Visual Deconstruction: Recognizing Articulated Objects

Tyng-Luh Liu and Davi Geiger

Courant Institute of Mathematical Sciences
251 Mercer Street
New York, NY 10012, USA

Abstract

We propose a deconstruction framework to recognize and find articulated objects. The deconstruction view decomposes the problem of finding a target object in an image into the one of (i) extracting key features in an image, (ii) detecting key points in the models, (iii) efficiently searching through possible image segmentations and (iv) comparing shapes. All of these subproblems can not be resolved independently. Together, they reconstruct the target object in the image.

In order to search for an optimal contour tracker, optimal shape comparison, optimal segmentation (combining feature extraction, contour tracking and shape comparison), we use and adapt the greedy and guaranteed Single Source Shortest Path Dijkstra algorithm.

1 Introduction

We investigate the problems of recognizing and finding articulated and deformable objects. In particular, we study human arm, leg articulations and objects like scissors, restricting the class of objects to the one that can be represented as contours. Still, it allows for a large variety of objects and applications. Our aims are for applications to aid video editing, database content retrieval, animation and medical imaging. In real world, articulated shapes are encountered almost everywhere. However, most of the (invariant-based) object recognition systems [17] do not have the capability to recognize articulated or deformable targets. Methods based on multiple views [21, 26, 27] can not account for large deformations or articulations. They lack a method that can realize the similarity between an object and its articulated/deformed counterparts.

The problem is formulated as follows: we give a model contour, Γ, and ask the system to find this model in an image I. The detected contour object, Γ^T, should be close to the given one in some similarity measure. We would like to derive a distribution of the form

$$P(\Gamma^T|\Gamma, I)\,, \tag{1}$$

and compute the optimal contour target $\Gamma^{T*} = arg\,max\, P(\Gamma^T|\Gamma, I)$.

Deconstruction approach: Our approach is a *Deconstruction* one. We break the problem of finding an object in the image that is most similar to the model into the one of (i) extracting key features in an image, (ii) detecting key points in the models, (iii) segmenting an image and (iv) comparing the shapes of the segmented image to the model one. These problems can be used to reconstruct the object in the image, and so the deconstruction is a process of breaking a model and an image then to reconstruct a solution (another "similar" object in the image).

We first briefly discuss (i) and (ii), how we extract model points (key points) and image features (key features) that are assumed to be in correspondence. This part of the work is needed to build the full recognition system, but we have not pursued this topic in depth. Rather, we have manually extracted model (key) points and used previous work in computer vision [22] to detect image (key) features.

Secondly, we focus on optimal search methods to detect snakes (active contours) and to compare contour shapes. The snake model was introduced in [12] and we follow a formulation based on dynamic programming [8]. In shape analysis we draw from the shape similarity work in [3], where a pseudo metric was established to measure articulations and some deformations. Our contribution on each problem, shape similarity and snake detection, is to identify a search method for the optimal solution that is greedy and guaranteed to find the global optimal. It is based on the Single Source Shortest Path (SSSP) Dijkstra's algorithm [1].

We finally integrate the efforts to deliver a global strategy that seeks a target contour model in the image. We require that model points match image features, preserving the model shape, and using the snake to delineate the solution (by grouping image edges). The search strategy is again greedy, but global, adapting the SSSP Dijkstra's algorithm. It works as follows: at each stage the top-down algorithm hypothesizes candidate matches between a pair of model points and a pair of image features. Then, feeds this information along with the contour model segment to the bottom-up (snake-shape) module. The snake-shape module seeks to link intensity edges (growing a snake) that best support the model. It feeds the result of this process to the top-down module to guide the search for a new match hypothesis. It can be formally thought as two levels of the Dijkstra algorithms. At the top level, the nodes of the graph represent hypothetical matches between model key points and image feature points. In this level the starting node (the first match) may be given as well as the final one. The values associated to the edges of the graph represent the optimal cost of the snake-shape module (see Figure-1(a)). To evaluate each edge cost a Dijkstra algorithm is used and this is the second level. In this case each node of the graph represent a lattice point. The first and final nodes (matches between image points and model points) are given by the hypothesis match (Figure-1(b)).

We present experimental results for recognizing articulated human shapes.

Previous work: Methods combining snakes with shapes have previously been discussed, e.g. [6]. Differently from their work, we model the deformations explicitly accounting for articulations. Moreover, we claim that our formulation can provide a foundation for object recognition. Interesting new ideas on recognition appeared in [2, 7, 23, 24, 26, 29] with related aspects to our approach. Previous work that describes objects by their parts can be found in [4, 14, 18]. The main ideas of representing object shapes via their reaction-diffusion deformations are in [13].

2 Key points and key features

We detect localized features in the image and represent them by a set $\{y_k : k = 1, ..., K\}$. They can represent corners, junctions, high curvature points, etc. Since our models are contour models, we use an ordered set of model features $[x_s : s = 1, ..., S]$ representing possibly high curvature points, or bending points.

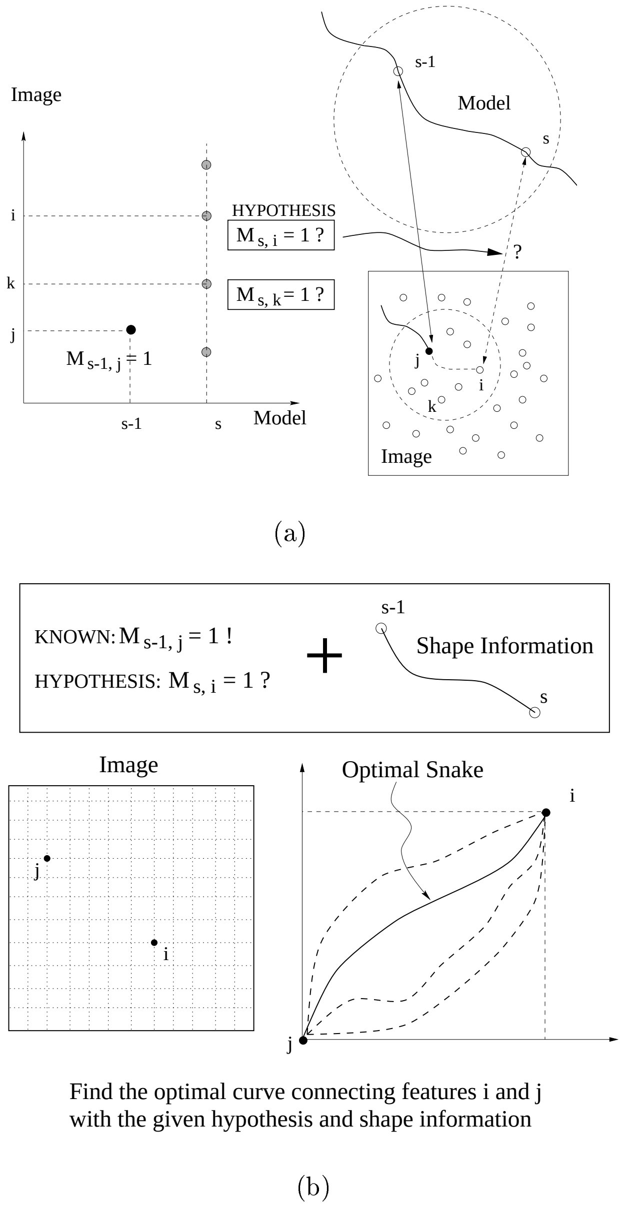


Figure 1: (a) The first level Dijkstra's algorithm has pulled off a possible solution from the priority queue with matching correspondence up to $M_{s-1,j}$, i.e., the $(s-1)^{th}$ key point on the shape model is matched to feature point j in the image. Then, to extend this possible solution, the top-level module will generate a set of hypotheses to feed the second (or bottom) level module to find the optimal contour. (b) Given a hypothesis, say $M_{s,i} = 1$, and the corresponding model shape information $\Gamma_{x_{s-1}}^{x_s}$, the snake-shape module will find the optimal contour.

2.1 Key model points

The key points of a model contour are points that convey a lot/most of information about the contour (see Figure-2(a)). These points connected by straight lines can give a good caricature of the contour, even when this contour undergoes deformations/articulations. All the points of a contour figure that are allowed to bend (articulation points) should be included.

Note that the key points of the model can, in a limit case, be all of the contour points. The distance between two points in the model, x_s and x_r are given by $d_S(x_s, x_r)$, or simply $d_S(s, r)$, and can be precomputed, following the contour model.

Ideally, we would like to derive a distribution for the key model points, $[x_s]$, given the contour Γ, i.e., $P([x_s]|\Gamma)$. Note that this distribution is independent of the image I. It is beyond the scope of this paper to derive/detect these points from an information-theoretic modeling. We will simply manually select them for each contour model. Moreover, we assume that the set of manually extracted model points $[x_s^{manual}]$ has no uncertainty, i.e. that the distribution is sharply peaked at $[x_s^{manual}]$. More precisely,

$$P([x_s]|\Gamma) \quad = \quad \prod_{s=1}^{S} \delta(x_s - x_s^{manual})\,, \tag{2}$$

where $\delta(x) = 1$ if $x = 0$ and $\delta(x) = 0$ otherwise.

2.2 Key image features

We would like to have access to the distribution of key image features that have arised from the imaging of the key model points, i.e., to know the distribution $P(\{y_k\}|[x_s], \Gamma, I)$. We would need a model of the geometric projection as well as illumination and reflectance. However, in our experiments we have neglected these issues and simply used the "SUSAN" feature detector [22] to generate a set of key features in the image. They are candidates for matching the model key points. These features usually capture corner points, junction points, and salient edges and we have found them useful for our experiments (see Figure-2(b)). We do require a reasonable amount of them to be capable of detecting the target. Roughly speaking, the feature resolution should be fine enough to sample a distribution that includes the one found by those model key features. We also make the assumption that the distribution over all possible features are sharply peaked at a subset of features from the set detected by SUSAN operator, say $\{y_k^{SUSAN}\}$.

Each x_s, model point, corresponds to at most one different y_k, feature. If occlusions are not considered, then each x_s always gives rise to a different y_k. Let us formalize these ideas.

We introduce binary matching variables $\{M_{s,k} \ : s = 1, ..., S : k = 1, ..., K\}$, so that

- $M_{s,k} = \begin{cases} 1, & \text{when } y_k \text{ is matched to } x_s, \\ 0, & \text{otherwise.} \end{cases}$
- $\sum_{s=1}^{S} M_{s,k} = 0$ or 1. 0 occurs when y_k is not generated by any model point contour.
- $\sum_{k=1}^{K} M_{s,k} = 1$.

In this paper we have not experimented with occlusions, where model points are not present in the image; otherwise, we would need to consider the case $\sum_{k=1}^{k} M_{s,k} = 0$. We are now equipped to derive the simplified distribution on the image features $\{y_k\}$. We propose

$$P(\{y_k\},\{M_{s,k}\}\,|\,[x_s],\Gamma,I) \;=\; \prod_{s=1}^{S}\sum_{k=1}^{K} M_{s,k}\delta(y_k - y_k^{SUSAN}(I))\,. \tag{3}$$

We still need to estimate $\{M_{s,k}\}$ to obtain the correspondence between key image features and key model points. This estimation process will be addressed when we integrate all the modules, section 4.

So far, our strong approximation assumes there is a subset of features detected in the image that exactly correspond to the model points. Note that, in the limit case where all image pixels are considered to be feature points, we are guaranteed to find such a subset. Of course, this case will lead to a very inefficient system. Ideally, better modeling of the process of generating the feature set $\{y_k\}$ from the model points $\{x_s\}$ should lead to a dramatic reduction on the computational time with increase on the quality of the results.

3 Snakes and Shape Similarity

Let us consider two feature points, y_j^{SUSAN} and y_i^{SUSAN} or simply y_j and y_i. We may hypothesize that they have arised due to the consecutive key model points x_{s-1} and x_s. We then consider a simple snake model connecting these two image feature points. The snake groups intensity gradients that are smooth to produce a single optimal contour between y_j and y_i. The shape similarity module compares the contour generated by the snake, $\Gamma_{y_j}^{y_i}$, with the associated one in the model, $\Gamma_{x_{s-1}}^{x_s}$. Let us now develop these two modules.

3.1 Snake

A snake model [12] is a deformable contour usually applied to detect contour boundaries (see Figure-2(c),(d) for results). Within a Bayesian rational, we need to construct the probability distribution of contours in images, $P(\Gamma^T)$, and the probability of generating an image given a contour, $P(I,|\Gamma^T)$, to obtain $P(\Gamma^T|I) \propto P(I|\Gamma^T)\,P(\Gamma^T)$. Assuming a uniform distribution over the set of images the proportionality becomes equality. When features $\{y_k\}$ and the matching units $\{M_{s,i}\}$ are available, we derive $P(\Gamma^T|I,\{y_k\},\{M_{s,i}\})$. We have considered the prior model of contours to be similar to the Elastica [16], i.e.,

$$P(\Gamma^T) \;=\; C_1 exp\Big\{-\int \left[\lambda_1|k(\Gamma^T(t)| + \lambda_2\right]\,dt\Big\}\,,$$

where the contour is parameterized by t, the parameters λ_1 and λ_2 have to be estimated, $k(\Gamma^T(t))$, or simply $k^T(t)$, are the contour curvature at coordinates $(x(t),y(t)) \in \Gamma^T$, and C_1, C_1' are normalization constants. This model differs from the Elastica model [16] in that we require the curvature term to be scale invariant.

For the image generation process, given the contour Γ^T, we have assumed that

$$P(I|\Gamma^T) \;=\; C_2 exp\Big\{-[A + \int \frac{1}{|\nabla I(\Gamma^T(t))|^2+\epsilon}\,dt]\Big\}\,,$$

where $|\nabla I(\Gamma^T(t))|$, or simply $|\nabla I(t)|$, is the image gradient at $(x(t),y(t)) \in \Gamma^T$. A is a constant reflecting a uniform distribution of intensity outside the contour boundary, and C_2 is a normalization constant. We then obtain the posterior distribution,

$$
\begin{aligned}
P(\Gamma^T|I) &= P(I|\Gamma^T)\,P(\Gamma^T) \\
&= C exp\Big\{-\int_0^T [\frac{1}{|\nabla I(t)|^2+\epsilon} + \lambda_1|k^T(t)| + \lambda_2]\,dt\Big\}
\end{aligned}
$$

where $C = C_1\,C_2\,e^{-A}$, T is the size of the contour, and we have used the prior assumption that $P(I)$ is uniform. When the image features, $\{y_k\}$, the matching between image features and model points, $\{M_{s,k}\}$, are available, we obtain the distribution

$$
\begin{aligned}
& P(\Gamma^T \mid I, \{M_{s,k}\}, \{y_k\}) \\
&= C\prod_{s=1}^{S}\prod_{i=1}^{K}\prod_{j=1}^{K} exp\left\{-M_{s,i}\,M_{s-1,j}\int_{\Gamma^T=y_j}^{y_i}\left[\tfrac{1}{|\nabla I(t)|^2+\epsilon} + \lambda_1|k^T(t)| + \lambda_2\right]\,dt\right\}. \quad (4)
\end{aligned}
$$

The matching units, $M_{s,k}$, help to guide the search through the ordering structure of the model contour, $s = 1, ..., S$.

Optimization and the search process: Montanari [15] has proposed a dynamic programming solution to this problem. On the experiments we have carry, where contour sizes are just a bit less than 300 pixels, the algorithm gave a speed up factor of about ten over similar contour-tracking algorithms using dynamic programming [15] [8].

3.2 Shape analysis

Our interest is to model $P^{shape}(\Gamma^T|\Gamma)$. In [3], it is argued that two contours are similar according to the amount of deformation, stretching and bending, to bring one into the other. It assumes that two shape contours are similar when their local components are similar. This assumption is particularly useful when modeling articulations, since an object undergoing articulations is untouched (similar) almost everywhere (see Figure-3(a)). Note that different correspondences can be established between two contours and a measure of similarity between them may depend on which correspondence is used, or on averaging over all possible correspondences (Bayesian view).

We use here the results from [3] where a measure of similarity satisfies the requirements for our work. For completeness, we first briefly review this work. We then point out in section B an information-theoretic derivation of a shape similarity measure, that is qualitatively similar to the one we have considered.

Formulation from previous work [3]: We can formalize this as follows: we parameterize the contours $\{\Gamma(s)\,;s = 1, ..., S\}$ and $\{\Gamma^T(t)\ : t = 1, ..., T\}$. Then, defining $t(s)$ to be the correspondence between the two contours, we have

$$
P^{shape}(\Gamma^T|\Gamma) = \sum_{\{t(s)\}} P(\{t(s)\}, \Gamma^T|\Gamma)\,.
$$

In order to test a distribution $P(\{t(s)\}, \Gamma^T|\Gamma)$, we can see the quality of the MAP estimate for a given Γ^T, i.e., the quality of

$$
\{t(s)^*\} = arg\max_{\{t(s)\}} P(\{t(s)\}, \Gamma^T_{given}|\Gamma)\,.
$$

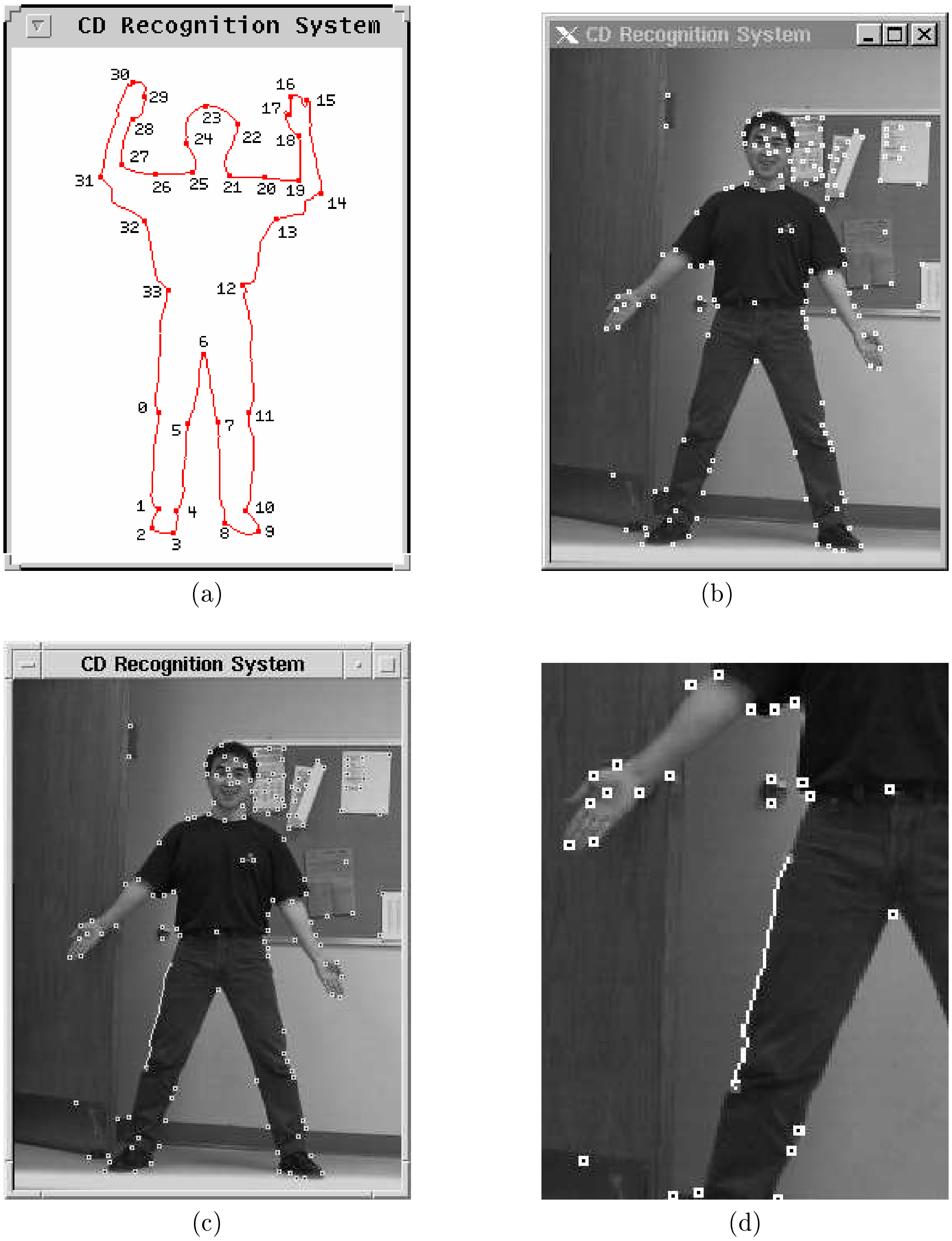
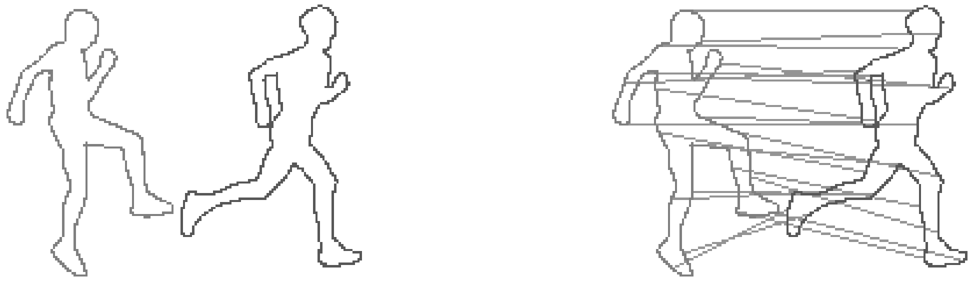


Figure 2: (a) Original model with its key points. (b) Original image with key features detected by "SUSAN" detector. (c) Snake result connecting two key features. (d) Same as (c), with a large scale of the snake region.



(a) (b)

Figure 3: (a) Two similar contours with articulations and stretchings. Note that locally both contours are similar almost everywhere. (b) Correspondences $\{t(s)\}$ derived from minimizing the spring model ($p = 2$) cost function.

Figure-3(b) illustrate one typical result to support the model we adopted. The spring model used is

$$P(\{t(s)\}, \Gamma^T|\Gamma) = exp\Big\{ - \int ds \frac{|(\theta^T)'(t(s)) - \theta'(s)|^p}{(|(\theta^T)'(t(s))| + |\theta(s)|)^{(p-1)}} + \lambda \frac{|t'(s) - 1|^p}{(t'(s) + 1)^{p-1}} \Big\} , \tag{5}$$

where $p > 1$ is constant (we use $p = 2$), $\theta'(s) = \frac{d\theta(s)}{ds} = k(s)$ and $(\theta^T)'(t(s)) = \frac{d\theta(t(s))}{ds} = k^T(t(s))t'(s)$, and $d\theta(s) = k(s)ds$ is the angle element accounting for the three points (x_{s-1}, x_s, x_{s+1}) (from the the circle described by these points).

Note that the first term accounts for bending, how much the matched angle elements are similar. This cost is tailored to deal with articulations in the sense that large curvature points are allowed to bend more with lesser increase in the cost (due to the division by the norm of the curvature). Notice that the cost is rotation and translation invariant, since it is solely based on angles and the measure for bending are scale invariant. The second term accounts for stretching, i.e., it prefers a uniform matching with $t'(s) = 1$. It is of the same form as the first term. The parameter λ weights the relative contributions of stretching and bending. The optimal solution $t^*(s)$ is illustrated for one example in figure 3.

Optimization and coarse approximation: In order to compute this distance we need to optimize for $t(s)$. This is possible using SSSP Dijkstra's algorithm. However efficient, Dijkstra's algorithm takes a few seconds to compute. This will cause the integration system, the deconstruction approach, to be very slow when calling the shape comparison many times. Moreover, we have not yet used the information contained in the features. The features are chosen such that the induced contour segments are smooth and yield $t'(s)$ uniform. We then devise a coarse and fast matching module, using the features, that has a shape comparison energy between the hypothesized matching segments $((\Gamma^S)_{s-1}^{s} \leftrightarrow (\Gamma^T)_{y_j}^{y_i})$ to be

$$E_{s,i,j}^{sh} = \frac{[\Delta\theta_S(s) - \Delta\theta_T(i)]^p}{[|\Delta\theta_S(s)| + |\Delta\theta_T(i)|]^{p-1}} + \lambda \frac{|t'^*(s,i,j) - 1|^p}{(t'^*(s,i,j) + 1)^{p-1}} , \tag{6}$$

where $\Delta\theta_S(s)$ and $\Delta\theta_T(i)$ are the angles, computed from the three points $(x_{s-1}, x_s x_{s+1})$ and $(y_{i-1}, y_i, y_i + 1))$, respectively. The stretching is computed by

$$t'^*(s,i,j) = \frac{d^S(x_s, x_{s-1})}{d^T(y_j, y_i)} ,$$

where $d^T(y_j, y_i)$ is a distance from connecting y_j to y_i, i.e., its computation depends on the choice of the snake $(\Gamma^T)_{y_j}^{y_i}$, while $d^S(x_s, x_{s-1})$ is a precomputed distance between x_s and x_{s-1} for the contour model. In this case the unknown matching is limited to the features and is represented by the unit variables $M_{s,i}$. The assumption above is that the matching between segments is assumed uniform.

Here we have had an abuse of notation: When referring to $t(s)$ and $t'(s)$, s represents any point in the contour Γ. While x_s gives the key model points, and so $s = 1, ..., K$ only describe the key model points. In this cases the correspondences are given by $\{M_{s,k} = 1\,;\, s = 1, ..., S\}$. So we have abused the notation by using s on both cases. However, one can distinguish both cases: (i) when we have an integral $\int(.)ds$ or $t(s)$, we mean the parameter s that accounts for all points in the contour and (ii) when we have $\prod_{s=1}^{K}(.)$ or $M_{s,k}$, we are just accounting for the key model points.

We can then rewrite the shape distribution in terms of the unknown correspondences $M_{s,k}$ as

$$\begin{aligned} P(\{t(s)\}, \Gamma^T \,|\, \Gamma) &= P(\{M_{s,i}\}, \Gamma^T \,|\, \Gamma, \{y_k\}, [x_s]) \\ &= \prod_{s=1}^{S} \prod_{i=1}^{K} \prod_{j=1}^{K} P_s(M_{s-1,j}, M_{s,i}, (\Gamma^T)_{y_j}^{y_i} \,|\, (\Gamma^S)_{s-1}^{s}) \\ &= \prod_{s=1}^{S} \prod_{i=1}^{K} \prod_{j=1}^{K} exp-[M_{s-1,j}\, M_{s,i}\, E_{s,i,j}^{sh}((\Gamma^S)_{s-1}^{s}, (\Gamma^T)_{y_j}^{y_i})] , \end{aligned} \tag{7}$$

4 Optimization: Top-down & Bottom-up

Let us first recall the distributions that we have derived on the previous sections so that we can "put them together".

(i) Manually detecting key model points, was described by (2), i.e.,

$$P([x_s]|\Gamma) \quad = \quad \prod_{s=1}^{S} \delta(x_s - x_s^{manual}(\Gamma)) .$$

(ii) For detecting image features, given key model points, we introduced the binary matching variables $M_{s,k} = 0, 1$, and described the distribution on image features by (3), i.e.,

$$P(\{y_k\}, \{M_{s,k}\} \,|\, [x_s], \Gamma, I) \quad = \quad \prod_{s=1}^{S} \sum_{k=1}^{K} M_{s,k} \delta(y_k - y_k^{SUSAN}(I))] .$$

(iii) From snakes we obtained (4), i.e.,

$$\begin{aligned} & P(\Gamma^T | I, \{M_{s,k}\}, \{y_k\}) \\ = C \prod_{s=1}^{S} \prod_{i=1}^{K} \prod_{j=1}^{K} & exp\left\{ -M_{s,i}\, M_{s-1,j} \int_{\Gamma^T = y_j}^{y_i} \left[\frac{1}{|\nabla I(t)|^2 + \epsilon} + \lambda_1 |k^T(t)| + \lambda_2 \right] dt \right\} . \end{aligned}$$

(iv) From shape comparison we obtained (7), i.e.,

$$
\begin{aligned}
& P(\{M_{s,i}\}, \Gamma^T | \Gamma, \{y_k\}, [x_s]) \\
&= \prod_{s=1}^{S} \prod_{i=1}^{K} \prod_{j=1}^{K} exp\Big\{ - M_{s-1,j}\, M_{s,i} \, [\tfrac{(\Delta\theta_S(s) - \Delta\theta_T(i))^p}{(|\Delta\theta_S(s)| + |\Delta\theta_T(i)|)^{p-1}} + \lambda \tfrac{|\frac{d^S(x_s, x_{s-1})}{d^T(y_j, y_i)} - 1|^p}{(\frac{d^S(x_s, x_{s-1})}{d^T(y_j, y_i)} + 1)^{p-1}}] \Big\} .
\end{aligned}
$$

4.1 Deconstructing

Our goal is to derive $P(\Gamma^T|\Gamma, I)$, and solve for the optimal Γ^T. Let us reconstruct $P(\Gamma^T|\Gamma, I)$ from (i)–(iv),

$$
\begin{aligned}
P(\Gamma^T|\Gamma, I) &= \sum_{\{M_{s,i}\},\{y_k\},[x_s]} P(\{M_{s,i}\}, \{y_k\}, [x_s], \Gamma^T \,|\, \Gamma, I) \\
&= \sum_{\{M_{s,i}\},\{y_k\},[x_s]} P(\Gamma^T \,|\, \{M_{s,i}\}, \{y_k\}, [x_s], \Gamma, I)\, P(\{M_{s,i}\}, \{y_k\}, [x_s] \,|\, \Gamma, I) \\
&= \sum_{\{M_{s,i}\},\{y_k\},[x_s]} P(\Gamma^T \,|\, \{M_{s,i}\}, \{y_k\}, [x_s], \Gamma, I)\, P(\{M_{s,i}\}, \{y_k\} \,|\, [x_s], \Gamma, I)\, P([x_s] \,|\, \Gamma, I) \\
&= \sum_{\{M_{s,i}\},\{y_k\},[x_s]} P^{snake}(\Gamma^T \,|\, \{M_{s,i}\}, \{y_k\}, I)\, P^{shape}(\Gamma^T \,|\, \{M_{s,i}\}, \{y_k\}, [x_s], \Gamma) \\
&\qquad P(\{M_{s,i}\}, \{y_k\} \,|\, [x_s], \Gamma, I)\, P([x_s] \,|\, \Gamma, I) \\
&= \sum_{\{M_{s,i}\}} P^{snake}(\Gamma^T \,|\, \{M_{s,i}\}, \{y_k^{susan}\}, I)\, P^{shape}(\Gamma^T, \{M_{s,i}\} \,|\, \{y_k^{susan}\}, [x_s^{manual}], \Gamma)\,,
\end{aligned}
$$

where we have fused the snake and shape models to give $P(\Gamma^T \,|\, \{M_{s,i}\}, \{y_k\}, [x_s], \Gamma, I)$. The sum over the features $\{y_k\}, [x_s]$ was easy, due to the peaked delta function distribution, and when combined with $P^{shape}(\Gamma^T \,|\, \{M_{s,i}\}, \{y_k\}, [x_s], \Gamma)$ gave $P^{shape}(\Gamma^T, \{M_{s,i}\} \,|\, \{y_k^{susan}\}, [x_s^{manual}], \Gamma)$.

We now assume the optimal snake Γ^{T*} to be obtained for the optimal $M^*_{s,k}$ (without summing over $\{M_{s,k}\}$). This leads to

$$
(\Gamma^{T*}, M^*) = arg \min_{\Gamma^T, M} \sum_{s=1}^{S} \sum_{i=1}^{K} \sum_{j=1}^{K} M_{s,i} M_{s,j-1} [E^{sn}_{i,j}((\Gamma^T)^{y_i}_{y_j}, I) + \ E^{sh}_{s,i,j}((\Gamma^T)^{y_i}_{y_j}, (\Gamma)^{x_s}_{x_{s-1}})] \, .
$$

We then approximate this solution by

$$
\begin{aligned}
M^* &= arg \min_{M} \sum_{s=1}^{S} \sum_{i=1}^{K} \sum_{j=1}^{K} [E^{sn}((\Gamma^{T*})^{y_i}_{y_j}) + E^{sh}((\Gamma^{T*})^{y_i}_{y_j})] \\
(\Gamma^{T*})^{y_i}_{y_j} &= arg \min_{(\Gamma^T)^{y_i}_{y_j}} E^{sn}_{i,j}((\Gamma^T)^{y_i}_{y_j}, I) \qquad \text{if } M^*_{s,i} = M^*_{s-1,j} = 1 \, .
\end{aligned} \tag{8}
$$

It assumes that we can obtain the snake between features independently of using the shape comparison module. In general it is not a valid assumption, but if the features are good representatives giving smooth contours in between them, then simply running the snakes between the features will provide a desired segmentation candidate (see Figure-1).

Solving for $(\Gamma^{T*})_{y_j}^{y_i}$: At the bottom-up level (see Figure-1 (b)), we can solve for $(\Gamma^{T*})_{y_j}^{y_i}$ using dynamic programming [8], and since the energy costs are all positive we can replace it with Dijkstra algorithm with a gain in efficiency of a factor 4 to 10 (see Figure-2(c)(d) for results).

Solving for $\{M_{s,k}^*\}$: At the top-down level we consider the Dijkstra algorithm to minimize for the set of matchings, i.e., to determine M^*. The advantage of using Dijkstra at the top-down level is that, due to its greedy characteristic, not all possible matches are considered. Indeed most matches will not be considered. The Dijkstra search at the top-down level will command which matching hypothesis to consider given its current state and evaluation. The bottom-up Dijkstra algorithm will simply compute the cost of each hypothesis. This is in a unique way the interaction between top-down and bottom-up algorithms (see Figure-1 (a)).

Rational: We now give an intuitive account, the rational, of the top-down & bottom-up processes. Following that we derive this rational from the optimization method, Dijkstra algorithm, to compute the solution.

From the top-down view of the problem, we need to select the optimal choice of M, i.e., the optimal matching correspondence between model key points and image key features. The algorithm must not search for all possible configurations of M to be efficient. The guidance of the top-down model is to decide which pairs, say $M_{s-1,j} = 1$ and $M_{s,i} = 1$, to consider, without searching for all possible ones.

The top-down module hypothesizes a pair of correspondences, say $M_{s-1,j} = 1$ and $M_{s,i} = 1$, and feeds the model contour Γ_{s-1}^s to the bottom-up module.

The bottom-up process test the hypothesis that (x_{s-1}, x_s) corresponds to (y_j, y_i), i.e., $M_{s-1,j}^* = 1$ and $M_{s,i}^* = 1$, taking the contour part Γ_{s-1}^s as input. There is freedom to select different contours $(\Gamma^T)_{y_j}^{y_i}$ connecting y_j to y_i. The bottom-up process returns the cost induced by the best contour choice, $(\Gamma^{T*})_{y_j}^{y_i}$. This cost takes into account the snake cost and the shape comparison cost with Γ_{s-1}^s.To simplify and approximate the solution, the bottom-up approach groups the edges between y_i and y_j, growing a snake and optimizing for E^{sn}, regardless the shape analysis measure E^{sh}. The bottom-up algorithm then computes E^{sh}, to deliver the total cost, $E^{sn} + E^{sh}$, for this pair of hypothesis.

The top-down will then decide which other pair of correspondences to hypothesize for its search.

An illustration of the control loop is given in Figure-4.

More precisely, the solution of the problem is to obtain a maximum a posteriori probability estimate (MAP) of

5 Experimental Results

In Figure-5 and 6, we show the experimental results of the snake-shape contour recognition method. Two kinds of model contours, a human shape and scissors contours (see Figure-5(a) and 6(a)), are used for carrying out the experiments. The test images with features detected by SUSAN operator are also shown in Figure-5 6. Note that all target objects in the test images are similar to the respective model contours except articulations and slightly change in scales. Satisfactory results are derived as seen in Figure-5 (c),(e) and Figure-6 (c),(e). In all tests, the target contour with articulations in each image is recovered successfully though there is still room for improvement such as in Figure-6 (e), the right hand is not picked up very well.

Currently, our methods have been implemented in two different versions: one for the SGI machines and the other for PC with Linux. On a Pentium 150, the algorithm takes about 5 minutes

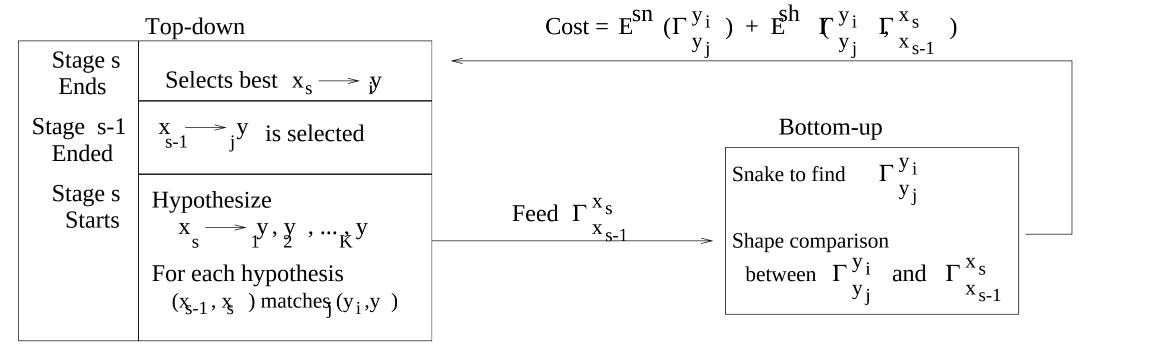


Figure 4: Control loop of the top-down and bottom-up algorithm. The stage $s-1$ has been resolved with x_{s-1} matching y_j, i.e., $M^*_{s-1,j} = 1$. Then the algorithm finds the best y_i such that $M^*_{s,j} = 1$. It also finds the contour connecting y_j and y_i, i.e., $(\Gamma^{T*})^{y_i}_{y_j}$.

to complete (including displaying the evolution of detected contours on the window interface of our software system).

6 Extensions and Conclusions

We have introduced a new method to recognize and detect shape contours for articulated and deformed objects. The basis of our approach is a carefully Bayesian-Information formulation and manipulation of the problem variables. We have relied on the use of Dijkstra algorithm as the main engine of computations. Combining shape and snakes is part of the system and we have shed light on combining top-down & bottom-up approaches to recognition in a unique way. The global optimization is approximately completed by optimization with "recognition by parts".

A limitation of our experiments and formulation is the use of contour models, 1 dimensional graphs, allowing the application of Dijkstra algorithms. As one moves to 3 dimensions, the use of surfaces would destroy the efficiency of Dijkstra algorithm. Another limitation of the approach is the non-use of symmetries along the contours. A natural way to overcome both limitations is to consider the axis of symmetry of an object. If we represent objects by their axes of symmetry, we naturally include the symmetries and represent 3D objects by 1D graphs (the axis of symmetry). The approach can then be easily extendible to 3D objects and account for symmetry properties.

Acknowledgments

The authors would like to thank Stuart Geman and Alan Yuille for interesting discussions. We thank the AFOSR grant #:F49620-96-1-0519 for sponsoring this research.

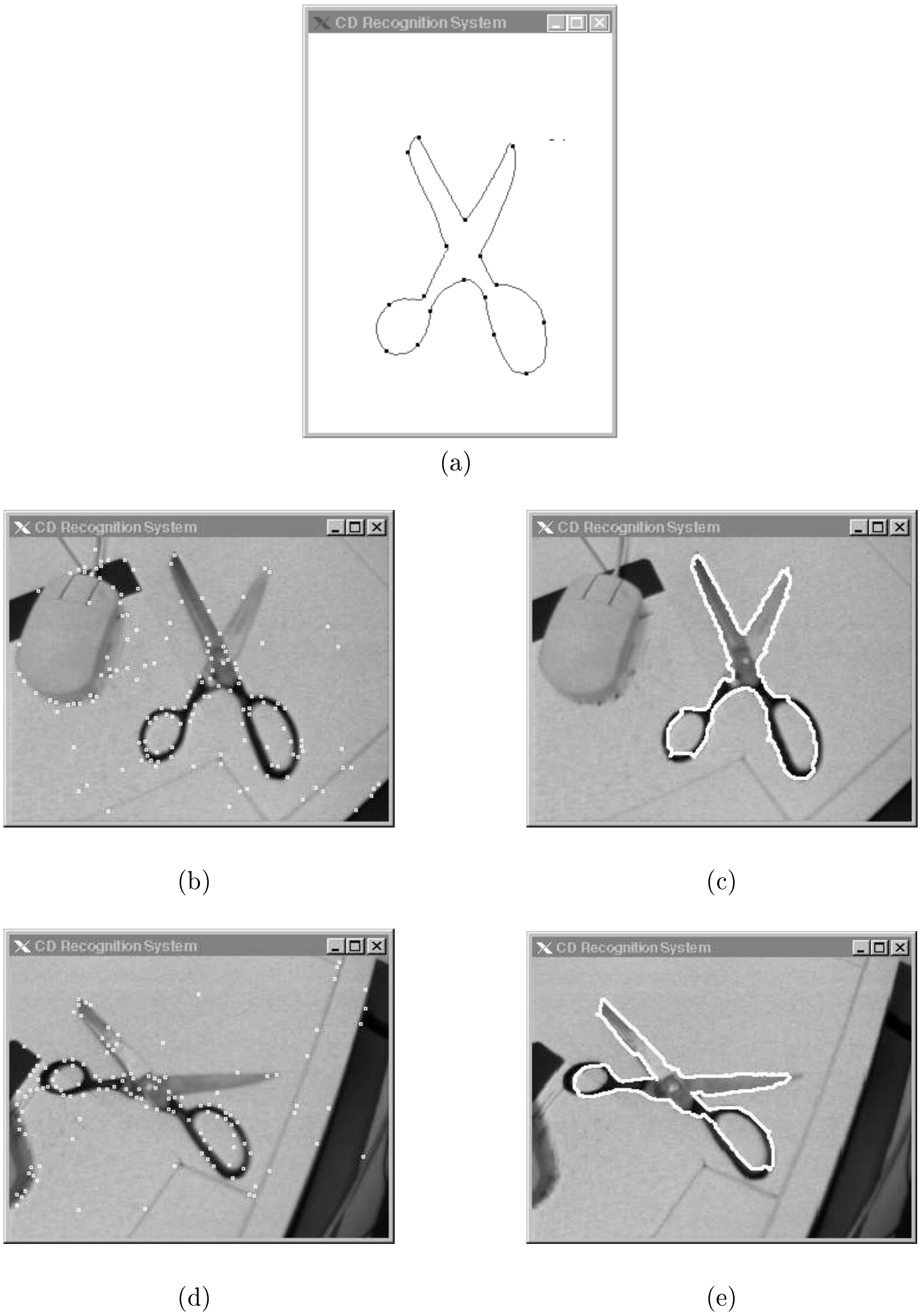
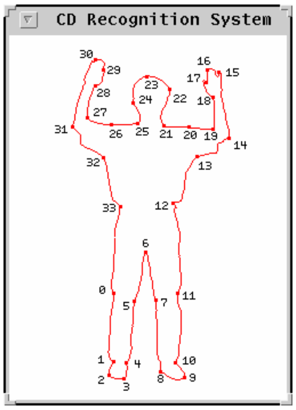
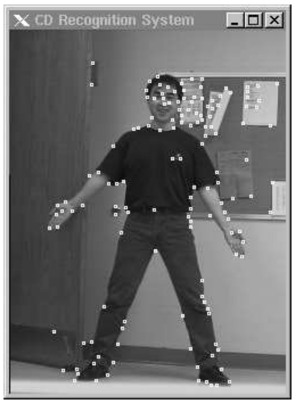


Figure 5: (a) The model contour of a pair of scissors. (b) and (d) are test images with detected features. (c) and (e) are the detected target contours.



(a)



(b)

(c)

(d)

(e)

Figure 6: (a) The model contour of a human shape. (b) and (d) are test images with detected features. (c) and (e) The detected target contours.

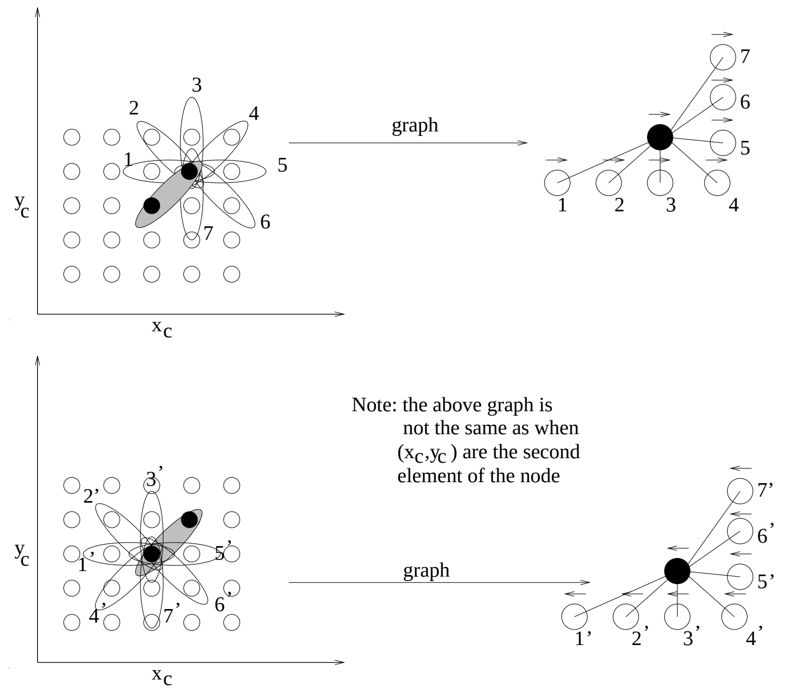


Figure 7: (a.) A pair of neighbor pixels yields two "oriented" nodes on the graph, and each node has seven (7) neighbor nodes. Note that node a arising from pixel (x_c, y_c) followed by its next neighbor $(x_c + 1, y_c + 1)$ has a different set of seven neighbors then node b arising from pixel $(x_c + 1, y_c + 1)$ followed by (x_c, y_c).

A The Snake algorithm

We use a greedy method, but guaranteed, based on the SSSP Dijkstra's algorithm. Moreover, because we are using second derivatives (the curvature term), we have to devise a special data structure, graph, to apply the Dijkstra algorithm. We will leave a detailed analysis of the algorithm to the appendix A.

In our proposed graph, each node represents two neighbor lattice pixels. The edges must connect different nodes with a common pixel. Given a pair of connected nodes, the curvature is immediately calculated (three consecutive pixels define a circle with the inverse radius given the curvature). Dijkstra's SSSP algorithm computes the shortest paths from pixel j to every coordinate reached by a directed graph with positive weights. The important constraint for the Dijkstra algorithm to work is that all edge costs must be positive which is valid in our case. For the sake of completeness, we reproduce the algorithm here as used in the system.

We start with a nodes $s = s_{j,i} = (y_j, y_i)$ where $i \in N_j$ with N_j representing the set of the eight neighbors to the lattice site y_j. The choice of the first neighbor, y_i, does not use the curvature term and so it is based on the gradient and length information, i.e., it is ad-hoc. One can also think this

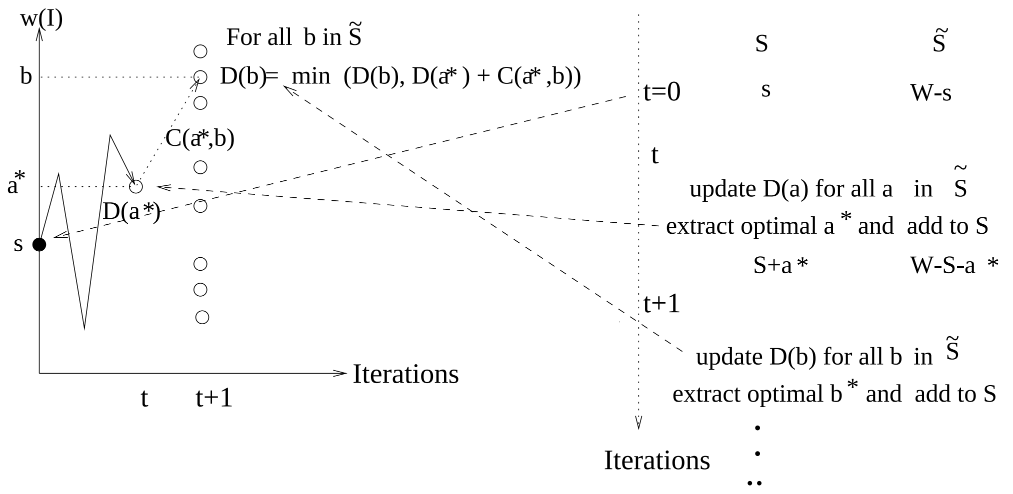


Figure 8: The schematic diagram for the SSSP algorithm. We have an image window w and start with node s, inserting it into an empty set S. $C[a, b]$ represents the transition cost, or the edge cost from node a to node b. If the second lattice site of node a is not the same as the first lattice site of node b the transition is not possible and $C[a, b] = \infty$. $D[a]$ represents the optimal cost to reach node a starting in node s and is updated at each iteration. We initialize all nodes costs to $D[a] = C[s = 1, a]$. $\bar{S}$ is the complement to S in the sense that $S + \bar{S} = Nodes(w)$. Assume at stage t, a^* is the optimal node in the path and we insert it into S and, consequently, remove it from $\bar{S}$. At stage $t + 1$, , for each node $b \in \bar{S}$, redefine $D[b] = min(D[b], D[a^*] + C[a^*, b])$. Clear is that one just needs to update the nodes b that are neighbors to a^*, since $C[a^*, b] = \infty$ otherwise. Create an array $P[b]$ that gives the best predecessor to node b, in this case $P[b] = a$ when $D[a] + C[a, b] < D[b]$. Then, choose the optimal node $b^* \in \bar{S}$, i.e., $b^* = arg \min_b D[b]$ and move it to the set S. Move to stage $t + 2$ ...

starting node as a boundary condition. So, we start with node $s_{j,i}$, or simply s, and collect it into a set S, i.e. $S = \{s\}$. Let us call $C[a, b]$ to be the transition cost, or the edge cost from node a to node b. One should think node a as a pair of neighbor lattice sites (7). In the case that curvature/second derivative is not used, a can simply be the lattice site. We say $C[a, b] = \infty$ when the transition is not possible, i.e., when there is no common pixel between node a and node b. For each node a there are seven neighbors (see figure 7). Let us also call $D[a]$ to be the cost to reach node a when starting from node $s_{j,i}$, and eventually will be the best cost. We initialize all nodes costs to $D[a] = C[s = 1, a]$. Let us create a set $\bar{S}$ to be the complement to S such that $S + \bar{S} = Nodes(w)$, i.e., all the nodes in the window w. Note that a window with $n \times n$ pixels yields about $56n^2$ valid edges for $8n^2$ nodes, since each pixel generates 8 nodes and each node has 7 neighbor nodes. Note that despite each node having two pixels, they are not counted twice. This is because a path going from one pixel to the next (one node) have seven (7) neighbors pixels (seven neighbor nodes) that are different than if the path had been going the other way (see figure 8a.). Thus, this representation of nodes is very much directional oriented. The algorithm consists of looping for all possible nodes and choosing a such that $D[a]$ is minimum and including a to S and, consequently, removing from $\bar{S}$. Moreover, at this stage, for each b in $\bar{S}$, redefine $D[b] = min(D[b], D[a] + C[a, b])$. Create, also, an array $P[b]$ that gives the best predecessor to node b, in this case $P[b] = a$ when $D[a] + C[a, b] < D[b]$. This algorithm is guaranteed to give the optimal path to the cost function we have proposed. The complexity of this algorithm is $O(wlogw)$.

B Shape similarity via the mutual information

We can derive a shape similarity criteria from a information-theoretic view. Let us consider each contour, Γ and Γ^T, to represent a set of samples, $s = 1, ..., S$, from a joint distribution $P(k^T(t(s)), k(s), \Delta s, \Delta t(s) \,|\, t(s))$ given the correspondence $t(s)$. Here $\Delta s, \Delta t(s)$ are segment distances along the contours. Let us define this distribution to be

$$P_s(k^T(t(s)), k(s), \Delta s, \Delta t(s) \,|\, t(s)) =$$

$$C\, e^{-\left[\left(k^T(t(s))t'(s)-k(s)\right)^2+\left(k^T(t(s))t'(s)\right)^2+\left(k(s)\right)^2+\lambda\left((\Delta s-\Delta t(s))^2+(\Delta s)^2+(\Delta t(s))^2\right)\right]} . \tag{9}$$

A rational for choosing this distribution is two fold (i) it gives the intuitive idea of requiring the joint distribution to encourage similar angles and small curvatures and small segments; (ii) it gives the Elastica model of contours as the marginal distribution. More precisely,

$$\begin{aligned} P_s(k^T(t(s)), \Delta t(s)|t(s)) &= \int_{-\infty}^{\infty} d\,k(s)\; d\,\Delta(s)\; P_s(k^T(t(s)), k(s), \Delta s, \Delta t(s)|t(s)) \\ &= C'\, e^{-\frac{1}{2}[(k^T(t(s))t'(s))^2+\lambda\Delta^2 t(s)]} . \end{aligned} \tag{10}$$

We can now compute the Mutual Information between the random variables $Y(t(s)) = (k^T(t(s)), \Delta t(s))$ and $X(s) = (k(s), \Delta s)$, given $t(s)$. Representing the integrals by sums, i.e., $d\,k(s)\; d\,\Delta(s) \to \sum_{X(s)}$, we write the mutual information as

$$\begin{aligned} I_{t(s)}(X(t(s)), Y(s)) &= \sum_{Y(t(s))} \sum_{X(s)} P_s(Y(t(s)), X(s) \,|\, t(s)) log[\frac{P_s(Y(t(s)), X(s) \,|\, t(s))}{P_s(X(s) \,|\, t(s))\, P_s(Y(t(s)) \,|\, t(s))}] \\ &= \sum_{k^T(t(s))} \sum_{\Delta t(s)} \sum_{k(s)} \sum_{\Delta s} P_s(k^T(t(s)), \Delta t(s), k(s), \Delta s \,|\, t(s)) \\ &\quad [\left(k^T(t(s))t'(s) - k(s)\right)^2 - \frac{1}{2}\left(k^T(t(s))t'(s)\right)^2 - \frac{1}{2}k^2(s) \\ &\quad +\lambda\left((\Delta s - \Delta t(s))^2 - \frac{1}{2}(\Delta s)^2 - \frac{1}{2}(\Delta t(s))^2\right) + 2logC' - logC] \\ &\approx 2logC' - logC - \frac{1}{S}\sum_{s=1}^{S} P_s(k^T(t(s)), k(s), \Delta s, \Delta t(s) \,|\, t(s)) \\ &\quad [\left(k^T(t(s))t'(s) - k(s)\right)^2 - \frac{1}{2}\left(k^T(t(s))t'(s)\right)^2 - \frac{1}{2}k^2(s) \\ &\quad +\lambda\left((\Delta s - \Delta t(s))^2 - \frac{1}{2}(\Delta s)^2 - \frac{1}{2}(\Delta t(s))^2\right)] . \end{aligned} \tag{11}$$

The last approximation uses the stationarity of the distribution and assumes that the number of samples (contour size) is large enough to be considered a statistical average. We can then maximize the mutual information with respect to $t(s)$ to find the optimal correspondence. This is to say that no further assumptions about $t(s)$ are known. It is clear that the above criteria is not identical to the cost function we have used, defined through the distribution in (7). However, it is similar in that the optimal correspondence $t^*(s)$ is penalized in the following ways: (i) Both have a bending error criteria due to the differences in curvatures (in angles); (ii) Both have the bending error diminished for larger curvatures (angles); (iii) Both have stretching costs that are analogous to the bending one.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, "The Design and Analysis of Computer Algorithms", Addison-Wesley, 1974.

[2] Y. Amit, D. Geman, and K. Wilder, "Recognizing Shapes from Simple Queries about Geometry," Technical Report, University of Chicago, 1995.

[3] B. Basri, L. Costa, D. Geiger, and D. Jacobs, "Determine Shape Similarity," *IEEE workshop in Physics Based Vision,* Boston, June 1995.

[4] I. Biederman, "Human Image Understanding: Recent Research and a Theory," *Computer Graphics, Vision, and Image Processing,* (32):29-73, 1985.

[5] M. C. Burl, P. Perona, "Recognition of Planar Object Classes," *IEEE Conf. on Computer Vision and Pattern Recognition,* pp. 223-230, San Francisco, June 1996.

[6] T. Cootes, C. Taylor, D. Cooper and J. Graham, "Active Shape Models - Their Training and Application," *Computer Vision and Image Understanding.,* Vol. 61, No 1, pp. 38-59, January. 1995.

[7] D. Forsyth, J. Malik, M. Fleck, Leung, Bregler, Carson, Greenspan, "Finding objects by grouping," *Proc. of Workshop on Object Representation, ECCV, England, 1996.*

[8] D. Geiger, A. Gupta, L. A. Costa, and J. Vlontzos, "Detecting, Tracking and Matching Elastic Contours with Dynamic Programming," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-17, March 1995.

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

[10] D. Huttenlocher, G. Klanerman, and W. Rucklidge, 1993, "Comparing Images Using the Hausdorff Distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **15**(9):850-863.

[11] D. Huttenlocher, J. Noh, and W. Rucklidge, 1993, "Tracking Non-Rigid Objects in Complex Scenes," *4th Int. Conf. on Computer Vision*:93–101.

[12] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models," *Proceedings of the First International Conf. on Computer Vision,* pp. 259-268, 1987.

[13] B. Kimia, A. Tannenbaum, and S. W. Zucker, "Shapes, Shocks, and Deformations I: The Components of Two-Dimensional Shape and the Reaction Diffusion Space" *Int. Journal of Computer Vision,* 15, 189-224, 1995.

[14] K. Kupeev and H. Wolfson, "On Shape Similarity," *Proceedings Int. Conf. on Pattern Recognition,* pp. 227-237, 1994.

[15] U. Montanari, "On the Optimal Detection of Curves in Noisy Pictures," *Communications of the ACM,* pp. 335-345, 1971.

[16] D. Mumford. Elastica and computer vision. In C. L. Bajaj, editor, *Algebraic Geometry and Its Applications.* Springer-Verlag, New York, 1993.

[17] J. L. Mundy and A. Zisserman (editors), "Geometric Invariance in Computer Vision," MIT Press, 1992.

[18] A. Pentland, "Recognition by Parts," *Proceedings of the First International Conf. on Computer Vision,* pp. 612-620, 1987.

[19] A. Pentland and B. Horowitz, "Recovery of Non-rigid Motion and Structure," *IEEE Trans. Pattern Analysis and Machine Intelligence,* **13**(7), 730-742, 1991.

[20] A. Pentland and S. Sclaroff, " Closed-form Solutions for Physically Based Shape Modeling and Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* **13**(7), 715-729, 1991.

[21] T. Poggio and D. Beymer, "Learning Networks for Face Analysis and Synthesis," *Proceedings of the International Workshop on Automatic Face- and Gesture-Recognition,* Zurich, Switzerland, June 26-28, 1995.

[22] S. Smith and J. Brady, "SUSAN - a New Approach to Low Level Image Processing," *Int. Journal of Computer Vision,* 1996, In publication.

[23] K. Siddiqi and B. Kimia, "Parts of Visual Form: Computational Aspects," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-17, March 1995.

[24] K. Siddiqi and B. Kimia, "Toward a Shock Grammar for Recognition," *IEEE Conf. on Computer Vision and Pattern Recognition,* 1996.

[25] M. Turk and A. Pentland, "Face Recognition Using Eigenfaces," *IEEE Conf. on Computer Vision and Pattern Recognition,* pp. 586-591, 1991.

[26] S. Ullman, "High-level vision", MIT Press, Cambridge, 1996.

[27] S. Ullman and R. Basri, "Recognition by Linear Combinations of Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-13 (10), pp. 992-1006, 1991.

[28] A. L. Yuille and P. W. Hallinan, "Deformable Templates for Recognition and Tracking," *Active Vision,* editors A. Blake and A. L. Yuille, MIT Press, Cambridge, 1992.

[29] S. C. Zhu and A.L. Yuille, "A Flexible Object Recognition and Modeling System, " *Proceedings International Conference on Computer Vision*, 1995. To appear in IJCV, 1996.