COMPUTER GRAPHICS AND IMAGE PROCESSING (1972) 1 (244–256)

An Iterative Procedure for the Polygonal Approximation of Plane Curves

URS RAMER
New York University School of Engineering and Science,
Department of Electrical Engineering and Computer Science,
University Heights, Bronx, New York 10453

Communicated by H. Freeman

Received August 20, 1972

The approximation of arbitrary two-dimensional curves by polygons is an important technique in image processing. For many applications, the apparent ideal procedure is to represent lines and boundaries by means of polygons with minimum number of vertices and satisfying a given fit criterion. In this paper, an approximation algorithm is presented which uses an iterative method to produce polygons with a small—but not minimum—number of vertices that lie on the given curve. The maximum distance of the curve from the approximating polygon is chosen as the fit criterion. The results obtained justify the abandonment of the minimum-vertices criterion which is computationally much more expensive.

1. INTRODUCTION

The efficient representation of irregular curves is an important problem in picture processing. In general, digitized pictures are generated in sampling the gray-value function over the domain of an image on a regular grid, usually on a square grid. Picture processing algorithms are then utilized for finding the boundaries or lines in the pictures. The resulting curves can be interpreted as polygons whose vertices lie on sampling points or whose edges coincide with cell boundaries [1]. The lengths of these polygon edges then are equal to the distances between two neighboring sampling points or equal to the length of one cell edge respectively. A natural and economic way of representing such curves generated on a uniform rectangular grid is Freeman's chain code [2,3].

This type of polygonal curve representation exhibits two important disadvantages. First the polygons contain a very large number of edges and, therefore, are not in as compact a form as possible. Second, the length of the edges is comparable in size to the noise introduced by quantization. This results in local concavities for polygons that represent locally convex curves.

Sklansky has written several papers on the subject of smoothing quantized contours [4,5,6]. He uses a minimum-perimeter polygon to describe the boundary of cellular complexes. These polygons smooth the digitization noise and conserve convexity for a large class of objects. Montanari has described an algorithm for finding minimum-length polygons with various minimum length properties [7]. Each boundary point is allowed to move within an uncertainty domain in order to obtain collinearity with other boundary points, and the polygon is found using nonlinear programming. Zahn [8] uses the deletion of local points of inflection to smooth a digitized boundary without destroying its features.

These approaches show interesting results as far as smoothing is concerned. However, often one is less concerned with smoothing than with simplifying the representation of a curve. What are wanted are polygons with few edges but which still retain the significant features of the curves they represent. The approaches mentioned above generally result in polygons with too many edges for this purpose.

Stone [9], Bellman [10], and Gluss [11,12] were concerned with the approximation of real, single-valued, continuous functions of one parameter by polygons with N segments. The criterion of best least-square fit was used for the treatment of three cases: (a) requiring the vertices of the polygons to lie on the curve, (b) requiring the vertices to lie on points of subdivisions of the curve, and (c) without constraints for the position of the vertices. These results cannot be applied to image processing since they deal with curves progressing in one direction only.

In a recent paper Jarvis [13] has demonstrated a method for fitting polygons with small numbers of edges to boundary lines while still preserving the overall features of the boundaries. A least-square fit is used under the constraint that the vertices of the polygons lie on a set of rays originating at the center of mass. An initial set of rays is found from peaks and troughs of the polar coordinates of the boundary relative to the center of mass. Additional rays are found iteratively at directions of maximum error until the fit criterion is satisfied. His polygons have few edges and describe the shape of nerve fibers usefully, but his method is not well suited for complex boundaries, since the computation time rises rapidly with increasing complexity of the boundary.

We shall now describe an algorithm that produces a polygon with a small number of edges for arbitrary two-dimensional digitized curves. The maximum distance of the curve points from the polygon is used as the fit criterion in a simple iterative approximation. A curve segment is approximated by a straight-line segment connecting its initium and terminus. If the fit is not acceptable the curve segment is broken into two segments at the curve point most distant from the straight-line segment. This segmentation is repeated until each curve segment can be approximated by a straight-line segment through its endpoints. The termini of all these curve segments then are the vertices of a polygon that satisfies the given maximum-distance approximation criterion.

2. THE SELECTION OF POLYGON VERTICES

An arbitrary two-dimensional open or closed curve can be represented digitally by an ordered set C of $N+1$ consecutive points along the curve (first and last points coincide for a closed curve). The points of the set C can be interpreted as the vertices p_i of an open or closed polygon P with N edges. A polygon P' with a reduced number N' of edges, whose vertices p'_k coincide with vertices of P, then corresponds to an ordered subset C' of points such that $C' \subseteq C$. The problem of interest is to find a subset C' of the set C such that the polygon P' with vertices $p'_k \in C'$ approximates the curve in a way suitable for a particular application. A desirable property for the subset C' is that it be as small as possible. This is important for the efficient execution of many algorithms and eases description of a curve.

The set C' of vertices p'_k divides the set C into ordered subsets S_k of the following form:

$$S_k = \{p_i, p_{i+1}, \ldots, p_j\};\ p_i = p'_{k-1},\ p_j = p'_k$$

where p_i, p_{i+1} are consecutive vertices in polygon P, $i = 0, 1, \ldots, N$ and p'_{k-1}, p'_k are consecutive vertices in polygon P', $k = 1, 2, \ldots, N$. The set C' also partitions the curve into curve segments, and S_k contains the points belonging to the k-th curve segment.

The following constraints hold among C, C', and S_k:

$$\bigcup_{k=1,N'} S_k = C$$

$$\Big(\bigcup_{k=1,N'-1} (S_k \cap S_{k+1})\Big) \cup \{p_o\} \cup \{p_N\} = C' \text{(for open polygons)},$$

$$\Big(\bigcup_{k=1,N'-1} (S_k \cap S_{k+1})\Big) \cup (S_{N'} \cap S_1) = C' \text{(for closed polygons)}.$$

Constraints for the generation of the polygon P' now can be embedded conveniently into conditions for the subsets S_k and C'. Conditions on the subsets S_k correspond to conditions for the approximation of a curve segment by a straight-line segment, conditions on C' can be regarded as global properties.

Reggiori [14] has described various criteria for the approximation of smooth, inflection-free curve segments by polygons while still conserving the overall features of the curves. Some of these are based on the following:

(1) maximum absolute deviation,
(2) mean-square deviation,
(3) absolute area between curve segment and approximating line segment.

If continuous curves contain points of inflection, the question of convexity and concavity arises. Unfortunately, in the process of digitizing a curve, local concavities and convexities are introduced which do not correspond to convexities and concavities in the original curve. There is a large class of problems in which minor concavities and convexities are unimportant. For such problems it is sufficient to approximate a curve according to some of the above criteria such that only major concavities and convexities are retained in the approximating polygon. Here we are not so much concerned with the approximation criteria themselves, but rather with the manner in which one partitions a digitized curve, such that each curve segment can be approximated by an edge of a polygon satisfying these criteria.

In general we are interested in the following problem:

Given a polygon with a set C of vertices p_i, $0 \leqslant i \leqslant N$, find a subset $C' \subseteq C$ of vertices with minimum number of elements satisfying a specified criterion function

$$f(S_k) \leqslant \alpha,$$

where $\alpha = \text{const}$.

In particular we are interested in the case where the criterion function represents a maximum distance, as follows:

$$f(S_k) = \max \, (\text{distance } (p_i \langle \overline{p'_{k-1} p'_k} \rangle) \leqslant \alpha,$$

where $p_i \in S_k$ and $\overline{p'_{k-1}\, p'_k}$ represents the straight-line segment from p'_{k-1} to p'_k.

The general solution of this problem with an unknown number of inequalities is complicated. Solutions could be found by graph searching techniques [15], but the cost would be very high. Also there is not, in general, a unique solution. It is, therefore, reasonable to simplify the problem by relaxing the minimum condition on the set C' and only ask that the number of vertices in C' be close to the minimum. A solution (naturally not a unique one) can then be found by iteration. In such an iteration the set C is divided into two subsets. Each subset is tested against the maximum-distance criterion. If the criterion is fulfilled for a subset, the subset is retained; otherwise the subset is divided and the resulting subsets are tested. This iteration continues until all subsets satisfy the maximum-distance criterion. The generated subsets can be interpreted as the nodes of a binary tree (see Fig. 2), where the leaves of the tree are the sought subsets S_k. To find the subsets in correct order the tree has to be searched in a depth-first fashion, which conveniently can be done using a first-in, last-out stack.

Various methods can be used to select the point where to divide a subset into two subsets. The division at the point of maximum distance from the approximating straight line results in a low number of vertices and very little extra computation. An algorithm is described in Fig. 1 which uses the maximum-distance criterion and divides subsets at the points of maximum distance.

The initial division of the set C of vertices of the original polygon P into two subsets must be different for open and closed polygons. For an open polygon the endpoints of the reduced polygon will be the same as those of the original polygon. Therefore, the given curve can be treated like any other curve segment and the set C like any other subset. However, for a closed polygon there are no vertices that are obvious candidates for endpoints. Any two distinct vertices could be selected arbitrarily for the initial division into subsets. For the best choice one should select two oppositely located extremal points, because there is a high probability that the algorithm would select these eventually anyway. The highest left-most point and the lowest right-most point are chosen for these extremal points by the algorithm of Fig. 1.

During the computation two push-down stacks are maintained for the vertices of the reduced polygon. The OPEN stack contains the found vertices

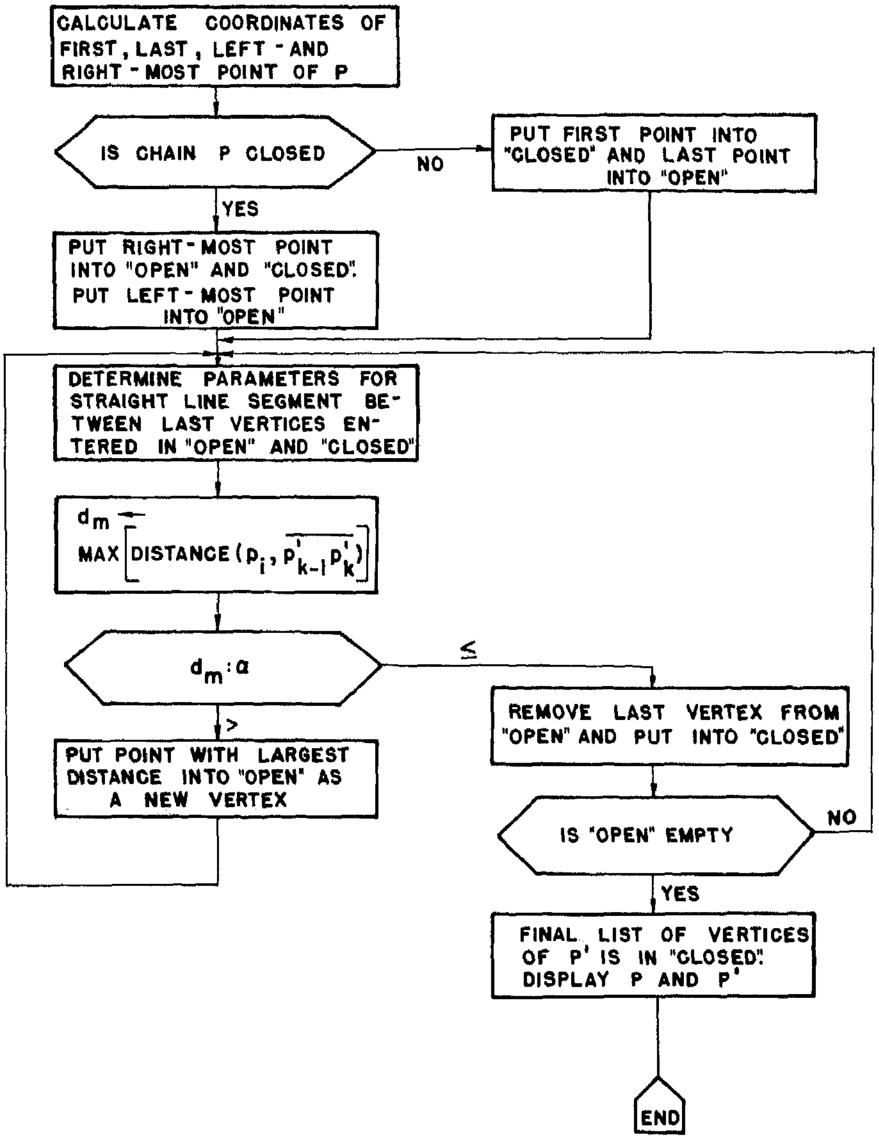
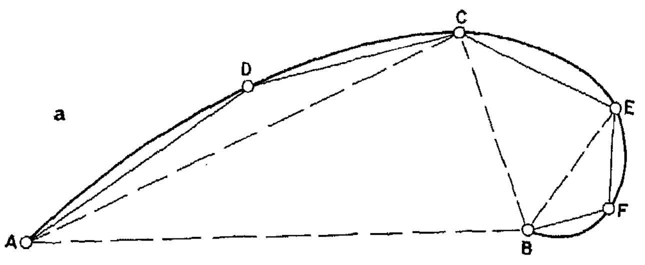

CALCULATE COORDINATES OF FIRST, LAST, LEFT - AND RIGHT - MOST POINT OF P
IS CHAIN P CLOSED
NO
PUT FIRST POINT INTO "CLOSED" AND LAST POINT INTO "OPEN"
YES
PUT RIGHT-MOST POINT INTO "OPEN" AND "CLOSED". PUT LEFT-MOST POINT INTO "OPEN"
DETERMINE PARAMETERS FOR STRAIGHT LINE SEGMENT BETWEEN LAST VERTICES ENTERED IN "OPEN" AND "CLOSED"
$d_m \leftarrow \max[\mathrm{DISTANCE}(p_i, \overline{p'_{k-1}p'_k})]$
$d_m : a$
$\leq$
$>$
PUT POINT WITH LARGEST DISTANCE INTO "OPEN" AS A NEW VERTEX
REMOVE LAST VERTEX FROM "OPEN" AND PUT INTO "CLOSED"
IS "OPEN" EMPTY
NO
YES
FINAL LIST OF VERTICES OF P' IS IN "CLOSED". DISPLAY P AND P'
END

FIG. 1. Polygon Generation Flow Diagram.

whose position in the sequence of vertices is not yet known (i.e., the endpoints of the subsets corresponding to nonterminal nodes in the tree to be searched). The CLOSED stack contains the vertices in sequence (i.e., the starting vertex and the endpoints of subsets corresponding to terminal nodes). For each vertex the coordinates and the location of the corresponding chain element are stored. An example is given in Fig. 2.



CLOSED	OPEN	Considered curve segment	Generated vertex
A	B	(AB)	C
A	B,C	(AC)	D
A	B,C,D	(AD)	–
A,D	B,C	(DC)	–
A,D,C	B	(CB)	E
A,D,C	B,E	(CE)	–
A,D,C,E	B	(EB)	F
A,D,C,E	B,F	(EF)	–
A,D,C,E,F	B	(FB)	–
A,D,C,E,F,B			

b

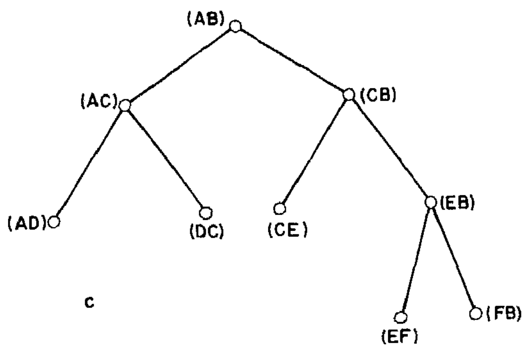


FIG. 2. Example of polygon generation: (a) curve with approximating straight-line segments (intermediate and final); (b) vertex generation sequence; (c) tree generated for Fig. 2a.

3. COMPUTATIONAL EFFICIENCY

Crucial for the speed of the algorithm is the calculation of the distance of a curve point from its associated line segment. Fortuitously the Euclidean distance need only be calculated for a small set of points. To find the point of maximum Euclidean distance it is sufficient to calculate the distance in the direction of one of the coordinate axis.

Consider the curve segment (AB) in Fig. 3. The most distant point in the Euclidean sense is found where the tangent to the curve is parallel to the straight-line segment $\overline{AB}$ (and does not intersect the curve between A and B). The difference between $\overline{AB}$ and points on (AB) in the direction of the y axis is Δy:

$$\Delta y = d/\cos\theta \quad ; \quad \cos\theta \neq 0,$$

where θ is the angle between $\overline{AB}$ and the x axis, and d is the Euclidean distance from $\overline{AB}$. For d_m it is sufficient to find the maximum Δy. The maximum Euclidean distance then is given by:

$$d_m = \Delta y_{\max} \cdot \cos\theta.$$

To prevent $\cos\theta$ from becoming too small, the distance in the direction of the x axis is used when $|\tan\theta| > 1$.

To estimate the time needed for the calculation, assume that the generated binary tree of curve segments is completely balanced. Then N' is a power of two and the number of distance calculations for curve points is as follows:

N'	h
1	$N-1$
2	$(N-1)+(N-2)$
4	$(N-1)+(N-2)+(N-4)$.

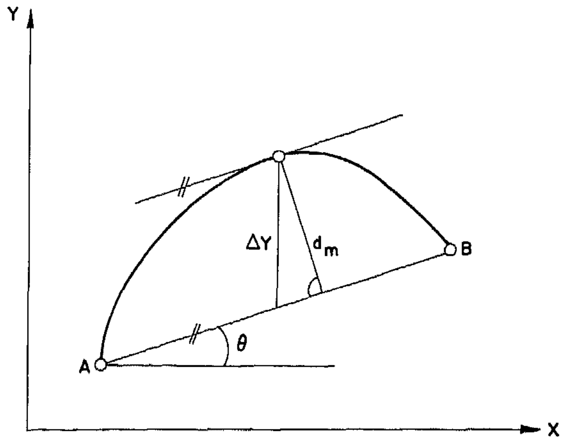


FIG. 3. Simplified distance calculation.

This leads to the following formula for h:

$$h = N \cdot (1 + \log_2 N') - 2N' + 1,$$

where h is the number of distance calculations.

Note that the above formula reflects the fact that the distances of the end-points of curve segments from the approximating straight-line segments are known to be zero. In unbalanced trees the number h can be larger or smaller, depending on what level the leaves corresponding to large curve segments can be found. The bounds on h are:

$$N - 1 \leqslant h \leqslant (N - 1) \cdot N'.$$

Experience with the algorithm shows that the formula for the balanced tree is a good approximation to the number of calculations needed for arbitrary curves.

4. SOME RESULTS AND APPLICATIONS

The algorithm was implemented in FORTRAN II on an ADAGE AGT 30 Graphics Terminal. This computer has a memory cycle time of 2μsec, no floating-point hardware, and no index registers. No particular effort was made to optimize the speed of the program. The distance calculation, for instance, uses floating-point operations although it could be programmed using integer arithmetic only, while still retaining the necessary precision. The manually measured processing times are given in Table I.

The curves used as examples were manually chain encoded and the resulting chains were used as input. The chain and the resulting polygon were displayed together. The polygons result in approximations comparable to those obtained by Jarvis [13] but the processing time is considerably shorter on a computer with similar speed. The algorithm is very robust and deals even with self-intersecting curves (Fig. 4).

The maple leaf in Fig. 5 is a figure with sharp features. For this kind of shape the algorithm is ideally suited because it tends to enhance concavities

TABLE I
COMPARISON OF EXAMPLES

Figure	Number of chain elements in P	Tolerance α in gridpoints	No. of vertices in polygon P'	Processing time (sec)
4	45	1.0	10	1
5a	120	1.0	21	2.5
5b	120	2.0	17	2.5
6a	328	1.0	29	3.5
6b	328	2.0	18	3.0
6c	328	4.0	12	2.5
7a	730	5.0	21	6
7b	730	3.0	34	8

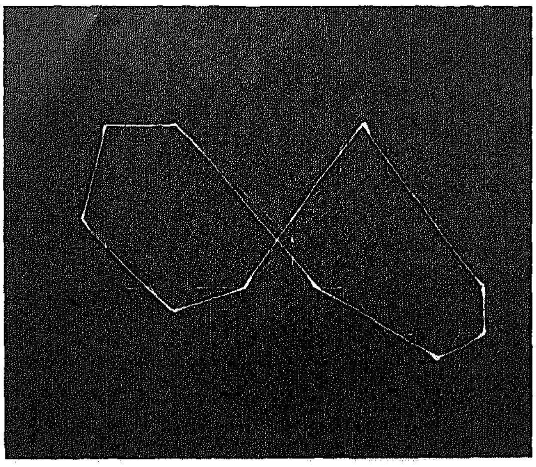


FIG. 4. Self-intersecting curve.

and convexities. Note the small difference between the polygons for a tolerance of 1.0 and 2.0.

Figure 6 shows the nucleus of a white blood cell, based on data taken from a paper by Ledley and Cheng [16]. They were interested in classifying cells according to significant concavities and convexities. The polygons obtained here are very suitable for this purpose. The selection of the tolerance is important. If the tolerance is selected too small, minor features are not discarded. If the tolerance is selected too large, important features may be lost. Therefore, the selection of the tolerance has to consider the noise in the class of pictures to be processed and the size of the features to be retained.

Polygons play an important role in the generalization of maps. In reducing the size of a map one is interested in discarding details. Figure 7 shows the approximation of the outline of the Seward Peninsula (data taken from Tobler [17]). In a further step this polygon could be edited, for instance, by replacing pairs of edges enclosing a small angle by single edges.

For curves with a large number of points the selection of the first few vertices takes a relatively long time because many distances must be calculated. This can be circumvented by the following modification of the algorithm. Apply the algorithm to the first, say, m points of the curve, thereby generating a number of vertices for P'. Discard the last two vertices thus obtained and apply the algorithm to the next m curve points starting from the last retained vertex. Repeat this procedure until the entire curve has been processed. This overlapping use of the algorithm allows arbitrary segmentation of the curve without severely affecting the selection of the vertices of P'.

5. EXTENSION TO n DIMENSIONS

The algorithm need not be restricted to planar curves. Except for a minor adjustment in the selection of the two extremal points for the initial subdivision of a closed curve into two subsets, the algorithm in Fig. 1 can be applied to arbitrary curves in a n-dimensional space. Naturally, the calcula-

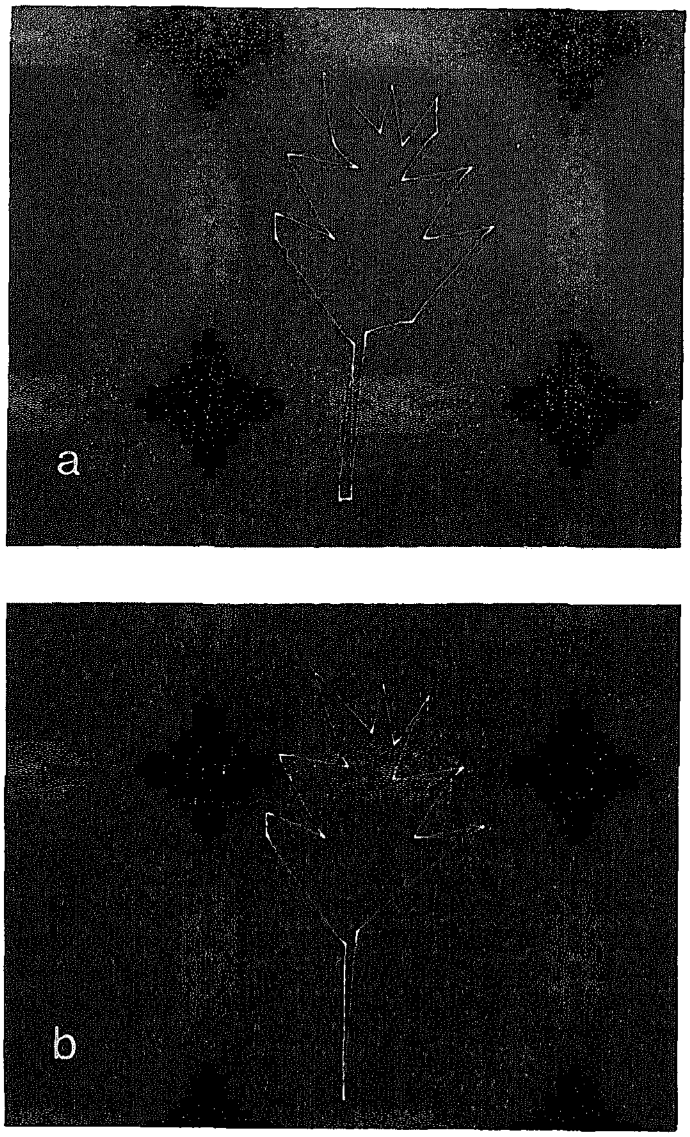


FIG. 5. Maple leaf: (a) tolerance 1.0; (b) tolerance 2.0.

tions of the distance of a curve point from the approximating straight-line segment then is more complicated than in two dimensions. Of major importance is the extension to curves in three-dimensional space, which can be conveniently represented in digital form using three-dimensional chain code [18].

6. CONCLUSION

An algorithm has been described for the approximation of arbitrary, digitized two-dimensional curves by polygons with a small number of edges. The

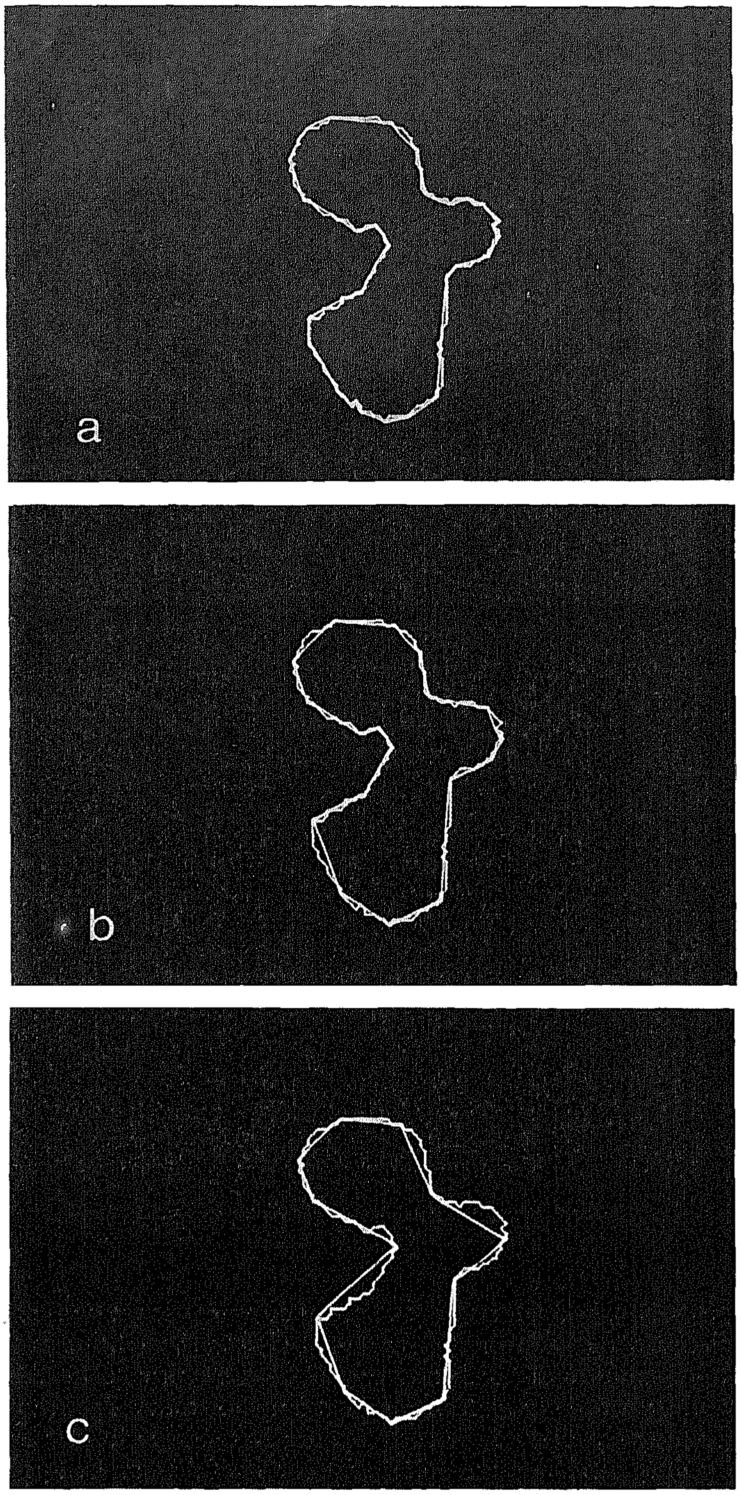


FIG. 6. Multi-lobed nucleus: (a) tolerance 1.0; (b) tolerance 2.0; (c) tolerance 4.0.

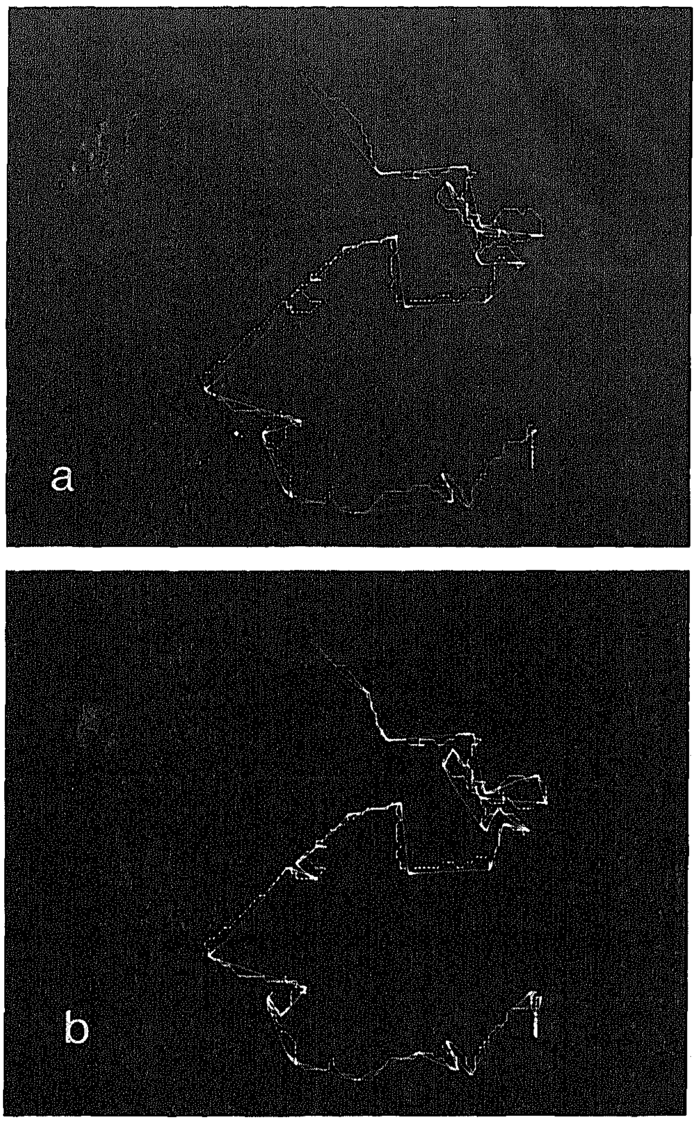


FIG. 7. Coast-line of seward peninsula: (a) tolerance 5.0; (b) tolerance 3.0.

approach uses iterative selection of a subset of points as vertices of a polygon, such that the distance of the curve from the polygon is bounded by a given tolerance. The maximum distance criterion could be replaced for certain applications by other criteria like for instance, the mean-square deviation criterion. According to our experience the convergence of the iteration is very fast. The algorithm is very robust and is believed to be useful in many applications in pattern recognition and processing of maps. The algorithm can be extended to approximation of curves in n-dimensional spaces.

ACKNOWLEDGMENTS

The author wishes to thank Professor H. Freeman for his suggestions and supervision of the research described in this report. This work was supported in part by the Directorate of Mathematical and Information Sciences, Air Force Office of Scientific Research, AFSC under Grant AF-AFOSR-70-1854, and in part by a scholarship from the Swiss National Foundation for Scientific Research. The computer science laboratory facility utilized for this work was provided by the National Science Foundation under grants GJ-19 and GJ-1009. This support is gratefully acknowledged.

REFERENCES

1. H. FREEMAN, Boundary Encoding and Processing, in *Picture Processing and Psychopictoris*, (B. Lipkin and A. Rosenfeld, Eds.) Academic Press, New York, 1970, 241–266.
2. H. FREEMAN, On the Encoding of Arbitrary Geometric Configurations, *IRE Trans. on Electronic Computers* **EC-10**, 1961, 260–268.
3. H. FREEMAN, On the Geometric Computer Classification of Geometric Line Patterns, *Proc. National Electronics Conference*, Chicago, Ill., 1962, Vol. 18, 312–324.
4. J. SKLANSKY, Recognition of Convex Blobs, *Pattern Recognition* **2**, 1970, 3–10.
5. J. SKLANSKY, R. L. CHAZIN, B. J. HANSEN, Minimum-Perimeter Polygons of Digitized Silhouettes, *IEEE Trans. on Computers* **C-21**, 1972, 260–268.
6. J. SKLANSKY, Measuring Concavity on a Rectangular Mosaic, School of Engrg., Univ. of California, Irvine, Calif., *Tech. Report TR-71-4*, 1971.
7. U. MONTANARI, A Note on Minimal Length Polygonal Approximation to a Digitized Contour, *Comm. ACM*, **13**, 1970, 41–47.
8. C. T. ZAHN, Two-Dimensional Pattern Description and Recognition via Curvature Points, Stanford Linear Accelerator Center, Stanford Univ., Stanford, Calif., *SLAC Report No. 70*, 1966.
9. H. STONE, Approximation of Curves by Line Segments, *Math. Comp.* **15**, 1961, 40–47.
10. R. BELLMAN, On the Approximation of Curves by Line Segments Using Dynamic Programming, *Comm. ACM* **4**, 1961, p. 284.
11. B. GLUSS, Further Remarks on Line Segment Curve-Fitting Using Dynamic Programming, *Comm. ACM* **5**, 1962, 441–443.
12. B. GLUSS, A Line Segment Curve-Fitting Algorithm Related to Optimal Encoding of Information, *Information and Control* **5**, 1962, 261–267.
13. C. L. JARVIS, A Method for Fitting Polygons to Figure Boundary Data, *Austral. Comput. J.*, **3**, 1971, 50–54.
14. G. REGGIORI, Digital Computer Transformations for Irregular Line Drawings, Doctoral Dissertation, Dept. of Elect. Engrg. and Computer Science, New York University, Bronx, N. Y., 1972.
15. N. J. NILSSON, *Problem-Solving Methods in Artificial Intelligence*, McGraw-Hill, New York, 1971.
16. R. S. LEDLEY AND G. C. CHENG, Automatic Recognition of White Blood Cells, *S.P.I.E. Journal*, **8**, 1970, 209–212.
17. W. R. TOBLER, An Experiment in the Computer Generalization of Maps, Dept. of Geography, Univ. of Michigan, Ann Arbor, Mich., *AD459953*, 1964.
18. K. RUTTENBERG, Digital Computer Analysis of Arbitrary Three-Dimensional Geometric Configurations, *Tech. Report 400–69*, New York University, Dept. of Elect. Engrg., October 1962.