

MULTIPLE-SCALE SEGMENTATION AND REPRESENTATION OF SOLID PLANE SHAPES

Ardeshir Goshtasby

Dept. of Computer Science, University of Kentucky

Abstract

A solid shape is a shape with possible holes, and therefore, not only does it possess an outer boundary but it may contain inner boundaries also. In this paper, a boundary contour is segmented at the points of peak curvature. Then, the obtained contour segments are approximated by parametric cubic curves in such a way that neighboring curves join smoothly. Representation of a contour at a lower scale is obtained by reducing the scale of the original shape and applying the same segmentation and representation process on the contour of the obtained shape.

1. Introduction

In problems related to image analysis, it is often required to represent shapes of regions in an image analytically for recognition, storage, and reconstruction purposes. Given the pixels in a shape, it is desired to find a representation that is fast in processing, economical in usage of storage, and accurate in reconstruction of the original shape.

We will be dealing with solid shapes in this paper. It is obvious, however, that pixels inside a shape between the outer boundary and the inner boundaries (holes) are insignificant. Because, if we can reconstruct the inner and outer boundaries of a shape, pixels between the boundaries become known automatically. Therefore, we will first concentrate on segmentation and representation of boundary contours and then consider how the inner and outer boundaries of a shape change as the scale of the shape changes.

To represent a contour analytically, we could segment the contour and replace each contour segment by a line or a curve segment. The problem then becomes: How should we segment the contour? Attneave¹ determined that peak curvature points in a contour carry most information about the contour and therefore, should be detected and preserved for later reconstruction. Yuille and Poggio² have also realized the importance of peak curvature points and have shown that it may be possible to reconstruct an original signal to within a constant factor if information about the peak curvature points of the signal at different scales are known.

Traditional contour segmentation techniques³⁻¹² do not guarantee preservation of the peak curvature points. In these techniques, starting at an arbitrary point on the contour and while following the contour, the error between the contour and the approximating line is measured. When the error reaches a bound, the contour is segmented at that point. Segmentation in this manner does not guarantee that a segmentation

point falls on a point of peak curvature and it may very well fall on a perfectly smooth area. A consequence of this would be that unnatural edges and vertices are obtained in the reconstructed contour. Other disadvantages of these techniques are: 1) dependency on the starting point, and 2) requirement of an error bound as the input data which depends on the scale of the contour and a novice user will have a hard time specifying it. In this paper, a contour is segmented at points of peak curvature. This preserves points with most information in the contour and results in a unique segmentation without dependency on the starting point.

Representation of boundary contours by line segments (polygons) works well as long as the underlying contours are polygonal. If the underlying contours are smooth and curved, the polygonal representation will perform poorly because a large number of line segments would be necessary to represent them. In the literature, circular arcs^{13,14} and conic sections¹⁵⁻¹⁷ have also been used for representing contours. These curves, however, cannot represent contour segments with inflections. In this paper, parametric cubic curves will be used to approximate contour segments. Parametric cubic curves are able to represent contour segments with inflections.

The organization of the rest of the paper is as follows. Section II discusses multiple-scale segmentation of boundary contours and Section III describes how to best fit a parametric cubic curve to a contour segment. Then Section IV measures the performance of the proposed segmentation and representation technique on reconstruction of solid plane shapes.

2. Segmentation

In this section, segmentation of a contour at the points of peak curvature is discussed. Peak curvature points in a contour are determined by computing the curvature values of points on the contour and locating the points that are locally maximum. Before doing so, however, we need to resolve one issue and that is: Do we want a shape representation that is sensitive to small details on the shape or that the small details are just the noise and we want to avoid them in our representation. In the following, both cases will be considered. We first describe a procedure that determines the peak curvature points of a contour in its highest scale (or finest resolution). Then we consider segmentation of the contour at a lower scale (or coarser resolution) such that small and noisy details in the contour are suppressed.

2.1. Segmentation in the highest scale

Let's consider a boundary contour that is constituted of n points P_1, P_2, \dots, P_n where P_{i+1} and P_{i-1} are neighbors of P_i (modulo n). If we look at point P_i in the contour, this point may be in any one of the four situations shown in Fig. 1. Each situation determines a curvature value. Therefore, if we encode each situation to a number between 1 and 4, we can represent curvature values on the contour by numbers between 1 and 4. A simple procedure that does this conversion is as follows. Start from an arbitrary point on the contour (say P_1) and shift a 3×3 window on the contour in such a way that the center of the window always lies on the contour and if we are interested in the curvature value at point P_i , then point P_i lies in the window center and P_{i-1} lies in the mid-bottom entry of the window, see Fig. 2. A curvature value between 1 and 4 is assigned to P_i depending on where P_{i+1} falls in the window, see Fig. 2. In this manner we can determine curvature values on the contour. Fig. 3.b shows determination of curvature values of contour of Fig. 3.a by this method. Once the curvature values are determined, detection of a peak curvature point is easy because we only need to determine the point that has a curvature value greater than curvature values of both of its neighbors.

Digitization of a curve could hide some of the peak curvature points. For example, consider the sequence of curvature values in a contour as: 112333211122111311. It is obvious that there are three peak curvature points in this sequence, two of them hidden. One of the hidden peaks is located somewhere in the sub-sequence 333 and the other is located in the sub-sequence 22. We put the peak curvature point for a hidden point at the middle of the sub-sequence. For example, in the above sequence, the peaks are set to be at locations shown by the asterisks in 1123*32111*2111*11. Peak curvature points of Figure 3.a are shown in Fig. 3.c.

2.2. Segmentation at a lower scale

The purpose of segmentation at a lower scale is twofold. One, we would like to have a representation which is not sensitive to small and noisy details in the shape and be able to represent general properties of the shape. Two, we would like to have the capability to represent and recognize a shape perceived at different scales (or in different resolutions). Consider the shape of an object in a scene. If the object moves towards the viewer, more details become visible and the shape will cover a larger area of the image and if

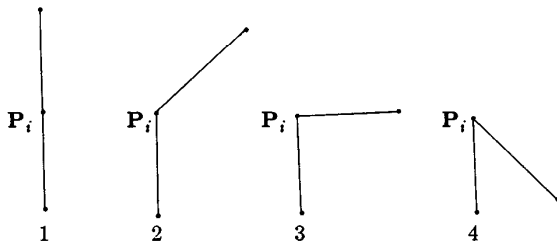


Fig. 1. Possible curvature values at a point P_i .

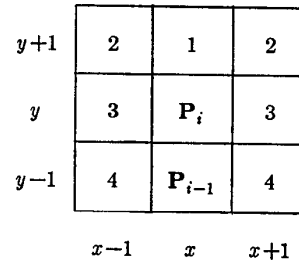


Fig. 2. Assignment of curvature value to point P_i depending on where P_{i+1} falls.

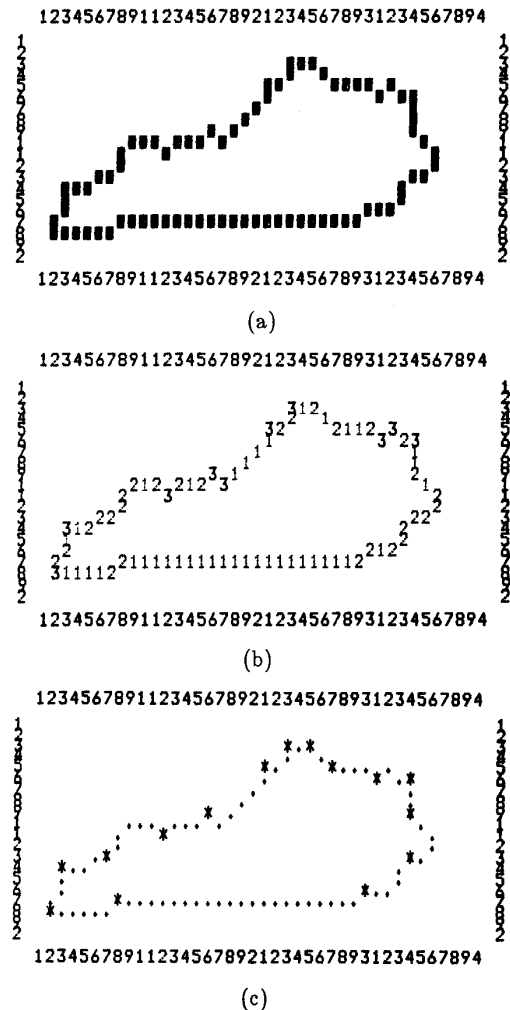


Fig. 3. (a) A contour, (b) its curvature values, and (c) its peak curvature points shown by the asterisks.

the object moves away from the viewer, it will cover a smaller area of the image and will lose some of its details. At some point, the object could completely change its shape and break into two or more parts. This shows that the boundary contour of a shape depends on the shape as a whole and we shouldn't treat it as an independent entity.

We expect that representation at a lower scale brings about the real shape of the object as if the object was perceived at a farther distance. In the past, lower scale representation has been achieved by specifying a smaller number of sides or a larger value for the error bound in the approximation of the polygons.³⁻¹² This will not fulfill any of the requirements for lower scale representation. First, the representation will still be sensitive to noisy details on the contour because the measurements are made on the original contour, and second, the obtained representation is not similar to the shape of the object when perceived at a farther distance. This is because the internal geometry of the shape is not considered in the scale reduction.

To obtain a representation at a lower scale, we first change the scale of the original shape and then determine a representation for it. To reduce the scale of a shape by a factor t , we transform the original shape to a new one by the following mapping function:

$$\begin{cases} X=tx \\ Y=ty \end{cases}$$

where (X,Y) and (x,y) are coordinates of corresponding shape points in the original and new coordinate

systems, respectively. To determine whether a point (x_1,y_1) in the new coordinate system belongs to the shape or not, we determine its corresponding point in the original coordinate system: $X=tx_1$ and $Y=ty_1$. Now, since a pixel in the new shape corresponds to a $t \times t$ block in the original shape, we examine the content of the $t \times t$ block centered at (tx_1,ty_1) in the original shape. If more than half of the pixels in the block belong to the shape, we let point (x_1,y_1) in the new coordinate system belong to the shape. Otherwise, we set the point as not belonging to the shape. This process, in effect, is smoothing the original shape and is reducing its scale. Fig. 4 shows reduction of the scale of a shape by factors of 3 and 5.

Once a shape has been reduced in scale, we can apply the segmentation procedure of Section II.A to it to segment it. Fig. 5 shows segmentation of boundary contours of shapes of Fig. 4. Note that by reducing the scale of a shape we have smoothed the shape and therefore have removed small and noisy details in the shape. Note also that we can do smoothing and scale reduction in this manner only when the underlying shapes are solid and bounded. Reducing the scale of a shape is like viewing the shape at a farther distance or in a lower resolution image.

Segmentation at a lower scale in this manner does not require specification of parameters such as the error bound or the number of required contour segments and the only parameter that the user has to specify is the reduction scaling factor.

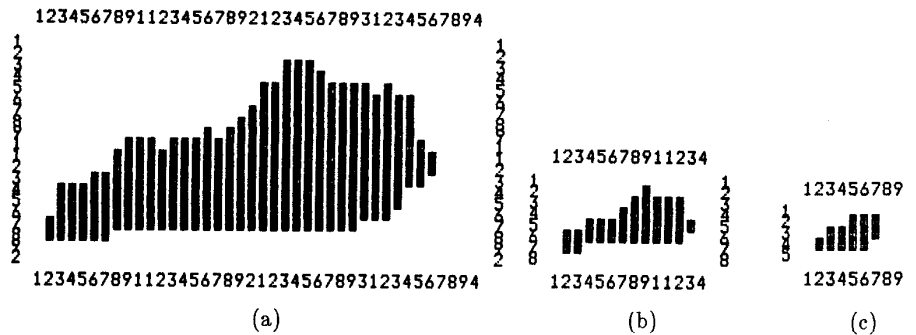


Fig. 4. Reducing the scale of a shape. (a) The original shape. (b) and (c) Shapes reduced in scale by factors of 3 and 5, respectively.

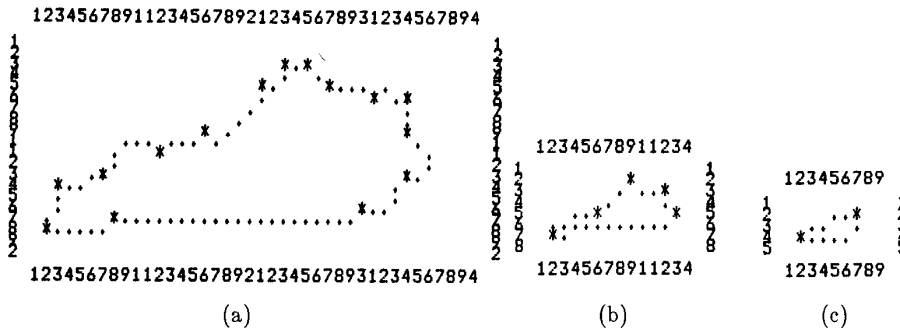


Fig. 5. Segmentation of boundary contours of shapes of Fig. 4.

Detection of the peak curvature points of a contour at different scales has been considered before.¹⁸⁻²¹ These techniques, however, use information from the original contour to determine the curvature values at lower scales without smoothing the original contour and this makes the obtained curvature values still dependent on noisy details on the contour.

3. Representation

Once a contour has been segmented, the next step is to approximate each segment by a line or curve segment. In the past, mostly line segments have been used to approximate contour segments.³⁻¹² Line segments, however, give the reconstructed contour a broken and unnatural appearance, especially when the original contour is smooth and curved. Circular arcs^{13,14} and conic sections¹⁵⁻¹⁷ have also been used to represent contour segments. These curves, however, being of degree two cannot represent contour segments with inflections. In this paper, parametric cubic curves will be used to represent contour segments because, these curves are able to represent contour segments with inflections.

A parametric cubic curve is defined by $P(s)=(x(s),y(s))$ where $x(s)$ and $y(s)$ are polynomials of degree three.²² There are many ways to express a polynomial of degree three. Most often it is expressed as a scaled sum of basis functions $1, s, s^2, s^3$, however, for our purposes we use the Hermite interpolant basis functions defined by,²²

$$\begin{aligned} h_1(s) &= 2s^3 - 3s^2 + 1 \\ h_2(s) &= -2s^3 + 3s^2 \\ h_3(s) &= s^3 - 2s^2 + s \\ h_4(s) &= s^3 - s^2 \end{aligned}$$

Using these basis functions, a parametric cubic curve that represents a contour segment with end points at P_i and P_{i+m} can be written as,

$$P(s) = h_1(s)P_i + h_2(s)P_{i+m} + h_3(s)D_i + h_4(s)D_{i+m} \quad (1)$$

where s is a parameter that changes between 0 and 1. Note that properties of the Hermite basis functions imply that when $s=0$ the curve is at point P_i and when $s=1$ the curve is at point P_{i+m} , and D_i and D_{i+m} are tangent vectors at points P_i and P_{i+m} , respectively. The tangent vector at a point P_i is calculated by,

$$D_i = \frac{1}{2}(P_{i+1} - P_{i-1}) = \frac{1}{2}[(P_{i+1} - P_i) + (P_i - P_{i-1})]$$

which is the average of chords $P_i - P_{i-1}$ and $P_{i+1} - P_i$ and can be obtained easily from points P_i and its two neighbors P_{i-1} and P_{i+1} on the contour. A tangent vector has direction and magnitude and if we let u_i and u_{i+m} be the unit tangent vectors and k_1 and k_2 be the tangent vector magnitudes at P_i and P_{i+m} , respectively, then we can replace D_i by k_1u_i and D_{i+m} by k_2u_{i+m} in (1) to obtain,

$$P(s) = h_1(s)P_i + h_2(s)P_{i+m} + h_3(s)k_1u_i + h_4(s)k_2u_{i+m} \quad (2)$$

The significance of the tangent vector magnitudes k_1 and k_2 is that by increasing or decreasing k_1 and k_2 appropriately we can change the shape and length of the curve without changing the direction of their

tangents at the end points. Fig. 6 depicts this fact.

This shows that we can keep the tangent vector directions of two curves that join at a point the same to ensure a smooth gradient at the joint but at the same time vary the magnitudes of the tangents at the two sides of the point separately, to change the shape and length of the curve. We will be determining k_1 and k_2 in such a way that the obtained curve lies as close as possible to the contour segment using the least-squares error criterion.

Considering the fact that there are $m-1$ points between P_i and P_{i+m} and if $(x(s),y(s))$ denotes points on the curve then,

$$\begin{aligned} x(s) &= h_1(s)x_i + h_2(s)x_{i+m} + h_3(s)k_1u_i + h_4(s)k_2u_{i+m}, \\ y(s) &= h_1(s)y_i + h_2(s)y_{i+m} + h_3(s)k_1u_i + h_4(s)k_2u_{i+m}, \end{aligned}$$

with u_i and u_{i+m} showing the x - and y -components of the unit tangent vector u_i and u_{i+m} , and u_i and u_{i+m} showing the x - and y -components of u_i and u_{i+m} , respectively, and if (x,y) denotes points on the contour segment, then the sum of squared errors between the curve and the contour segment is defined by,

$$E_2 = \sum_{j=i}^{i+m} \left\{ [x_j - x(s_j)]^2 + [y_j - y(s_j)]^2 \right\} \quad (3)$$

where $s_j = (j-i)/m$. We determine parameters k_1 and k_2 by minimizing this error quantity. To minimize E_2 with respect to k_1 and k_2 , we determine the partial derivatives of E_2 with respect to k_1 and k_2 and set them equal to zero. Doing so, we obtain a system of two linear equations in k_1 and k_2 as follows:

$$\begin{cases} Ak_1 + Bk_2 + C = 0 \\ Dk_1 + Ek_2 + F = 0 \end{cases} \quad (4)$$

where,

$$A = -(u_i^2 + u_{i+m}^2) \sum_j h_3(s_j)$$

$$B = -(u_{i+m}u_i + u_iu_{i+m}) \sum_j h_4(s_j)$$

$$\begin{aligned} C &= u_i \sum_j x_j - u_{i+m} \sum_j y_j - (u_{i+m}x_i + u_iy_{i+m}) \sum_j h_1(s_j) \\ &\quad - (u_ix_{i+m} + u_{i+m}y_i) \sum_j h_2(s_j) \end{aligned}$$

$$D = -(u_{i+m}u_i + u_iu_{i+m}) \sum_j h_3(s_j)$$

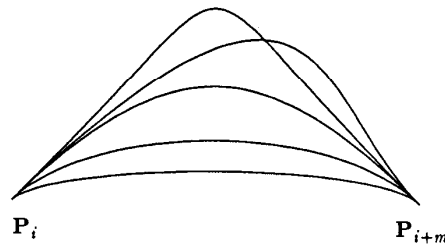


Fig. 6. Hermite parametric cubic curves having the same tangent vector directions but different magnitudes at the end points.

$$E = -(u_{i+m_x}^2 + u_{i+m_y}^2) \sum_j h_4^2(s_j)$$

$$F = u_{i+m_x} \sum_j^{k_4(s_j)} x_j + u_{i+m_y} \sum_j^{k_4(s_j)} y_j - (u_{i+m_x} x_i + u_{i+m_y} y_i) \sum_j h_1(s_j) h_4(s_j) - (u_{i+m_x} x_{i+m} + u_{i+m_y} y_{i+m}) \sum_j h_2(s_j) h_4(s_j)$$

From the system of equations of (4) we find

$$k_1 = \frac{BF - CE}{AE - BD}$$

$$k_2 = \frac{CD - AF}{AE - BD}$$

conditioned that $AE - BD \neq 0$.

Tangent vector magnitudes k_1 and k_2 determined in this manner make the Hermite cubic curve of (2) fit the contour segment $P_i P_{i+m}$ in such a way that the sum of squared errors defined by (3) is minimum. Fig. 7 shows least-squares fitting of Hermite cubics to contour segments of Fig. 5.

The above technique reconstructs the outer boundary of a shape. If a shape has inner boundaries too, we should reconstruct them in the same manner, to recover the whole solid shape.

Multiple-scale representation of contours has been studied by Mokhtarian and Mackworth²³ also. In their approach, a contour was convolved with different sized 1D Gaussian filters to obtain representation at multiple scales. It is worth mentioning that their representation is valid as long as the underlying contour can be considered as a one-dimensional structure. If the contour is reduced from a two-dimensional structure, for example from the boundary of an object in an image, then the multiple-scale representation of Mokhtarian and Mackworth would not truly simulate perception of a shape at multiple resolutions (or perception of an object at different distances to the viewer). The multiple-scale representation which was described in this paper, on the other hand, considers a contour as a boundary of a 2D shape, reduces the scale of the shape as a whole, and then takes the boundary of the reduced shape and represents it by Hermite cubics.

4. Performance Measure

To evaluate the performance of the proposed shape representation technique, we need to measure the speed, the accuracy, and the storage requirements of the technique.

Computation time for representation of a contour involves: 1) determination of the peak curvature points and 2) fitting of contour segments between consecutive peak curvature points by parametric cubic curves. Computational complexity of either of these steps is n , the number of points in the given contour. Therefore, the computational complexity of the technique is n .

Fitting parametric cubic curves to a digital contour involves error that can be measured by (3). For example, the root-mean-square error measured in representation of curves of Fig. 7 was 0.31, 0.32, and 0.45 units, where by a unit it is meant the length of a side of a pixel. Error is also obtained when reducing the scale of a shape. Each $t \times t$ block on the boundary of an original shape is mapped to a pixel on the boundary of the new shape. A block could have $t^2 + 1$ different values depending on whether all pixels in the block are 0, all are 1, or something in between. All these values are mapped to only two values of 0 and 1 in the new shape, causing loss of shape information. We shouldn't, however, count this kind of error as the representation error because a shape in a lower scale possesses less information than it did when in a higher scale.

The amount of memory required to store a representation for a contour depends on the number of peak curvature points in the contour. Assuming there are N peak curvature points in a contour, for each peak curvature point we need to store: 1) the unit tangent vector at that point, 2) the incoming and outgoing tangent vector magnitudes at the point, and 3) the peak curvature point itself. This would require six memory locations, and in total $6N$ memory locations would be needed to store information about the whole contour. We needed 90, 30, and 12 memory locations to store information about contours of Fig. 7. This is reduction in data by factors of 1.8, 5.3, and 13.3 compared to memory needed to store pixel locations in the original contour.

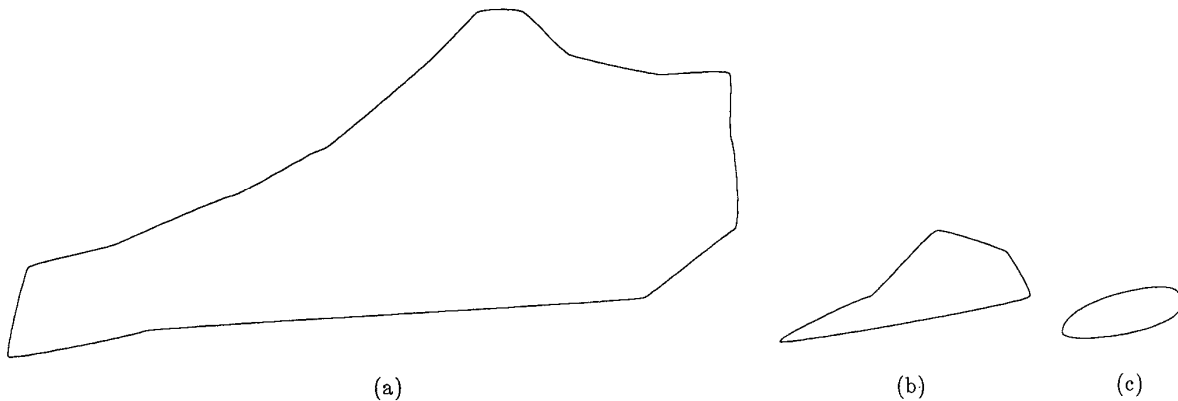


Fig. 7. Representation of boundary contours of Fig. 5 by Hermite cubic curves.

5. Conclusion

A technique for segmentation and representation of solid plane shapes was given. The technique is capable of representing a shape at multiple-scales as if the shape was viewed at different distances to the viewer. At the highest scale the representation is sensitive to small details in the shape while at lower scales the representation is insensitive to small details and rather it represents the general properties of the shape.

A solid shape is modeled by its (inner and outer) boundary contours. Previous studies¹ have shown that information in a contour is mostly concentrated at the peak curvature points of the contour. To preserve these points, a contour was segmented at the peak curvature points. Segmentation in this manner also has the advantage of providing a unique segmentation independent of the starting point and without user interface. After segmenting a contour, each contour segment was approximated by a Hermite parametric cubic curve. Parameters of the curve were determined by constraining the neighboring curves join smoothly and the obtained curve fit the contour segment by an error metric based on least-squares. Simple examples were given to exhibit different stages of the technique.

ACKNOWLEDGEMENT

The help of Jane Spanyer in preparation of this paper is greatly appreciated.

References

1. F. Attneave, "Some Informational Aspects of Visual Perception," *Psychological Review*, Vol. 61, No. 3, 1954, pp 183-193.
2. A. L. Yuille and T. Poggio, "Scaling Theorems for Zero-Crossings," *Proc. Workshop on Computer Vision: Representation and Control*, 1984, pp 3-7.
3. D. E. McClure, "Problems and Methods of Nonlinear Feature Generation in Pattern Analysis," *Eighth Annual Princeton Conference on Information Sciences and Systems*, 1974, pp 244-247.
4. T. Pavlidis, "The Use of Algorithms of Piecewise Approximations for Picture Processing Applications," *ACM Trans. Mathematical Software*, Vol. 2, No. 4, Dec. 1976, pp 305-321.
5. T. Pavlidis and S. L. Horowitz, "Segmentation of Plane Curves," *IEEE Trans. Computers*, Vol. C-23, No. 8, Aug. 1974, pp 860-870.
6. T. Pavlidis, "Optimal Piecewise Polynomial L_2 Approximation of Functions of One and Two Variables," *IEEE Trans. Computers*, Jan. 1975, pp 98-102.
7. T. Pavlidis, "Polygonal Approximations by Newton's Method," *IEEE Trans. Computers*, Vol. C-26, No. 8, Aug. 1977, pp 800-807.
8. U. Ramer, "An Iterative Procedure for the Polygonal Approximation of Plane Curves," *Computer Graphics and Image Processing*, Vol. 1, 1972, pp 244-256.
9. J. Sklansky and V. Gonzalez, "Fast Polygonal Approximation of Digitized Curves," *Pattern Recognition*, Vol. 12, 1980, pp 327-331.
10. I. Tomek, "Two Algorithms for Piecewise-Linear Continuous Approximation of Functions of One Variable," *IEEE Trans. Computers*, April 1974, pp 445-447.
11. C. M. Williams, "An Efficient Algorithm for the Piecewise Linear Approximation of Planar Curves," *Computer Graphics and Image Processing*, Vol. 8, 1978, pp 286-293.
12. R. L. Kashyap and B. J. Oommen. "Scale Preserving Smoothing of Polygons," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, No. 6, Nov. 1983, pp 667-671.
13. T. Pavlidis, "Curve Fitting as a Pattern Recognition Problem," *Sixth Int. Conf. Pattern Recognition*, 1982, pp 853-859.
14. H. Asada and M. Brady, "The Curvature Primal Sketch," *Proc. Workshop Computer Vision: Representation and Control*, 1984, pp 9-14.
15. Y. Liao, "A Two-Stage Method of Fitting Conic Arcs and Straight-line Segments to Digitized Contours," *IEEE Proc. Pattern Recognition and Image Processing*, 1981, pp 224-229.
16. T. Pavlidis, "Curve Fitting with Conic Splines," *ACM Trans. Graphics*, Vol. 2, No. 1, Jan. 1983, pp 1-31.
17. F. L. Bookstein, "Fitting Conic Sections to Scattered Data," *Computer Graphics and Image Processing*, Vol. 9, 1979, pp 56-71.
18. A. Rosenfeld and E. Johnston, "Angle Detection on Digital Curves," *IEEE Trans. Computers*, Sept. 1973, pp 875-878.
19. A. Rosenfeld and J. S. Weszka, "An Improved Method of Angle Detection on Digital Curves," *IEEE Trans. Computers*, Sept. 1975, pp 940-941.
20. L. S. Davis, "Understanding Shape: Angles and Sides," *IEEE Trans. Computers*, Vol. C-26, No. 3, March 1977, pp 236-242.
21. H. Freeman and L. S. Davis, "A Corner-Finding Algorithm for Chain-Coded Curves," *IEEE Trans. Computers*, March 1977, pp 297-303.
22. M. E. Mortenson, *Geometric Modeling*, John Wiley & Sons, New York, 1985, pp 34-37.
23. F. Mokhtarian and A. Mackworth, "Scale-Based Description and Recognition of Planar Curves and Two-Dimensional Shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, No. 1, 1986, pp 34-43.