PERGAMON

Pattern Recognition 35 (2002) 2539–2548

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Efficient algorithms for 3-D polygonal approximation based on LISE criterion

Kuo-Liang Chung[a,*], Wen-Ming Yan[c], Wan-Yue Chen[b]

[a] *Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, No. 43, Section 4, Keelung Road, Taipei 10672, Taiwan, ROC*

[b] *Department of Information Management, National Taiwan University of Science and Technology, No. 43, Section 4, Keelung Road, Taipei 10672, Taiwan, ROC*

[c] *Department of Computer Science and Information Engineering, National Taiwan University, Taipei 10764, Taiwan, ROC*

Received 5 October 2000; accepted 19 October 2001

Abstract

Given a polygonal curve P in three-dimensional (3-D) space, the polygonal approximation (PA) problem in this research is to find a polygon P' to approximate P either with the minimal polygonal segments under a given error or, conversely, with the minimal error under a specified number of segments allowable. The former PA problem is called the PA-# problem; the latter PA problem is called the PA-ε problem. Given a 3-D P with n nodes, under the local integral square error criterion, this paper first presents an $O(n^2)$-time algorithm for solving the PA-# problem using $O(n)$ space. Then we present an $O(n^2 \log n)$-time algorithm for solving the PA-ε using $O(n^2)$ space. Finally, a sampling technique is employed to reduce the memory requirement from $O(n^2)$ to $O(n)$. Some experiments are carried out to confirm the theoretical analysis. © 2002 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Algorithms; Dynamic programming; Integral square error criterion; 3-D polygonal approximation; Sampling technique; Shortest path

1. Introduction

Given a polygonal curve P in two-dimensional (2-D) or 3-D space, the polygonal approximation (PA) problem is to find a polygon P' to approximate P with n points either with the fewest polygonal segments, say #, under an error tolerance criterion or with the least error, say ε, under a specified number of segments allowable. The former PA problem is called the PA-# problem; the latter PA problem is called the PA-ε problem. Solving PA-# and PA-ε problems is very important in the field of shape representation [1]. In some sense, P' can be viewed as an compressed representation of P. For different levels of errors, the corresponding P's can be used in binary image progressive transmission (IPT) [2]. Thus, the results of this research can be used in IPT.

There are many sub-optimal algorithms developed, for solving the PA problem in $O(n)$ time. In Ref. [3], Rosin presented various techniques for evaluating different heuristic algorithm. However, this research focuses on the case of the optimal solution mentioned in the above paragraph.

Under different error metrics, many efficient algorithms for solving the PA problems have been developed. In what follows, we survey some previous published results under different error criteria. Most of them focus on 2-D domain. Throughout the paper, we assume that the starting point is given. For the case of closed curve where no starting point is given all the time complexity mentioned below must be multiplied by n. We only discuss on the case of open curve. Let the error be defined as the difference between the perimeters of P and P'. Sato [4] presented an $O(n^3)$-time algorithm for solving the PA-ε problem. Using the city block, i.e. L_1, as the error metric, an $O(n^2)$-time algorithm was presented by Pikaz and Dinstein [5]. Sharaiha

* Corresponding author. Fax: +886-2-273-76777.
E-mail address: klchung@cs.ntust.edu.tw (K.-L. Chung).

PII: S0031-3203(01)00226-6

and Cristofides [6] presented an $O(n^2)$-time algorithm for solving the PA-# problem. But in Ref. [6], the PA edges are restricted to approximate only digital arcs that contain one rectangular direction only. Using the tolerance zone criterion, Dunham [7] presented the cone intersection method and an $O(n^3)$-time algorithm was presented for solving the PA-# problem. Imai and Iri [8] improved Dunham's result and presented an $O(n^2 \log n)$-time ($O(n^2 \log^2 n)$-time) algorithm for solving the PA-# (PA-ε) problem using $O(n^2)$ space. Independently, Melkman and O'Rourke [9] presented the same results. Further, Chan and Chin [10] presented an $O(n^2)$-time ($O(n^2 \log n)$-time) algorithm for solving the PA-# (PA-ε) problem while using $O(n^2)$ space. Recently, Chen and Daescu [11] reduced the space complexity to $O(n)$.

For 3-D domain, using the L_1 and L_∞ metrics, Barequet et al. [12] presented an $O(n^2)$-time ($O(n^2 \log n)$-time) algorithm for solving the PA-# (PA-ε) problem. Using the L_2 metric, the PA-# (PA-ε) problem can be solved in $O(n^2 \log n)$-time ($O(n^2 \log^3 n)$-time). Using the Chebyshev error, Hakimi and Echmeichel [13] presented an $O(n^2)$-time ($O(n^2 \log n)$-time) algorithm for solving the PA-# (PA-ε) problem. Using the parallel-strip error criterion, Eu and Toussaint [14] improved the results of Imai and Iri [15] and presented an $O(n^2)$-time ($O(n^2 \log n)$-time) algorithm using $O(n^2)$ space for solving the 2-D PA-# (PA-ε) problem. For 3-D case, using the L_1 and L_{inf} error metrics, they presented an $O(n^2)$-time ($O(n^3)$-time) algorithm for solving the PA-# (PA-ε) problem. Using the L_2 metric, the PA-# (PA-ε) problem can be solved in $O(n^3)$-time ($O(n^3 \log n)$-time). All the algorithms in Ref. [14] need $O(n^2)$ space.

Using the global integral square error (ISE) criterion, Perez and Vidal [16] presented an $O(sn^2)$-time algorithm for solving the PA-ε problem, where s denotes the number of specified polygonal segments. Using the local ISE (LISE) criterion, Ray and Ray [17] presented a heuristic $O(n)$-time algorithm for finding a PA, but their solution is only sub-optimal for solving the PA-# problem. Basically, the LISE metric can avoid the loss of peak information when compared to the global ISE. Among these error metrics used previously, the LISE metric [3] is a useful metric in PA. Following the LISE criterion, the motivation of this research is to develop efficient algorithms for solving the PA-# problem and the PA-ε problem with optimal solutions.

Using the LISE criterion, this paper first presents an $O(n^2)$-time algorithm to solve the 3-D PA-# problem using $O(n)$ space. Then an $O(n^2 \log n)$-time algorithm is presented to solve the 3-D PA-ε using $O(n^2)$ space. Further, a sampling technique is employed to reduce the memory requirement from $O(n^2)$ to $O(n)$. To the best of our knowledge, this is the first time that such 3-D PA-# and PA-ε algorithms are presented. Specifically, following our results, the PA-# (PA-ε) problems in 2-D domain under the same error criterion can be solved using $O(n^2)$-time ($O(n^2 \log n)$-time) and $O(n)$ space. The techniques used in our proposed algorithms are some different from the previous algorithms since different error criteria often lead to different methods for solving the PA problems.

The rest of this paper is organized as follows. Section 2 presents the proposed algorithm for solving the 3-D PA-# problem. Section 3 presents the proposed algorithm for solving the 3-D PA-ε problem. Some experimental results are illustrated in Section 4. Some concluding remarks are addressed in Section 5.

2. The proposed 3-D PA-# algorithm

Suppose the given 3-D curve with n points is represented by the set $\{P_k = (x_k, y_k, z_k)$ for $k = 1, 2, 3, \ldots, n\}$. We first derive some nontrivial formulas to compute the LISE between a segment and a set of points apart from it in an incremental way.

Let

$$d_{ij} = \sum_{k=i+1}^{j-1} d^2(P_k, \text{line}(P_iP_j))$$

be the LISE between the line passing through two arbitrary points, P_i and P_j, say line(P_iP_j), and the set of points $P_{i+1}, P_{i+2}, \ldots, P_{j-1}$, where

$$d(P_k, \text{line}(P_iP_j))$$

is the Euclidean distance from the point P_k to the line line(P_iP_j).

Since line(P_iP_j) can be represented by the following parametric form:

$$x = x_i + \alpha_{ij}t,$$

$$y = y_i + \beta_{ij}t,$$

$$z = z_i + \gamma_{ij}t,$$

where $\alpha_{ij} = x_j - x_i$, $\beta_{ij} = y_j - y_i$, and $\gamma_{ij} = z_j - z_i$. The square error (SE) between the point P_k to the line(P_iP_j) is the minimal value of

$$\begin{aligned} f(t) &= (x_k - x_i - \alpha_{ij}t)^2 + (y_k - y_i - \beta_{ij}t)^2 \\ &\quad + (z_k - z_i - \gamma_{ij}t)^2 \\ &= (\alpha_{ij}^2 + \beta_{ij}^2 + \gamma_{ij}^2)t^2 - 2[(x_k - x_i)\alpha_{ij} \\ &\quad + (y_k - y_i)\beta_{ij} + (z_k - z_i)\gamma_{ij}]t \\ &\quad + [(x_k - x_i)^2 + (y_k - y_i)^2 + (z_k - z_i)^2]. \end{aligned}$$

Let $a = (\alpha_{ij}^2 + \beta_{ij}^2 + \gamma_{ij}^2)$, $b = [(x_k - x_i)\alpha_{ij} + (y_k - y_i)\beta_{ij} + (z_k - z_i)\gamma_{ij}]$, and $c = (x_k - x_i)^2 + (y_k - y_i)^2 + (z_k - z_i)^2$. Plugging the three parameters, a, b, and c, into $f(t)$, we have

$$\begin{aligned} f(t) &= at^2 - 2bt + c \\ &= a\left[\left(t - \frac{b}{a}\right)^2 + \frac{ac - b^2}{a^2}\right]. \end{aligned}$$

When $t = b/a$, we have the minimal value of $f(t)$ and the corresponding SE is equal to

$$\begin{aligned}\frac{ac-b^2}{a} &= [(x_k-x_i)^2+(y_k-y_i)^2+(z_k-z_i)^2] \\ &\quad - \frac{1}{\alpha_{ij}^2+\beta_{ij}^2+\gamma_{ij}^2}[\alpha_{ij}(x_k-x_i)+\beta_{ij}(y_k-y_i) \\ &\quad + \gamma_{ij}(z_k-z_i)]^2 \\ &= A_{ij}x_k^2 + B_{ij}y_k^2 + C_{ij}z_k^2 + D_{ij}x_k y_k + E_{ij}y_k z_k \\ &\quad + F_{ij}x_k z_k + G_{ij}x_k + H_{ij}y_k + I_{ij}z_k + J_{ij},\end{aligned}$$

where

$$A_{ij} = 1 - \frac{\alpha_{ij}^2}{\alpha_{ij}^2+\beta_{ij}^2+\gamma_{ij}^2},$$

$$B_{ij} = 1 - \frac{\beta_{ij}^2}{\alpha_{ij}^2+\beta_{ij}^2+\gamma_{ij}^2},$$

$$C_{ij} = 1 - \frac{\gamma_{ij}^2}{\alpha_{ij}^2+\beta_{ij}^2+\gamma_{ij}^2},$$

$$D_{ij} = -\frac{2\alpha_{ij}\beta_{ij}}{\alpha_{ij}^2+\beta_{ij}^2+\gamma_{ij}^2},$$

$$E_{ij} = -\frac{2\beta_{ij}\gamma_{ij}}{\alpha_{ij}^2+\beta_{ij}^2+\gamma_{ij}^2},$$

$$F_{ij} = -\frac{2\gamma_{ij}\alpha_{ij}}{\alpha_{ij}^2+\beta_{ij}^2+\gamma_{ij}^2},$$

$$G_{ij} = -2x_i + 2\frac{\alpha_{ij}(\alpha_{ij}x_i+\beta_{ij}y_i+\gamma_{ij}z_i)}{\alpha_{ij}^2+\beta_{ij}^2+\gamma_{ij}^2},$$

$$H_{ij} = -2y_i + 2\frac{\beta_{ij}(\alpha_{ij}x_i+\beta_{ij}y_i+\gamma_{ij}z_i)}{\alpha_{ij}^2+\beta_{ij}^2+\gamma_{ij}^2},$$

$$I_{ij} = -2z_i + 2\frac{\gamma_{ij}(\alpha_{ij}x_i+\beta_{ij}y_i+\gamma_{ij}z_i)}{\alpha_{ij}^2+\beta_{ij}^2+\gamma_{ij}^2},$$

$$J_{ij} = (x_i^2+y_i^2+z_i^2) - \frac{(\alpha_{ij}x_i+\beta_{ij}y_i+\gamma_{ij}z_i)^2}{\alpha_{ij}^2+\beta_{ij}^2+\gamma_{ij}^2}.$$

Return to the calculation of LISE. As a result, we have

$$\begin{aligned}d_{ij} &= \sum_{k=i+1}^{j-1} d^2(P_k, \text{line } P_iP_j) \\ &= A_{ij}S_1(i,j) + B_{ij}S_2(i,j) + C_{ij}S_3(i,j) + D_{ij}S_4(i,j) \\ &\quad + E_{ij}S_5(i,j) + F_{ij}S_6(i,j) + G_{ij}S_7(i,j) + H_{ij}S_8(i,j) \\ &\quad + I_{ij}S_9(i,j) + (j-i-1)J_{ij},\end{aligned}$$

where

$$S_1(i,j) = \sum_{k=i+1}^{j-1} x_k^2, \quad S_2(i,j) = \sum_{k=i+1}^{j-1} y_k^2,$$

$$S_3(i,j) = \sum_{k=i+1}^{j-1} z_k^2, \quad S_4(i,j) = \sum_{k=i+1}^{j-1} x_k y_k,$$

$$S_5(i,j) = \sum_{k=i+1}^{j-1} y_k z_k, \quad S_6(i,j) = \sum_{k=i+1}^{j-1} x_k z_k,$$

$$S_7(i,j) = \sum_{k=i+1}^{j-1} x_k, \quad S_8(i,j) = \sum_{k=i+1}^{j-1} y_k,$$

$$S_9(i,j) = \sum_{k=i+1}^{j-1} z_k,$$

with the boundary equation $S_l(i,i+1) = 0$ for $l = 1,2,\ldots,9$.

The calculation of $S_l(i+1,j)$, $S_l(i-1,j)$, $S_l(i,j-1)$, and $S_l(i,j+1)$ can be done using operations from $S_l(i,j)$ for $l = 1,2,\ldots,9$. The related incremental formulas are shown below:

$$S_1(i,j+1) = S_1(i,j) + x_j^2, \quad S_2(i,j+1) = S_2(i,j) + y_j^2,$$

$$S_3(i,j+1) = S_3(i,j) + z_j^2, \quad S_4(i,j+1) = S_4(i,j) + x_j y_j,$$

$$S_5(i,j+1) = S_5(i,j) + y_j z_j, \quad S_6(i,j+1) = S_6(i,j) + z_j x_j,$$

$$S_7(i,j+1) = S_7(i,j) + x_j, \quad S_8(i,j+1) = S_8(i,j) + y_j,$$

$$S_9(i,j+1) = S_9(i,j) + z_j,$$

for $j < n$;

$$S_1(i,j-1) = S_1(i,j) - x_{j-1}^2,$$

$$S_2(i,j-1) = S_2(i,j) - y_{j-1}^2,$$

$$S_3(i,j-1) = S_3(i,j) - z_{j-1}^2,$$

$$S_4(i,j-1) = S_4(i,j) - x_j y_{j-1},$$

$$S_5(i,j-1) = S_5(i,j) - y_j z_{j-1},$$

$$S_6(i,j-1) = S_6(i,j) - z_j x_{j-1},$$

$$S_7(i,j-1) = S_7(i,j) - x_{j-1},$$

$$S_8(i,j-1) = S_8(i,j) - y_{j-1},$$

$$S_9(i,j-1) = S_9(i,j) - z_{j-1},$$

for $j-1>i$;

$$S_1(i-1,j)=S_1(i,j)+x_i^2,\quad S_2(i-1,j)=S_2(i,j)+y_i^2,$$

$$S_3(i-1,j)=S_3(i,j)+z_i^2,\quad S_4(i-1,j)=S_4(i,j)+x_iy_i,$$

$$S_5(i-1,j)=S_5(i,j)+y_iz_i,\quad S_6(i-1,j)=S_6(i,j)+z_ix_i,$$

$$S_7(i-1,j)=S_7(i,j)+x_i,\quad S_8(i-1,j)=S_8(i,j)+y_i,$$

$$S_9(i-1,j)=S_9(i,j)+z_i,$$

for $i>1$;

$$S_1(i+1,j)=S_1(i,j)-x_{i+1}^2,$$

$$S_2(i+1,j)=S_2(i,j)-y_{i+1}^2,$$

$$S_3(i+1,j)=S_3(i,j)-z_{i+1}^2,$$

$$S_4(i+1,j)=S_4(i,j)-x_jy_{i+1},$$

$$S_5(i+1,j)=S_5(i,j)-y_jz_{i+1},$$

$$S_6(i+1,j)=S_6(i,j)-z_jx_{i+1},$$

$$S_7(i+1,j)=S_7(i,j)-x_{i+1},$$

$$S_8(i+1,j)=S_8(i,j)-y_{i+1},$$

$$S_9(i+1,j)=S_9(i,j)-z_{i+1},$$

for $i+1<j$.

Given the values of $S_l(i,j)$ for $l=1,2,\ldots,9$, x_i, y_i, z_i, x_j, y_j, and z_j, the LISE $d_{ij}=\sum_{k=i+1}^{j-1}d^2(P_k,\mathrm{line}(P_iP_j))$ can be obtained in $O(1)$ time using the following procedure:

$$\alpha\leftarrow x_j-x_i,$$

$$\beta\leftarrow y_j-y_i,$$

$$\gamma\leftarrow z_j-z_i,$$

$$A\leftarrow 1-\frac{\alpha^2}{\alpha^2+\beta^2+\gamma^2},$$

$$B\leftarrow 1-\frac{\beta^2}{\alpha^2+\beta^2+\gamma^2},$$

$$C\leftarrow 1-\frac{\gamma^2}{\alpha^2+\beta^2+\gamma^2},$$

$$D\leftarrow -\frac{2\alpha\beta}{\alpha^2+\beta^2+\gamma^2},$$

$$E\leftarrow -\frac{2\beta\gamma}{\alpha^2+\beta^2+\gamma^2},$$

$$F\leftarrow -\frac{2\gamma\alpha}{\alpha^2+\beta^2+\gamma^2},$$

$$G\leftarrow -2x_i+2\frac{\alpha(\alpha x_i+\beta y_i+\gamma z_i)}{\alpha^2+\beta^2+\gamma^2},$$

$$H\leftarrow -2y_i+2\frac{\beta(\alpha x_i+\beta y_i+\gamma z_i)}{\alpha^2+\beta^2+\gamma^2},$$

$$I\leftarrow -2z_i+2\frac{\gamma(\alpha x_i+\beta y_i+\gamma z_i)}{\alpha^2+\beta^2+\gamma^2},$$

$$J\leftarrow (x_i^2+y_i^2+z_i^2)-\frac{(\alpha x_i+\beta y_i+\gamma z_i)^2}{\alpha^2+\beta^2+\gamma^2},$$

$$\begin{aligned}d_{ij}\leftarrow{}& AS_1(i,j)+BS_2(i,j)+CS_3(i,j)+DS_4(i,j)\\&+ES_5(i,j)+FS_6(i,j)+GS_7(i,j)+HS_8(i,j)\\&+IS_9(i,j)+(j-i-1)J.\end{aligned}$$

Before presenting the proposed algorithm for solving the PA-# problem, we first describe how to construct a directed, weighted graph $G=(V,E)$, where the set of nodes is denoted by $V=\{P_1,P_2,\ldots,P_n\}$, i.e. $|V|=n$, such that for each edge $(P_i,P_j)\in E$, $1\leqslant i<j\leqslant n$, the LISE d_{ij} is equal to or less than the prespecified error tolerance ε.

Considering the first point P_1, we connect all pairs P_1 and P_i, for $2\leqslant i\leqslant n$, if the LISE $d_{ij}\leqslant\varepsilon$. We also assign the weight value 1 on the edge. From the above incremental formulas, it takes $O(n)$ time to compute d_{ij} for these feasible edges. By the same arguments, for $2\leqslant i\leqslant n-1$, we connect all the related feasible edges. The edges set E is the union of these connected edges. As a result, the directed graph G can be constructed in $O(n^2)$ time using $O(n^2)$ space.

We then apply Dijkstra's algorithm [18] to solve the single-source shortest-path problem on this directed, weighted graph G, where the source node in G is the starting point P_1 in P and the target node in G is the final point P_n in P. The shortest-path finding algorithm takes $O(n^2)$ time. We thus have a shortest path from point P_1 to point P_n. Because the path length of the obtained shortest path is minimal among all possible paths from P_1 to P_n, this shortest path is indeed the desired approximate polygon P' with the minimal number of segments, i.e. edges, under the specified LISE. Since it takes $O(n^2)$ time for finding the shortest-path, so given an ε, the 3D PA-# problem can be solved in $O(n^2)$ time and using $O(n^2)$ space.

We now combine the above two phases, (1) constructing the directed, weighted graph G and (2) applying the shortest-path algorithm for finding the path from point P_1 to point P_n, into one phase. This leads to a reduction in space from $O(n^2)$ to $O(n)$.

Before presenting the formal algorithm, we outline the key concept. Suppose currently the weight of the shortest path from P_1 to P_i for $2\leqslant i<j$ is known and is saved in $W[i]$. We also maintain an array L for linking the points in the shortest path. The contents $L[2],L[3],\ldots,L[j-1]$ are

such that if

$$L^1[i] = L[i],$$

$$L^{k+1}[i] = L[L^k[i]] \quad \text{for } k \geqslant 1, \tag{1}$$

then $P_{L^{W[i][i]}} \cdots P_{L^2[i]} P_{L[i]} P_i$ is a shortest path from P_1 to P_i for all $2 \leqslant i < j$. For example, suppose we have a P_1–P_5–P_7–P_9 shortest path. Then we have $L[9]=7$, $L[7]=L[L[9]]=L^2[9]=5$, $L[5]=L[L[L[9]]]=L^3[9]=1$.

Initially, for $j=3$, let $W[1]=0, W[2]=1$, and $L[2]=1$. When $j>3$, suppose we have obtained the values of $W[1], W[2], \ldots, W[j-1], L[2], L[3], \ldots$, and $L[j-1]$ which satisfy the conditions in Eq. (1). For $1 \leqslant i < j$, if $d_{i,j} < \varepsilon$, we set the weight 1 on the edge connecting P_i and P_j; otherwise, do nothing. For any specific t, $1 \leqslant t < j$, if the following two conditions hold:

1. $d_{t,j} < \varepsilon$,
2. if $d_{i,j} < \varepsilon$ and $1 \leqslant i < j$ then $W[t] \leqslant W[i]$,

then we know that $P_{L^{W[t][t]}} \cdots P_{L^2[t]} P_{L[t]} P_t P_j$ is a shortest path from P_1 to P_j. We thus assign $L[j]=t$ and $W[j]=W[t]+1$. So, the shortest path $P_{L^{W[j][j]}} \cdots P_{L^2[j]} P_{L[j]} P_j$ is identical to the path $P_{L^{W[t][t]}} \cdots P_{L^2[t]} P_{L[t]} P_t P_j$.

Since $S_l(j-1, j) = 0$ for all $l = 1, 2, \ldots, 9$, we have $d_{(j-1)j}=0$ and we can compute d_{ij} for $i=j-1, j-2, \ldots, 1$ in $O(j)$ time using the following procedure:

```
S1(j − 1, j) = 0, S2(j − 1, j), ..., S9(j − 1, j) = 0,
d(j−1)j = 0
for i ← j − 2 downto 2
    {the values of S1(i + 1, j), S2(i + 1, j), ..., S9(i + 1, j)
    are known}
    Compute S1(i, j), S2(i, j), ..., S9(i, j) {it takes O(1) time
    in an incremental way}
    Compute dij {it takes O(1) time}
endfor
```

Hence, given $W[1], W[2], \ldots, W[j-1], L[2], L[3], \ldots$, and $L[j-1]$, we can compute $d_{(j-1)j}, d_{(j-2)j}, \ldots, d_{1j}$ in $O(j)$ time. Furthermore, we can compute $W[j]$ and $L[j]$ in $O(j)$ time using the following procedure:

```
m ← W[j − 1],
t ← j − 1
for i ← j − 2 downto 2
    if dij < ε then
      if W[i] < m then
        m ← W[i];   t ← i
      endif
    endif
endfor
W[j] ← m + 1;  t ← j;  L[j] ← t
```

Since at most $O(n)$ space and $O(n)$ time are required to update the current node P_j, it takes $O(n)$ space and $O(n^2)$ time to find the path-length of the path P_1–P_n under the specified LISE. Finally, using the backtracking technique, the desired approximate polygon P' is obtained. The formal algorithm is listed below.

Algorithm 1. PA-#

Input: the LISE ε and the 3-D polygon P with n points, $\{P_k = (x_k, y_k, z_k)$ for $k = 1, 2, 3, \ldots, n\}$.

Output: the approximate 3-D polygon P' with $P'_1 = P_1$ and $P'_m = P_n$, where m denotes the number of points in P'.

```
Function find(ε) {return the solution, the number of seg-
ments}
W[1] ← 0;  L[2] ← 1;  W[2] ← 1
for j ← 3 to n
    m ← W[j − 1]
    t ← j − 1
    for k ← j − 1 downto 2
      i ← k − 1
      Compute S1(i, j), S2(i, j), ..., and S9(i, j)
      Compute dij
      if dij < ε then
        if W[i] < m then
          m ← W[i];  t ← i
        endif
      endif
    endfor
      W[j] ← m + 1;  t ← j;  L[j] ← t
endfor
m ← W[n]
t ← n
for i ← m + 1 downto 1
  P'i ← Pt
  t ← L[t]
endfor
return m

begin
find(ε) {function call}
end
```

According to the above description and the algorithm, the first main result is given below.

Theorem 1. *Given an LISE of* ε, *the* 3-*D PA-# problem can be solved in* $O(n^2)$ *time and using* $O(n)$ *space.*

Suppose we are given a 2-D curve with n points and the coordinate of the point P_k is (x_k, y_k) for $k=1, 2, 3, \ldots, n$. We now discuss the calculation of the LISE $\sum_{k=2}^{i-1} d^2(P_k, \text{line}(P_1P_i))$, where $d(P_k, \text{line}(P_1P_i))$ is the Euclidean distance from the point P_k to the line line(P_1P_i). Since the equation of the line P_1P_i can be represented by $a_i x + b_i y + c_i = 0$, where $a_i = y_i - y_1$, $b_i = x_1 - x_i$ and $c_i = x_i y_1 - x_1 y_i$. Hence the SE from the point P_k to the line

P_1P_i is equal to

$$\frac{(a_i x_k + b_i y_k + c_i)^2}{a_i^2 + b_i^2}.$$

Hence, the LISE can be written as

$$\begin{aligned}
&\sum_{k=2}^{i-1} d^2(P_k, \text{line}\, P_1P_i) \\
&= \frac{1}{a_i^2 + b_i^2} \sum_{k=2}^{i-1} (a_i x_k + b_i y_k + c_i)^2 \\
&= \frac{1}{a_i^2 + b_i^2} \sum_{k=2}^{i-1} [a_i^2 x_k^2 + b_i^2 y_k^2 + c_i^2 \\
&\quad + 2a_i b_i x_k y_k + 2a_i c_i x_k + 2b_i c_i y_k] \\
&= \frac{1}{a_i^2 + b_i^2} \left[a_i^2 \sum_{k=2}^{i-1} x_k^2 + b_i^2 \sum_{k=2}^{i-1} y_k^2 + (i-2)c_i^2 \right. \\
&\quad \left. + 2a_i b_i \sum_{k=2}^{i-1} x_k y_k + 2a_i c_i \sum_{k=2}^{i-1} x_k + 2b_i c_i \sum_{k=2}^{i-1} y_k \right].
\end{aligned}$$

By the same arguments used for solving the 3-D PA-# problem, the following result is followed.

Corollary 1. *Given an LISE ε, the 2-D PA-# problem can be solved in $O(n^2)$ time and using $O(n)$ space.*

3. The proposed 3-D PA-ε algorithm

Given the number of segments allowable, say s, consider a 3-D curve with n points again. In this section, an $O(n^2 \log n)$-time and $O(n)$-space algorithm will be presented for solving the PA-ε problem.

Following the results in Refs. [8,19], it is not hard to derive the following result.

Lemma 1. *Let ε_A, ε_B be two LISEs, with ε_A to obtain an approximate curve P'_A of m_A segments and with ε_B to obtain an approximate curve P'_B of m_B segments, then (1) if $\varepsilon_A < \varepsilon_B$, then $m_A \geqslant m_B$; (2) if $m_A < m_B$, then $\varepsilon_A > \varepsilon_B$.*

From the discussion in Section 2, for $1 \leqslant i < j \leqslant n$, all the LISEs $d_{ij} = \sum_{k=i+1}^{j-1} d^2(P_k, \text{line}(P_iP_j))$, each of which is equal to or less than the prespecified error, can be obtained in $O(n^2)$ time.

We first apply the sorting algorithm [18] to sort these $O(n^2)$ LISEs in an increasing order and save these sorted LISEs in an array. Suppose these sorted LISEs are denoted by $err_1, err_2, \ldots,$ and err_q, where $q = O(n^2)$.

Based on the binary search method, we first fix $err_{q/2}$ as the specified LISE and apply the proposed PA-# algorithm described in Section 2. Then the approximate polygon with the minimal number of segments is obtained and the number of segments required, say m', is known. If m' is equal to the given number of segments, s, we indeed solve the PA-ε problem. If m' is greater than (less than) s, by Lemma 1, we fix $err_{q/4}$ ($err_{3q/4}$) as the specified LISE and apply the previous PA-# algorithm again. Continuing this way, it takes at most $O(\log n)$ search steps. By Theorem 1, each search step takes $O(n^2)$ time, so totally it takes $O(n^2 \log n)$ time and $O(n^2)$ space for solving the PA-ε problem, where $O(n^2)$ space is used for saving those sorted LISEs, $err_1, err_2, \ldots,$ and err_q. The algorithm is listed below.

Algorithm 2. PA-ε

Input: the number of segments allowable in P', say s, and the given 3-D polygon P with n points, say $\{P_k = (x_k, y_k, z_k)$ for $k = 1, 2, 3, \ldots, n\}$.

Output: the approximate 3-D polygon P' with $P'_1 = P_1$ and $P'_m = P_n$ with the minimal LISE.

```
begin
l ← 1;   r ← q
repeat
  t ← (l + r)/2
  m ← find(ε_t)
  if m ⩽ s then
    if m < s then
      r ← t − 1
    else
      r ← t
  else
      l = t + 1
  endif
until l = r
return(ε_l)
end
```

Finally, we want to reduce the memory required in Algorithm 2 to $O(n)$ by using the sampling method [11], but preserve the same time bound, $O(n^2 \log n)$.

For the ith point in P, $1 \leqslant i \leqslant n$, suppose the constructed set of feasible edges under the specified LISE in the weighted, directed graph G is denoted by Err_i and $Err_i = O(n)$. In G, we have $Err_1, Err_2, \ldots,$ and Err_{n-1}.

In order to keep the memory requirement bounded in $O(n)$, we process Err_1 first. That is, we sort it using $O(n \log n)$ time and $O(n)$ space. Then we select the $j\sqrt{n}$th elements for $1 \leqslant j \leqslant \sqrt{n}$ in the sorted Err_1 and these $\sqrt{n}$ selected elements form the set ERR_1 with size $O(\sqrt{n})$. By the same arguments, we process Err_2 and the corresponding ERR_2 with size $O(\sqrt{n})$ is obtained. It still takes $O(n \log n)$ time and $O(n)$ space. We continue to process the remaining $\sqrt{n} - 2$ Err_i's one by one. Finally, we obtain $ERR_1, ERR_2, \ldots,$ and $ERR_{\sqrt{n}}$, each ERR_i with size $O(\sqrt{n})$. We further sort these $\sqrt{n}$ ERR_i's and obtain the sorted list with size $O(n)$. We then select the $j\sqrt{n}$th elements for $1 \leqslant j \leqslant \sqrt{n}$ in the sorted list and these $\sqrt{n}$ selected ele-

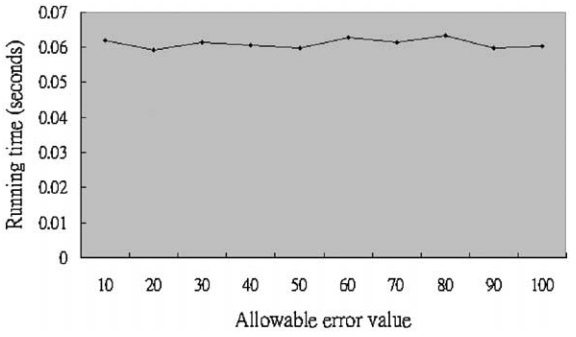


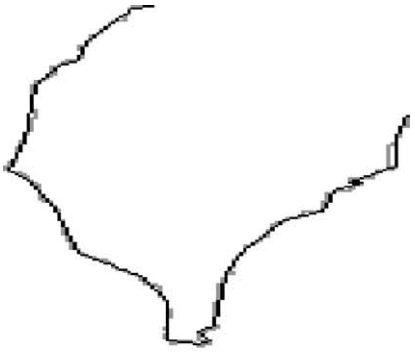
Fig. 1. The running time performance for our proposed algorithm for solving PA-# problem.

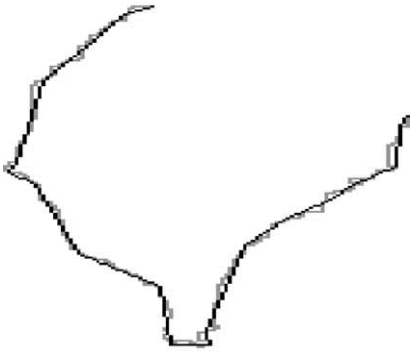
ments form the set S_1 with size $O(\sqrt{n})$. According to the result in Ref. [11], there are $O(n)$ err_i's between any two consecutive elements in S_1. The time required for obtaining S_1 is bounded by $O(n\sqrt{n}\log n)$ and the space is bounded by $O(n)$. Up to now, the size of S_1 is $O(\sqrt{n})$.

Very similar to obtain S_1, we can obtain $S_1, S_2, \ldots$, and $S_{\sqrt{n}}$ using $O(n^2 \log n)$ time and $O(n)$ space. We then sort these $O(\sqrt{n})$ S_i's and obtain the sorted set S' with size $O(n)$. Based on the binary search method, by Theorem 1, we can locate two consecutive elements in S', say x and y, using $O(n^2 \log n)$ time and $O(n)$ space such that the given number of segments, s, is within range from x to y. By the same argument, there are $O(n\sqrt{n})$ err_i's between x and y in S'.

Next, we use x and y as a filter and $err_1, err_2, \ldots$, and err_q are fed into the filter one by one. There are $O(n\sqrt{n})$ err_i's retained, but they will not be saved explicitly. In real implementation, an incremental approach is still adopted. The previous $O(n)$ err_i's in the $O(n\sqrt{n})$ retained err_i's are fed into the first group. Then the first group is sorted. We then select the $j\sqrt{n}$th elements for $1 \leqslant j \leqslant \sqrt{n}$ in the first sorted group. These $\sqrt{n}$ selected elements form a new set T_1 with size $O(\sqrt{n})$. We do the same thing incrementally and obtain the new sets, $T_2, T_3, \ldots$, and $T_{\sqrt{n}}$. Collecting these $\sqrt{n}$ new sets, we sort these n elements. We then select the $j\sqrt{n}$th elements for $1 \leqslant j \leqslant \sqrt{n}$ in the sorted list. There exist $O(n)$ err_i's within the interval of any two consecutive elements. Applying the binary search method associated with Algorithm 1 on the sorted list, two consecutive elements are located and are used as a filter again. Then $err_1, err_2, \ldots$, and err_q are fed into the filter one by one. There are $O(n)$ err_i's retained, and they are saved explicitly. Finally, applying algorithm 2 (only using $O(n)$ space), the desired solution can be obtained. We thus have the following result.

Theorem 2. *Suppose the number of segments allowable in P' is specified by s, the 3D PA-ε problem can be solved in*


(a)


(b)


(c)


(d)

Fig. 2. The resulting PAs for PA-# problem: (a) PA for LISE $\varepsilon = 10$; (b) PA for LISE $\varepsilon = 20$; (c) PA for LISE $\varepsilon = 30$; (d) PA for LISE $\varepsilon = 40$.

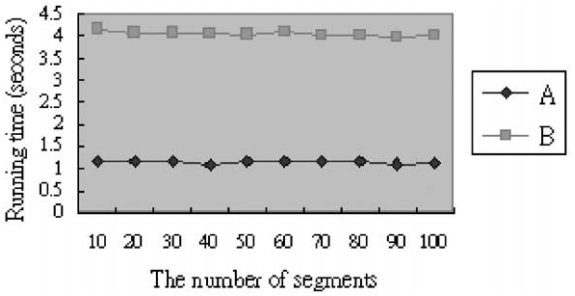


Fig. 3. The running time performance for our proposed two PA-ε algorithms.

$O(n^2 \log n)$ time and using $O(n)$ space under the minimal LISE.

By the same arguments mentioned in solving the 2-D PA-# problem, it is not hard to derive the following result.

Corollary 2. *Suppose the number of segments allowable in P' is specified by s, the 2-D PA-ε problem can be solved in $O(n^2 \log n)$ time and using $O(n)$ space under the minimal LISE.*

4. Experimental results

We implement the related algorithms using Borland C++ Builder 4.0 language and the Pentium 133 PC with 48 MB RAM. The digital map of the southern Taiwan with 233 segments is used for evaluating the performance. For solving the PA-# problem, Fig. 1 depicts the running time performance for our proposed algorithms, where the detailed time compleity is rather stable and denoted by $1.1 \times 10^{-6} n^2$.

Suppose the PA-# problem. LISEs are set to be $\varepsilon = 10$, 20, 30, and 40. Applying our proposed algorithm for solving minimal number of the PA-# problem, the resulting form PAs are illustrated in Fig. 2(a), (b), (c), and (d), respectively, with the minimal segments $\# = 27$, 19, 15, and 13.

For solving the PA-ε problem, Fig. 3 depicts the running time performance for our proposed two algorithms, where the curve A denotes the proposed algorithm using $O(n^2 \log n)$ time and $O(n^2)$ space; the curve B denotes the proposed algorithm employing the sampling technique and using only $O(n)$ space. Taking the leading constant factor in the big-O notation, we have that the time complexity of the algorithm A (B) is rather stable and denoted by $2.023 \times 10^{-6} n^2 \log n$ ($7.155 \times 10^{-6} n^2 \log n$).

Given the number of segments, say $\# = 6$, 11, 21, and 31, applying our proposed algorithm for solving the PA-ε problem, the resulting PAs are illustrated in Fig. 4(a), (b), (c), and (d), respectively, with LISEs $\varepsilon = 650.6$, 70.0, 15.3, and 8.7.

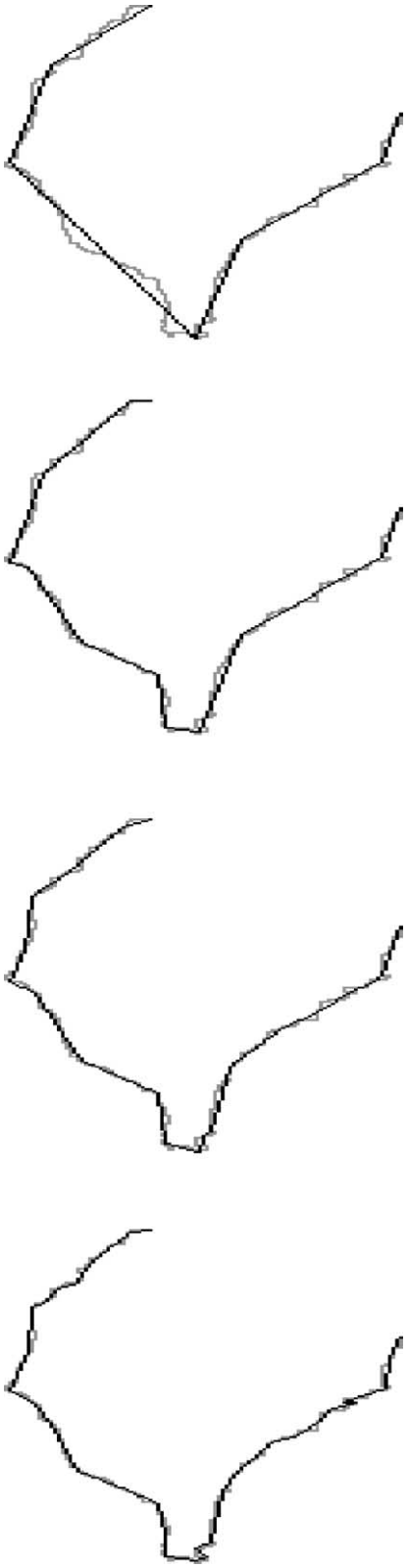


Fig. 4. The resulting form PAs for PA-ε problem.

5. Conclusions

Solving the PA problem is very fundamental in shape representation. The LISE is a practical error metric in the PA. We have presented three efficient algorithms for solving the PA-# problem and the PA-ε problem. To the best of our knowledge, this is the first time that such three algorithms are presented under the LISE criterion. In fact, the results of this research can be used in image progressive transmission.

The nontrivial lower bound proof for the PA problem based on the LISE criterion is an open problem which can tell us the gap between the proposed algorithms and the theoretical lower bound. Sometimes the ratio n/m can be viewed as the compression ratio. How to applying the results of this paper to design efficient algorithms using the normalized figure of merit ($=n/m \times ISE$) metric [6,20,21] is an interesting research problem.

Acknowledgements

The authors appreciate the anonymous referees, the Managing Editor Mrs. Blaire V. Mossman, and Prof. T.C. Woo for their valuable comments that lead to the improved version of this paper. This research is supported by the National Science Council of R.O.C. under contract NSC89-2213-E011-061.

References

[1] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Section 8.1.2: Polygonal Approximations, Addison-Wesley, New York, 1992.

[2] C.L.B. Jordan, T. Ebrahimi, M. Kunt, Pregressive content-based shape compression for retrieval of binary images, Comput. Vision Image Understanding 71 (2) (1998) 198–212.

[3] P.L. Rosin, Techniques for assessing polygonal approximations of curves, IEEE Trans. Pattern Anal. Mach. Intell. 19 (6) (1997) 659–666.

[4] Y. Sato, Piecewise linear approximation of plane curves by perimeter optimization, Pattern Recognition 25 (12) (1992) 1535–1543.

[5] A. Pikaz, I. Dinstein, Optimal polygonal approximation of digital curves, Pattern Recognition 28 (3) (1995) 373–379.

[6] Y.M. Sharaiha, N. Christofides, An optimal algorithm for the straight segment approximation of digital arcs, CVGIP: Graphical Models Image Process. 55 (5) (1993) 397–407.

[7] J.G. Dunham, Optimization piecewise linear approximation of planar curves, IEEE Trans. Pattern Anal. Mach. Intell. 8 (1) (1986) 67–75.

[8] H. Imai, M. Iri, Computational-geometric methods for polygonal approximations of a curve, Comput. Vision Graphics Image Process. 36 (1986) 31–41.

[9] A. Melkman, J. O'Rourke, On polygonal chain approximation, in: G.T. Toussaint (Ed.), Computational Morphology, North-Holland, Amsterdam, 1988, pp. 87–95.

[10] W.S. Chan, F. Chin, Approximation of polygonal curves with minimum number of line segments or minimum error, Int. J. Comput. Geom. Appl. 6 (1996) 59–77.

[11] D.Z. Chen, O. Daescu, Space-efficient algorithms for approximating polygonal curves in two dimensional space, in: W.L. Hsu, M.Y. Kao (Eds.), The Fourth Annual International Conference COCOON'98, 1998, pp. 45–54.

[12] G. Barequet, D.Z. Chen, O. Daescu, M.T. Goodrich, J. Snoeyink, Efficiently approximating polygonal paths in three and higher dimensions, Proceedings of the 14th Annual ACM Symposium on Computational Geometry, 1998, pp. 317–326.

[13] S.L. Hakimi, E.F. Schmeichel, Fitting polygonal functions to a set of points in the plane, CVGIP: Graph. Models Image Process. 53 (2) (1991) 132–136.

[14] D. Eu, G.T. Toussaint, On approximating polygonal curves in two and three dimensions, CVGIP: Graph. Models Image Process. 56 (3) (1994) 231–246.

[15] H. Imai, M. Iri, Polygonal approximations of a curve—formulations and algorithms, in: G.T. Toussaint (Ed.), Computational Morphology, North-Holland, Amsterdam, 1988, pp. 71–86.

[16] J.C. Perez, E. Vidal, Optimum polygonal approximation of digitized curves, Pattern Recognition Lett. 15 (1994) 743–750.

[17] B.K. Ray, K.S. Ray, A non-parametric sequential method for polygonal approximation of digital curves, Pattern Recognition Lett. 15 (1994) 161–167.

[18] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, Section 25.2, Dijkstra's Algorithm, The MIT Press, Cambridge, MA, 1990.

[19] G.T. Toussaint, On the complexity of approximating polygonal curves in the plane, Proceedings of the IASTED International Symposium on Robotics and Automation, 1985.

[20] A. Held, K. Abe, C. Arcelli, Towards a hierachical contour description via dominant point detection, IEEE Trans. Systems Man Cybernet. 24 (1994) 942–949.

[21] P.L. Rosin, G.A.W. West, Non-parametric segmentation of curves into various representations, IEEE Trans. Pattern Anal. Mach. Intell. 17 (1995) 1140–1153.

About the Author—KUO-LIANG CHUNG received the B.S., M.S., and Ph.D. degrees in Computer Science and Information Engineering from National Taiwan University in 1982, 1984, and 1990, respectively. From 1984 to 1986, he was a soldier. From 1986 to 1987, he was a research assistant in the institute of Information Science, Academic Sinica. Now he is a professor in the department of Computer science and Information Engineering, National Taiwan University of Science and Technology. Prof. Chung received the Distinguished Professor Award from the Chinese Institute of Engineers in May 2001. His research interests include image compression, image processing, computer graphics, coding theory, and algorithms.

About the Author—WEN-MING YAN received the B.S. and M.S. degrees in Mathematics from National Taiwan University. Now he is an associate professor in the department of Computer science and Information Engineering, National Taiwan University. His research interests include image compression, matrix computations, and algorithms.

About the Author—WAN-YUE CHEN received the B.S. and M.S. degrees in Information Management from National Taiwan University of Science and Technology in 1999 and 2001, respectively. Her research interests include image compression, image processing, and algorithms.