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. In particular we are interested in human arm and leg articulations.
The deconstruction view of recognition naturally decomposes the problem of finding an object in an image, 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 shapes. All of these subproblems can not be resolved independently. Together, they reconstruct the object in the image. We briefly address (i) and (ii) to focus on solving together *shape similarity* and *segmentation*, combining top-down & bottom-up algorithms.
We show that the *visual deconstruction* approach is derived as an optimization for a Bayesian-Information theory, and that the whole process is naturally generated by the guaranteed Dijkstra optimization algorithm.

1 Introduction

We investigate the problems of recognizing and finding articulated and deformable objects. In particular we study human arm and leg articulations, restricting the class of objects to the one that can be represented as contours. Our approach is extendible to three dimensional objects, which we discuss, but our main focus is on two dimensional contours. Still, it allows for a large variety of objects and applications. Our aims are 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 [12] do not have the capability to recognize articulated or deformable targets. Methods based on multiple views [16, 21, 22] are interesting but, 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.

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, (iv) comparing shapes. These problems can be used to reconstruct the object in the image, and so the deconstruction is this process of breaking a model and an image to reconstruct a solution (another "similar" object in the image).

To illustrate our approach we briefly discuss (i) and (ii) then focus on the integration of two methods, snakes (active contours) and shape comparison. The ability of judging the degree of similarity between two shapes (contours) provides a global shape property not encountered on previous snake model. The snake model was introduced in [9] and we follow a formulation based on dynamic programming [7]. In shape analysis we draw from the shape similarity work in [2], where a pseudo metric was established to measure articulations and some deformations.

From (i) and (ii) we extract model points and features in images that are believed to have a correspondence. The top-down method guides the search for reconstructing the object in the image. It hypothesizes candidate matches between a pair of model points and a pair of image features. Then, feeds the bottom-up module with the object part that connects the pair of model points. The bottom-up method extracts edges, then groups edges to link the hypothesized pair of image features (growing a snake), and finally compares the snakes with the object parts given by the top-down module. It feeds the result of this process to the top-down module to guide the search for a new match hypothesis. We show that these methods are derived as an optimization for a Bayesian-Information theory, and that the whole process is naturally generated by the guaranteed Dijkstra optimization algorithm.

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, our formulation can provide a foundation for object recognition. Interesting new ideas on recognition appeared in [1, 5, 18, 19, 21, 24] with related aspects to our approach.

To reduce the computation complexity of our method, we adopt an "integration by parts" technique to approximately solve the global optimization problem. Our rational is that the shape similarity measure [2] prefers deformations that preserve the part structures of objects over deformations that modify the parts. Previous work that describes objects by their parts can be found in [3, 10, 13]. We present experimental results for recognizing articulated human shapes.

2 Deconstructing Recognition

Given an image I and model Γ, we first establish the difference and similarity between the processes of recognizing and of finding Γ in I.

Definition 1: Recognizing an object Γ in the image is equivalent to responding "yes" to the question " Is an object model present in the image ?" , i.e.,

$$P(\Gamma|I) \; > \; 0.5 \quad ?$$

Definition 2: Finding an object Γ in the image is equivalent to computing among all possible objects in the image the one closest to Γ, i.e., to find

$$\Gamma^{T*} = arg \max_{\Gamma^T} P(\Gamma^T|\Gamma, I)\,.$$

Note that, if $P(\Gamma^T, \Gamma|I)$ is given, we can compute both Γ^{T*} and $P(\Gamma|I)$, since

$$\Gamma^{T*} = arg \max_{\Gamma^T} P(\Gamma^T, \Gamma|I) = arg \max_{\Gamma^T} P(\Gamma^T|\Gamma, I)$$

$$P(\Gamma|I) = \sum_{\Gamma^T} P(\Gamma^T, \Gamma|I) \approx P(\Gamma^{T*}, \Gamma|I)\,,$$

where the last approximation is only valid if the distribution $P(\Gamma^T, \Gamma|I)$ is sharply peaked on Γ^{T*}.

2.1 Deconstructing with key points and features

The focus now is to compute the optimal Γ^{T*}. We introduce the possibility of detecting localized features in the image represented 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.

The key points of a model contour are points that convey a lot/most of information about the contour (see Figure-1(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) are included. 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.

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, s and r are given by $d_S(s, r)$ and can be precomputed by following the contour. We then decompose $P(\Gamma^T|\Gamma, I)$ as

$$\begin{aligned}
P(\Gamma^T|\Gamma, I) &= \sum_{[x_s],\{y_k\}} P(\Gamma^T, [x_s], \{y_k\}|\Gamma, I) \\
&= \sum_{[x_s],\{y_k\}} P(\Gamma^T|[x_s], \{y_k\}, \Gamma, I)\ P([x_s], \{y_k\}|\Gamma, I) \\
&= \sum_{[x_s],\{y_k\}} P(\Gamma^T|[x_s], \{y_k\}, \Gamma, I)\ P(\{y_k\}|[x_s], \Gamma, I)\ P([x_s]|\Gamma, I) \\
&\approx \sum_{[x_s],\{y_k\}} P^{shape}(\Gamma^T|[x_s], \{y_k\}, \Gamma)\ P^{snake}(\Gamma^T|\{y_k\}, I) \\
&\qquad P^{features}(\{y_k\}|[x_s], \Gamma, I)\ P^{model-points}([x_s]|\Gamma)\,, \qquad (1)
\end{aligned}$$

where in the approximation we did consider the model points, $[x_s]$, to be independent of the image I, i.e., $P^{model-points}([x_s]|\Gamma) = P^{model-points}([x_s]|\Gamma, I)$. $P(\Gamma^T|[x_s], \{y_k\}, \Gamma, I)$ is decomposed into a shape model, $P^{shape}(\Gamma^T|[x_s], \{y_k\}, \Gamma)$

and a snake model $P^{snake}(\Gamma^T|\{y_k\}, I)$. We say the whole process is a deconstruction because we not only decompose the recognition into various terms, but later we reconstruct the final object in the image through this decomposition.

Ideally, the image features $\{y_k\}$ are originated from the model points $[x_s]$. A model of the geometric projection as well as illumination and reflectance is necessary. However, in our experiments we have neglected this issues and simply used the "SUSAN" feature detector [17] to generate a set of key features in the image. They are candidates for matching the model key features. These features usually capture corner points, junction points, and salient edges and we have found them useful for our experiments (see Figure-1(b)). Clearly, the total number of detected features affect the algorithm complexity and 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 similar distribution like the one found by those model key features.

Formally, we are making the assumption that the distribution over all possible features are sharply peaked at a set of features detected by SUSAN operator, say $\{y_k^{SUSAN}\}$. Moreover, we will manually produce $[x_s]$, say $[x_s^{manual}]$ and assume the distribution over all possible $[x_s]$ is also sharply peaked. These two assumptions can be formally written as

$$
\begin{aligned}
P^{features}(\{y_k\}|[x_s], \Gamma, I) &= \prod_{k=1}^{K} \delta(y_k - y_k^{SUSAN}) \,, \\
P^{model-points}([x_s]|\Gamma) &= \prod_{s=1}^{S} \delta(x_s - x_s^{manual})
\end{aligned} \tag{2}
$$

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

Matching key points and features: In the above model we have not made explicit that the model points , $[x_s]$, have a correspondence in the feature set $\{y_k\}$. We can make so by introducing 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 part of the model contour.
- $\sum_{k=1}^{K} M_{s,k} = 1$.

In this paper, we have not experimented with occlusions; otherwise we would need to consider the case $\sum_{k=1}^{k} M_{s,k} = 0$. We can now rewrite (1) as

$$
\begin{aligned}
& P(\Gamma^T|\Gamma, I) \\
\approx & \sum_{[x_s],\{y_k\}} \{ \prod_{s,i,j=1}^{S,K,K} [(1 - M_{s,i}M_{s-1,j}) + M_{s,i}M_{s-1,j} P^{shape}_{s,i,j}((\Gamma^T)^{y_i}_{y_j}|(\Gamma)^{x_s}_{x_{s-1}}) \\
& P^{snake}_{i,j}((\Gamma^T)^{y_i}_{y_j}|y_i, y_j, I)\ P^{features}_{s,i}(y_i|x_s, \Gamma, I)\ P^{modelpoints}_{s}(x_s|\Gamma)\ ]\} \qquad (3) \\
\approx & \prod_{s,i,j=1}^{S,K,K} [(1 - M_{s,i}M_{s-1,j}) \\
& + M_{s,i}M_{s-1,j} P^{shape}_{s,i,j}((\Gamma^T)^{y_i}_{y_j}|(\Gamma)^{x_s}_{x_{s-1}}) P^{snake}_{i,j}((\Gamma^T)^{y_i}_{y_j}|y_i, y_j, I)]\ ,
\end{aligned}
$$

where in the last step we have used (2), that is, the assumption (approximation) that image features obtained by SUSAN and the manually detected model points are optimal and sharply peaked. We also simply use $\{y_k\}, [x_s]$ to refer to $\{y_k^{SUSAN}\}, [x_s^{manual}]$.

Thus, the focus of our work is reduced to the snake and shape integration (decomposition and reconstruction). One could criticize having spent this effort on writing about probabilities of features and not using them. We argue that by following the approach fully, and the making of this assumption, helps to (1) clearly understand where the assumptions are being made; (2) In the future, extend the work to account for these probabilities, including a system that could evaluate feature detectors based on the recognition system; (3) The main technical difficulty of decomposing and reconstructing are left present to analyze the two remaining probabilities, for shape analysis and snakes.

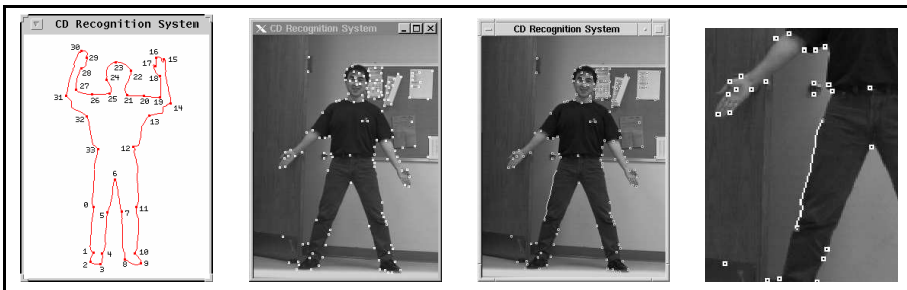
3 Snake and Shape Models

In this section, we establish both the snake model, $P(\Gamma^T|I, \{y_k\})$ which is a simple model for detecting contours, and the shape model, $P(\Gamma|\Gamma^T, [x_s], \{y_k\})$.

3.1 Snake

A snake model [9] is a deformable contour usually applied to detect contour boundaries (see Figure-1(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 images and features given contours, $P(I, \{y_k\}|\Gamma^T)$ to obtain $P(\Gamma^T|I, \{y_k\}) = P(I, \{y_k\}|\Gamma^T)\, P(\Gamma^T)$. We have considered the prior model of contours to be similar to the Elastica [11], i.e.

$$
\begin{aligned}
P(\Gamma^T) &= C_1 e^{-\int_{\Gamma^T} [\lambda_1 |k|_{\Gamma^T}(x(t),y(t)) + \lambda_2]\, dt} \\
&\text{or} \\
P_{i,j}((\Gamma^T)^{y_i}_{y_j}) &= {C'}_1 e^{-\int_{\Gamma^T = y_j}^{y_i} [\lambda_1 |k(t)| + \lambda_2]\, dt}\ , \qquad (4)
\end{aligned}
$$



(a) (b) (c) (d)

Fig. 1. (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.

where the contour is parameterized by t, the parameters λ_1 and λ_2 have to be estimated, $k_{\Gamma^T}(x(t), y(t))$, or simply $k(t)$, are the contour curvature at $(x(t), y(t)) \in \Gamma^T$, and C_1 is a normalization constant. This model differs from the elastica model [11] in that we require the curvature term to be scale invariant.
For the image generation, given the contour Γ^T, we have assumed that

$$P(I|\Gamma^T) = C_2 e^{-[A+\int_{\Gamma^T} \frac{1}{|\nabla I|^2_{(x(t),y(t))}+\epsilon}\, dt]}$$

or

$$P_{ij}(I, y_i, y_j|\Gamma^T) = C'_2 e^{-[A+\int_{\Gamma^T=y_j}^{y_i} \frac{1}{|\nabla I|^2_{(x(t),y(t))}+\epsilon}\, dt]}, \tag{5}$$

where $|\nabla I|_{(x(t),y(t))}$, 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,

$$P(\Gamma^T, \{y_k\}|I) = C \prod_{i,j=1}^{K,K} e^{-\int_{\Gamma^T=y_j}^{y_i} \left[\frac{1}{|\nabla I|^2(t)+\epsilon}+\lambda_1|k(t)|+\lambda_2\right] dt}, \tag{6}$$

where $C = C'_1\, C'_2\, e^{-A}$. In order to maximize this probability alone, one can use the Dijkstra algorithm (single source optimal path).

3.2 Shape analysis and Kullback Leibler distance

In [2], 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

(a) (b)

Fig. 2. (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.

undergoing articulations is untouched (similar) almost everywhere (see Figure-2(a)).
Finally we consider the Dijkstra algorithm at the top-down level, to minimize for the set of matchings, i.e., to determine M^*.

Our interest is to model $P^{shape}(\Gamma^T|\Gamma)$ or $P^{shape}_{s,i,j}((\Gamma^T)^{y_i}_{y_j}|(\Gamma)^{x_s}_{x_{s-1}})$. So, this is to say that given a contour Γ the probability of generating a "similar" contour Γ^T should be high. We notice that different correspondences can be established between two contours, say Γ and Γ^T, and a measure of similarity between them may depend on which correspondence is used. We can formalize this as follows: we parameterize the contours $\Gamma(s)\ ; s = 1, ..., S$ and $\Gamma^T(t)\ : t = 1, ..., T$. Then,

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

The sum over all possible correspondences can sometimes be reduced to the optimal correspondence, i.e., we may consider

$$P^{shape}(\Gamma^T|\Gamma) \approx P(\{t(s)^*\}, \Gamma^T|\Gamma)\,,$$

where $\{t(s)^*\} = arg \max_{\{t(s)\}} P(\{t(s)\}, \Gamma^T|\Gamma)$. Given the correspondence $t(s)$, we can then express the Kullback Leibler distance between the "true" contour, $\Gamma(s)$, and the detected one, $\Gamma^T(t(s))$, as

$$\mathcal{D}_{\{t(s)\}}(\Gamma^T||\Gamma) = \sum_{s=1}^{S} P(\Gamma(s)) log \frac{P(\Gamma(s))}{P(\Gamma^T(t(s)))}\,,$$

where we are indeed considering $\{\Gamma(s)\ :\ s = 1, 2, ..., S\}$ to be a sequence of samples from the distribution of the "true" contour. Analogously, $\{\Gamma^T(t(s))\ :\ s = 1, ..., S\}$ are the sequence samples from the "targeted" contour. Assuming again the prior (4) distribution on Γ, we obtain

$$\mathcal{D}_{\{t(s)\}}(\Gamma^T||\Gamma) = \int \{e^{-[|k(s)|+\lambda]}[|k^T(t(s))|dt - |k(s)|ds + \lambda(dt-ds)]\}$$
$$= \int ds\{e^{-[|k(s)|+\lambda]}[|k^T(t(s))|t'(s) - |k(s)| + \lambda(t'(s)-1)]\}. \quad (7)$$

where $t' = dt/ds$, $\lambda = \frac{\lambda_2}{\lambda_1}$. The problem is again to find $t^*(s)$ that minimizes $\mathcal{D}_{\{t(s)\}}(\Gamma^T||\Gamma)$.

This is a non-symmetric shape comparison measure, which may be desired since there is a "true" contour, the model contour Γ and it is possible that human perception does not utilize symmetric measures. Another (symmetric) distance we have considered [2] is

$$E^{shape}_{\{t^*(s)\}}(\Gamma, \Gamma^T)$$
$$= \min_{\{t(s)\}} E(\Gamma, \Gamma^T, \{t(s)\}) = \int ds \frac{|k^T t'^*(s) - k|^p}{(|k^T t'^*(s)| + |k|)^{(p-1)}} + \lambda \frac{|t'^*(s)-1|^p}{(t'^*(s)+1)^{p-1}}, \quad (8)$$

where p is constant (we use $p = 2$). The first term in the integral (8) is the bending cost and the second one is the stretching cost. Notice that the cost is rotation and translation invariant, since it is solely based on the curvature measure of the contour. The bending cost is scale invariant. The parameter λ weights the relative contributions of stretching and bending. One important property of the model is accounting of articulations. Since articulations occur at high curvature points, it will cost relatively less to bend them (because of the denominator of the bending cost).

In order to compute this distance we need to optimize for $t(s)$. This is possible using Dijkstra algorithm (single source optimal path).

Coarse computation Since the Dijkstra algorithm efficiency may still leads to long computations (when this module is called many times), we investigate a coarse approach to shape similarity. The shape energy can be computed in a coarse way as

$$E^{sh}_{s,i,j} = \frac{[\Theta_S(s) - \Theta_T(i)]^p}{[|\Theta_S(s)| + |\Theta_T(i)|]^{p-1}} + \lambda \frac{|t'(s,i,j) - 1|^p}{(t'(s,i,j)+1)^{p-1}}, \quad (9)$$

where $\Theta_S(s)$ and $\Theta_T(i)$ are the angles of the contours at the points x_s and y_i, respectively. The stretching is computed by

$$t'^*(s) = t'^*((\Gamma^S)^s_{s-1}, (\Gamma^T)^{f_i}_{f_j}) = \frac{d^S(s, s-1)}{d^T(y_j, y_i)},$$

where $d^T(y_j, y_i)$ is computed from the contour connecting y_j and y_i, i.e., depends on the choice of $(\Gamma^T)^{y_i}_{y_j}$, while $d^S(s, s-1)$ is a precomputed distance between x_s and x_{s-1} in the contour model (following the contour). These computations are done much faster than with Dijkstra algorithm. We have considered

$$P(\Gamma^T|\Gamma) = C \prod_{s,i,j=1}^{S,K,K} e^{-E^{sh}_{s,i,j}}\,.$$

4 Optimization: Top-down & Bottom-up

From (3) we have obtained

$$P(\Gamma^T|\Gamma, I) \approx \prod_{s,i,j=1}^{S,K,K} e^{-M_{s,i}M_{s-1,j}[E^{sn}_{i,j}((\Gamma^T)^{y_i}_{y_j},I)+\beta\ E^{sh}_{s,i,j}((\Gamma^T)^{y_i}_{y_j},(\Gamma)^{x_s}_{x_{s-1}})]}\,, \quad (10)$$

where

$$E^{sn}_{i,j}((\Gamma^T)^{y_i}_{y_j}, I) = \int_{(\Gamma^T)^{y_i}_{y_j}} \left[\frac{1}{|\nabla I|^2(t)+\epsilon} + \lambda_1 k^T(t) + \lambda_2\right] dt\,,$$

and

$$E^{sh}_{s,i,j} = \frac{[\Theta_S(s) - \Theta_T(i)]^p}{[|\Theta_S(s)| + |\Theta_T(i)|]^{p-1}} + \lambda \frac{|t'(s,i,j) - 1|^p}{(t'(s,i,j)+1)^{p-1}}\,.$$

Notice that $\{M_{s,i}\}$ have been rearranged, moved to the top of the exponential, without any change in the theory. When $M_{s,i} = 0$ the correspondence is not established and the costs do not contribute for the probability estimation.

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.

4.1 The rational

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_{s-1} 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_{s-1} as input. There is freedom to select different contours $(\Gamma^T)^{y_i}_{y_j}$ connecting y_j to y_i. The bottom-up process returns the cost induced by the best contour choice, $(\Gamma^{T*})^{y_i}_{y_j}$. This cost takes into account the snake cost and the shape comparison cost with Γ^s_{s-1}.

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

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, $E^{sn} + \beta E^{sh}$, for this pair of hypothesis. An illustration of the control loop is given in Figure-3.

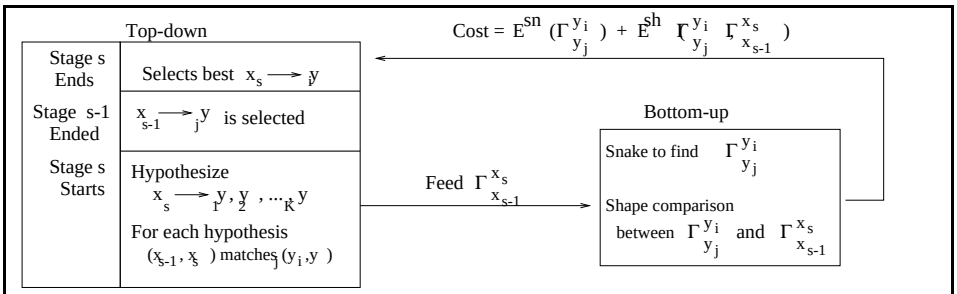


Fig. 3. Control loop of the top-down and bottom-up algorithm. The stage $s-1$ has been resolved with x_{s-1} matching y_j, i.e., $M^*_{s-1,j} = 1$. Then the algorithm finds the best y_i such that $M^*_{s,j} = 1$. It also finds the contour connecting y_j and y_i, i.e., $(\Gamma^{T*})^{y_i}_{y_j}$.

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

$$(\Gamma^{T*}, M^*) = arg \min_{\Gamma^T, M} logP(\Gamma^T | \Gamma, I)$$

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

We then approximate this solution by

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

$$(\Gamma^{T*})^{y_i}_{y_j} = arg \min_{(\Gamma^T)^{y_i}_{y_j}} E^{sn}_{i,j}((\Gamma^T)^{y_i}_{y_j}, I) \qquad \text{if } M^*_{s,i} = M^*_{s-1,j} = 1\,. \tag{11}$$

4.2 Different levels of Dijkstra algorithm

At the bottom-up level, we can solve for $(\Gamma^{T*})^{y_i}_{y_j}$ using dynamic programming [7], 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-1(c)(d) for results).

At the next level, still bottom-up, a Dijkstra algorithm has also been used to test the shape similarity model, i.e. to find $\{t^*(s)\}$ (see Figure-2 (a),(b)).

Finally we consider the Dijkstra algorithm at the top-down level, 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.

In order to justify the algorithm we use the dynamic programming principle (and the fact that the costs are all positive) since it is easier to convey the idea.

The dynamic programming algorithm works as follows (as shown in Figure-4 where each vertex is a possible matching combination, e.g., $M_{s,i} = 1$ indicates that a feature point y_i is matched to the model contour point $x_S(s)$.)

(a) Top-down: For each match candidate, $M_{s,i} = 1$, we compute all the transition cost $E^{sn} + \beta E^{sh}((\Gamma^{T*})_{y_j}^{y_i})$ for $j = 1, 2, ..., N$ of going from any of the pairs $(x_S(s-1), y_j)$ $(M_{s-1,j} = 1)$ to the current candidate $(x_S(s), y_i)$ (We don't need to consider all key features, but only the ones within reasonable distances from the current one.)

(b) Bottom-up: This transition cost is obtained by running a snake algorithm between the two key features, using Dijkstra algorithm, to obtain $(\Gamma^{T*})_{y_j}^{y_i}$. Then the shape comparison is used to estimate $t'(s)$, or one could use Dijkstra again to estimate $\{t(s)\}$ at more computational time.

(c) we repeat the computations for all key points, i.e., loop for $s = 1, ..., S$

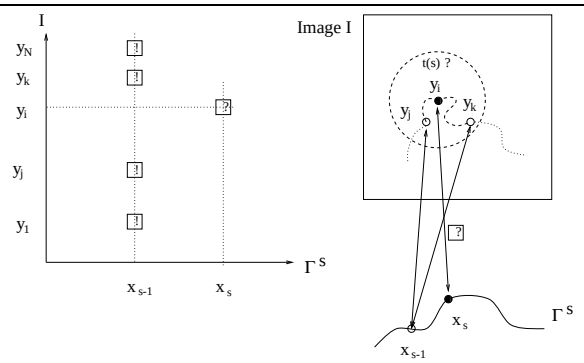


Fig. 4. Dynamic programming model

4.3 Further simplifications:

We discuss two further simplifications of the algorithm.

Optimization with Integration by Parts: We now consider to represent the objects by parts to simplify further the computations. A part is a partial model contour between two consecutive partition points that we have marked as *partition*-points (see Figure-5(b)). The set of partition-points is a subset of all key points from the model.

The globally optimal solution becomes a sum of optimizations done on each part of the model contour and we refer this process as "integration by parts". Whenever a snake recovered a new part in the optimization (and the solution has been verified by the shape comparison), we content that a new part is found and it is added into the previously recovered optimal solution. We then start a new optimization process from the last vertex recovered. We have used this simplification in our experiments.

Verifying the correctness of a target contour: Observe that if the articulations and deformations of a human shape do not introduce self-occlusions, the region enclosed by the polygon when orderly connecting the partition-points can be considered as a rigid part. Also, most of human articulations are caused by movements of parts such as hands, legs or head. This provides us a good measurement of how correctly a derived target contour is. For example, one can simply compute then compare the enclose rigid areas for the model and target contour to explicitly test a possible solution. We have not used this simplification in our experiments.

5 Experimental Results

In Figure-5, we show the results of testing our proposed shape contour recognition method. Given an input image as shown in Figure-5, we carried out the algorithm with respect to three different model contours as in Figure-5 (a), (b) and (c). Note that contour (a) is part of the whole human shape model contour (b). Satisfactory results are derived as seen in Figure-5 (e), (f), (h), (k) and (l). It is worth to point out that the image being tested does contain a complex background and our method has successfully recovered the corresponding contour of human shape in each experiment though there is still room for improvement such as in result (h), the right hand was 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 to complete (including displaying the evolution of detected contours on the window interface of our softwares).

6 Extensions and Conclusions

We have introduced a new method to recognize and detect shape contours for articulated and deformed objects. The basis of the approach is a carefully Bayesian-Information formulation and manipulation of the problem variables. We have

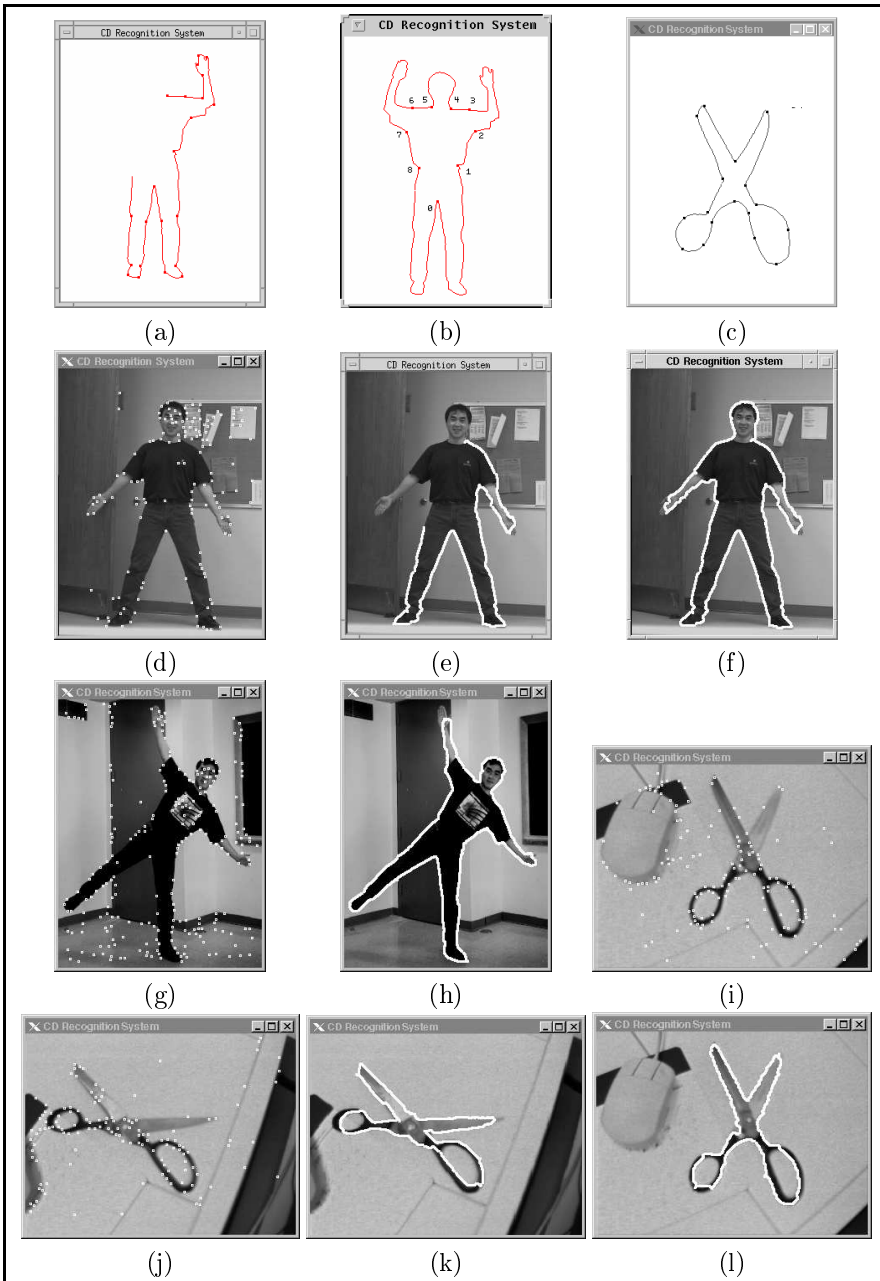


Fig. 5. (a), (b) and (c) are the model contours. (d), (g), (i) and (j) are test images with detected features. (e) (with model (a)), (f)(h) (with model (b)) and (k)(l) (with model (i)) are the corresponding detected shape boundaries.

relied on the use of Dijkstra algorithm as the main engine of computations. Combining shape and snakes have been part of the system and we have shed light on combining top-down & bottom-up approaches to recognition in unique way. The global optimization is approximately completed by optimization with "integration by parts".

A limitation of our experiments and formulation is the use of contour models, 1 dimensional graphs, allowing the use 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.

References

1. Y. Amit, D. Geman, and K. Wilder, "Recognizing Shapes from Simple Queries about Geometry," Technical Report, University of Chicago, 1995.
2. B. Basri, L. Costa, D. Geiger, and D. Jacobs, "Determine Shape Similarity," *IEEE workshop in Physics Based Vision,* Boston, June 1995.
3. I. Biederman, "Human Image Understanding: Recent Research and a Theory," *Computer Graphics, Vision, and Image Processing,* (32):29-73, 1985.
4. 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.
5. D. Forsyth, J. Malik, M. Fleck, Leung, Bregler, Carson, Greenspan, "Finding objects by grouping," *Proc. of Workshop on Object Representation, ECCV, England, 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. 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.
8. U. Grenander, Y. Chow and D. M. Keenan, " Hands: a Pattern Theoretic Study of Biological Shapes," Springer-Verlag, 1991.
9. M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models," *Proceedings of the First International Conf. on Computer Vision,* pp. 259-268, 1987.
10. K. Kupeev and H. Wolfson, "On Shape Similarity," *Proceedings Int. Conf. on Pattern Recognition,* pp. 227-237, 1994.

11. D. Mumford. Elastica and computer vision. In C. L. Bajaj, editor, *Algebraic Geometry and Its Applications.* Springer-Verlag, New York, 1993.
12. J. L. Mundy and A. Zisserman (editors), "Geometric Invariance in Computer Vision," MIT Press, 1992.
13. A. Pentland, "Recognition by Parts," *Proceedings of the First International Conf. on Computer Vision,* pp. 612-620, 1987.
14. A. Pentland and B. Horowitz, "Recovery of Non-rigid Motion and Structure," *IEEE Trans. Pattern Analysis and Machine Intelligence,* **13**(7), 730-742, 1991.
15. 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.
16. 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.
17. S. Smith and J. Brady, "SUSAN - a New Approach to Low Level Image Processing," *Int. Journal of Computer Vision,* 1996, In publication.
18. K. Siddiqi and B. Kimia, "Parts of Visual Form: Computational Aspects," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-17, March 1995.
19. K. Siddiqi and B. Kimia, "Toward a Shock Grammar for Recognition," *IEEE Conf. on Computer Vision and Pattern Recognition,* 1996.
20. M. Turk and A. Pentland, "Face Recognition Using Eigenfaces," *IEEE Conf. on Computer Vision and Pattern Recognition,* pp. 586-591, 1991.
21. S. Ullman, "High-level vision", MIT Press, Cambridge, 1996.
22. 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.
23. 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.
24. 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.

This article was processed using the LaTeX macro package with LLNCS style