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

Kuo-Liang Chung\textsuperscript{a,}\textsuperscript{*}, Wen-Ming Yan\textsuperscript{c}, Wan-Yue Chen\textsuperscript{b}

\textsuperscript{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
\textsuperscript{b}Department of Information Management, National Taiwan University of Science and Technology, No. 43, Section 4, Keelung Road, Taipei 10672, Taiwan, ROC
\textsuperscript{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-\( /SI \) 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-\( /SI \) 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 \( e \), under a specified number of segments allowable. The former PA problem is called the PA-\# problem; the latter PA problem is called the PA-\( e \) problem. Solving PA-\# and PA-\( e \) problems is very important in the field of shape representation \cite{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) \cite{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. \cite{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 \cite{4} presented an \( O(n^3) \)-time algorithm for solving the PA-\( e \) 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 \cite{5}. Sharaiha

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

0031-3203/02/$22.00 \copyright \ 2002 \ Pattern \ Recognition \ Society. \ Published \ by \ Elsevier \ Science \ Ltd. \ All \ rights \ reserved.

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 $O(n^2)$-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-$\epsilon$) 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 ($O(n^2 \log n)$) algorithm for solving the PA-$\#$ (PA-$\epsilon$) 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 $L_\infty$ metrics, Barequet et al. [12] presented an $O(n^2)$-time ($O(n^2 \log n)$)-time algorithm for solving the PA-$\#$ (PA-$\epsilon$) problem. Using the $L_2$ metric, the PA-$\#$ (PA-$\epsilon$) problem can be solved in $O(n^2 \log n)$-time ($O(n^2 \log^2 n)$-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-$\epsilon$) 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-$\epsilon$) problem. For 3-D case, using the $L_1$ and $L_{ass}$ error metrics, they presented an $O(n^2)$-time ($O(n^2)$)-time algorithm for solving the PA-$\#$ (PA-$\epsilon$) problem. Using the $L_2$ metric, the PA-$\#$ (PA-$\epsilon$) problem can be solved in $O(n^2)$-time ($O(n^2 \log n)$)-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 $O(n^2)$-time algorithm for solving the PA-$\epsilon$ 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-$\epsilon$ 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-$\epsilon$ 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-$\epsilon$ algorithms are presented. Specifically, following our results, the PA-$\#$ (PA-$\epsilon$) 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-$\epsilon$ 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 $\{P_k = (x_k, y_k, z_k) \mid k = 1, \ldots, n\}$. 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

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

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

$$d(P_k, \text{line}(P_i, P_j))$$

is the Euclidean distance from the point $P_k$ to the line line$(P_i, P_j)$.

Since line$(P_i, P_j)$ can be represented by the following parametric form:

$$x = x_i + a(t - i),$$
$$y = y_i + b(t - i),$$
$$z = z_i + c(t - i),$$

where $a = x_j - x_i$, $b = y_j - y_i$, and $c = z_j - z_i$. The square error (SE) between the point $P_k$ and the line line$(P_i, P_j)$ is the minimal value of

$$f(t) = (x_k - x_i - at)^2 + (y_k - y_i - bt)^2 + (z_k - z_i - ct)^2$$

$$+ [(x_k - x_i)^2 + (y_k - y_i)^2 + (z_k - z_i)^2].$$

Let $a = (x_j^2 + y_j^2 + z_j^2)$, $b = [(x_k - x_i)x_j + (y_k - y_i)y_j + (z_k - z_i)z_j]$, 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

$$f(t) = at^2 - 2bt + c$$

$$= a \left[ \frac{t - b}{a} \right]^2 + \frac{ac - b^2}{a^2}.$$
When \( t = b/a \), we have the minimal value of \( f(t) \) and the corresponding SE is equal to

\[
\frac{a c - b^2}{a} = [(x_k - x_i)^2 + (y_k - y_i)^2 + (z_k - z_i)^2]
\]

\[- \frac{1}{\gamma_i} [(x_k - x_i) + \beta_i (y_k - y_i)]
\]

\[+ \gamma_i (z_k - z_i)]^2
\]

\[= A_{ij} x_i^2 + B_{ij} y_i^2 + C_{ij} z_i^2 + D_{ij} x_k y_k + E_{ij} y_k z_k
\]

\[+ F_{ij} x_k z_k + G_{ij} x_k + H_{ij} y_k + I_{ij} z_k + J_{ij},
\]

where

\[A_{ij} = 1 - \frac{x_i^2}{x_{ij}^2 + \beta_{ij}^2 + \gamma_{ij}^2},
\]

\[B_{ij} = 1 - \frac{\beta_{ij}}{x_{ij}^2 + \beta_{ij}^2 + \gamma_{ij}^2},
\]

\[C_{ij} = 1 - \frac{\gamma_{ij}}{x_{ij}^2 + \beta_{ij}^2 + \gamma_{ij}^2},
\]

\[D_{ij} = -\frac{2x_i \beta_{ij}}{x_{ij}^2 + \beta_{ij}^2 + \gamma_{ij}^2},
\]

\[E_{ij} = -\frac{2 \beta_{ij} \gamma_{ij}}{x_{ij}^2 + \beta_{ij}^2 + \gamma_{ij}^2},
\]

\[F_{ij} = -\frac{2 \gamma_{ij} x_i}{x_{ij}^2 + \beta_{ij}^2 + \gamma_{ij}^2},
\]

\[G_{ij} = -2x_i + \frac{2 \beta_{ij} (x_i x_i + \beta_{ij} y_i + \gamma_{ij} z_i)}{x_{ij}^2 + \beta_{ij}^2 + \gamma_{ij}^2},
\]

\[H_{ij} = -2 y_i + \frac{2 \beta_{ij} (x_i x_i + \beta_{ij} y_i + \gamma_{ij} z_i)}{x_{ij}^2 + \beta_{ij}^2 + \gamma_{ij}^2},
\]

\[I_{ij} = -2 x_i + \frac{2 \gamma_{ij} (x_i x_i + \beta_{ij} y_i + \gamma_{ij} z_i)}{x_{ij}^2 + \beta_{ij}^2 + \gamma_{ij}^2},
\]

\[J_{ij} = (x_i^2 + y_i^2 + z_i^2) - \frac{(x_i x_i + \beta_{ij} y_i + \gamma_{ij} z_i)^2}{x_{ij}^2 + \beta_{ij}^2 + \gamma_{ij}^2}.
\]

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

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

\[= A_{ij} S_1(i, j) + B_{ij} S_2(i, j) + C_{ij} S_3(i, j) + D_{ij} S_4(i, j)
\]

\[+ E_{ij} S_5(i, j) + F_{ij} S_6(i, j) + G_{ij} S_7(i, j) + H_{ij} S_8(i, j)
\]

\[+ I_{ij} S_9(i, j) + J_{ij}.
\]

where

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

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

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

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

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

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:

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

\[S_3(i, j + 1) = S_3(i, j) + z_i^2, \quad S_4(i, j + 1) = S_4(i, j) + x_i y_i,
\]

\[S_5(i, j + 1) = S_5(i, j) + y_i z_i, \quad S_6(i, j + 1) = S_6(i, j) + z_i x_i,
\]

\[S_7(i, j + 1) = S_7(i, j) + x_i, \quad S_8(i, j + 1) = S_8(i, j) + y_i,
\]

\[S_9(i, j + 1) = S_9(i, j) + z_i,
\]

for \( j < n; \)

\[S_1(i, j - 1) = S_1(i, j) - x_i^2,
\]

\[S_2(i, j - 1) = S_2(i, j) - y_i^2,
\]

\[S_3(i, j - 1) = S_3(i, j) - z_i^2,
\]

\[S_4(i, j - 1) = S_4(i, j) - x_i y_i,
\]

\[S_5(i, j - 1) = S_5(i, j) - y_i z_i,
\]

\[S_6(i, j - 1) = S_6(i, j) - z_i x_i,
\]

\[S_7(i, j - 1) = S_7(i, j) - x_i,
\]

\[S_8(i, j - 1) = S_8(i, j) - y_i,
\]

\[S_9(i, j - 1) = S_9(i, j) - z_i,
\]
for $j - 1 > i$;

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

\[ S_3(i, 1, j) = S_3(i, j) + z_i^2, \quad S_4(i, 1, j) = S_4(i, j) + x_i y_i, \]

\[ S_5(i, 1, j) = S_5(i, j) + z_i x_i, \quad S_6(i, 1, j) = S_6(i, j) + y_i, \]

\[ S_7(i, 1, j) = S_7(i, j) + z_i, \]

for $i > 1$;

\[ S_1(i + 1, j) = S_1(i, j) - x_i y_{i+1}, \]

\[ S_2(i + 1, j) = S_2(i, j) - y_i z_{i+1}, \]

\[ S_3(i + 1, j) = S_3(i, j) - z_i x_{i+1}, \]

\[ S_4(i + 1, j) = S_4(i, j) - x_{i+1}, \]

\[ S_5(i + 1, j) = S_5(i, j) - y_{i+1}, \]

\[ S_6(i + 1, j) = S_6(i, j) - z_{i+1}, \]

for $i + 1 < j$. Given the values of $S_l(i, j)$ for $l = 1, 2, \ldots, 9$, we can obtain the shortest path. The contents of these connected edges. As a result, the directed graph is the shortest path. The contents can be constructed in $O(1)$ time using the following procedure:

\[ \alpha \leftarrow x_i - x_i, \]

\[ \beta \leftarrow y_i - y_i, \]

\[ \gamma \leftarrow z_i - z_i, \]

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

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

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

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

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

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

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

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

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

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

\[ d_{ij} \leftarrow AS(i, j) + BS(i, j) + CS(i, j) + DS(i, j) + ES(i, j) + FS(i, j) + GS(i, j) + HS(i, j) + IS(i, j) + (j - i - 1)J. \]

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 $(P_i, P_j) \in E$, $1 \leq i < j \leq n$, the LISE $d_{ij}$ is equal to or less than the prespecified error tolerance $\epsilon$. Considering the first point $P_1$, we connect all pairs $P_1$ and $P_i$, for $2 \leq i \leq n$, if the LISE $d_{ij} \leq \epsilon$. 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 $2 \leq i \leq n-1$, 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 $\epsilon$, 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 $2 \leq i < j$ 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
\[ L^1[i] = L[i], \]
\[ L^{k+1}[i] = L[L^k[i]] \quad \text{for } k \geq 1, \quad (1) \]
then \( P_{W[i]} \cdots P_{i,j} \cdots P_{i,j} P_i P_i P_i \) is a shortest path from \( P_i \) to \( P_j \) for all \( 2 \leq i < j \). For example, suppose we have a \( P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4 \rightarrow P_5 \) shortest path. Then we have \( L^9[9] = 7, L^7[7] = L[9^9] = L^9[9] = 5, L[5] = L[L^9[9]] = L^9[9] = 1. \)

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 \( W[1], W[2], \ldots, W[j-1], L[2], L[3], \ldots, L[j-1] \) which satisfy the conditions in Eq. (1). For \( 1 \leq i < j \), 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 \( t, 1 \leq t < j \), if the following two conditions hold:

1. \( d_{i,j} < \varepsilon \),
2. if \( d_{i,j} < \varepsilon \) and \( 1 \leq i < j \) then \( W[t] \leq W[i] \),

then we know that \( P_{W[t]} \cdots P_{j,i} P_{t,j} P_t P_j P_i P_i \) is a shortest path from \( P_i \) to \( P_j \). We thus assign \( L[j] = t \) and \( W[j] = W[t] + 1 \). So, the shortest path \( P_{W[t]} \cdots P_{j,i} P_{t,j} P_t \) to the path \( P_{W[t]} \cdots P_{j,i} P_{t,j} P_t P_j P_i P_i \).

Since \( S_1(j - 1, j) = 0 \) for all \( l = 1, 2, \ldots, 9 \), we have \( d_{i(j-1)} = 0 \) and we can compute \( d_{i,j} \) for \( i = j - 1, j - 2, \ldots, 1 \) in \( O(j) \) time using the following procedure:

\[ S_1(j - 1, j) = 0, S_2(j - 1, j), \ldots, S_8(j - 1, j) = 0, \]
\[ d_{i(j-1)} = 0 \]
for \( i \leftarrow j - 2 \) downto 2
\{the values of \( S_1(i + j, i + j), S_2(i + j, i + j), \ldots, S_8(i + j, i + j) \) \}
Compute \( S_1(i, j), S_2(i, j), \ldots, S_8(i, j) \) \{it takes \( O(1) \) time in an incremental way\}
Compute \( d_{i,j} \) \{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 \( d_{i(j-1)}, d_{i(j-2)}, \ldots, d_{i,j} \) in \( O(j) \) time. Furthermore, we can compute \( W[j] \) and \( L[j] \) in \( O(j) \) time using the following procedure:

\[ m \leftarrow W[j - 1], t \leftarrow j - 1 \]
for \( i \leftarrow j - 2 \) downto 2
\{if \( d_{i,j} < \varepsilon \) then\}
if \( W[i] < m \) then
\[ m \leftarrow W[i]; t \leftarrow i \]
endif
endif
endfor
\[ W[j] \leftarrow m + 1; t \leftarrow j; L[j] \leftarrow t \]

Since at most \( O(n) \) space and \( O(n) \) time are required to update the current node \( P_i \), it takes \( O(n) \) space and \( O(n^2) \) time to find the path-length of the path \( P_1 \cdots P_n \) 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 \( \varepsilon \) 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 \( P^\prime \) with \( P^\prime_1 = P_1 \) and \( P^\prime_m = P_n \), where \( m \) denotes the number of points in \( P^\prime \).

**Function** find(\( \varepsilon \)) \{return the solution, the number of segments\}

\[ W[1] \leftarrow 0; L[2] \leftarrow 1 \; W[2] \leftarrow 1 \]
for \( j \leftarrow 3 \) to \( n \)
\[ m \leftarrow W[j - 1] \]
\[ t \leftarrow j - 1 \]
for \( k \leftarrow j - 1 \) downto 2
\[ i \leftarrow k - 1 \]
Compute \( S_1(i, j), S_2(i, j), \ldots, S_8(i, j) \)
Compute \( d_{i,j} \) \{if \( d_{i,j} < \varepsilon \) then\}
if \( W[i] < m \) then
\[ m \leftarrow W[i]; t \leftarrow i \]
endif
endfor
\[ W[j] \leftarrow m + 1; t \leftarrow j; L[j] \leftarrow t \]
endfor
\[ m \leftarrow W[n] \]
\[ t \leftarrow n \]
for \( i \leftarrow m + 1 \) downto 1
\[ P^\prime_i \leftarrow P_t \]
\[ t \leftarrow L[i] \]
endfor
return \( m \)

begin
find(\( \varepsilon \)) \{function call\}
end

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

**Theorem 1.** Given an LISE of \( \varepsilon \), 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 \( \sum_{k=1}^{n-1} d^2(P_k, \text{line}(P_k, P_{k+1})) \), where \( d(P_k, \text{line}(P_k, P_{k+1})) \) is the Euclidean distance from the point \( P_k \) to the line \( \text{line}(P_k, P_{k+1}) \). Since the equation of the line \( P_kP_l \) can be represented by \( a_i x + b_i y + c_i = 0 \), where \( a_i = y_l - y_k, b_i = x_k - x_l, \) and \( c_i = x_k y_l - x_l y_k \). Hence the SE from the point \( P_k \) to the line...
Lemma 1. Let \( P_1P_2 \) be equal to
\[
(a_i x_k + b_i y_k + c_i)^2 \over a_i^2 + b_i^2.
\]
Hence, the LISE can be written as
\[
\sum_{k=2}^{n-1} d^2(P_k, \text{line } P_1P_j).
\]

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

3. The proposed 3-D PA-\( \varepsilon \) 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-\( \varepsilon \) problem.

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

Lemma 1. Let \( \varepsilon_4, \varepsilon_5 \) be two LISEs, with \( \varepsilon_4 \) to obtain an approximate curve \( P'_4 \) of \( m_4 \) segments and with \( \varepsilon_5 \) to obtain an approximate curve \( P'_5 \) of \( m_5 \) segments, then
(1) if \( \varepsilon_4 < \varepsilon_5 \), then \( m_4 \geq m_5 \); (2) if \( m_4 < m_5 \), then \( \varepsilon_4 > \varepsilon_5 \).

From the discussion in Section 2, for \( 1 \leq i < j \leq n \), all the LISEs \( d_{ij} = \sum_{k=2}^{i-1} d^2(P_k, \text{line}(P_iP_j)) \), 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 \( err_1, err_2, \ldots, err_q \), 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-\( \varepsilon \) 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-\( \varepsilon \) 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-\( \varepsilon \) 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-\( \varepsilon \) problem, where \( O(n^2) \) space is used for saving those sorted LISEs, \( err_1, err_2, \ldots, err_q \). The algorithm is listed below.

Algorithm 2. PA-\( \varepsilon \)

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
\( l \leftarrow 1; \quad r \leftarrow q \); repeat
\( t \leftarrow (l + r)/2 \);
\( m \leftarrow \text{find}(e_t) \);
if \( m \leq s \) then
\( r \leftarrow t - 1 \);
else
\( r \leftarrow t \);
end if
until \( l = r \);
return(\( e_t \))
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 \( i \)th point in \( P, 1 \leq i \leq n \), suppose the constructed set of feasible edges under the specified LISE in the weighted, directed graph \( G \) is denoted by \( Err \), 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 \( 1 \leq j \leq \sqrt{n} \) 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 \( 1 \leq j \leq \sqrt{n} \) in the sorted list and these \( \sqrt{n} \) selected ele-
ments form the set \( S_1 \) with size \( O(\sqrt{n}) \). According to the result in Ref. [11], there are \( O(n) \) erri’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 \( S_1; S_2; \ldots; S_\sqrt{n} \) 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 \( O(n\sqrt{n}) \) erri’s between \( x \) and \( y \) in \( S' \).

Next, we use \( x \) and \( y \) as a filter and \( \text{err}_1, \text{err}_2, \ldots, \text{err}_q \) are fed into the filter one by one. There are \( O(n\sqrt{n}) \) erri’s retained, but they will not be saved explicitly. In real implementation, an incremental approach is still adopted. The previous \( O(n) \) erri’s in the \( O(n\sqrt{n}) \) retained erri’s are fed into the first group. Then the first group is sorted. We then select the \( j\sqrt{n} \)th elements for \( 1 \leq j \leq \sqrt{n} \) 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, \( T_2, T_3, \ldots, T_\sqrt{n} \). Collecting these \( n \sqrt{n} \) new sets, we sort these \( n \) elements. We then select the \( j\sqrt{n} \)th elements for \( 1 \leq j \leq \sqrt{n} \) in the sorted list. There exist \( O(n) \) erri’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 \( \text{err}_1, \text{err}_2, \ldots, \text{err}_q \) are fed into the filter one by one. There are \( O(n) \) erri’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-\( \varepsilon \)-problem can be solved in
$O(n^2 \log n)$ 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-$\varepsilon$ 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 complexity 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 $\# = 27, 19, 15, \text{and} 13$.

For solving the PA-$\varepsilon$ 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-$\varepsilon$ 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.
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 × 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


About the Author—KOU-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.