In Proc. CVPR'2000, pp. 10-17

Fixed Topology Skeletons

Polina Golland and W. Eric. L. Grimson
Artificial Intelligence Laboratory,
Massachusetts Institute of Technology,
Cambridge, MA 02139, USA
{polina,welg}@ai.mit.edu

Abstract

In this paper, we present a novel approach to robust skeleton extraction. We use undirected graphs to model connectivity of the skeleton points. The graph topology remains unchanged throughout the skeleton computation, which greatly reduces sensitivity of the skeleton to noise in the shape outline. Furthermore, this representation naturally defines an ordering of the points along the skeleton. The process of skeleton extraction can be formulated as energy minimization in this framework. We provide an iterative, snake-like algorithm for the skeleton estimation using distance transform.

Fixed topology skeletons are useful if the global shape of the object is known ahead of time, such as for people silhouettes, hand outlines, medical structures, images of letters and digits. Small changes in the object outline should be either ignored, or detected and analyzed, but they do not change the general structure of the underlying skeleton. Example applications include tracking, object recognition and shape analysis.

1. Introduction

Skeletons, or medial axis transforms, have been used in computer vision for several decades. As the name implies, a skeleton is a set of curves that approximates the local symmetry axis of the shape. Several definitions of skeletons have been proposed in the literature. One of the first [2] was based on a "grass fire" model, i. e. , a moving wavefront generated by an inward motion of an outline curve with constant speed along a normal vector at every point on the curve. The skeleton is the set of points at which the wavefront crosses itself. It can be shown that each skeletal point is the center of an inscribed circle that touches the outline in more that one point.

An alternative definition for skeletons is based on a distance transform. Distance transform, or distance map, $D(x, y)$ is a function that for any point inside a shape is equal to the distance from the point to the closest point on an outline [2, 15]. A skeleton of a shape can be defined as the set of ridge points of the distance map. It can be proved that the two definitions are equivalent, and furthermore, a value of a distance map at any skeletal point is equal to the radius of the inscribed circle associated with it.

Numerous algorithms have been developed for skeleton extraction [1, 3, 12, 13], using wavefront propagation or distance transform. Another group of algorithms uses Voronoi diagrams for skeleton computation [4, 16]. The main drawback of traditional skeletons is their high sensitivity to noise in the boundary: small errors in segmentation of the object can drastically change the structure of the subsequently derived skeleton. This becomes a serious problem when the shapes are not defined by smooth curves or surfaces, but extracted from digital images. Several methods have been proposed to stabilize the skeleton extraction, mostly by pruning "false" branches that are believed to be caused by noise in the outline [4, 12, 13]. A different approach, based on self-similarity of a smooth outline curve, was demonstrated in [14].

Skeletons provide an intuitive, compact representation of a shape, which made them appealing for many applications in computer vision. One of the important features of the medial axis representation is separation of the shape's topological properties (sub-parts and connectivity between them) from its geometric properties (the location of the curves and the shape width at every skeletal point). If the object of interest undergoes non-rigid transformations, this property becomes important for modeling the shape. Examples include articulated motion (e.g., people, non-rigid objects), as well as differences in shape between different instances of the same object that cannot be explained by the similarity transformation (for example, natural shape variability of anatomical structures). Recent examples of applications using skeletons include modeling of articulated motion in tracking (a human body was modeled using a simplified version of a skeleton for a tracking application in [8]),

shape modeling of anatomical structures for segmentation and registration (a modified version of skeletons, so called *cores* was used in [6] for this purpose) and statistical shape analysis (shape features were extracted using skeletons for corpus callosa and used for classification in [10]).

In this paper, we propose and develop an approach to robust estimation of the geometrical properties of the shape and its skeleton in cases when the topology of the skeleton is known *a priori*.

1.1. Fixed Topology Skeletons

In many applications, the global shape of the object is known ahead of time, and one would like to either ignore small changes in the shape (for example, in tracking of articulated objects), or detect and study them (such as shape analysis of anatomical structures). This information can help in extracting the skeleton more reliably. We propose a new representation, *fixed topology skeletons*, to be used as a framework for incorporating constraints on shape topology into the skeleton extraction algorithm. We model a skeleton using an undirected graph whose global structure does not change during the computation, while its location is adjusted to approximate the main ridges of the distance map. An important question is, "What are the situations when the information on the topology of the skeleton is available and can be used for robust estimation of its geometrical properties?" We discuss several such examples in this section and provide the results for some of them in Section 5.

In tracking human motion, we know the general shape of the object and would like to estimate its location, size and the position of the articulated parts relative to each other. By fixing the topology of the skeleton, we utilize information available to us besides the input image to improve the accuracy of the skeleton estimation. Moreover, the considerations of speed in real time applications force us to use images of low resolution, which causes even greater quantization error and more noise in the outline extraction. Therefore the proposed approach can offer a significant improvement by using additional information to stabilize the skeleton. The resulting skeleton can be used to study the object motion, e.g., measuring the angles between branches of the skeleton and their change over time, or detecting periodicity of the motion.

Other examples are medical applications such as virtual endoscopy and vessel connectivity estimation. One of the tasks of virtual endoscopy [9, 17] is to generate a smooth fly-through path between two points inside a tubular structure, such as bronchi or a colon, that is as close to the middle of the tube as possible[1]. In the second example, the goal is to create a graph-like representation of the segmented set of tubular structures, such as blood vessels, for therapeutic planning and surgical navigation. While traditional skeletons are too sensitive to noise to produce a satisfactory result, the algorithm proposed in this paper can be easily extended to handle 3D tubular structures (whose skeletons are curves), and therefore can be used in those applications.

[1] Any contact of the path with the walls of the structure will cause poor visualization and will increase chances of tissue damage if the system is used to drive an endoscope.

1.2. Active Contours

A graph representation of a skeleton lends itself naturally to a snake-like algorithm. Snakes, or active contours, were introduced by Kass, Witkin and Terzopoulos [11] and have been extensively used in computer vision for segmentation. This approach casts the problem of boundary localization into a curve evolution framework. The curve is evolved in a potential energy field (intensity gradient in the case of segmentation) under a set of smoothness constraints. Fua and Brechbuhler [7] proposed a method for incorporating geometric constraints (angle values, distances, etc.) into the active contour algorithm.

If we use the distance transform as the potential energy function in this formulation, the snake algorithm can be used for skeleton extraction. In fact, Leymarie and Levine [13] used the active contour algorithm on the distance transform to simulate the grass fire wavefront propagation and estimate the shape skeleton. The points where the wavefront crossed itself were identified as skeletal points. A post-processing step of parsing the resulting snake and estimating the graph structure from the collapsed contour was proposed, as well as a pruning technique that could be incorporated into the algorithm.

The main difference of the algorithm used in this paper for estimation of the skeleton location from other snake based methods is that we operate on a graph of a general structure (as an opposite to a set of closed curves). In our implementation, the branches of the graph are driven in a snake-like fashion towards the ridges of the distance map, while the connectivity between the branches is fixed. Use of prior information allows us to eliminate the steps of topology estimation and pruning required in [13]. This points to an additional advantage of using a graph representation: it defines a natural ordering of the points along the skeleton curves. This can be important if the skeletons are used to establish correspondence between similar shapes, or to extract shape features for further analysis. It can be difficult to infer the connectivity structure in places where several skeleton branches merge together, as many points seem to be good candidates for a junction node. In the proposed representation, the connectivity is fixed and we optimize for the position of the junction.

The remainder of the paper is organized as follows. In the next section, we define the graph representation and in-

troduce necessary notation. Section 3 contains a review of the traditional active contours algorithm, followed by a description of a modified snake algorithm we developed for skeleton extraction. Then the results of applying this technique and testing of the algorithm's sensitivity to initialization are presented, followed by concluding remarks.

2. Graph Representation

We use undirected graphs to model skeletons. A node in a graph corresponds to a point on the skeleton, an edge establishes a neighborhood relationship between two points of the skeleton. There are three types of nodes: leaves, junction nodes and internal nodes. A leaf has exactly one neighbor and is used to model an endpoint of a skeleton. A junction node has more than two neighbors and corresponds to a merging point of several branches of a skeleton. An internal node has exactly two neighbors and is used to approximate points on the branches of the skeleton. Formally, a skeleton $\mathcal{S} = (V, E, X)$ is defined as following:

$$\begin{aligned} &V = \{i | 1 \le i \le N\}, \\ &E \subseteq V \times V, \\ &X = \{\mathbf{x}_i\}_{i \in V}, \end{aligned} \tag{1}$$

where V is the set of nodes, E is the set of edges of the graph and X is the set of node positions $\mathbf{x}_i = (x_i, y_i)$ in the image plane. We use $\mathcal{N}_i$ to denote a set of neighbors of node i

$$\mathcal{N}_i \stackrel{\triangle}{=} \{j | (i,j) \in E\}. \tag{2}$$

A *reduction* operation on an internal node is defined as removing the node, while adding a new edge between its neighbors. If reduction is applied repeatedly to the graph until there are no internal nodes left in the graph, we call the resulting graph an *r-graph* (for "reduced graph"):

$$\begin{aligned} &\mathcal{R}(V, E, X) = (V_r, E_r, X_r), \\ &V_r = \{i | |\mathcal{N}_i| \neq 2\}, \\ &E_r = \left\{ (i,j) \;\middle|\; \begin{array}{l} \text{there exists a path from} \\ i \text{ to } j \text{ in } \mathcal{S} \text{ passing thru} \\ \text{internal nodes only} \end{array} \right\}, \\ &X_r = \{\mathbf{x}_i\}_{i \in V_r}, \end{aligned} \tag{3}$$

We say that two undirected graphs have the same topology if the corresponding *r-graphs* are isomorphic. It is easy to see that the topology of the skeleton is fully determined by the connectivity between its leaves and its junction nodes. Therefore, we can guarantee that the topology of the graph does not change if we restrict our modifications of the graph to reductions and insertions of internal nodes of the graph.

We observe that the nodes of the three types play different roles in the graph representation of the skeleton: the internal nodes are used to approximate the curves of the skeleton (which are not necessarily straight lines), and the the leaves and the junction nodes determine the skeleton topology.

3. Active Contours

A common physical model used by active contours is an elastic band with mass density ρ and elasticity κ moving in the potential energy field. The band is parametrized by arclength as $\mathbf{x}(\xi) = (x(\xi), y(\xi)), 0 \le \xi < 1$. The problem of ridge extraction of an image function can be formulated as an energy minimization:

$$\mathcal{E}_{total} = \int_0^1 \left[\rho \mathcal{E}_p(x(\xi), y(\xi)) + \mathcal{E}_d(\xi)\right] d\xi, \tag{4}$$

where $\mathcal{E}_p(x, y)$ is the potential energy function, and

$$\mathcal{E}_d(\xi) = \kappa \left[\left(\frac{\partial x}{\partial \xi} \right)^2 + \left(\frac{\partial y}{\partial \xi} \right)^2 \right] \tag{5}$$

is the energy of the elastic deformation. Using Euler-Lagrange equations, we arrive at the dynamics of the curve (defined by two forces: the gradient of the potential energy and the elastic deformation force):

$$\mathbf{x}_t(\xi) = -\rho \nabla \mathcal{E}_p(\mathbf{x}) + \kappa \frac{\partial^2 \mathbf{x}}{\partial \xi^2}. \tag{6}$$

Geodesic snakes algorithm [5] modifies this equation to restrict the evolution of the curve to the normal direction at every point on the curve:

$$\mathbf{x}_t(\xi) = \langle (-\rho \nabla \mathcal{E}_p(\mathbf{x}) + \kappa \frac{\partial^2 \mathbf{x}}{\partial \xi^2}) \cdot \mathbf{N}(\xi) \rangle \mathbf{N}(\xi), \tag{7}$$

where $\langle \mathbf{u} \cdot \mathbf{v} \rangle$ is an inner product of vectors $\mathbf{u}$ and $\mathbf{v}$, and $\mathbf{N}(\xi)$ is a unit length normal pointing inwards. This approach postulates that displacements along the tangent directions affect the parameterization of the curve, but not its shape, and should therefore be excluded from the evolution dynamics.

In practice, a discrete approximation is used, and Eq. (7) becomes

$$\begin{aligned} \Delta \mathbf{x}_i^t &\stackrel{\triangle}{=} \mathbf{x}_i^{t+1} - \mathbf{x}_i^t \\ &= \langle (-\rho \nabla \mathcal{E}_p(\mathbf{x}_i^t) + \kappa \sum_{j \in \mathcal{N}_i} \mathbf{x}_j^t - \mathbf{x}_i^t) \cdot \mathbf{N}_i^t \rangle \mathbf{N}_i^t, \end{aligned} \tag{8}$$

where $\mathbf{x}_i^t$ is the location of point i on the discrete approximation of the snake at iteration t and $\mathbf{N}_i^t$ is the normal vector at that location estimated using the point's neighbors.

Since we are interested in extracting the ridges of the distance map, we use the negated distance transform $D(x, y)$ as a potential energy function:

$$\Delta \mathbf{x}_i^t = \langle (\rho \nabla D(\mathbf{x}_i^t) + \kappa \sum_{j \in \mathcal{N}_i} \mathbf{x}_j^t - \mathbf{x}_i^t) \cdot \mathbf{N}_i^t \rangle \mathbf{N}_i^t. \tag{9}$$

The process is stopped when the curve starts oscillating around the ridge. We can see that this process essentially simulates a smoothed version of a wavefront propagation process.

4. Skeleton Estimation

Now we are ready to describe an algorithm for skeleton estimation that consists of three main steps. First, we estimate the positions of the leaf nodes along the outline and initialize the graph. Then we use a snake-like algorithm to "drive" the graph along the gradient of the distance map while keeping the leaf nodes fixed. Once the graph has settled onto the distance map ridges, the leaf node positions are adjusted to minimize the reconstruction error. This section describes each of the three steps in details.

4.1. Leaf Position Estimation

In order to initialize the skeleton graph, we first estimate positions of all leaf nodes along the outline. Since the leaf positions will be refined in the later steps of the algorithm, the precision of this step is not crucial, as long as the initial estimates are in the "right segments" of the outline (i.e., they can be off the optimal position, but not swapped with another leaf on the outline). Section 5 examines the method's sensitivity to errors in initialization of the leaves.

It has been shown [12, 13, 15] that the endpoints of the skeleton correspond to maxima of positive curvature along the outline. This provides us with a way of estimating the leaf positions. Fig. 1 shows an example shape, an initial graph, and a plot of the outline curvature for that shape. The five maximum points were used to initialize the skeleton.

In some applications, the initialization can use *a priori* knowledge about the shape. For example, for people tracking, we can get a fairly accurate guess at the initial position and orientation of the skeleton based on the bounding box of the silhouette. Section 5 contains an example of a tracking application and provides the details of the leaf position estimation. In tracking in general, the results from the previous frames can be used to initialize the skeleton in the next frame, so that only the first frame needs to be initialized using additional information. If the motion is estimated from a video sequence, some combination of the predicted positions for the leaf nodes in the next frame and the estimation results from the previous frame can be used for initialization.

Medical images are another example where additional information on the shape orientation might be available. In this case, we usually know an approximate position and orientation of a body in the scanner, and this might be sufficient for initial estimation. Section 5 contains an example of corpus callosum, when the initialization was performed

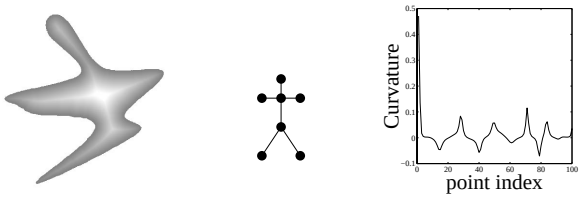


Figure 1. Skeleton extraction, example: (a) original distance map; (b) initial skeleton graph; (c) curvature.

using one point on the left side of the shape and one point on the right side of the shape. Since the skeleton structure is fairly simple, this was sufficient for reliable skeleton estimation.

Once the leaf node positions on the outline are estimated, the junction nodes are assigned arbitrary locations along the outline. Then branches are constructed to follow the outline. This constrains the graph to the inside of the shape, since we are using the gradient of the distance transform to "drive" the graph.

The algorithm is quite insensitive to the initial position of the junction nodes. We have tested the algorithm for different initialization points, and it converged to the same solution independently of the initial positions of the junction nodes, while the number of iterations varied slightly depending on the initialization point. For example, we tested 20 different initialization points for the junction nodes selected randomly along the outline for the shape in Fig. 1. The number of iterations required for the skeleton to converge varied between 405 and 440. Using the first skeleton as reference, in all trials the distance between any point on the resulting skeleton and the closest point on the reference skeleton was under 1 pixel. Fig. 2 shows the progress of the algorithm and the final solution for two different initialization points for that shape.

4.2. Graph Position Estimation

The branches of the graph evolve using the standard snake update rule of Eq. (9) (we use $\rho = 1, \kappa = 0.5$ in our experiments), while the leaf nodes remain fixed on the outline. This evolution allows us to update the positions of the internal nodes. For technical reasons, it is desirable to maintain close to uniform sampling along the curves, and therefore the branches of the skeleton need to be re-sampled (re-parametrized) every few iterations. Note that this does not change the topology of the corresponding *r-graph*, as we only operate on internal nodes.

The junction nodes serve as connectors between the branches, their main purpose is to preserve the structure of

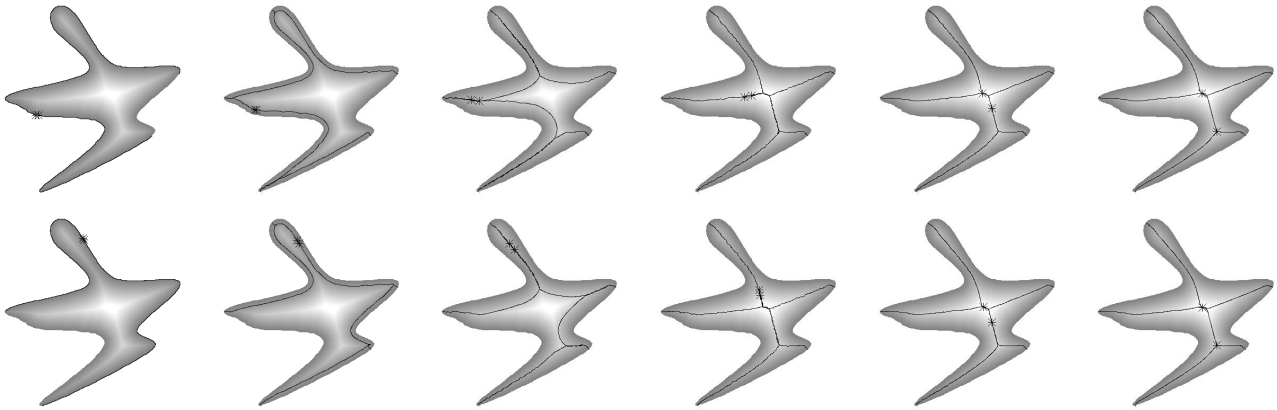


Figure 2. Skeleton location after 0, 5, 15, 200, 300 and 400 iterations for two different initializations of the junction nodes. Stars indicate junction nodes. The skeleton is first folded along the outline with both junction nodes placed at the same point. As the algorithm progresses, the skeleton unfolds and settles onto the ridges of the distance map.

the skeleton. Thus the update rule for a junction node places it in the center of the polygon defined by its neighbors:

$$\mathbf{x}_i^{t+1} = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mathbf{x}_j^t. \tag{10}$$

This rule is somewhat similar to Eq. (9), except that the junction nodes are not affected by the distance map.

The algorithm stops when no node is moving significantly. Fig. 2 shows several snapshots of the skeleton during this process.

4.3. Leaf node position re-estimation

In order understand how the optimal positions of the leaves are estimated, let's first consider reconstructing the outline from its estimated skeleton. This can be achieved by creating a circle around each skeletal point with the radius equal to the value of the distance map at that point and computing an envelope of all the circles. If the skeleton were precise, the reconstructed outline would be identical to the original one. But since we are computing a constrained version of the skeleton, the reconstructed outline is only an approximation of the original outline. We define a distance from point $\mathbf{u}$ to outline $\mathcal{O}$ to be the shortest distance from that point to any point on the outline:

$$d(\mathbf{u}; \mathcal{O}) = \min_{\mathbf{v} \in \mathcal{O}} \|\mathbf{u} - \mathbf{v}\|, \tag{11}$$

and the reconstruction error to be an average distance from the points on the original outline $\mathcal{O}$ to the reconstructed outline $\mathcal{O}'$:

$$e_r(\mathcal{O}, \mathcal{O}') = \frac{1}{|\mathcal{O}|} \sum_{\mathbf{u} \in \mathcal{O}} d(\mathbf{u}; \mathcal{O}'). \tag{12}$$

Note that we can bound the distances $d(\mathbf{u}; \mathcal{O}')$ and e_r using the points of the skeleton directly, without reconstructing the outline (a triangle inequality):

$$d(\mathbf{u}; \mathcal{O}') \leq \min_{\mathbf{x}_i \in \mathcal{S}} (\|\mathbf{u} - \mathbf{x}_i\| + D(\mathbf{x}_i)). \tag{13}$$

We say that point $\mathbf{u}$ on the original outline is *explained* by point $\mathbf{x}_i$ on the skeleton if the closest point to $\mathbf{u}$ on the reconstructed outline was generated by the circle centered at $\mathbf{x}_i$. We will say that a segment of an outline is *explained* by a particular branch of the skeleton if every node in the segment is explained by some node on the branch.

Now we can describe the algorithm for estimating an optimal position for each leaf node. First, we estimate a segment of the outline that is explained by the leaf branch (which consists of nodes between the leaf and the closest junction node). This is achieved by scanning the outline from the leaf node in each direction until we encounter a node on the outline that is explained by the node on the skeleton that does not belong to the leaf branch. Then we allow the leaf node to slide along the outline (first in the clockwise and then in the counterclockwise direction), and for every new leaf position, we re-estimate the position of the leaf branch using the update rule of Eq. (9). Since the position of the leaf node doesn't change much in every step, re-estimation of the branch location takes only a few iterations. For every new position of the leaf node, we compute the reconstruction error e_r for the outline segment explained by the current leaf branch. The process stops when a local minimum is found, or when the leaf reaches the boundary of the explained outline segment. Fig. 3 shows the outline segments explained by the corresponding leaf branches and the final positions of the leaf branches, as well as the resulting skeleton and the reconstructed outline.

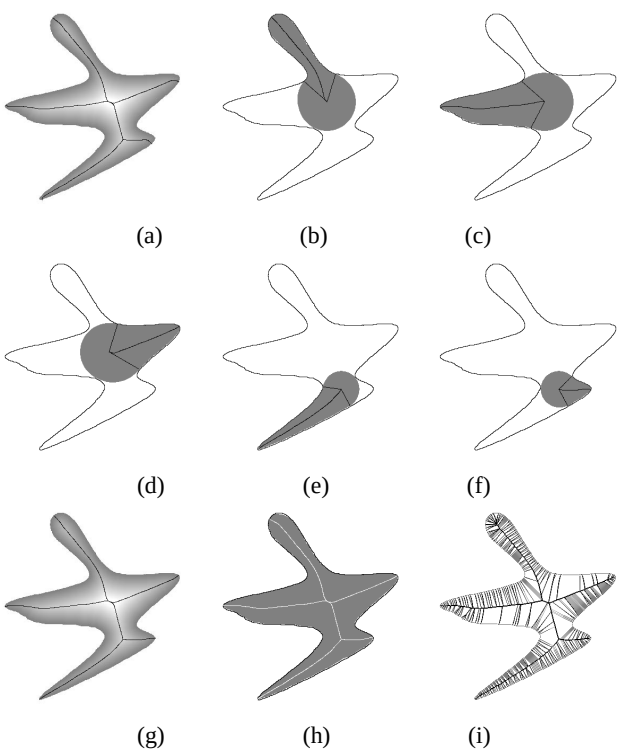


Figure 3. Final leaf positions: (a) fixed topology skeleton before leaf position re-estimation; (b)-(f) leaf position re-estimation: the shaded area corresponds to the part of the shape that is explained by the leaf branch, only that area is used in the final leaf position estimation; (g) fixed topology skeleton, the final result; (h) the final reconstruction (gray) with the original outline (black), compare the leaf node positions to (a); (i) compare to the traditional skeleton.

Note that this procedure finds a leaf position that corresponds to a local minimum of the reconstruction error. If the run time is not a concern, an exhaustive search can be performed for every leaf inside the outline segment explained by that leaf.

5. Experimental Results

We have applied the algorithm to both artificial and real images from different domains. Since it relies on the initial positions of the leaf nodes, we have tested the algorithm's sensitivity to the initialization. The results are reported in this section.

Fig. 4 shows an example of another artificial image skeletonization. The initial leaf positions have been estimated using the extrema of the curvature points. The "traditional" skeletons shown for comparison in this paper were computed by extracting local maxima of the distance transform.

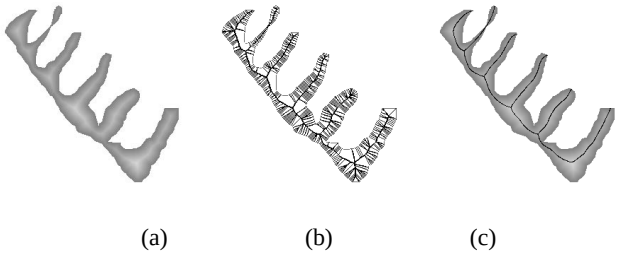


Figure 4. Skeleton extraction, artificial example: (a) original distance map; (b) traditional skeleton; (c) fixed topology skeleton.

Real Images. Fig. 5 demonstrates application of fixed topology skeletons to tracking data. We show the estimated skeleton overlaid on top of the grayscale images (64×64 pixels) of a walking person. The tracking software developed in our group [18] was used to segment the images, then the largest connected component was extracted and the resulting binary image was given as an input to the skeletonization algorithm. The leaves were initialized as following:

- head: the highest point of the silhouette,
- arms: $(x_{\min,\max}, \frac{1}{2}(y_{\min} + y_{\max}))$,
- legs: $(x_{\min,\max}, y_{\min})$,

where $(x_{\min}, x_{\max}, y_{\min}, y_{\max})$ define the bounding box of the silhouette. For each of the five points above, the closest point on the outline was found and used as the initial position for the corresponding leaf.

We can see that the algorithm estimated well the features that were present in the images and made reasonable interpolation for the occluded features (arms). Since we did not introduce any knowledge on the geometry of the skeleton, the absence of the arms did not prevent the algorithm from "detecting" them[2]. We do not expect the branches of the skeleton to exactly mimic the position of the limbs of the person, since we have not incorporated such geometric information, but we do expect the leaf associated with each extremum to be close to the appropriate contour boundary point. As can be seen, this generally occurs for these cases, even when the arm is not visible as a boundary point. This demonstrates that we can use fixed topology skeletons to model non-rigidly moving bodies for such problems as

[2] If such behavior is undesirable, additional constraints have to be introduced. For example, if removing a particular branch of the skeleton does not increase the reconstruction error significantly, we might want to conclude that the corresponding feature is absent from the image. This has to be done on per application basis, as requirements on the inference can change from problem to problem.

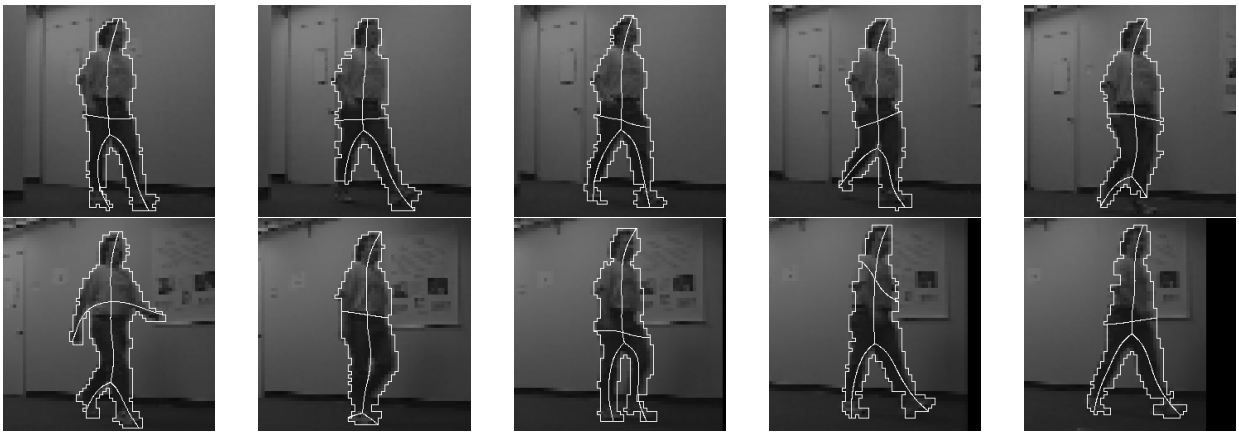


Figure 5. Skeleton extraction for two sequences of five frames each.

gait recognition, detecting periodicity of the motion, measuring motion parameters, etc. The method uses the results of tracking (segmentation) to produce a simple graph description of the moving shape. If we further want to extract limb articulations, then adding geometric information to the skeleton would be necessary.

Shape Analysis. We have used fixed topology skeletons for feature extraction in shape analysis of corpus callosum (a cross-section of the fiber tract that connects the two hemispheres in the brain). Fig. 6 shows an example of skeletonization of corpus callosum from a segmented MRI slice of a brain. Once the skeleton was extracted, the curvature and the width of the shape were measured at a set of discrete points along the skeleton and used as features for detecting statistical differences in shape between a group of schizophrenia patients and a group of normal control patients [10]. The algorithm was used to extract a skeleton for 66 different images of corpus callosum in the study.

In this application, the skeleton was assumed to be a simple string (no junction nodes), and the initialization was obtained from *a priori* knowledge on the shape orientation: one point was "guessed" to be in the left half of the image, and another leaf was assumed to be in the right half of the image. Then the closest points on the outline were automatically found and used as the initial positions of the two leaves. We can see that the initialization is far from the optimal position of the leaf nodes found by the algorithm. The corpus callosum case presents a serious challenge to the conventional methods for skeleton extraction, as the shape size is comparable to the pixel size, and using segmented images results in a highly noisy skeleton. By fixing the structure of the skeleton, we avoid the problem of false branches.

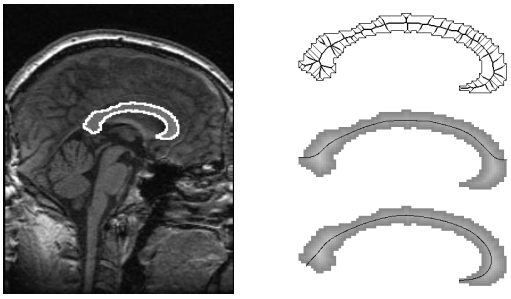
Robustness. To test the algorithm's sensitivity to initialization, we performed the following experiment: given the shape and the optimal leaf positions (found by the algorithm and verified by the user), we moved every leaf away from the optimal position. Then we performed the skeletonization algorithm and measured the maximal distance from any point on the original outline to the reconstructed outline:

$$e_{\max}(\mathcal{O}, \mathcal{O}') = \max_{\mathbf{u}\in\mathcal{O}} d(\mathbf{u}; \mathcal{O}') = \max_{\mathbf{u}\in\mathcal{O}} \min_{\mathbf{v}\in\mathcal{O}'} \|\mathbf{u} - \mathbf{v}\| \quad (14)$$

In addition to distance $e_{\max}(\mathcal{O}, \mathcal{O}')$ between the outlines, we also report distance $e_{\max}(\mathcal{S}, \mathcal{S}')$ between two skeletons: the skeleton estimated using the optimal leaf positions and the one estimated after perturbing the leaves. These two measures indicate how robust the algorithm is to the initial conditions.

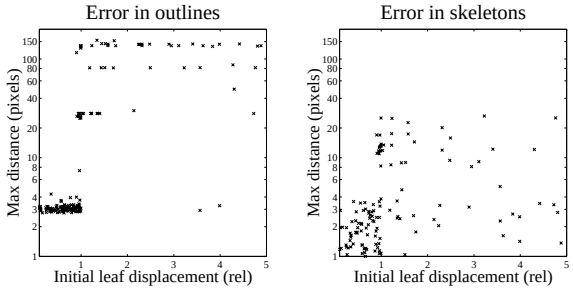
Fig. 7 shows the results of the experiment for the shape in Fig. 1. Every data point in the graph corresponds to a single run with all five leaves perturbed from the optimal position; a total of 200 runs is reported. The first step of the algorithm is to place the leaves on the outline, thus it is reasonable to report the results based on the distance between the points along the outline (rather than the Euclidean distance between them). Furthermore, since different leaves explain different portions of the outline, we normalize the distance by the appropriate outline segment length. For example, if a new leaf position is 50 pixel lengths away from its optimal position in the clockwise direction, we divide 50 by the distance (along the outline) between the optimal leaf position and the last node on the outline in the clockwise direction that is explained by the corresponding leaf branch. This normalization allows us to compare results from different branches of the skeleton. In Fig. 7, we show the maximal distance $e_{\max}$ between the outlines and the skeletons as a function of the largest (among the five leaves) normalized displacement.

As we can see, the algorithm is quite robust if all the leaves are initialized inside the corresponding outline segments (normalized displacement is smaller than 1). The distance between the outlines is between 3 and 4 pixels for all the runs and the distance between the skeletons is under 4



(a) (b)

Figure 6. Skeleton extraction, corpus callosum: (a) a slice of an MR scan with the corpus callosum outlined in white; (b) skeletonization results: traditional skeleton (top), fixed topology skeleton before leaf re-estimation (middle) fixed topology skeleton, final result (bottom).



(a) (b)

Figure 7. Maximal distance $e_{\max}$ (in logarithmic scale) as a function of relative leaf displacement (a) between the original and the reconstructed outlines; (b) between the skeletons before and after the initial leaf positions were perturbed. See text for more details.

pixels. Note that this is the maximal distance, and most of the points on the resulting skeleton (83% to 87% for the runs with the normalized displacement of all the five leaves smaller than 1) are less than 1 pixel away from the original skeleton. As we consider runs with the normalized initial displacement greater than 1 (at least one leaf was initialized outside of its corresponding range), the error increases. Several distinct values for the distance between the outlines emerge, which corresponds to the algorithm erring on different leaves.

6. Concluding remarks

We have presented a new approach to incorporating *a priori* information into skeleton estimation. We use a graph representation for the skeleton. Robustness is achieved by constraining the graph topology and introducing smoothness constraints on the skeleton branches. We use an iterative snake-like algorithm for estimating the location of the branches.

Fixed topology skeletons can be useful in applications where some information on the global shape of the object is available. They provide a way to constrain the skeleton, restricting the space of possible solutions. Example applications that could benefit from this approach include object recognition, articulate shape modeling, shape feature extraction.

Acknowledgements. The authors would like to thank Doron Shaked and Freddy Bruckstein for comments and suggestions on this work and Chris Stauffer for providing the tracking data.

This research was supported by Mitsubishi Electric Research Laboratories and NSF IIS 9610249 grant.

References

[1] C. Arcelli and G. Sanniti di Baja. A Width Independent Fast Thinning Algorithm. In *IEEE Trans. PAMI*, 7:463-474, 1985.

[2] H. Blum. A Transformation for Extracting New Descriptors of Shape. In *Models of the perception of Speech and Visual Form*, MIT Press, Cambridge, MA, 1967.

[3] F. L. Bookstein. The Line Skeleton. *CGIP: Comp. Graphics and Image Proc.*, 11:123-137, 1979.

[4] J. W. Brandt and V. R. Algazi. Continuous Skeleton Computation by Voronoi Diagram. *CVGIP: Image Understanding*, 55:329-338, 1994.

[5] V. Caselles, R. Kimmel and G. Sapiro. Geodesic Active Contours. *IJCV*, 22(1):61–79, 1997.

[6] D. S. Fritsch *et al*. Stimulated Cores and their Applications in Medical Imaging. In *Proc. IPMI'95*, 365-368, 1995.

[7] P. Fua and C. Brechbuhler. Imposing Hard Constraints on Soft Snakes. In *Proc. ECCV'96*, 2:495-506, 1996.

[8] H. Fujiyoshi and A. J. Lipton. Real-time Human Motion Analysis by Image Skeletonization. In *Proc. IEEE Workshop on Appl. Comp. Vision*, 15-21, 1998.

[9] F. A. Jolesz *et al*. Interactive Virtual Endoscopy. *American J. Radiology*, 169:1229-1235 1997.

[10] P. Golland, W.E.L. Grimson and R. Kikinis. Statistical Shape Analysis Using Fixed Topology Skeletons: Corpus Callosum Study. In *Proc. IPMI'99*, LNCS 1613:382-387, 1999.

[11] M. Kass, A. Witkin and D. Terzopoulos. Snakes: Active Contour Models. *IJCV*, 1(4):321-331, 1988.

[12] R. Kimmel *et al*. Skeletonization via Distance Maps and Level Sets. *CVIU: Comp. Vision and Image Understanding*, 62(3):382-391, 1995.

[13] F. Leymarie and M. D. Levine. Simulating the Grassfire Transform Using an Active Contour Model. In *IEEE Trans. PAMI*, 14(1):56-75, 1992.

[14] T.L. Liu, D. Geiger and R.V. Kohn. Representation and Self-Similarity of Shapes. In *Proc. ICCV'99*, 1129-1135, 1999.

[15] U. Montanari. Continuous Skeletons from Digitized Images. *J. Assoc. Comp. Machinery*, 16(4):534-549, 1969.

[16] R.L. Ogniewicz. Discrete Voronoi Skeletons. Hartung-Gore, 1993.

[17] R. Shahidi *et al*. Assessment of Several Virtual Endoscopy Techniques Using Computed Tomography and Perspective Volume Rendering. In *Proc. VBC'96*, LNCS 1131:521–526, 1996.

[18] C. Stauffer and W.E.L. Grimson. Adaptive Background Mixture Models for Real-Time Tracking. In *Proc. CVPR'99*, 246-252, 1999.