Multi-Scale Object Segmentation for Robust Recognition

Farzin Mokhtarian
Centre for Vision, Speech, and Signal Processing
Department of Electronic & Electrical Engineering
University of Surrey, Guildford, England GU2 5XH
Email: F.Mokhtarian@ee.surrey.ac.uk

Abstract

Effective local segmentation of contours is an important problem which arises in occluded object recognition as well as other areas. For any recognition system to perform successfully, the segmentation procedure used must be robust in presence of noise and local distortions of shape. Furthermore, it should be based on geometric invariants so that the segmentation will not be affected by arbitrary choices.

This paper proposes a new multi-scale segmentation routine for planar contours (and space curves) which is based on the curvature scale space (and torsion scale space) representations. Curvature (and torsion) zero-crossing segments extracted from a continuum of scales are utilized for robust segmentation of planar contours (and space curves). Curvature (and torsion) zero-crossing points are effectively tracked across increasing scales to ensure that the same segment is extracted only once. This approach is more robust than techniques which attempt to recover features/segments from a stable *scale and, as a result, risk over- or under-segmentation of the input contour.*

1 Introduction

Robust segmentation of a contour is an important problem which occurs in occluded object recognition among other areas. For such an object recognition system to perform reliably, the segmentation algorithm must be robust with respect to noise and local variations in object shape. A large amount of work has been carried out on contour segmentation. Much of the earlier work focused on polygon approximation of contour data [1]. Various criteria have been utilized to determine the number as well as the location of polygon vertices. A sequence of polygons with improving fit to the contour data can be obtained by gradually reducing the acceptable error of fit. This polygon sequence can be considered a multi-scale representation of the contour data. However, the locations of the chosen vertices tend to be arbitrary and as a result, this technique can not be considered robust with respect to noise and local shape distortions.

A newer class of algorithms segment an input contour by first smoothing it to remove some noise and then extracting the points where the absolute value of the curvature function has a local maximum. These local maxima are also referred to as *corners*. An attempt is then made to identify a *stable* scale at which the corresponding corners can be extracted [10, 11]. This group of algorithms is generally more robust than the polygon fitting algorithms described earlier. However an important problem arises since feature points are always extracted from a single scale whereas features can in general occur at multiple scales on free-form contours. As a result, the segmentation can still suffer from noise or from loss of structure due to over-smoothing.

This paper proposes a new multi-scale segmentation procedure for 2-D contours based on the curvature scale space (CSS) image, and for space curves based on the torsion scale space (TSS) image. The CSS/TSS image is an organization of curvature/torsion zero-crossing points extracted from an input contour at a continuum of scales. Section 2 presents a brief overview of the CSS technique and section 3 presents a short overview of the TSS method. Section 4 describes in detail the proposed multi-scale segmentation method for 2-D contours, and section 5 describes the adaptation of that multi-scale segmentation technique for space curves. The segmentation method described in section 4 was used in conjunction with an occluded object recognition system presented in [7]. A brief review of the system is given in section 6. Section 7 presents the results of the 2-D multi-scale segmentation technique applied to two contours as well as the results of the object recognition system based on the 2-D segmentation method. Section 8 contains the concluding remarks.

2 The CSS representation

A curvature scale space representation is a multi-scale organization of the invariant geometric features (curvature zero-crossing points and/or extrema) of a planar curve (here, only curvature zero-crossings were used). The CSS representation of a planar curve represents that curve uniquely modulo scaling and a rigid motion [5]. To compute it, the curve Γ is first parametrized by the arc length parameter u:

$$\Gamma(u) = (x(u), y(u)).$$

It is assumed that the input curve is initially represented by a polygon with possibly many vertices. An *evolved version* Γ_σ of Γ is defined by [3]:

$$\Gamma_\sigma = (\mathcal{X}(u,\sigma), \mathcal{Y}(u,\sigma))$$

where

$$\mathcal{X}(u,\sigma) = x(u) \otimes g(u,\sigma)$$

$$\mathcal{Y}(u,\sigma) = y(u) \otimes g(u,\sigma)$$

where $\otimes$ is the convolution operator and $g(u,\sigma)$ denotes a Gaussian of width σ. Note that σ is also referred to as the *scale* parameter. The process of generating evolved versions of Γ as σ increases from 0 to ∞ is referred to as the *evolution* of Γ. This technique is suitable for removing noise from a planar curve. Evolving contours can be considered an early form of active contours (snakes) [2] since they are similar in behaviour to snakes without any external constraints.

The CSS representation contains curvature zero-crossings or extrema extracted from evolved versions of the input curve. In order to find such points, one needs to compute curvature accurately and directly on an evolved version Γ_σ of a planar curve. It can be shown that curvature κ on Γ_σ is given by [8]:

$$\kappa(u,\sigma) = \frac{\mathcal{X}_u(u,\sigma)\mathcal{Y}_{uu}(u,\sigma) - \mathcal{X}_{uu}(u,\sigma)\mathcal{Y}_u(u,\sigma)}{(\mathcal{X}_u(u,\sigma)^2 + \mathcal{Y}_u(u,\sigma)^2)^{1.5}}$$

where

$$\mathcal{X}_u(u,\sigma) = \frac{\partial}{\partial u}(x(u) \otimes g(u,\sigma)) = x(u) \otimes g_u(u,\sigma)$$

$$\mathcal{X}_{uu}(u,\sigma) = \frac{\partial^2}{\partial u^2}(x(u) \otimes g(u,\sigma)) = x(u) \otimes g_{uu}(u,\sigma)$$

$$\mathcal{Y}_u(u,\sigma) = y(u) \otimes g_u(u,\sigma)$$

and

$$\mathcal{Y}_{uu}(u,\sigma) = y(u) \otimes g_{uu}(u,\sigma).$$

The function defined implicitly by $\kappa(u,\sigma) = 0$ is the CSS image of Γ. Note that as its name suggests, the

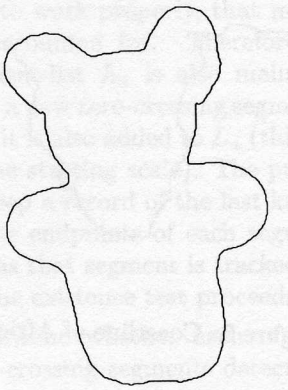

Figure 1: Outline of a bear

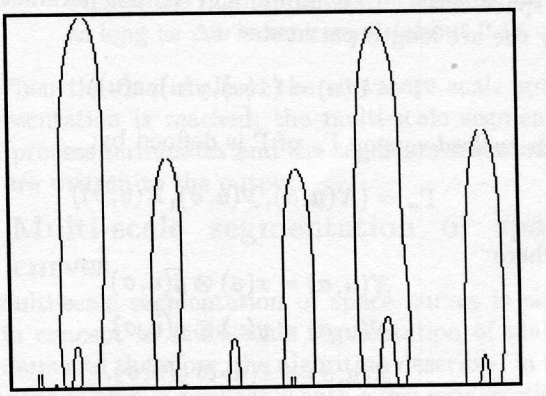

Figure 2: CSS representation of the Bear

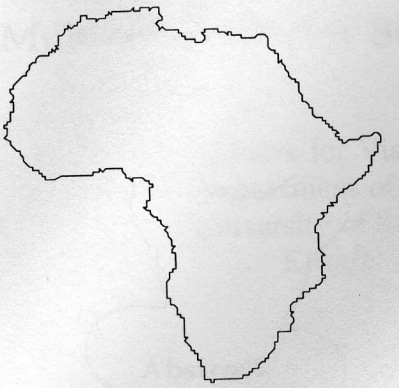

Figure 3: Coastline of Africa

CSS image is stored as a binary image in which each row corresponds to a specific value of σ and each column corresponds to a specific value of u. For all values of σ larger than a σ_c, evolved curves Γ_σ will be simple and convex. This suggests that the computation can stop as soon as σ_c is reached [9, 6].

Figure 1 shows the outline of a bear and figure 2 shows the CSS image of the bear. Figure 3 shows the coastline of Africa and figure 4 shows the CSS image of Africa. For further examples of CSS images, see [8].

3 The TSS Representation

The TSS representation is a multi-scale organization of torsion zero-crossing points and/or extrema of a space curve. To compute it, Γ is first parametrized by the arc length parameter u:

$$\Gamma = \Gamma(u) = (x(u), y(u), z(u))$$

An *evolved* version Γ_σ of Γ is defined by:

$$\Gamma_\sigma = (\mathcal{X}(u,\sigma), \mathcal{Y}(u,\sigma), \mathcal{Z}(u,\sigma))$$

where

$$\mathcal{X}(u,\sigma) = x(u) \otimes g(u,\sigma)$$
$$\mathcal{Y}(u,\sigma) = y(u) \otimes g(u,\sigma)$$
$$\mathcal{Z}(u,\sigma) = z(u) \otimes g(u,\sigma).$$

Torsion on evolved curve Γ_σ is given by:

$$\tau = \frac{\dot{\mathcal{X}}(\ddot{\mathcal{Y}}\breve{\mathcal{Z}} - \ddot{\mathcal{Z}}\breve{\mathcal{Y}}) - \dot{\mathcal{Y}}(\ddot{\mathcal{X}}\breve{\mathcal{Z}} - \ddot{\mathcal{Z}}\breve{\mathcal{X}}) + \dot{\mathcal{Z}}(\ddot{\mathcal{X}}\breve{\mathcal{Y}} - \ddot{\mathcal{Y}}\breve{\mathcal{X}})}{(\dot{\mathcal{Y}}\ddot{\mathcal{Z}} - \dot{\mathcal{Z}}\ddot{\mathcal{Y}})^2 + (\dot{\mathcal{Z}}\ddot{\mathcal{X}} - \dot{\mathcal{X}}\ddot{\mathcal{Z}})^2 + (\dot{\mathcal{X}}\ddot{\mathcal{Y}} - \dot{\mathcal{Y}}\ddot{\mathcal{X}})^2}$$

where $\breve{}$ denotes the third derivative. The first three derivatives of $\mathcal{X}(u,\sigma)$, $\mathcal{Y}(u,\sigma)$, and $\mathcal{Z}(u,\sigma)$ are computed as shown in the previous section.

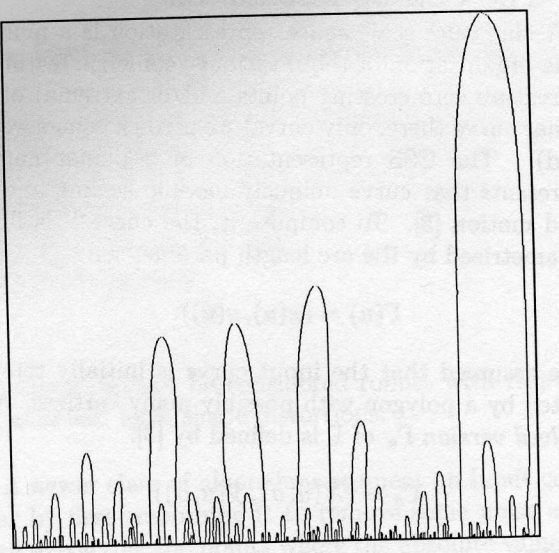

Figure 4: CSS representation of Africa

The function defined implicitly by $\tau(u,\sigma) = 0$ is the TSS image of Γ. For examples of evolution of a space curve and its TSS image, see [4].

4 Multi-scale segmentation of planar curves

The purpose of the multi-scale segmentation procedure described here is to divide a 2-D contour into basic segments to be used by a local matching algorithm for occluded object recognition. Curvature zero-crossing points are used as inherent feature points to segment the contour since their locations are invariant to rotation, uniform scaling, and translation of the contour. (Note, however, that corners (or maxima of the absolute value of curvature) are also inherent feature points which can be combined with curvature zero-crossing points, if necessary, to obtain a richer segmentation.) Since curvature zero-crossing points can be extracted from a contour at different scales, the choice of an appropriate scale to be used for feature detection is a crucial one. On free-form contours, features exist at different scales and therefore attempts to discover a *natural* scale to be used for feature extraction will be misguided. Some heuristics may be used to select a specific scale but if the scale chosen is too small, the segmentation will be affected by noise and local distortions of shape, and if it is too large, important structure on the contour may be lost.

The solution used here was motivated by the main

underlying concept of the curvature scale space representation: utilize information from multiple scales rather than prefer a single scale. The adaptation of that concept to the problem considered here necessitates the extraction of curvature zero-crossing segments from different scales. This process would ensure robustness to noise and local shape distortions as well as small features that may be part of the model object but missing from a similar input object. The multi-scale segmentation procedure is as follows:

- Choose a low scale to start the segmentation process. This scale should be selected such that most or all of the camera as well as discretization noise is smoothed out. Its value was experimentally determined: input contours were sampled at 300 points and the starting scale was $\sigma = 2.0$ with the filter covering 10σ sampled points.

- Locate all curvature zero-crossing points on the input contour at the starting scale (as described in section 2) and add all zero-crossing segments to the main segment-list L_m. Also add each such segment to the auxiliary segment-list L_a (to be explained shortly). A zero-crossing segment is defined as a segment on the input contour delimited by two consecutive curvature zero-crossing points referred to as its *endpoints*. Each endpoint is specified uniquely by the value of the arc-length parameter at that point which is relative to the starting point for arclength parametrization.

- Repeat the following steps until the evolving contour becomes simple and convex.

  - Determine the next higher scale for segment extraction. This can be achieved by adding a small increment $\Delta\sigma$ (about 0.1) to the previous value of σ.

  - Detect all curvature zero-crossing points at the current scale. Note that since new curvature zero-crossings will not be created at higher scales on a simple contour, zero-crossing detection can be accomplished more efficiently by *tracking* the zero-crossing points detected at the previous scale: the corresponding filter is convolved with the data only in a small neighborhood (two points on each side) of each existing zero-crossing point.

  - Add each of the zero-crossing segments detected at the current scale to L_m if it does not already exist in that list. In order to establish existence, two zero-crossing segments are considered to be the same if the difference between the arc-length values of their corresponding endpoints is small. Note however, that curvature zero-crossing points can move across scales and for the existence test to work properly, that movement must be accounted for. Therefore an auxiliary segment-list L_a is also maintained. Each time a new zero-crossing segment is added to L_m, it is also added to L_a (this also happens at the starting scale). The purpose of L_a is to keep a record of the last known locations of the endpoints of each segment added to L_m as that segment is tracked across scales. So the existence test proceeds as follows:

    - Determine whether each of the curvature zero-crossing segments detected at the current scale is the same as an existing segment in L_a. Let the segment under consideration be $S[e_1, e_2]$ with endpoints e_1 and e_2, and let $L_a = \{S_1, S_2, ..., S_n\}$ where S_i has endpoints S_{i1} and S_{i2}. Segment S is considered to already exist in L_a (and therefore L_m) if $|e_1 - e_{i1}| < j$ and $|e_2 - e_{i2}| < j$ for some $i \in [1, n]$. If this is the case, simply update the locations of the endpoints of S_i in L_a to be the same as those of segment S. Otherwise, segment S is a new segment: it is added to L_m as well as L_a. In both lists, its endpoints will be e_1 and e_2. Note that the value of j was chosen to be 3 which will work well even if the number of samples points varies as long as $\Delta\sigma$ is kept small (about 0.1).

When the final scale of the curvature scale space representation is reached, the multi-scale segmentation process terminates and the segments recorded in L_m are written to the output.

5 Multi-scale segmentation of space curves

Multi-scale segmentation of space curves is similar in concept to multi-scale segmentation of planar contours and therefore, the algorithm described in the previous section is applicable with a few modifications as described below:

- Torsion zero-crossing segments rather than curvature zero-crossing segments are used throughout the segmentation process.

- Torsion zero-crossing points can be created at higher scales in a TSS image. As a result, track-

ing of the zero-crossings is no longer applied: at each scale, the filter is convolved with the curve at every point.

- If torsion zero-crossing points are created at a higher scale, the corresponding segment will initially be very small. As a result, L_m now records the maximum length of each torsion zero-crossing segment: when the updated location of a torsion zero-crossing segment is recorded in L_a, its length is compared to the length of the corresponding segment in L_m. If the segment in L_a is longer, the segment in L_m is updated to be the same as the segment in L_a.

- The segmentation process terminates when the number of torsion zero-crossing points stabilizes at a low value.

6 A system for object recognition with occlusion

A robust system for object recognition with occlusion based on the multi-scale segmentation technique described in the previous section was presented in [7]. Note that only a brief summary of the multi-scale segmentation scheme presented in this paper appeared in [7]. That object recognition system consisted of the following stages:

6.1 Candidate generation and filtering

The multi-scale segmentation technique explained in the previous section was applied to several model and image contours in order to obtain a set of segments at multiple scales from each contour. All possible local matches need to be considered. Note however that object indexing [12] is employed to render the initial search more efficient. When considering a pair of model-image contour segments, the model contour segment is registered to the image contour segment in order to measure the average pointwise distance of the two segments. Iterative transformation parameter optimization is utilized to obtain the best registration. Candidates with relatively low distances are selected for the next stage.

6.2 Candidate merging

Initial candidates correspond to segments delimited by neighboring curvature zero-crossing points. However, it is possible for the visible boundary of an object in an input image to be divided into several neighboring or even overlapping segments. As a result, it is necessary to merge compatible candidates. Several criteria were utilized to measure compatibility such as both candidates corresponding to the same model and the resulting candidate also having a relatively low distance. Candidate merging is applied until no two candidates can be found which satisfy the merging criteria.

6.3 Candidate extention

In general, the intersection point of two object boundaries gives rise to a corner on the corresponding image contour. In order to to find the exact location of such intersection points, it is necessary to gradually extend the contour segments associated with the merged candidates as long as an acceptable fit between the image and model segments can be observed. The outcome of this stage is to extend each merged candidate to (or very close to) the intersection point at either end of the corresponding image contour segment.

6.4 Candidate grouping

The next step in the system is to group compatible but disjoint candidates. The criteria applied to determine compatability are the same as the criteria used in section 4.2 except that the candidates considered are no longer neighboring or overlapping. This stage is necessary since, due to occlusion, an object in the scene may appear as two or more disjoint components in the input image.

6.5 Candidate selection

The final stage is to select the *best* candidates using an appropriate criterion. As stated earlier the distance measure associated with each candidate is the average pointwise distance of the corresponding segments when registered. Candidate *support* is defined as the length of the image contour segment associated with the candidate. The *cost* of each candidate is defined as its distance measure divided by its support. This criterion will prefer candidates with low distance measures and large supports, since a lower cost candidate is considered a better candidate. So the candidate selection process consists of sorting the candidates according to their costs, selecting the lowest cost candidate, removing all image contour data corresponding to the chosen candidate, selecting the next lowest cost candidate (that remains valid: its corresponding image contour segment has not been removed), and repeating this process until no valid candidates remain. Note that this procedure is independent of the number of objects in the input image.

7 Results and discussion

This section demonstrates the result of the application of the multi-scale segmentation algorithm described earlier. Figure 5 shows all the curvature zero-crossing segments detected at the starting scale on the outline of the bear shown in figure 1. Figure 6 shows the additional curvature zero-crossing segments

detected at all higher scales. These curvature zero-crossing segments have been displayed by marking their endpoints on the *original* contour. Note that most segments added at higher scales will be convex segments. Concave segments can be added at higher scales only when a curvature zero-crossing contour that is nested inside another zero-crossing contour disappears. In particular, the segment added in figure 6(a) is concave, in figure 6(b) it is concave, in figure 6(c) it is convex, in figure 6(d) it is concave, and segments added in figures 6(e) and 6(f) are convex. This example demonstrates that the initial segmentation of the bear (shown in figure 5) is not satisfactory since parts of the object are oversegmented but that useful segments are added at higher scales. Starting the segmentation at a higher scale may have removed some useful segments as well as noise.

Figure 7 shows all the curvature zero-crossing segments detected at the starting scale on the coastline of Africa shown in figure 3. Figure 8 shows the additional curvature zero-crossing segments detected at all higher scales. Note that if two curvature zero-crossing segments are detected at different scales, then they are displayed on separate contours. Note that figures 8(a) and 8(c) show that two segments are added at each of the scales $\sigma = 2.3$ and $\sigma = 2.8$. As figures 7 and 8 show, most curvature zero-crossing segments are detected at the starting scale with a few added at higher scales. Again, most of the segments added at higher scales are convex, including the segments shown in figures 8(m) to 8(o). This example also demonstrates that multi-scale segmentation is immune to both noise and possible loss of useful structure in parts of the input contour due to over-smoothing.

The multi-scale contour segmentation algorithm demonstrated here was used to segment a number of model and image contours used as input to an object recognition with occlusion system. The system was tested using 18 model contours and seven input images. Figure 9 shows one of the images on which the system was tested. Figure 10 shows the outermost contour recovered from the input image and figure 11 shows the recognition result. Model objects have been registered with the image data. Only the outermost contour was used in order to demonstrate that the system could still recognize the objects correctly. The challenging nature of the recognition problem is highlighted by the fact that actual 3-D objects were used for the experiments which gave rise to some perspective distortions.

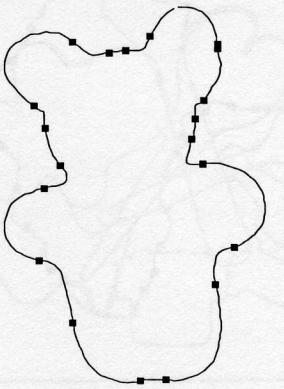

Figure 5: Curvature zero-crossing segments of Bear at the initial scale

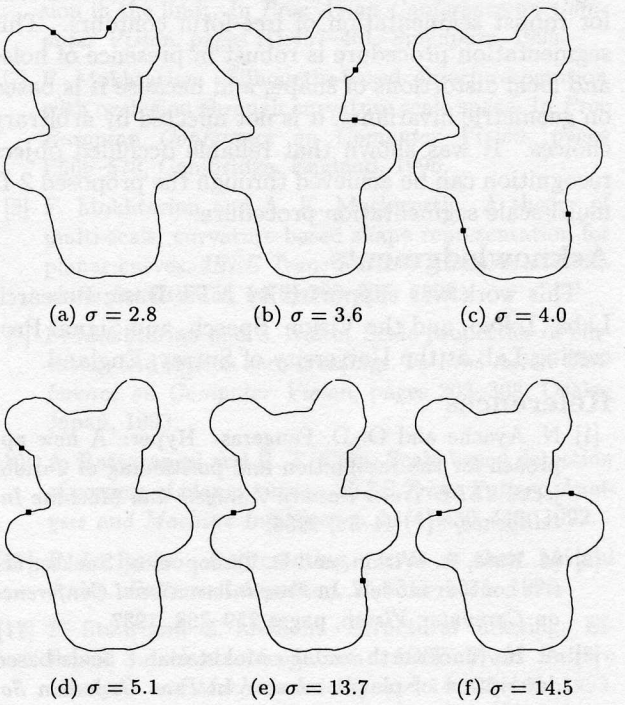

Figure 6: Curvature zero-crossing segments of Bear at higher scales

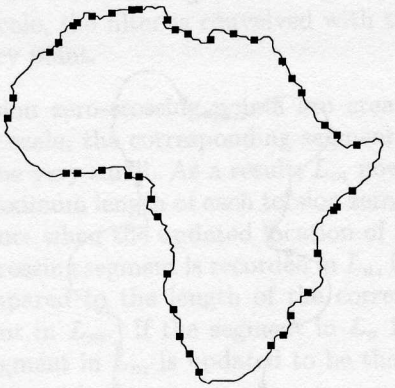

Figure 7: Curvature zero-crossing segments of Africa at the initial scale

8 Conclusions

This paper proposed a new multi-scale segmentation procedure for 2-D curves based on the CSS representation and for 3-D contours based on the TSS representation. Curvature (and torsion) zero-crossing segments were extracted from multiple scales and used for robust segmentation of free-form contours. This segmentation procedure is robust in presence of noise and local distortions of shape, and because it is based on geometric invariants, it is not affected by arbitrary choices. It was shown that reliable occluded object recognition can be achieved through the proposed 2-D multi-scale segmentation procedure.

Acknowledgments

This work was supported by NTT Basic Research Labs, Tokyo, and the Vision, Speech, and Signal Processing Lab at the University of Surrey, England.

References

[1] N. Ayache and O. D. Faugeras. Hyper: A new approach for the recognition and positioning of 2-d objects. *IEEE Trans Pattern Analysis and Machine Intelligence*, 8(1):44–54, 1986.

[2] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. In *Proc International Conference on Computer Vision*, pages 259–268, 1987.

[3] A. K. Mackworth and F. Mokhtarian. Scale-based description of planar curves. In *Proc Canadian Society for Computational Studies of Intelligence*, pages 114–119, London, Ontario, 1984.

[4] F. Mokhtarian. Multi-scale description of space curves and 3-d objects. In *Proc IEEE Conference on Computer Vision and Pattern Recognition*, pages 298–303, Ann Arbor, Michigan, 1988.

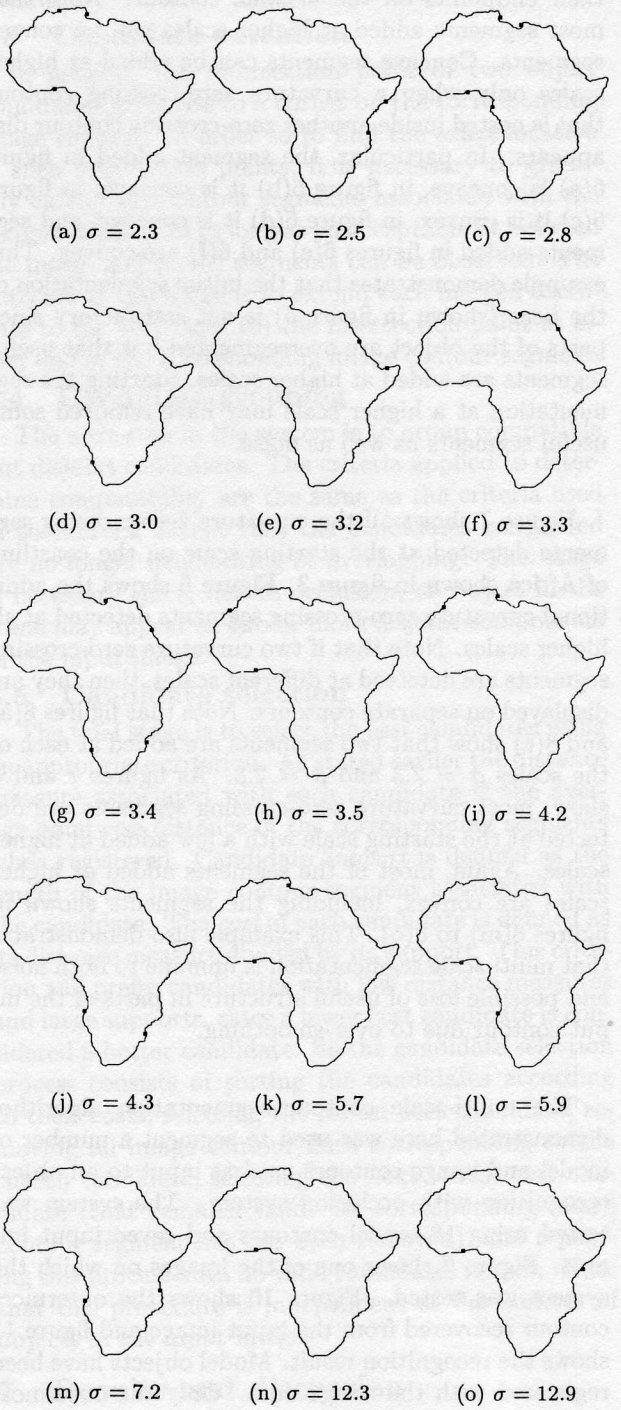

Figure 8: Curvature zero-crossing segments of Africa at higher scales

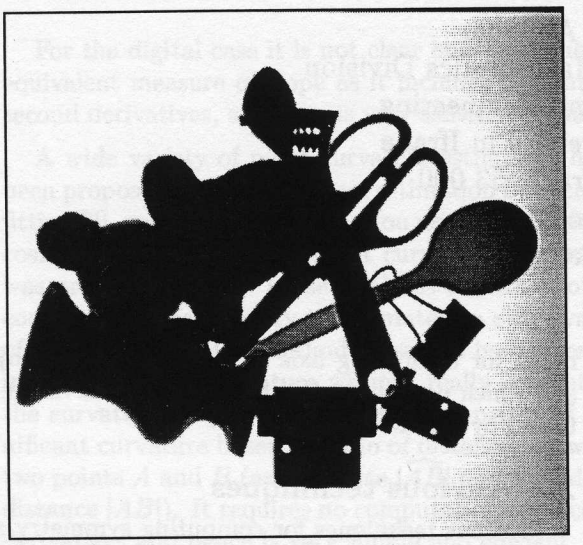

Figure 9: Input test image

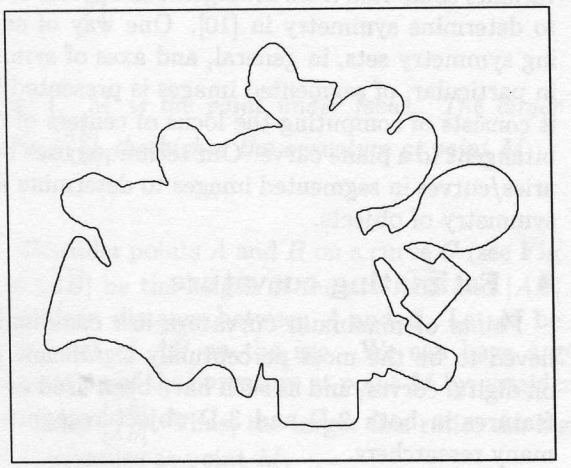

Figure 10: Outermost contour from input image

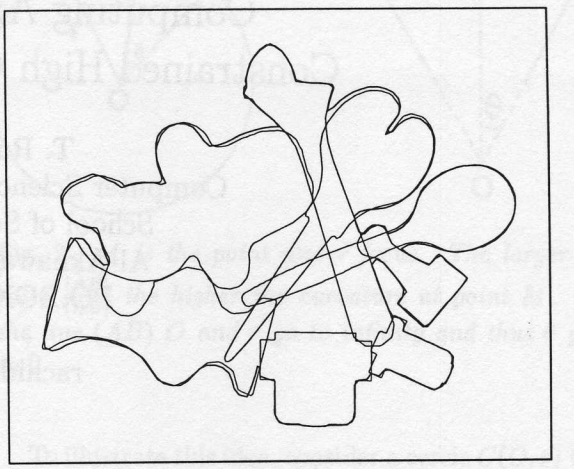

Figure 11: Recognition result

[5] F. Mokhtarian. Fingerprint theorems for curvature and torsion zero-crossings. In *Proc IEEE Conference on Computer Vision and Pattern Recognition*, pages 269–275, San Diego, CA, 1989.

[6] F. Mokhtarian. Zero-crossings of curvature and torsion in the limit. In *Proc Asian Conference on Computer Vision*, pages III:457–461, Singapore, 1995.

[7] F. Mokhtarian. Silhouette-based object recognition with occlusion through curvature scale space. In *Proc European Conference on Computer Vision*, pages I:566–578, Cambridge, England, 1996.

[8] F. Mokhtarian and A. K. Mackworth. A theory of multi-scale, curvature-based shape representation for planar curves. *IEEE Trans Pattern Analysis and Machine Intelligence*, 14(8):789–805, 1992.

[9] F. Mokhtarian and S. Naito. Scale properties of curvature and torsion zero-crossings. In *Proc Asian Conference on Computer Vision*, pages 303–308, Osaka, Japan, 1993.

[10] A. Rattarangsi and R. T. Chin. Scale-based detection of corners of planar curves. *IEEE Trans Pattern Analysis and Machine Intelligence*, 14(4):430–449, 1992.

[11] P. L. Rosin. Representing curves at their natural scales. *Pattern Recognition*, 25:1315–1325, 1992.

[12] F. Stein and G. Medioni. Structural indexing: Efficient 3-d object recognition. *IEEE Trans Pattern Analysis and Machine Intelligence*, 14:125–145, 1992.