *generic* contours in images, such as [3, 19].) Other graph-search algorithms related to dynamic programming are used to perform template matching, such as [12], and also [10], which searchs automatically for the optimal match given a set of local features extracted from an image.

In this paper we show that dynamic programming can, in principle, be used to detect deformable objects in reasonable time *without initialization.* We test this result with a Bayesian deformable hand template (an earlier, non-Bayesian version of this work appears in [6]). Adaptive quantization is used for the continuous variables. This enables the algorithm to find a hand in six minutes in a 388 × 512 image. The use of pruning techniques allows us to speed the algorithm up by an order of magnitude. Our current algorithm runs in 25 seconds on a Pentium 150, and as few as 19 seconds for simple images. Adding an occlusion process allows the algorithm to find the hand even when it is significantly occluded (and to identify the location of the occlusion), and multiple hands can be detected in the same image.

## 2 The Deformable Hand Template

Our deformable template is designed to detect planar views of a hand (or multiple hands) in a grayscale image. The template consists of four elements: a geometric prior, an occlusion process, an imaging model (all three of which make up the Bayesian model) and a dynamic programming optimization algorithm. The geometric prior describes probabilistically what configurations (shapes) the hand template is likely to assume, and the imaging model together with the occlusion process describe probabilistically how any particular configuration will appear in a grayscale image. Given an image, the Bayesian model assigns an a posteriori probability to each possible hand configuration. A dynamic programming algorithm searches for the maximum a posteriori (MAP) configuration, the configuration that optimally fits the image data.

### 2.1 The Geometric Prior

The first part of the Bayesian model is the geometric prior, which assigns a probability to each hand configuration. The configuration is represented by a chain of points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ in the plane which trace the boundary of the 2-dimensional hand shape, and by an associated chain $\theta_1, \theta_2, \cdots, \theta_N$ of normal orientations which describe the direction of outward-pointing normal vectors at the points (see Figure 1). Each point $\mathbf{x}_i$ has two components $x_i$ and $y_i$ and is quantized to the image pixel lattice. For brevity we define $\mathbf{q}_i = (\mathbf{x}_i, \theta_i)$.

The geometric prior is defined so as to assign high probabilities to configurations $\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_N$ which are hand-like and low probabilities to configurations that are not. This is achieved by comparing a configuration to a fixed prototype shape $\tilde{\mathbf{q}}_1, \tilde{\mathbf{q}}_2, \cdots, \tilde{\mathbf{q}}_N$ (constructed manually from a representative photograph of a hand) and penalizing any deviation in shape between the two, irrespective of global rotation and translation. (The scale of the prototype is fixed and we assume knowledge of this scale when we execute our algorithm.)

Deviations in shape are measured by two kinds of shape similarities. First, the relative orientation of $\theta_i$ and $\theta_{i+1}$ should be similar to that of $\tilde{\theta}_i$ and $\tilde{\theta}_{i+1}$. Expressed in terms of probabilities, this similarity is written

$$P(\theta_{i+1}|\theta_i) = G(\theta_{i+1} - \theta_i - (\tilde{\theta}_{i+1} - \tilde{\theta}_i); \sigma_{a,i}) \tag{1}$$

where $G(x;\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}}$ and $\sigma_{a,i}$ is the standard deviation governing the angular relationship between stage $i$ and stage $i+1$ of the configuration.[1] This expression holds for all but three values of $i$, the ones corresponding to points $\mathbf{x}_{j_1}, \mathbf{x}_{j_2}, \mathbf{x}_{j_3}$ which precede the "hinges" between the index fingers. At these special values of $i = j_1, j_2 or j_3$, we define the conditional probability to be uniform between certain limits: $P(\theta_{i+1}|\theta_i) = 1/\delta_j$ if $\theta_{i+1} - \theta_i$ is between $\pi$ and $\pi + \delta_j$ and $P(\theta_{i+1}|\theta_i) = 0$ otherwise. Here the "hinge angle" between adjacent fingers is assumed to range uniformly from 0 (i.e. fingers together) to $\delta_j$ (maximum spread angle in radians).

Second, we expect the geometric relationship of $\mathbf{x}_i, \theta_i$ and $\mathbf{x}_{i+1}$ to be similar to that of $\tilde{\mathbf{x}}_i, \tilde{\theta}_i$ and $\tilde{\mathbf{x}}_{i+1}$. Specifically, the displacement vector $\Delta\tilde{\mathbf{x}}_i = \tilde{\mathbf{x}}_{i+1} - \tilde{\mathbf{x}}_i$ that connects $\tilde{\mathbf{x}}_i$ to $\tilde{\mathbf{x}}_{i+1}$ need only be rotated the proper amount to make a prediction $\Delta\mathbf{x}_i^p$ of the unknown displacement $\Delta\mathbf{x}_i = \mathbf{x}_{i+1} - \mathbf{x}_i$. As Figure 2 suggests, $\Delta\mathbf{x}_i^p = \mathbf{R}(\theta_i - \tilde{\theta}_i)\Delta\tilde{\mathbf{x}}_i$ where $\mathbf{R}(\theta)$ denotes counterclockwise rotation by $\theta$. In our model we allow $\mathbf{x}_{i+1}$ to deviate from $\mathbf{x}_{i+1}^p = \mathbf{x}_i + \Delta\mathbf{x}_i^p$ more freely in the normal direction perpendicular to $\Delta\mathbf{x}_i^p$ than along the tangent direction

[1] The argument of the Gaussian is to be evaluated modulo $2\pi$ since it is an angle. As a result, the standard normalization factor $\frac{1}{\sqrt{2\pi}\sigma}$ is only approximately correct.

parallel to $\Delta \mathbf{x}_i^p$. The rationale is that the length of $\Delta \mathbf{x}_i$ is determined by the scale of the template and should not change much across stages, whereas the direction of $\Delta \mathbf{x}_i$ varies as the contour bends. This may be expressed as follows:

$$P(\mathbf{x}_{i+1}|\mathbf{x}_i, \theta_i) = G([\Delta \mathbf{x}_i - \Delta \mathbf{x}_i^p] \cdot \Delta \hat{\mathbf{x}}_i^p; \sigma_{t_i}) G([\Delta \mathbf{x}_i - \Delta \mathbf{x}_i^p]^\perp \cdot \Delta \hat{\mathbf{x}}_i^p; \sigma_{n_i}) \tag{2}$$

where $\Delta \hat{\mathbf{x}}_i^p = \frac{\Delta \mathbf{x}_i^p}{|\Delta \mathbf{x}_i^p|}$ and $(x,y)^\perp = (-y,x)$ represents rotation of a vector by 90 degrees.[2] Here $\sigma_{t,i}$ and $\sigma_{n,i}$ are standard deviations along the tangent and normal directions, respectively ($\sigma_{t,i} < \sigma_{n,i}$). The dependence of these standard deviations, as well as the form of $P(\theta_{i+1}|\theta_i)$, on stage $i$ allows the prior to "bend" more easily at certain stages, such as the "hinge" between two fingers.

Combining these two elements of the shape prior yields

$$P(\mathbf{q}_{i+1}|\mathbf{q}_i) = P(\theta_{i+1}|\theta_i) P(\mathbf{x}_{i+1}|\mathbf{x}_i, \theta_i), \tag{3}$$

and the prior of the entire configuration is

$$P(\mathbf{q}_1 \cdots \mathbf{q}_N) = P(\mathbf{q}_1) \prod_{i=1}^{N-1} P(\mathbf{q}_{i+1}|\mathbf{q}_i) \tag{4}$$

where $P(\mathbf{q}_1)$ is a uniform constant. Notice that the prior is a Markov chain invariant under global translations and rotations, and the behavior of the prior is similar to the deformations that a piece of wire bent into the shape of a hand contour may assume. The values of the standard deviations were chosen experimentally by statistically sampling the prior, i.e. generating samples from the prior distribution to illustrate what shapes have high probability (see Figures (3,4 for examples). Learning techniques such as Minimax Entropy Learning [28] could be employed to determine more accurate values.

[2] Since $\mathbf{x}_{i+1}$ is quantized on a lattice, the normalization of the Gaussians will again differ from their standard form.

# 3 The Occlusion Process

An occlusion process which interacts with the imaging model (see next section) has been added to the prior to allow the deformable template to detect a hand even when part of the hand is occluded or obscured by noise. The occlusion process is a chain of occlusion state variables $h_1, h_2, \ldots, h_N$ associated with the configuration points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$. Each occlusion state variable $h_i$ may assume one of three values: 0, for non-occlusion; 1/2, denoting a triple point (i.e. corner or T-junction); and 1, for occlusion. These three occlusion states have distinct image properties (e.g. there is much less image information about the hand in an occluded region than in an unoccluded region), and using an occlusion process allows us to exploit these differences in the deformable template, rather than confound them.

We model the occlusion process as a Markov chain:

$$P(h_1 \cdots h_N) = P(h_1) \prod_{i=1}^{N-1} P(h_{i+1}|h_i) \tag{5}$$

The transition probabilities $P(h_{i+1}|h_i)$ are chosen to be independent of $i$: $P(h_{i+1} = b|h_i = a) = W_{ba}$ where $W = \begin{pmatrix} 1-\epsilon & \beta & 0 \\ \epsilon & 0 & \alpha \\ 0 & 1-\beta & 1-\alpha \end{pmatrix}$. The entry $W_{ba}$ represents the transition probability from state $a$ to state $b$, where the first, second and third rows represent states 0, 1/2 and 1 respectively (and similarly for columns). These transition probabilities are shown in Figure 5. $\epsilon$ is the probability that state 0 will jump to state 1/2 and is chosen small, corresponding to the fact that occlusions are relatively rare. State 1/2 acts as a bridge between the non-occlusion state 0 and occlusion state 1, representing the fact that the transition from a non-occlusion to an occlusion is usually signaled by a triple point, where the occluder and the object being occluded each produce edges at distinct orientations. (As the matrix indicates, our model forbids direct transitions between the 0 and 1 states.) $\beta$ is the transition probability from state 1/2 to 0, and $\alpha$ is the probability that state 1 will jump back to the triple point state 1/2. The starting probability $P(h_1)$ is set to $(1, 0, 0)$, i.e. the first point is assumed to be non-occluded. Finally, the occlusion process is independent of the geometric prior, so that the overall prior for the deformable template is given by:

$$P(\mathbf{s}_1, \mathbf{s}_2, \cdots \mathbf{s}_N) = P(\mathbf{q}_1 \cdots \mathbf{q}_N) P(h_1 \cdots h_N), \tag{6}$$

where the notation $\mathbf{s}_i$ is used to denote $(\mathbf{q}_i, h_i)$.

As we will describe in more detail in Section (4), we will have a specific cornerness measure $C(\mathbf{x})$ which is statistically associated with the triple points which often occur at occlusions. $C(\mathbf{x})$ measures the presence of a secondary edge at or near $\mathbf{x}$ with an orientation different from the primary edge at $\mathbf{x}$, as occurs at corners and so-called “T-junctions ” (see [20, 17, 15] for details). Figure 7 shows an example of an image and its cornerness map. The cornerness measure provides evidence of occlusions: at most occlusion points there is a T-junction formed where the occluder and the object being occluded produce edges at distinct orientations. We will see how to exploit this information in the next section.

# 4 The Imaging Model

A simple imaging model explains what image data may be expected given a specific configuration. Rather than explicitly modeling the grayscale intensities $I(\mathbf{x})$ at each pixel location $\mathbf{x}$ we model three sets of data derived from the Nitzberg corner/edge detector [20]. Our choice of this detector, and its parameters, was determined by statistical evaluation of its performance, see subsection (4.1). The output of the detector was the edge map $I_e(\mathbf{x})$, the cornerness map $C(\mathbf{x})$, and the image gradient orientation map $\theta_I(\mathbf{x})$. For simplicity, we will use the symbol $\mathbf{D}$ to represent the entire edge map, the cornerness map and the image gradient orientation map jointly, so that $\mathbf{D}(\mathbf{x}) = (I_e(\mathbf{x}), C(\mathbf{x}), \theta_I(\mathbf{x}))$.

Since $\mathbf{D}(\mathbf{x})$ depends heavily on the presence or absence of object boundaries in the image, our model accordingly divides the pixel lattice into three classes: the ”on” points that are on the hand boundary in the occlusion state 0, the ”triple” points that are on the template in the occlusion state 1/2, and the "off" points everywhere else (all points on the hand boundary in the occlusion state 1 and all points off the hand boundary). Each set of points is described by its own imaging model. The imaging model factors over all pixels in the lattice[3]:

$$P(\mathbf{D}|\mathbf{s_1}\cdots\mathbf{s_N}) = \prod_{\substack{\text{all pixels}\\ \mathbf{y}}} \mathbf{P}(\mathbf{D}(\mathbf{y})|\mathbf{s_1}\cdots\mathbf{s_N}) \tag{7}$$

where $P(\mathbf{D}(\mathbf{y})|\mathbf{s_1}\cdots\mathbf{s_N})$ is set to $P_{on}(\mathbf{D}(\mathbf{y})|\theta_\mathbf{i})$, $P_{triple}(\mathbf{D}(\mathbf{y}))$ or $P_{off}(\mathbf{D}(\mathbf{y}))$. $P_{on}$ is chosen if there exists an $i$ from 1 to $N$ such that $\mathbf{y} = \mathbf{x_i}$ and $h_i = 0$; $P_{triple}$ if there is an $i$ for which $\mathbf{y} = \mathbf{x_i}$ and $h_i = 1/2$; and $P_{off}$ if there is no $i$ for which $\mathbf{y} = \mathbf{x_i}$, or if there is an $i$ for which $\mathbf{y} = \mathbf{x_i}$ and $h_i = 1$. Note that the classification of points in an input image into "on", "triple" and "off" is specified by the configuration $\mathbf{s_1}\cdots\mathbf{s_N}$, which is determined by the dynamic programming algorithm described in Section 5.

We assume the three imaging models factor into separate probabilities on the edge map, the cornerness map and the image gradient orientation map:

$$P_{on}(\mathbf{D}(\mathbf{y})|\theta_i) = P^e_{on}(I_e(\mathbf{y}))P^c_{on}(C(\mathbf{y}))P^a_{on}(\theta_i - \theta_I(\mathbf{y})), \tag{8}$$

$$P_{triple}(\mathbf{D}(\mathbf{y})) = P^e_{triple}(I_e(\mathbf{y}))P^c_{triple}(C(\mathbf{y}))P^a_{triple}(\theta_i), \tag{9}$$

and

$$P_{off}(\mathbf{D}(\mathbf{y})) = P^e_{off}(I_e(\mathbf{y}))P^c_{off}(C(\mathbf{y}))P^a_{off}(\theta_i). \tag{10}$$

The probabilities $P^e_{on}, P^c_{on}, P^e_{triple}, P^c_{triple}, P^e_{off}$ and $P^c_{off}$ were measured using histogramming techniques on a dataset of images (on which hand edges and triple points were selected

[3] Since the value of $\mathbf{D}(\mathbf{x})$ at each point $\mathbf{x}$ is derived from image intensities $I(\mathbf{x}')$ in a neighborhood of $\mathbf{x}$, the independence of $\mathbf{D}(\mathbf{x})$ from point to point is only an approximation.

manually). $P^a_{on}$ was modeled as $P^a_{on}(\theta - \theta_I) = (1/2)[G(\theta_I - \theta; \sigma_O) + G(\theta_I + \pi - \theta; \sigma_O)]$, reflecting the fact that the orientation estimated from the image gradient usually points towards or 180 degrees against the true normal direction. Finally, $P^a_{triple}$ and $P^a_{off}$ were assumed constant (i.e. an assumption of isotropy).

There are a large variety of edge and corner detectors to choose from. Our choice of the Nitzberg detector was motivated by the following experimental results.

## 4.1 Evaluating Detectors using Chernoff Information

In this paper, we are concerned with discriminating between edges/corners and non-edges/non-corners in the image. The task of edge detection is to determine whether the value of a filtered image $\mathbf{I}_\phi(x)$ at a specific point $x$ is more likely caused by an edge or by a non-edge. From our perspective, this task can be formulated in terms of discriminating between two distributions [12] – the distribution of filter responses when an edge is present, $P_{on}(y) = Prob(\mathbf{I}_\phi(x) = y | x$ is on an edge) and the distribution of responses when there is no edge $P_{off}(y) = Prob(\mathbf{I}_\phi(x) = y | x$ is off an edge).

The existence of probability distributions $P_{on}, P_{off}$ which can reliably be determined from image data is crucial to the success of our approach. Ideally such distributions would change relatively little between images within similar domains. The justification for $P_{on}, P_{off}$ is based on our statistical analysis of edge detectors on two classes of images: (i) images of hands in indoor scenes (described in this paper), and (ii) images of birds in outdoor scenes (used as a consistency check).

We considered several types of edge detector operator. It is clearly desirable to use operators whose on-edge and off-edge statistics differ as much as possible. It has been argued [27] that a suitable measure of difference for such problems is the *Chernoff Information* which determines the error rates (i.e. rates of false positives and false negatives) in making the optimal statistical decision (using criteria motivated by the Neyman-Pearson lemma) (see

[7]). This is a good measure of difference *because it defines the least error rate possible for distinguishing between data from the two distributions using a log-likelihood test with optimal choice of threshold.* Moreover, Chernoff Information also appeared naturally in our analysis of fundamental bounds and expected time convergence rates of search algorithms described elsewhere [27].

The first edge detector we considered was the modulus of the image gradient – i.e. $\phi(I(x)) = \left|\vec{\nabla} I(x)\right|$. This was computed at a range of scales obtained by convolving the image with a Gaussian filter before taking the gradient.

It was then necessary to determine reliable on-edge and off-edge histogram statistics from our images. For off-edge we tried two approaches which gave similar answers. Firstly, we computed histograms over the entire image reasoning that the vast majority of pixels would be off-edge and so the edges would only weakly contaminate the histograms. Secondly, we randomly sampled off-edge pixels by clicking with a mouse. Both histograms were very similar. The effect of convolving the image with a Gaussian was, not suprisingly, merely to smooth out the histograms. The $P_{off}$ were then calculated by normalizing the histograms.

To determine the on-edge statistics we first had to locate reliable edges in the images. These edges should not be those most visually saliant because this would obviously compromise the statistics. Our technique was to select a long smooth edge of an object in the scene and fit this edge to a parameterized curve such as a spline. We could then gather statistics along the spline. The length, and smoothness, of the edge meant that we were getting a representative sample of edge strengths (i.e. the modulus gradients were high at some points on the curve but were low at others). In addition, to take into account errors in fitting the spline, we explored the edge strengths in neighbourhoods a few pixels away from the spline and perpendicular to it. Changing the neighbourhood size did not significantly alter our results. We then computed the histograms and calculated the $P_{on}$ as before.

The results of this process are shown in Figure (8). Empirically, the general shapes of the $P_{on}, P_{off}$ and their log likelihood ratios are similar from image to image in datasets of hand images, and similar results have been obtained on bird images.

We also repeated this process for other edge detectors and rated them based on the Chernoff information of their $P_{on}, P_{off}$. Of the operators we considered, the one with best Chernoff information was the edge detector devised by Nitzberg.

In addition, studies of statistics of natural outdoor scenes and underwater images have demonstrated qualitative similarity between the edge and non-edge statistics in these two different domains and have related them to the receptive field properties of the retinas of animals [2]. Finally, we observe that the work of Zhu and Mumford [29] demonstrated that the statistics of certain edge detection filters such as first and second derivatives was very similar in form between different images. Since most of the points in images are non-edges these statistics will be dominated by the off-edge terms and so provide additional evidence that the $P_{off}$ are similar between images. More extensive tests on large image databases also confirm the consistency of $P_{on}$ and $P_{off}$ [18].

# 5 Dynamic Programming With Pruning

Having constructed the prior $P(\mathbf{s_1} \cdots \mathbf{s_N})$ and imaging model $P(\mathbf{D}|\mathbf{s_1} \cdots \mathbf{s_N})$, Bayes theorem may be used to find the a posteriori probability: $P(\mathbf{s_1} \cdots \mathbf{s_N}|\mathbf{D}) = \frac{\mathbf{P(D|s_1 \cdots s_N)P(s_1 \cdots s_N)}}{\mathbf{P(D)}}$. Fortunately we do not need to undertake the difficult calculation of $P(\mathbf{D})$, since the MAP estimate is equivalent to

$$MAP = \mathop{argmax}_{\mathbf{s_1 \cdots s_N}} \{\prod_{i=1}^{N} \frac{P_i^{on}(\mathbf{s_i})}{P_{off}(\mathbf{s_i})}\} P(\mathbf{s_1} \cdots \mathbf{s_N}), \tag{11}$$

where $P_i^{on}(\mathbf{s_i})$ denotes $P_{on}(\mathbf{D}(\mathbf{x_i})|\theta_\mathbf{i})$ if $h_i = 0$, $P_{triple}(\mathbf{D}(\mathbf{x_i}))$ if $h_i = 1/2$, and $P_{off}(\mathbf{D}(\mathbf{x_i}))$ if $h_i = 1$. This result assumes all the members of the chain $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ to be distinct (i.e. non-self-intersecting hand contour).

Since the coupling of the variables $\mathbf{s}_i$ in the MAP expression is chain-like — $\mathbf{s}_i$ is directly coupled by the prior only to $\mathbf{s}_{i-1}$ and $\mathbf{s}_{i+1}$ — dynamic programming (DP) may be applied to find the MAP, as long as the $\mathbf{s}_i$ are quantized to finite sets. The occlusion process variables $h_i$ are discrete and the position variables $\mathbf{x}_i$ are naturally quantized on the image lattice, but it less clear how to quantize the real variables $\theta_i$. Rather than quantize the $\theta_i$ densely from 0 to $2\pi$ and incur an enormous multiplication in the number of nodes to search in each stage of DP, we experimented with various *adaptive quantization* techniques to select only a small number of $\theta_i$'s to consider for each $h_i$ and $\mathbf{x}_i$. Of the several variations we experimented with, the most successful one selects two $\theta_i$ candidates for each $h_i$ and $\mathbf{x}_i$. The first is the angle predicted by the prior at stage $i$ (i.e. $argmax(\theta_i)P(\theta_i|\theta_{i-1})$ where $\theta_{i-1}$ is the best predecessor of $\mathbf{s}_i$ chosen by DP); the second is either $\theta_I(\mathbf{x}_i)$ or $\theta_I(\mathbf{x}_i) + \pi$, whichever is closer to the $\theta_i$ predicted by the prior.

We review the DP procedure briefly and introduce some new notation. DP searches for the best path, i.e. best sequence of values $\mathbf{s_1}, \mathbf{s_2}, \cdots, \mathbf{s_N}$ (see Figure 9), in a stage-by-stage procedure. At each stage $i$ the best path to each $\mathbf{s_i}$ is determined on the basis of previous computations of the best paths to each $\mathbf{s_{i-1}}$; by best path we mean the one that maximizes the score $f_i(\mathbf{s_i}) = \{\prod_{\mathbf{j=1}}^{\mathbf{i}} \frac{\mathbf{P_j^{on}(s_j)}}{\mathbf{P_{off}(s_j)}}\}\mathbf{P}(\mathbf{s_1}, \mathbf{s_2}, \cdots, \mathbf{s_i})$. (Note that, although we are describing the procedure in terms of probabilities, we perform the calculations on the computer in terms of sums of log probabilities to avoid the numerical instabilities that occur when many probabilities are multiplied together.) In the final stage, the best $\mathbf{s_N}$ is determined and the other variables $\mathbf{s_{N-1}}, \mathbf{s_{N-2}}, \cdots, \mathbf{s_1}$ are successively recovered.

We use a linked-list data structure to represent the partial paths in the computer, allowing the computer to allocate memory dynamically at each stage to "grow" only those paths that have survived pruning. This technique saves a considerable amount of memory compared to standard dynamic programming implementations in which memory is allocated for every possible path at the start of the algorithm.

We also exploit the limited fan-out from one stage to the next: $P(\mathbf{s_{i+1}}|\mathbf{s_i})$ is negligible (or zero) for most pairs $\mathbf{s}_i$ and $\mathbf{s}_{i+1}$, and so the spatial component of the DP search can be narrowed. Given $\mathbf{s}_i$, only a fairly small region of spatial candidates $\mathbf{x}_{i+1}$ need be considered. This small region of pixels, whose size we denote $R$, is much smaller than the total number of lattice pixels $P$, reducing the computational complexity to $2 \times 3 \times PN \times 2 \times 2R = 24RPN$. We now turn to a discussion of two pruning methods used to speed up the DP algorithm.

## 5.1 Relative Pruning

The addition of a *relative pruning* technique, inspired by pruning techniques developed in speech processing [16] and natural language processing [13], reduces the running time to as little as 25 seconds (clocked on a Pentium 150 when the occlusion process is turned off). In this technique, all scores $f_i(\mathbf{s}_i)$ at stage $i$ are compared to the top score $f_i^*$ at this stage, and all $\mathbf{s}_i$ with scores $f_i(\mathbf{s}_i) < \epsilon_p f_i^*$ are eliminated from further consideration ($\epsilon_p = e^{-10}$). As we will show next, the scores are approximately proportional to partial path probabilities and so we are essentially removing all paths at each stage with probability sufficiently smaller than the most probable path.

If we make the approximation $P(\mathbf{s}_1 \ldots \mathbf{s}_N) \approx \prod_{i=1}^N P(\mathbf{s}_i)$, then we can calculate the probability of a partial path conditioned on the image data. From Bayes rule we have

$$P(\mathbf{s}_1 \ldots \mathbf{s}_m | \mathbf{D}) = \sum_{\mathbf{s}_{m+1} \ldots \mathbf{s}_N} P(\mathbf{D} | \mathbf{s}_1 \ldots \mathbf{s}_N) P(\mathbf{s}_1 \ldots \mathbf{s}_N) / P(\mathbf{D}) \tag{12}$$

and since $P(\mathbf{D})$ and $\prod_{\text{all pixels } \mathbf{y}} P_{off}(\mathbf{D}(\mathbf{y}))$ depend only on $\mathbf{D}$,

$$P(\mathbf{s}_1 \ldots \mathbf{s}_m | \mathbf{D}) \propto \sum_{\mathbf{s}_{m+1} \ldots \mathbf{s}_N} \prod_{i=1}^{N} \frac{P_i^{on}(\mathbf{s_i})}{P_{off}(\mathbf{s_i})} P(\mathbf{s}_i) \tag{13}$$

$$= \{\prod_{i=1}^{m} \frac{P_i^{on}(\mathbf{s_i})}{P_{off}(\mathbf{s_i})}\} \sum_{\mathbf{s}_{m+1} \ldots \mathbf{s}_N} \{ \prod_{i=m+1}^{N} \frac{P_i^{on}(\mathbf{s_i})}{P_{off}(\mathbf{s_i})} P(\mathbf{s}_i)\} \tag{14}$$

$$\propto P(\mathbf{s}_1 \ldots \mathbf{s}_m) \prod_{i=1}^{m} \frac{P_i^{on}(\mathbf{s_i})}{P_{off}(\mathbf{s_i})} \tag{15}$$

since $\sum_{\mathbf{s}_{m+1}\ldots\mathbf{s}_N}\{\prod_{i=m+1}^{N}\frac{P_i^{on}(\mathbf{s_i})}{P_{off}(\mathbf{s_i})}P(\mathbf{s}_i)\}$ is independent of $\mathbf{s}_1\ldots\mathbf{s}_m$.

## 5.2 Absolute Pruning

We devised another pruning technique, motivated by the probabilistic bounds on log likelihoods from [27], called *absolute pruning.* This technique prunes a partial path if the image data along it is deemed inconsistent with the partial path being on the true hand. More specifically, the $P_{on}$ and $P_{off}$ image data distributions are used to decide if the summed log likelihood image data along a partial path belongs to an edge along the true hand or not.

Defining

$$L = log(P_{on}(I_e)/P_{off}(I_e)) \tag{16}$$

as the log likelihood value from a single data sample (here we are ignoring the cornerness measure $C(\mathbf{x})$ and image gradient orientation $\theta_I(\mathbf{x})$ and omitting the superscript $P^e$ in $P^e_{on}$ and $P^e_{off}$), we can regard $L$ as a random variable which is distributed according to $\tilde{P}_{on}(L)$ if the sample belongs to an edge along the true hand or according to $\tilde{P}_{off}(L)$ if not. (Note that the edge strength distributions induce the log likelihood distributions $\tilde{P}_{on}(L)$ and $\tilde{P}_{off}(L)$.) Since $I_e$ was quantized to produce $P_{on}(I_e)$ and $P_{off}(I_e)$, $L$ can only assume a finite set of values and so $\tilde{P}_{on}(L)$ and $\tilde{P}_{off}(L)$ are sums of delta functions, depicted as weighted spikes in Figure 10.

If we sample $L$ repeatedly and independently $m$ times along a path $\mathbf{s}_1\ldots\mathbf{s}_m$ then the sum $L_m$ of the $L$'s is also a random variable. To calculate the distribution $\tilde{P}_{on}(L_m)$ we use the fact that the probability distribution of a sum of two independent random variables equals the probability distribution of one random variable convolved with the probability distribution of the other, i.e. $P_z(z = x + y) = (P_x \star P_y)\mid_{z=x+y}$. Thus $\tilde{P}_{on}(L_m)$ equals $\tilde{P}_{on}(L)$ convolved with itself $m$ times.

We can calculate $\tilde{P}_{on}(L_m)$ numerically by representing the function $\tilde{P}_{on}(L)$ as a discrete sequence of equally-spaced samples (all zeros except for a small subset of values representing the delta functions) and convolving the sequence with itself $m$ times to obtain $\tilde{P}_{on}(L_m)$. Similarly, we can calculate $\tilde{P}_{off}(L_m)$, which is shown side-by-side with $\tilde{P}_{on}(L_m)$ in Figure 11. Noting that the peaks in $\tilde{P}_{on}(L_m)$ and $\tilde{P}_{off}(L_m)$ have little overlap for large enough $m$, we can devise a method for deciding whether to reject (i.e. prune) or accept a partial path of $m$ segments: reject the partial path if $L_m < t_m$ and accept it otherwise. Here $t_m$ is defined such that

$$\tilde{P}_{on}(L_m < t_m) = \int_{-\infty}^{t_m} dL_m \tilde{P}_{on}(L_m) = \epsilon, \tag{17}$$

where $\epsilon$ is a small failure rate equal to the probability that the correct path has $L_m < t_m$.

We experimented with absolute pruning with the occlusion process switched off and found that it pruned paths successfully but much more conservatively than the relative pruning technique over a range of failure rates ($\epsilon = 10^{-4}, 10^{-3}, 10^{-2}$). The main reason for this conservative behavior lies in the limitation of the geometric prior, which should be included in the absolute pruning criterion but is too weak to prevent multiple DP paths from being geometrically distorted so as to cling to true edges. Thus many false paths have a disproportionate number of points along true edges, and their log likelihood evidence is comparable to that of the true path, making it difficult to prune these false paths. Even if the prior were strengthened, however, one limitation would remain: although the $\tilde{P}_{on}(L_m)$ and $\tilde{P}_{off}(L_m)$ distributions are quite distinct, many $L_m$ values reflect paths which are partially on *and* partially off the true hand, and it is unclear how to estimate the probability that the pruning rule fails to reject partially on/partially off paths.

# 6 Multiple Hands in a Single Image

We will set up a model for $M$ non-overlapping hands in a single image, and show that the previous model for a single hand can be adapted with slight modification to find the MAP multiple-hand estimate.

The prior for M independent, non-overlapping hands $(\mathbf{s}_1^\mu, \mathbf{s}_2^\mu, \dots \mathbf{s}_N^\mu)$, where $\mu$ labels different hands $\mu = 1, 2, \dots, M$, is:

$$P_M(\{\mathbf{s}_i^\mu\}) = \prod_{\mu=1}^{M} P(\mathbf{s}_1^\mu \dots \mathbf{s}_N^\mu) \tag{18}$$

where $P(\mathbf{s}_1^\mu \dots \mathbf{s}_N^\mu)$ is the single-hand prior from before. Similarly, the imaging model for $M$ independent, non-overlapping hands is modified only slightly from the single-hand imaging model:

$$P(\mathbf{D}|\{\mathbf{s}_i^\mu\}) = \{\prod_{\text{all pixels } \mathbf{y}} P_{off}(\mathbf{D}(\mathbf{y})\} \prod_{\mu=1}^{M} \prod_{i=1}^{N} \frac{p_i^{on}(\mathbf{s}_i)}{P_{off}(\mathbf{s}_i)}. \tag{19}$$

Therefore, assuming all $NM$ pixel locations $\{\mathbf{x}_i^\mu\}$ are distinct (i.e. no overlap between hands and no self-intersection within hands) we get the MAP multiple-hand estimate:

$$MAP = argmax_{\{\mathbf{s}_i^\mu\}} \prod_{\mu=1}^{M} [\{\prod_{i=1}^{N} \frac{P_i^{on}(\mathbf{s}_i)}{P_{off}(\mathbf{s}_i)}\} P(\mathbf{s}_1^\mu \dots \mathbf{s}_N^\mu)]. \tag{20}$$

Because the multiple-hand MAP estimate factors into $M$ independent pieces, the M hands may be recovered using the single-hand model as follows. The first hand is the MAP estimate according to the single-hand model, i.e. $argmax_{\{\mathbf{s}_i\}} P(\mathbf{s}_1 \dots \mathbf{s}_N|\mathbf{D})$; the second hand equals the MAP estimate of the single-hand model that doesn't intersect hand 1; the third hand equals the MAP estimate of the single-hand model that doesn't intersect hand 1 or 2; and so forth for all $M$ hands. In other words, the $M$ hands may be detected by finding the top $M$ non-overlapping DP paths. Note that, in the absence of pruning, finding the top $M$ hands takes no additional time compared to finding the top hand alone, since the complexity scales with the number of pixels in the image, not $M$. (Of course, if pruning is used then the complexity will increase somewhat with $M$.)

To implement this, we run our standard dynamic programming algorithm to recover the top-scoring path as the first hand. Then we find the next best-scoring path whose pixel locations come no closer than $d = 3$ pixel units of any of the pixels $\{\mathbf{x}_1^1, \mathbf{x}_2^1, \dots \mathbf{x}_N^1\}$. The third hand is the next best-scoring path after the second whose pixel locations $\{\mathbf{x}_1^2, \mathbf{x}_2^2, \ldots \mathbf{x}_N^2\}$ come no closer than $d$ pixels of any member of the set $\{\mathbf{x}_1^1, \mathbf{x}_2^1, \ldots \mathbf{x}_N^1, \mathbf{x}_1^2, \mathbf{x}_2^2, \ldots \mathbf{x}_N^2\}$, and similarly for more hands (see Figure 15).

# 7 Results

Two prototype shapes $\tilde{\mathbf{q}}_1, \tilde{\mathbf{q}}_2, \cdots, \tilde{\mathbf{q}}_N$ were used, one constructed with $N = 83$ (for speed) and the other with $N = 101$ (for reliability). $\sigma_{a,i}$ was set to 0.03 (radians), and at the hinge points we set $\delta_j = 0.5$. The values $\sigma_{t,i}$ and $\sigma_{n,i}$ were scaled in proportion to the scale of the prototype and will not be quoted here; their values were also increased at hinge points and the ratio $\sigma_{n,i}/\sigma_{t,i}$ was fixed at 2.3. The orientation map standard deviation $\sigma_O$ was set to 0.1. The prototype was rescaled manually to match the scale of each input image.

Our template has been tested on a variety of grayscale images, some of which are shown in Figure 12. To reduce computational complexity, all calculations were performed on a decimated lattice (decimated by a factor of 4 in both dimensions) derived from the edge, corner and orientation maps computed on the original image lattice. Figure 13 demonstrates the ability of the algorithm to cope with hinge deformation.

When the occlusion process was enabled, the algorithm was able to handle severe occlusions, as demonstrated in Figure 14. In these figures, the occlusion state at each point is designated as follows: a '+' means state 0, a '$\lambda$' means state 1/2, and a circle means state 1. We are able to detect multiple hands in a single image by finding the top few non-overlapping DP paths, as shown in Figure 15 (with a run time of about two minutes on a Pentium 150).

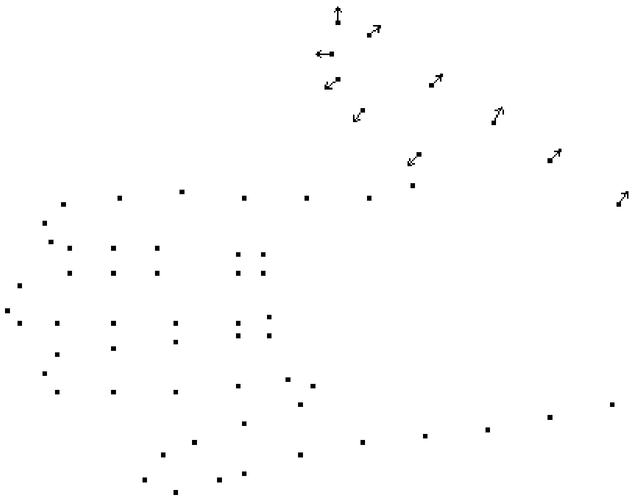

Figure 1: Sample configuration points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ with arrows showing normal orientations $\theta_1, \theta_2, \cdots, \theta_N$ on thumb.

# 8 Conclusion

We have described a promising algorithm for detecting flexible objects in real images. The algorithm can handle substantial hinge deformations, occlusions and the presence of multiple hands.

It is desirable to speed up the algorithm further and to deal with certain limitations of the model. In particular, the first-order Markov geometric prior seems to be the weakest

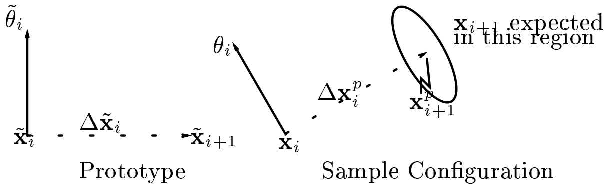

Figure 2: The prototype displacement $\Delta\tilde{\mathbf{x}}_i$ need only be rotated by $\theta_i - \tilde{\theta}_i$ to make a displacement prediction $\Delta\mathbf{x}_i^p$. $\mathbf{x}_{i+1}$ is expected in a neighborhood of $\mathbf{x}_{i+1}^p = \mathbf{x}_i + \Delta\mathbf{x}_i^p$.

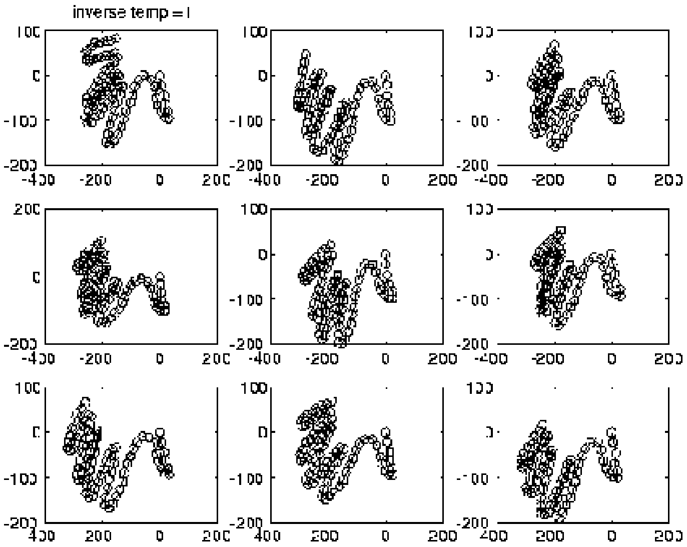

Figure 3: Samples of the hand prior with high standard deviations $\sigma_{a,i}$, $\sigma_{a,i}$ and $\sigma_{a,i}$. (For simplicity the hinges were removed). Observe that an excessive range of deformations is permitted: several samples have overlapping fingers.

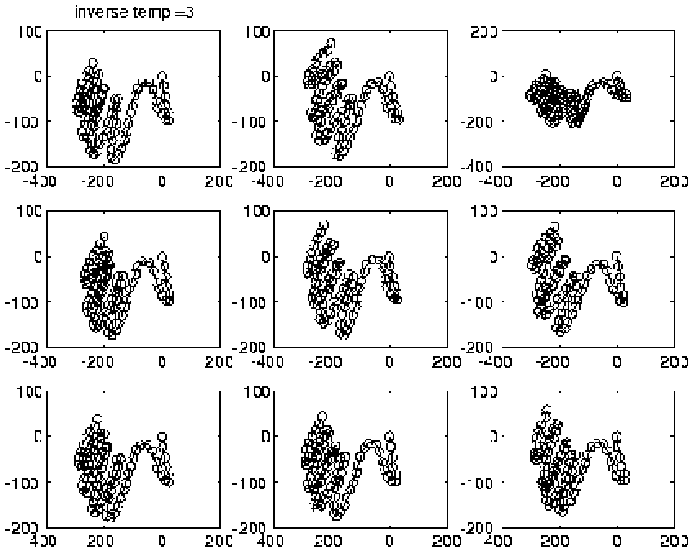

Figure 4: Samples of the hand prior with medum standard deviations $\sigma_{a,i}$, $\sigma_{a,i}$ and $\sigma_{a,i}$ (hinges removed). The samples are fairly realistic and show a good range of variability.

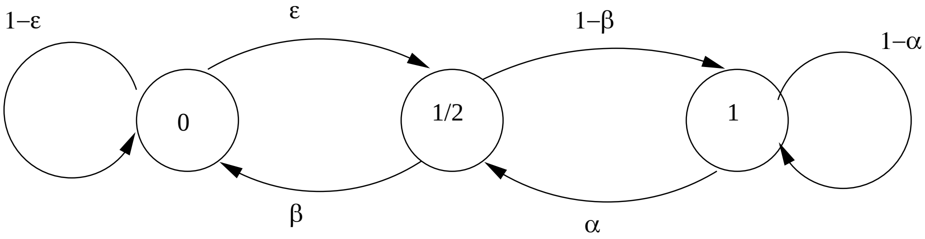

Figure 5: Transition probabilities among states 0, 1/2 and 1.

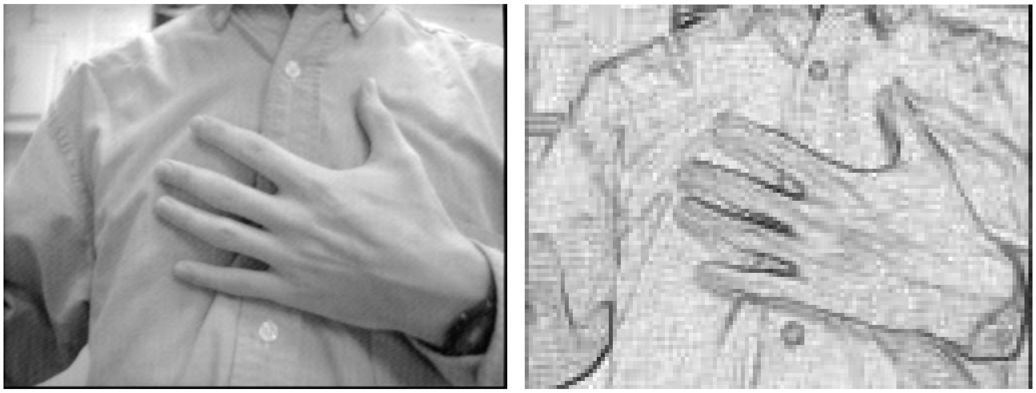

Figure 6: Original image on left, edge map on right.

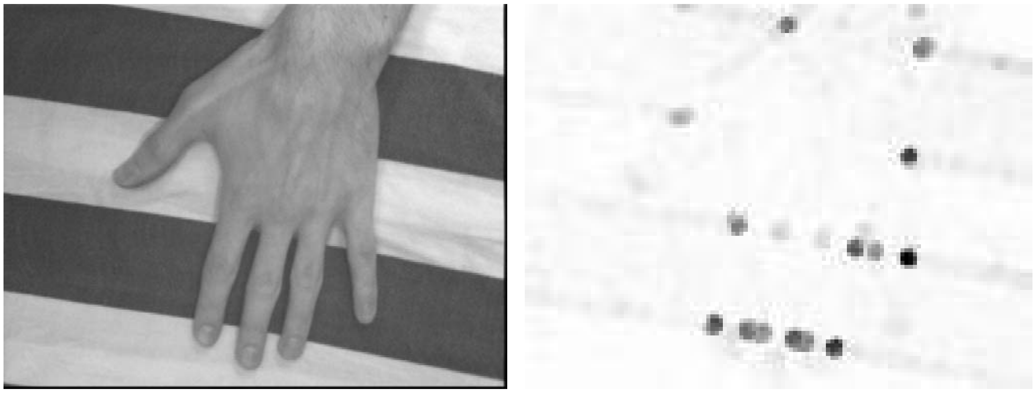

Figure 7: Original image on left, corner map on right.

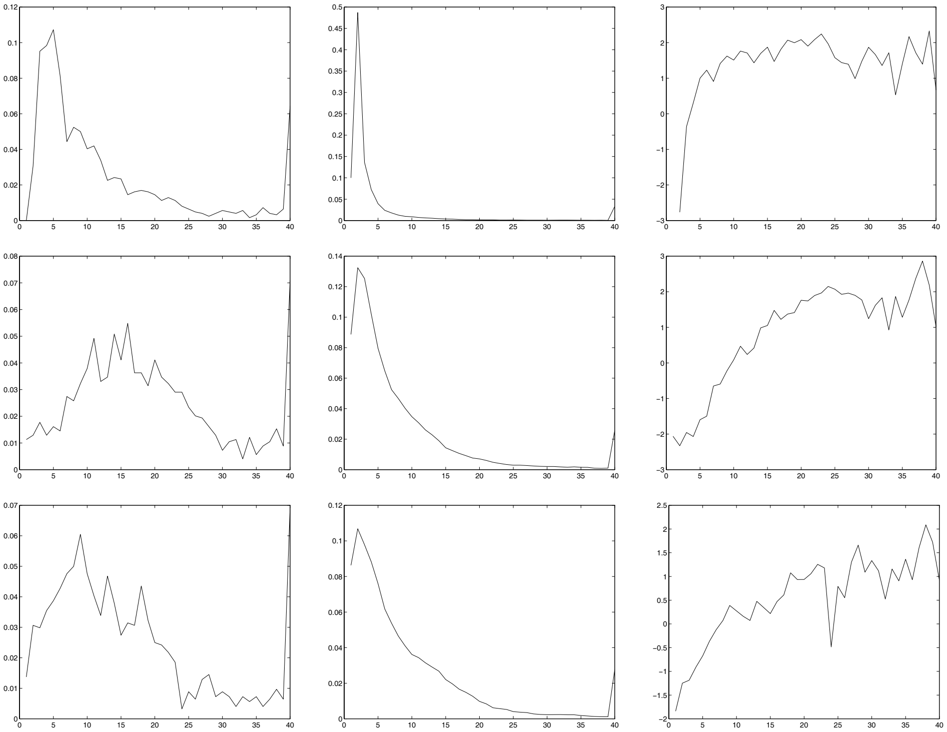

Figure 8: On-edge probabilities $P_{on-edge}$ (left), off-edge probabilities $P_{off-edge}$ (center) and log likelihood ratios $log(P_{on-edge}/P_{on-edge})$ (right). Each row represents results from a different edge filter. Top: Nitzberg filter, Chernoff information=0.3541. Middle: magnitude of the image gradient (after smoothing with a Gaussian of $\sigma = 2$), Chernoff information=0.2504. Bottom: magnitude of the image gradient (after smoothing with a Gaussian of $\sigma = 4$), Chernoff information=0.0877. The horizontal axis is the bin number (1-40).

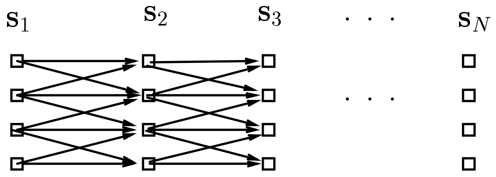

Figure 9: Schematic of DP algorithm showing limited fan-out from one stage to the next.

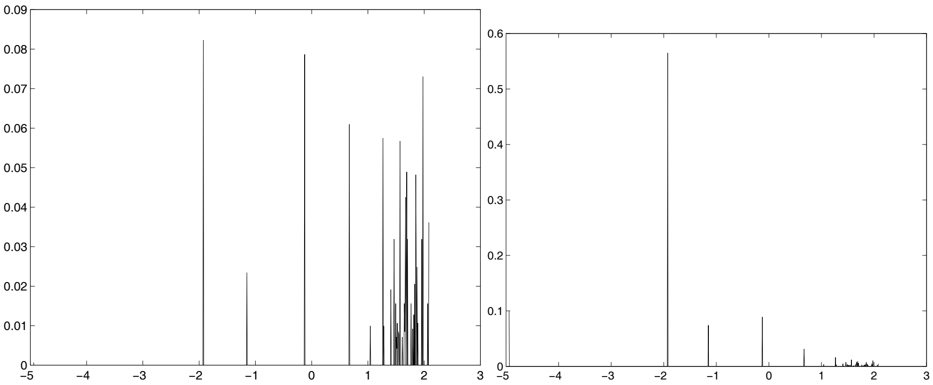

Figure 10: $\tilde{P}_{on}(L)$ on left and $\tilde{P}_{off}(L)$ on right.

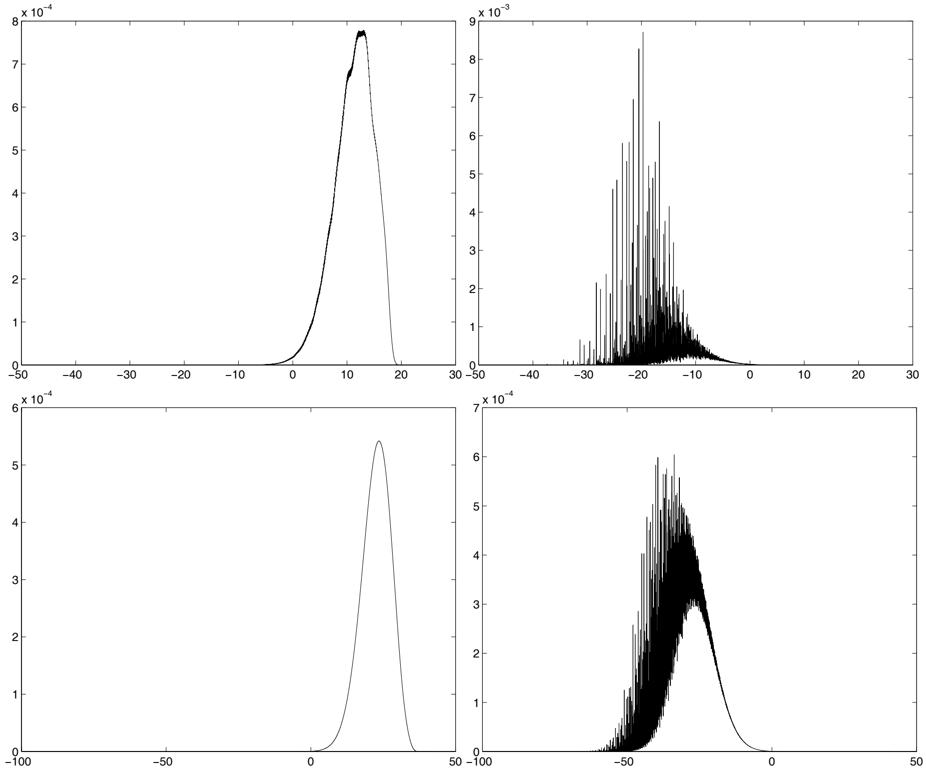

Figure 11: $\tilde{P}_{on}(L_{10})$ and $\tilde{P}_{off}(L_{10})$ on top row, $\tilde{P}_{on}(L_{20})$ and $\tilde{P}_{off}(L_{20})$ on bottom.

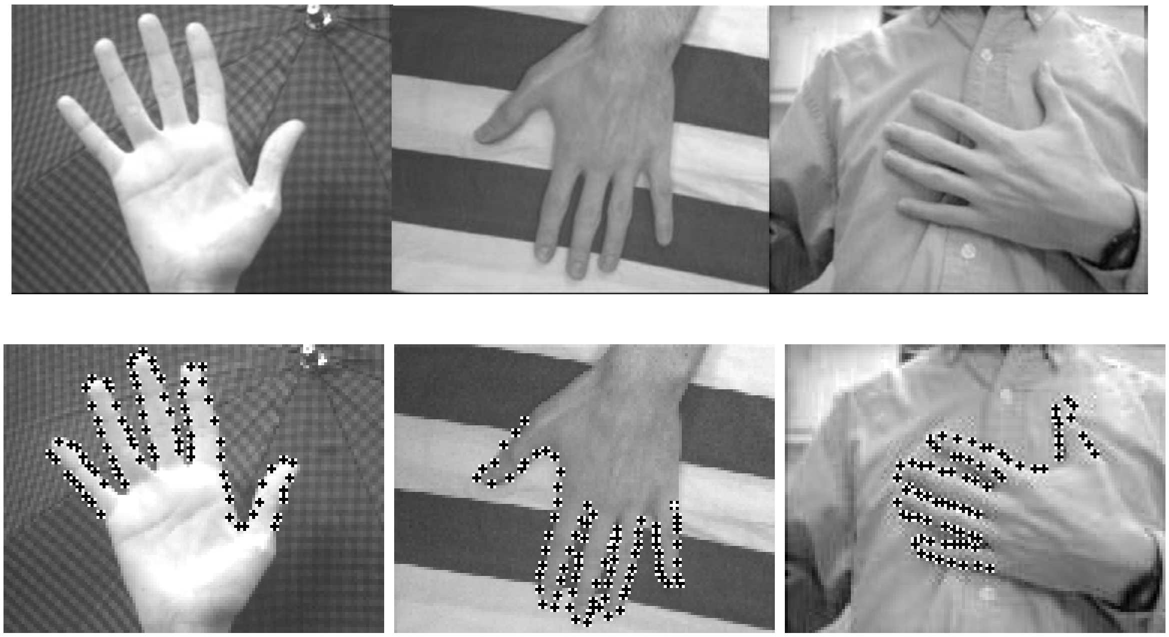

Figure 12: Inputs on top, outputs on bottom.

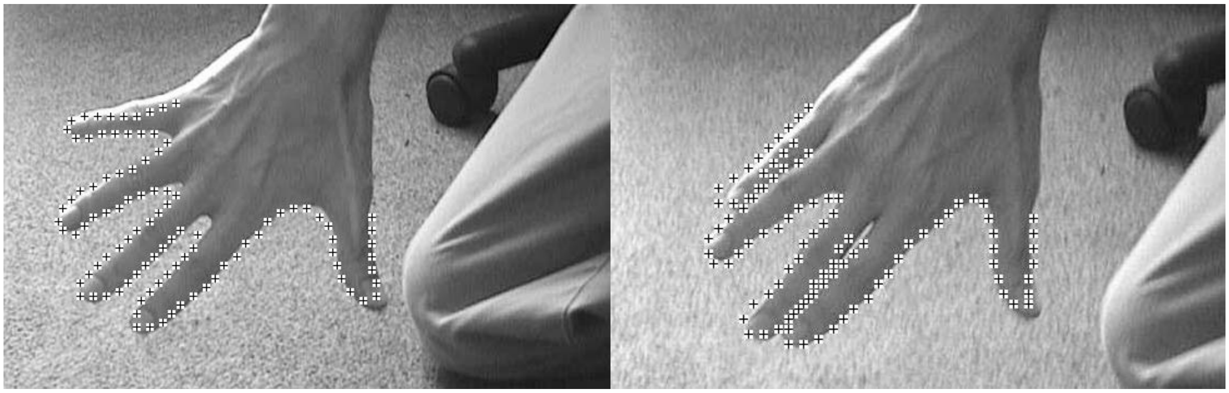

Figure 13: Sample hinge deformations.

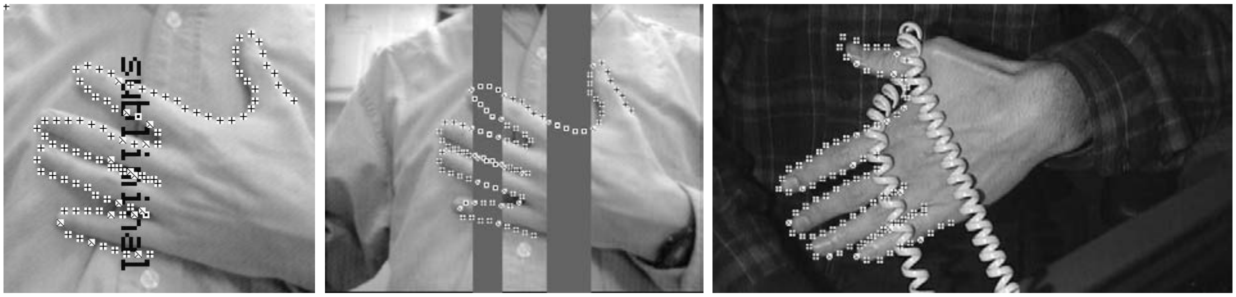

Figure 14: Examples of occlusions. In these figures, the occlusion state at each point is designated as follows: a '+' means state 0 (i.e. unoccluded), a '$\lambda$' means state 1/2 (triple point), and a circle means state 1 (occluded).

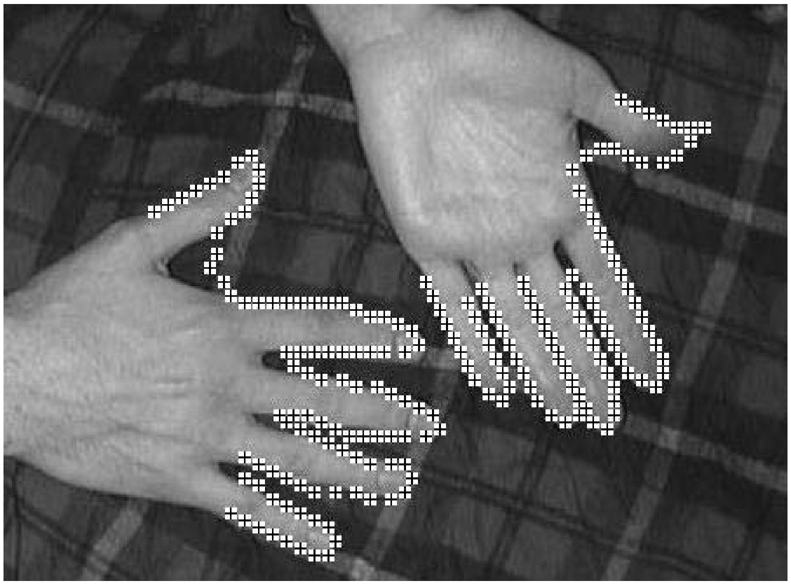

Figure 15: Two hands in one image.

component of the Bayesian model. The local interactions of the prior are insufficient to prevent such problems as self-intersection and unrealistic deformations. These local interactions mean that points that are spatially close together, e.g. two points near the same knuckle on opposite sides of a finger, may be separated by many points on the Markov chain and have no direct influence on each other. Also, the current approach is only able to deal with limited scale variations. We anticipate that more sophisticated representations involving symmetry axes [11] and key features will help solve both problems. Learning the prior systematically (and not merely relying on stochastic sampling experiments), as was done for the imaging model, will improve the reliability of the model. In addition, there are many other pruning techniques which we have not yet explored which offer more principled ways to speed up the algorithm [21, 26].

### Acknowledgements

We would like to thank Joshua Goodman, Richard Szeliski and Guy-Adrien Mounier for their useful comments and suggestions. Support was provided by NSF Grant IRI 92-23676 and the Center for Imaging Science Grant ARO DAAH049510494.

## References

[1] Y. Amit and A. Kong, “Graphical Templates for Image Matching,” Technical Report No. 373, Department of Statistics, University of Chicago, August 1993.

[2] R. Balboa. PhD Thesis. University of Allicante. Spain. 1997.

[3] M. Barzohar and D. B. Cooper, “Automatic Finding of Main Roads in Aerial Images by Using Geometric-Stochastic Models and Estimation,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 459-464, 1993.

[4] R. E. Bellman, *Applied Dynamic Programming,* Princeton University Press, 1962.

[5] T. F. Cootes and C. J. Taylor, “Active Shape Models - ’Smart Snakes’,” *British Machine Vision Conference,* pp. 266-275, Leeds, UK, September 1992.

[6] J. Coughlan. “Global Optimization of a Deformable Hand Template Using Dynamic Programming.” Harvard Robotics Laboratory. Technical report. 95-1. 1995.

[7] T.M. Cover and J.A. Thomas. **Elements of Information Theory**. Wiley Interscience Press. New York. 1991.

[8] M.A. Fischler and R.A. Erschlager. “The Representation and Matching of Pictorial Structures”. *IEEE. Trans. Computers.* C-22. 1973.

[9] D. Geiger, A. Gupta, L. A. Costa, and J. Vlontzos, “Dynamic Programming for Detecting, Tracking, and Matching Deformable Contours,” *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. 17, no. 3, March 1995.

[10] D. Geiger and T-L Liu. “Visual Deconstruction: Recognizing Articulated Objects”. In *Proceedings EMMCVPR’ 97.* Venice. Italy. 1997.

[11] D. Geiger, T-L Liu and A.L. Yuille. “Segmenting by seeking the symmetry axis.” GOAT, 1998.

[12] D. Geman. and B. Jedynak. “An active testing model for tracking roads in satellite images”. *IEEE Trans. Patt. Anal. and Machine Intel.* Vol. 18. No. 1, pp 1-14. January. 1996.

[13] J. Goodman, “Global Thresholding and Multiple-Pass Parsing,” In Proceedings of the Second Conference on Empirical Methods in Natural Language Processing, pages 11-25, Providence, RI, August, 1997.

[14] U. Grenander, Y. Chow and D. M. Keenan, *Hands: a Pattern Theoretic Study of Biological Shapes,* Springer-Verlag, 1991.

[15] C. G. Harris and M.J. Stephens. A Combined Corner and Edge Detector. In Proc. 3rd Alvey Vision Conference, 147-152. 1998

[16] F. Jelinek. **Statistical Methods for Speech Recognition**. MIT Press. Cambridge, MA. 1997.

[17] L. Kitchen and A. Rosenfeld. Gray-level corner detection, Technical Report TR-887, Computer Science Center, University Of Maryland, College Park, 1980.

[18] S. Konishi et al, in progress.

[19] U. Montanari, "On the Optimal Detection of Curves in Noisy Pictures," Commun. ACM, May 1972, pp. 335-345.

[20] M. Nitzberg, D. Mumford, and T. Shiota, *Filtering, Segmentation and Depth,* Springer-Verlag, 1993.

[21] J. Pearl, "Heuristics: Intelligent Search Stategies for Computer Problem Solving". Addison-Wesley Press. 1984.

[22] B. D. Ripley. "Classification and Clustering in Spatial and Image Data". In **Analyzing and Modeling Data and Knowledge**. Eds. M. Schader. Springer-Verlag. Berlin. 1992.

[23] L.H. Straib and J.S. Duncan. "Parametrically deformable contour models". *Proceedings of Computer Vision and Pattern Recognition*, pp 98-103. San Diego, CA. 1989.

[24] L. Wiscott and C. von der Marlsburg, "A Neural System for the Recognition of Partially Occluded Objects in Cluttered Scenes". Neural Computation. 7(4):935-948. 1993.

[25] A. L. Yuille, “Deformable Templates for Face Recognition”. *Journal of Cognitive Neuroscience.* Vol 3, Number 1. 1991.

[26] A.L. Yuille and J. Coughlan. “Twenty Questions, Focus of Attention, and A*: A theoretical comparison of optimization strategies.” In *Proceedings EMMCVPR’ 97*. Venice. Italy. 1997.

[27] A.L. Yuille and J. Coughlan. “Visual Search: Fundamental Bounds, Order Parameters, Phase Transitions and Convergence Rates.” 1998.

[28] S.C. Zhu, Y. Wu, and D. Mumford. “Minimax Entropy Principle and Its Application to Texture Modeling”. Neural Computation. To appear. 1997.

[29] S.C. Zhu and D. Mumford. “Prior Learning and Gibbs Reaction-Diffusion”. IEEE Trans. on PAMI vol. 19, no. 11. Nov. 1997.