Polygonal approximation of closed discrete curves

Alexander Kolesnikov\textsuperscript{a,b,}\textsuperscript{*}, Pasi Fränti\textsuperscript{a}
\textsuperscript{a}Department of Computer Science, University of Joensuu, Joensuu, Finland
\textsuperscript{b}Institute of Automation and Electrometry, Novosibirsk, Russia

Received 8 December 2005; received in revised form 2 August 2006; accepted 5 September 2006

Abstract

Optimal polygonal approximation of closed curves differs from the case of open curve in the sense that the location of the starting point must also be determined. Straightforward exhaustive search would take \(N\) times more time than the corresponding optimal algorithm for an open curve, because there are \(N\) possible points to be considered as the starting point. Faster sub-optimal solution can be found by iterating the search and heuristically selecting different starting point at each iteration. In this paper, we propose to find the optimal approximation of a cyclically extended closed curve of double size, and to select the best possible starting point by search in the extended search space for the curve. The proposed approach provides solution very close to the optimal one using at most twice as much time as required by the optimal algorithm for the corresponding open curve.

© 2006 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

Keywords: Polygonal approximation; Closed contour; Dynamic programming

1. Introduction

Approximation of polygonal discrete curves aims at finding a subset of the original vertices so that a given objective function is minimized. The problem can be formulated in two ways:

(a) \textit{Minimum-distortion problem}: given an \(N\)-vertex polygonal curve \(P\), approximate it by another polygonal curve \(Q\) with a given number of segments \(M\) so that the approximation error is minimized.

(b) \textit{Minimum-rate problem}: given an \(N\)-vertex polygonal curve \(P\), approximate it by another polygonal curve \(Q\) with minimum number of segments so that the approximation error does not exceed a given maximum tolerance \(\varepsilon\).

Sometimes the problems are referred as \textit{min-c} and \textit{min-}\#\textit{ problem}, respectively [1]. In the case of open curve, the problems can be solved by dynamic programming (DP) algorithm [1–9], or by \textit{A}\*-search algorithm [10]. The time complexity of these algorithms varies from \(O(N^2)\) to \(O(N^3)\). Heuristic algorithms also exist: \textit{split} [11–13], \textit{merge} [14–20], \textit{dominant points detection} [21–24], \textit{sequential tracing} [25–27], \textit{genetic algorithms} [28–31], \textit{tabu search} [31,32], \textit{ant colony methods} [33,34], \textit{particle swarm method} [35]. These algorithms are fast but they lack the optimality.

In the case of closed curve, we have to find optimal allocation of all approximation nodes including also the starting point. A straightforward solution is to try all \(N\) vertices of \(P\) as a starting point for an algorithm designed for open curve, and to choose the one with the minimal number of segments [6] or the minimal error [8]. Actually, for the minimum-distortion problem the number of vertices to be tested is \((N - M)\) [8], but this is still \(O(N)\) for small \(M\).

It was shown in Ref. [2] that the minimum-rate problem for closed curve can be solved as the all-pairs shortest path problem for a graph of \(N\) vertices by an algorithm of complexity \(O(N^3)\). Considering that the time complexity of the optimal minimum-rate algorithm for open curve is \(O(N^2)\) [2], we have the same proportion between the processing time for closed and open curve, which is \(O(N)\).
In Refs. [4,9] the minimum-rate problem with $L_2$ [4] and $L_1$ [9] error measures were considered, and algorithms for open curve based on the shortest path problem were proposed. For the closed curve, they proposed to test each vertex of a segment of the curve as a starting point. In this way, the number of vertices to be tested is reduced from $O(N)$ to $O(N/M)$, on average, but this is still large, especially for the problems with a small values of $M$. On the other hand, the optimal selection of the starting point is more critical especially for small values of $M$.

Two fast heuristic methods were proposed in Refs. [36,37] for the starting point selection for minimum-distortion problem. First, polygonal approximation with the optimal algorithm for an open curve [8] is performed using a random starting point. A new starting point is selected among the approximating nodes of $Q$ based on certain rules, and this point is used for the second run of the optimal algorithm [8].

Another possibility would be to ignore the optimality at the first place and use the optimal algorithm developed for open curve without considering the problem of starting point.

To sum up, existing heuristics for starting point selection are sub-optimal [36,37] whereas the optimal choice is time consuming [2,4,6,8,9]. Thus, the problem of finding the optimal approximation efficiently for a closed contour is still unsolved. The existing heuristic approaches can be summarized by the fact that they iterate the search by using one of the approximation nodes as a new starting point for the subsequent iterations. After every run, however, the information from the previous iteration is lost, and the search starts from scratch.

In this paper, we propose to use the information from the previous run by performing dynamic programming search in an extended state space constructed for a twofold curve (see Fig. 1). All parts of the shortest path in the extended state space have be analyzed to find such a sub-path with conjugate states that provides minimum for the cost function in question (approximation error or the number of segments).

Pair of conjugate states in the extended state space defined by two points on the twofold curve which are produced by the same point on the input closed curve. It is expected that the proposed approach provides better solution than the existing heuristic algorithms in a comparable time because of using all available information.

The proposed approach is illustrated in Fig. 1. Approximation with DP algorithm for open curve with the vertex 1 as the starting point is given by segments 1-2-4-7-8. Optimal starting point and the corresponding approximation of the original contour can now be found by analyzing the approximation of the twofold curve $P_2$. In this example, we obtain the solution as the segments 2-4-7-2.

The rest of the paper is organized as follows. In Section 2, we give the problem formulation, recall the dynamic programming approach of [8] for minimum-distortion approximation of open curves, including approximation algorithm with reduced search [38], and present a new algorithm for the optimization of the starting point by searching in the extended state space. In Section 3, we apply the proposed approach to the case of minimum-rate problem. Experimental results and discussions are then given in Section 4, and conclusions are drawn in Section 5.

2. Minimum-distortion problem for closed curve

2.1. Problem formulation

We define a closed $N$-vertex polygonal curve $P$ in two-dimensional space as the ordered set of vertices $P = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, where the last vertex coincides with the first one: $p_N = p_1$. The problem is stated as follows: approximate the closed polygonal curve $P$ by another closed polygonal curve $Q$ with a given number of linear segments $M$ so that the total distortion (approximation error) $E(P, M)$ is minimized. The optimal approximation $Q$ of a closed curve $P$ is the
set of nodes \( \{q_1, q_2, \ldots, q_{M+1}\} \) that minimizes the cost function \( E(P, M) \) with \( L_2 \) measure:

\[
E(P, M) = \min_{\{q_m\}} \left\{ \sum_{m=1}^{M} e_2(q_m, q_{m+1}) \right\},
\]

(1)

Here \( q_{M+1} = q_1 = p_1 = p_N \). The problem also can be referred as rate-constrained problem. Error of the approximation of curve segment \( \{p_1, \ldots, p_j\} \) with the corresponding linear segment \( (q_m, q_{m+1}) \) is the sum of squared Euclidean distances from each vertex \( p_k \in \{p_1, \ldots, p_j\} \) to the line segment \( (q_m, q_{m+1}) \):

\[
e_2(p_k, p_j) = \sum_{k=i}^{j-1} d^2(k; p_i, p_j),
\]

(2)

where \( q_m = p_i \) and \( q_{m+1} = p_j \).

The Euclidean distance \( d(k; p_i, p_j) \) from a point \( p_k = (x_k, y_k) \) to the approximating line segment \( y = a_{i,j}x + b_{i,j} \) can be calculated by the following expression:

\[
d(k; i, j) = \frac{|y_k - a_{i,j}x_k - b_{i,j}|}{\sqrt{1 + a^2_{i,j}}}.
\]

(3)

The coefficients \( a_{i,j} \) and \( b_{i,j} \) of the line are calculated from the coordinates of the end points \( p_i \) and \( p_j \):

\[
a_{i,j} = (y_j - y_i)/(x_j - x_i),
\]

\[
b_{i,j} = y_i - a_{i,j}x_i.
\]

(4)

To solve the optimization task we first recall the optimal dynamic programming algorithm of Perez and Vidal [8] for open curves.

2.2. Dynamic programming approach

Optimal approximation of an open \( N \)-curve \( P_1 \) with \( M \) line segments is the set nodes \( \{q_2, q_3, \ldots, q_M\} \) that minimizes the cost function \( E(P_1, M) \):

\[
E(P_1, M) = \min_{\{q_m\}} \left\{ \sum_{m=1}^{M} e_2(q_m, q_{m+1}) \right\},
\]

(5)

here we assume that the first and the last approximation nodes are fixed: \( q_1 = p_1, q_{M+1} = p_N \).

Let us define a discrete two-dimensional state space \( \Omega_1 = \{\omega(n, m) : n = 1, \ldots, N; m = 0, \ldots, M\} \). Every point \( \omega(n, m) \) in the state space \( \Omega_1 \) represents the sub-problem of approximating of an \( n \)-vertex polygonal curve \( \{p_1, p_2, \ldots, p_n\} \) by \( m \) line segments. The state space \( \Omega_1 \) is bounded by left \( L(m) \), right \( R(m) \), bottom \( B(n) \) and top \( T(n) \) borders as follows (see Fig. 2):

\[
L(m) = \begin{cases} 
m + 1, & m = 0, 1, \ldots, M - 1, \\
N, & m = M,
\end{cases}
\]

\[
R(m) = \begin{cases} 
1, & m = 0, \\
N - M + m, & m = 1, 2, \ldots, M,
\end{cases}
\]

\[
B(n) = \begin{cases} 
0, & n = 1, \\
1, & n = 2, \ldots, N - M, \\
N - N + M, & n = N - M + 1, \ldots, N,
\end{cases}
\]

\[
T(n) = \begin{cases} 
M - 1, & n = 1, \ldots, M, \\
M, & n = N.
\end{cases}
\]

The complete problem is represented by the goal state \( \omega(N, M) \) (see dark grey square on Fig. 2). In the state space \( \Omega_1 \), we define a cost function \( C(n, m) \) as the error of the optimal approximation for the \( n \)-vertex curve \( \{p_1, \ldots, p_n\} \) by \( m \) line segments.

For solving the problem in question we have to find the weighted shortest path from the start state \( \omega(1, 0) \) to the goal state \( \omega(N, M) \). The optimization problem can be solved by the dynamic programming algorithm with the following recursive expressions for cost function \( C(n, m) \) for all \( \omega(n, m) \in \Omega_1 \):

\[
C(n, m) = \min_{L(m-1) \leq j < n} \{C(j, m - 1) + e^2(p_j, p_n)\},
\]

\[
A(n, m) = \arg \min_{L(m-1) \leq j < n} \{C(j, m - 1, j) + e^2(p_j, p_n)\}.
\]

(6)

Here \( A(n, m) \) is the parent state that provides the minimum value of the cost function \( C(n, m) \) for a state \( \omega(n, m) \), and \( e^2(p_j, p_n) = e^2(q_m, q_{m+1}) \) is the approximation error of the curve segment \( \{p_1, \ldots, p_j\} \) with the corresponding line segment \( (q_m, q_{m+1}) \). The dynamic programming algorithm is detailed in Fig. 3. The optimal path \( H(m) \) in the state space \( \Omega_1 \) is restored by backtracking from the goal state \( \omega(N, M) \) to the start state \( \omega(1, 0) \).

The time complexity of the DP algorithm is \( O(M(N - M)^2) \). For the sake of simplicity, we refer it as \( O(MN^2) \) because usually \( M \ll N \). The space complexity is defined by the area of the state space, which is \( O(MN) \).

2.3. Reduced search approach

The main idea behind the reduced search (RS) algorithm [38] is to reduce the time consuming search in the state space by exploring only a small but relevant part of the state space. To define which part should be explored, an initial solution is generated using any fast heuristic algorithm. The solution
defines a reference path in the state space $\Omega_1$. A bounding corridor of width $W$ is then constructed along this path, and the minimum cost path is searched within the corridor using DP algorithm (see Figs. 4 and 5).

Any fast heuristic algorithm for the minimum-distortion problem can be used to construct the reference path $H_0$.

We use algorithm $\text{Merge-L}_2$ [19,20] to construct a reference path because this algorithm gives smaller approximation error with measure $L_2$. For closed curve the advantage of the merge algorithm is that the result does not depend on the choice of the starting point because any two adjacent segments can be merged. The complexity of the merge algorithm is $O(N \log N)$.

The time complexity of the near-optimal RS algorithm is $O(W^2 N^2 / M)$, and the speed-up is proportional to $(W/M)^2$ in comparison to the time complexity of the full search. The space complexity is $O(WN)$. We set the corridor width as $W = \sqrt{M}$ to keep the processing time fixed to $O(N^2)$ for all $M$.

2.4. Solution for closed curve

In heuristic approach [36,37] for closed curve approximation, a new starting point is selected among the nodes obtained at the preliminary run of the optimal approximation algorithm [8]. This provides a solution that is close to the optimal one, especially for a large number of segments. However, for a small number of $M$, the solution can be far from the optimal. The main reason for non-optimality is that after the first run, the search starts again from scratch and loses the information of the previous run.

To solve the problem, we offer to perform the dynamic programming search in state space $\Omega_2$ constructed for an open $(2N - 1)$-vertex curve $P_2$, obtained from the input curve $P$. In other words, we continue the search along closed contour $P$ beyond the end point until we reach the end point second time. To reduce the processing time we apply the reduced search approach [38].

The algorithm consists of four steps:

1. Find approximation $Q_R$ for $P$ with $\text{Merge-L}_2$ algorithm, select a starting point for $P_2$ among the nodes of $Q_R$, and get a reference solution;
2. Construct the bounding corridor in the state space $\Omega_2$ using the reference solution;
3. Perform DP search in the bounding corridor to construct solutions for all sub-problems;
4. For every state in the second half of the bounding corridor ($n \geq N$) backtrack the shortest path to find a such sub-path with conjugate states that provides minimum difference of the cost function between the states. The starting point is the first node of the found sub-path.

Let us consider the last step of the algorithm in details. We backtrack the optimal path $H(\omega_g) = \{h(1), \ldots, h(m_g)\}$ from the state $\omega_g = (n_g, m_g) \in \Omega_2$ and $n_g \geq N$; $m_g \geq M$. We refer two states $\omega_1 = \omega(n_1, m_1)$ and $\omega_2 = \omega(n_2, m_2)$ on the optimal path $H(\omega_g)$ as conjugate states if (1) they represent two vertices $p_{n_1}$ and $p_{n_2}$ of twofold curve $P_2$ produced by the vertex $p_{n_1}$ in the input.
Among all shortest paths there is at least one path with conjugate states: path for the state $O(n)$ to check a single path.

The complexity of the search for optimal sub-path with conjugate states is $O(WMN)$, which is defined by the total number of goal states $O(NW)$, and the number of steps $O(M)$ to check a single path.

3. Minimum-rate problem for closed curve

3.1. Problem formulation

The minimum-rate problem is stated as follows: approximate the $N$-vertex closed polygonal curve $P$ by another closed polygonal curve $Q$ with a minimum number of line segments $M$ so that the approximation error $E(P)$ is less than a given error bound $\varepsilon$. The approximation error $E$ with measure $L_\infty$ is defined as the maximum Euclidean distance from the vertices of the curve $P$ to the approximation line segments of curve $Q$:

$$E(P) = \max_{1 \leq m \leq M} \{d(q_m, q_{m+1})\}.$$  

where $d(p_i, p_j) = \max_{1 \leq k \leq j} \{d(k; p_i, p_j)\}$, and $q_m = p_i$, $q_{m+1} = p_j$. The problem can be referred as distortion-constrained problem.

3.2. Solution for fixed end points

To find minimum-rate approximation for the closed curve $P$ with a fixed starting point, a feasibility graph $G_1 = G(P, \varepsilon)$ is constructed on vertices of the curve $P$ for the given error tolerance $\varepsilon$ [1–7] (see Fig. 6). Nodes $V = \{v_1, v_2, \ldots, v_N\}$ of the graph $G_1$ are vertices $\{p_1, p_2, \ldots, p_N\}$ of the curve $P$. A pair of nodes $u_i$ and $v_j$ is connected by an edge $e_{i,j}$ if the approximation error $d(p_i, p_j)$ for the curve segment $\{p_i, p_{i+1}, \ldots, p_j\}$ by the line segment $(p_i, p_j)$ is less than a given error tolerance: $d(p_i, p_j) \leq \varepsilon$.

The solution for the problem is the shortest path in the feasibility graph $G_1$. To find the shortest path in directed acyclic graph, we introduce one-dimensional discrete state space $\Omega_1 = \{\omega(n) : n = 1, \ldots, N\}$ of size $N$ (see Fig. 7). Every point $\omega(n)$ in the state space $\Omega_1$ represents the
Fig. 6. Feasibility graph $G_1 = G(P, \varepsilon)$ constructed on the test shape #1 with a fixed starting point for error tolerance $\varepsilon = 30$ (left), and the solution with $M = 6$ line segments as the shortest path in the graph $G_1$ (right).

**Initialization:**

C(1) = 0

**Recursion:**

FOR n = 2 TO N DO

R (n) = \infty

FOR j = n-1 TO 1 DO

IF edge(j,n) ∈ G

THEN

IF C(n) > C(j) + 1

THEN

C(n) = C(j) + 1

A(n) = j

ENDIF

ENDIF

END

END

**Backtracking to find the optimal path H:**

M = R(N)

H(M+1) = N

FOR m = M+1 TO 2 DO

H(m−1) = A(H(m))

END

Fig. 7. Pseudo code of the dynamic programming algorithm for the shortest path in the feasibility graph $G_1$.

sub-problem of the shortest path finding from the first node $v_1$ of $G_1$ to the node $v_n$. The cost function $C(n) \equiv C(\omega(n))$ for state $\omega(n)$ is given as the minimum number of edges in the shortest path. The minimum number of approximating line segments $M$ is given by cost function value for the goal state $\omega(N) : M = C(N)$. The cost function $C(n)$ is calculated for all $n = 1, \ldots, N$ by dynamic programming:

$$C(n) = \min_{1 \leq j < n} \{ C(j) + 1 \}, \quad \text{(10a)}$$

$$A(n) = \arg \min_{1 \leq j < n} \{ C(j) + 1 \}. \quad \text{(10b)}$$

After completing the DP search, optimal solution $H$ is backtracked using the array of parent states $A(n)$ (see Figs. 7, 8). The complexity of the minimum-rate algorithm is defined by the complexity of the algorithm for the feasibility graph construction, which is $O(N^2)$ [2], and by the complexity of the shortest path construction, which is also $O(N^2)$.

### 3.3. Solution for closed curve

With heuristic approach [36,37] the optimal result can be obtained by two iterations of the algorithm for open curve, and the corresponding processing time is $T_H = 2T_1$, where $T_1$ is processing time for one run of the optimal algorithm for the open curve. To make the new algorithm competitive...
The proposed algorithm for closed curve consists of three steps:

1. Construct an extended feasibility graph $G_2$ for $z$-fold curve $P_z$.
2. Construct the shortest paths to all goal nodes in the graph $G_2$.
3. For every goal node $v_k \in G_2$ backtrack the shortest path $H(v_k)$ and find such a pair of nodes $v_i$ and $v_j$ on the path $H(v_k)$ that corresponding states $o(i)$ and $o(j)$ are conjugate and the number of edges in the sub-path connecting the nodes $v_i$ and $v_j$ is minimal.

Consider the shortest path $H(v_k)$ for some goal node $v_k \in G_2$ and $k > N$. The number of edges in the sub-path between nodes $v_i$ and $v_j$ of the path is defined as the difference of the cost function for the corresponding states $o(i)$ and $o(j)$: $M = C(j) - C(i)$. In such a way, the number of approximating line segments for the closed curve $P$ for starting point $p_i$ is defined as the difference of cost function for conjugate states $o(j)$ and $o(i)$.

To solve the minimum-rate problem, we have to test the shortest paths to all goal states $v_k \in G_2$ to find such conjugate states $o(i)$ and $o(i + N - 1)$ for nodes $v_i, v_{i+N-1} \in H(v_k)$ that the difference of cost function values is minimal:

$$M = \min_{v_i, v_{i+N-1} \in H(v_k)} \{C(i + N - 1) - C(i)\}.$$

The algorithm is illustrated by Fig. 10 for test shape #1. The found pair of conjugate states $o(5)$–$o(28)$ for $v_5, v_{28} \in H(v_k)$ gives the minimal difference of cost functions: $M = C(28) - C(5) = 5$, the optimal starting point is the vertex $p_5$.

The complexity of the search for the conjugate states is defined by the number of goal states to be checked, which is $O(N)$, and the number of nodes in the shortest path. $O(M)$. In total, this yields to $O(MN)$, which is much smaller than the complexity of the core algorithm $O(N^2)$.

During the search, we have to backtrack shortest paths $H(v_k)$ for all goal nodes $v_k \in G_2$ and $k > N$. If the fold-factor $z$ is large enough, some node $v_k^*$ can belong to the optimal solution $H^{(opt)}$. Following back from the goal node
to the first node \( v_1 \) of the graph \( G_x \) we could restore the optimal solution \( H^{(\text{opt})} \) as a sub-path of the path \( H(v_1^* k) \). However, the first nodes of the restored path \( H(v_1^* k) \) do not necessarily belong to the globally optimal path \( H^{(\text{opt})} \) because its location can be distorted by non-optimal selection of the starting point, i.e. \( v_1 \notin H^{(\text{opt})} \).

The influence of the starting point selection on the subsequent approximating nodes is diminishing when we go further from the node \( v_1 \). In other words, when we continue the search in the extended graph \( G_x \), we increase the probability for finding such a goal state \( v_1^* k \in H^{(\text{opt})} \) that beginning from a node of \( H(v_1^* k) \) the location of other nodes is not affected anymore by non-optimality of the starting point, and the pair of conjugate states for nodes \( v_i, v_{i+N-1} \in H(v_1^* k) \) gives global minimum of the number of segments.

4. Results and discussion

Experiments are provided for the set of test shapes shown in Fig. 11. The test set includes images of different type (digitized contour and vector maps), smoothness and size. The shape #1 is used mainly to illustrate the algorithm. The shapes #2 and #3 are examples of a noisy and a smooth contour, respectively. The shape #4 is a large vector contour of France of size \( N = 6663 \).

The optimality of the solutions for minimum-distortion problem is measured by the fidelity \((F)\), and for minimum-rate problem by the corresponding efficiency parameters [39]. For the shapes #2 and #3, the average fidelity and efficiency are calculated with respect to the optimal solution obtained by trying all possible starting points. For the shape #4, the optimal result have been estimated by testing 200 random starting points.

4.1. Minimum-distortion problem

We compare the following algorithms for the minimum-distortion approximation:

1. FS-1: one run of the full search DP algorithm [8] with a random starting point;
2. FS-2: two runs of full search DP algorithm [8] with the starting point selection as in [36];
3. RS-\(\Omega_2\): proposed algorithm with bounded search in the extended state space.

In FS-1, the starting point is chosen randomly and the results presented here are averages over all tested starting points. In FS-2, the starting point for the second run is chosen heuristically as proposed in Ref. [36].

Experiments in Figs. 12–14 and Tables 1–3 show that the heuristic methods (FS-1 and FS-2) find the optimal result only occasionally, and the average fidelity is usually less than 100% for small \( M \).

The results of the RS-\(\Omega_2\) are significantly better and the average fidelity of the solution is usually 100%. Thus, the algorithm finds the optimal solution with rare exceptions (see Fig. 12). The superiority of the proposed algorithm RS-\(\Omega_2\) over the heuristic ones is most noticeable for small values of \( M \) (see Figs. 12–14). As for greater number of segments \((M > 20)\), when the heuristic algorithm FS-2 is more efficient, the proposed algorithm is much faster than the FS-2 and yet it provides 100% fidelity.

Fig. 11. The set of test shapes: #1: simulated contour; #2: simulated noisy contour; #3: digitized contour Leaf; #4: contour of France.

Fig. 12. The average fidelity of the minimum-distortion approximation for the shape #2 with the algorithms FS-2 and RS-\(\Omega_2\).
Fig. 13. Average fidelity of minimum-distortion approximation for the shape #3 with the algorithms FS-2 and RS-Ω2.

Fig. 14. The average fidelity of minimum-distortion approximation for the shape #4 with the algorithms FS-2 and RS-Ω2.

Table 1
The shape #2: average fidelity (F) and processing time (T) for the algorithms FS-1, FS-2, and RS-Ω2

<table>
<thead>
<tr>
<th>M</th>
<th>FS-1</th>
<th>FS-2</th>
<th>RS-Ω2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F (%)</td>
<td>T (s)</td>
<td>F (%)</td>
</tr>
<tr>
<td>5</td>
<td>82.0</td>
<td>0.012</td>
<td>95.1</td>
</tr>
<tr>
<td>10</td>
<td>91.5</td>
<td>0.023</td>
<td>99.4</td>
</tr>
<tr>
<td>15</td>
<td>88.3</td>
<td>0.035</td>
<td>99.9</td>
</tr>
<tr>
<td>20</td>
<td>97.1</td>
<td>0.045</td>
<td>99.98</td>
</tr>
<tr>
<td>30</td>
<td>97.2</td>
<td>0.058</td>
<td>99.97</td>
</tr>
</tbody>
</table>

So, performing search in the extended state space for twofold curve we can analyse much more sub-solutions to find the best one. On the other hand, restricting search in the space by the bounding corridor we reduce the processing time.

Table 2
The shape #3: average fidelity (F) and processing time (T) for the algorithms FS-1, FS-2, and RS-Ω2

<table>
<thead>
<tr>
<th>M</th>
<th>FS-1</th>
<th>FS-2</th>
<th>RS-Ω2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F (%)</td>
<td>T (s)</td>
<td>F (%)</td>
</tr>
<tr>
<td>5</td>
<td>61.2</td>
<td>0.07</td>
<td>97.9</td>
</tr>
<tr>
<td>10</td>
<td>72.7</td>
<td>0.14</td>
<td>97.0</td>
</tr>
<tr>
<td>15</td>
<td>86.5</td>
<td>0.22</td>
<td>99.6</td>
</tr>
<tr>
<td>20</td>
<td>90.4</td>
<td>0.29</td>
<td>100.0</td>
</tr>
<tr>
<td>30</td>
<td>95.5</td>
<td>0.44</td>
<td>99.95</td>
</tr>
</tbody>
</table>

Table 3
The shape #4: average fidelity (F) and processing time (T) for the algorithms FS-1, FS-2, and RS-Ω2

<table>
<thead>
<tr>
<th>M</th>
<th>FS-1</th>
<th>FS-2</th>
<th>RS-Ω2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F (%)</td>
<td>T2 (s)</td>
<td>F (%)</td>
</tr>
<tr>
<td>6</td>
<td>65.5</td>
<td>3.5</td>
<td>98.1</td>
</tr>
<tr>
<td>10</td>
<td>91.8</td>
<td>5.4</td>
<td>99.4</td>
</tr>
<tr>
<td>15</td>
<td>88.9</td>
<td>8.8</td>
<td>99.9</td>
</tr>
<tr>
<td>20</td>
<td>93.6</td>
<td>12.8</td>
<td>99.7</td>
</tr>
<tr>
<td>50</td>
<td>98.0</td>
<td>34.3</td>
<td>99.99</td>
</tr>
</tbody>
</table>

The question whether we can obtain the optimal solution for any curve and for any value of M is still open. Even though this is rare and usually happens only with very small values of M, the search can get stuck in a local minimum. Even when extending the search to space ΩK for K-fold curve (K=3, 4, ...), we cannot theoretically guarantee 100% fidelity of the solution. From practical point of view, however, the proposed algorithm gives solution very close to the optimal one for 500–5000-vertex curves in 0.1–10 s using our current computer with 3 GHz processor.

4.2. Minimum-rate problem

The second series of experiments was performed using the proposed algorithm for the minimum-rate problem. We compare the following algorithms:

(1) FS-1: one run of the full search DP in the state space Ω1 for a random starting point in P;
(2) FS-2: two runs of the full search DP in the state space Ω1 with the starting point as in [36];
(3) FS-Ω2: proposed algorithm with the DP search in the extended state space Ω2 (see Fig. 10).

For the heuristic method FS-2 with two iterations of the DP algorithm, the processing time is always 2T1, where T1 is the processing time of one run. For the tested shapes, the minimum number of segments was obtained in most cases.

In the case of the proposed algorithm, FS-Ω2, the trade-off between the processing time and efficiency can be controlled.
by changing the parameter $\alpha$ within the range from 1.0 (no optimization) up to 2.0. In practice, for the tested shapes the global minimum was achieved for $\alpha \leq 1.5$ including the worst case when a large error tolerance value $\varepsilon$ was used (see Figs. 15–17 and Tables 4–6). For smaller values of $\varepsilon$, and large number of segments, respectively, the fold-factor $\alpha$ can be even less than 1.5 (see Figs. 15–17).

5. Conclusions

We have introduced a new approach for polygonal approximation of closed curves based on the corresponding optimal dynamic programming algorithm for open curves. The information from the previous run is exploited by performing dynamic programming search in an extended state space constructed for a twofold curve.

For minimum-distortion approximation, dynamic programming search in extended state space finds the optimal approximation with a high probability. On the other hand, reducing the search by bounding corridor along a reference solution we can keep the processing time small.

Then proposed approach, DP search in the extended state space, was also propagated on the case of minimum-rate problem. The processing time of the new algorithm for closed curve is about 1.5 of the processing time for the open curve. The trade-off between the processing time and efficiency can be controlled by changing the number of processed vertices in the second run along the curve.
Fig. 17. Average efficiency of the proposed algorithm FS-Ψ for the minimum-rate problem for the shape #4 (left), and the sample result for ε=0.4 (right).

Table 4
The shape #2: average efficiency (Eff.) and processing time (T) for the algorithms FS-1, FS-2, and FS-Ψ (ε = 1.5)

<table>
<thead>
<tr>
<th>ε</th>
<th>M_min</th>
<th>FS-1</th>
<th>FS-2</th>
<th>FS-Ψ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Eff. (%)</td>
<td>T1 (s)</td>
<td>Eff. (%)</td>
<td>T2 (s)</td>
</tr>
<tr>
<td>20</td>
<td>96.1</td>
<td>0.009</td>
<td>100</td>
<td>0.017</td>
</tr>
<tr>
<td>30</td>
<td>92.6</td>
<td>0.017</td>
<td>100</td>
<td>0.036</td>
</tr>
<tr>
<td>40</td>
<td>88.9</td>
<td>0.023</td>
<td>100</td>
<td>0.048</td>
</tr>
<tr>
<td>60</td>
<td>76.5</td>
<td>0.040</td>
<td>97.1</td>
<td>0.080</td>
</tr>
</tbody>
</table>

Table 5
The shape #3: average efficiency (Eff.) and processing time (T) for the algorithms FS-1, FS-2, and FS-Ψ (ε = 1.5)

<table>
<thead>
<tr>
<th>ε</th>
<th>M_min</th>
<th>FS-1</th>
<th>FS-2</th>
<th>FS-Ψ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Eff. (%)</td>
<td>T1 (s)</td>
<td>Eff. (%)</td>
<td>T2 (s)</td>
</tr>
<tr>
<td>1.0</td>
<td>98.1</td>
<td>0.014</td>
<td>100</td>
<td>0.027</td>
</tr>
<tr>
<td>2.0</td>
<td>97.2</td>
<td>0.036</td>
<td>100</td>
<td>0.071</td>
</tr>
<tr>
<td>4.0</td>
<td>95.6</td>
<td>0.060</td>
<td>100</td>
<td>0.122</td>
</tr>
<tr>
<td>6.0</td>
<td>94.0</td>
<td>0.078</td>
<td>100</td>
<td>0.160</td>
</tr>
</tbody>
</table>

Table 6
The shape #4: average efficiency (Eff.) and processing time (T) for the algorithms FS-1, FS-2, and FS-Ψ (ε = 1.5)

<table>
<thead>
<tr>
<th>ε</th>
<th>M_min</th>
<th>FS-1</th>
<th>FS-2</th>
<th>FS-Ψ</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Eff. (%)</td>
<td>T1 (s)</td>
<td>Eff. (%)</td>
<td>T2 (s)</td>
</tr>
<tr>
<td>0.1</td>
<td>98.9</td>
<td>1.5</td>
<td>100</td>
<td>3.0</td>
</tr>
<tr>
<td>0.2</td>
<td>97.2</td>
<td>7.5</td>
<td>100</td>
<td>15.0</td>
</tr>
<tr>
<td>0.5</td>
<td>94.3</td>
<td>34.7</td>
<td>100</td>
<td>69.7</td>
</tr>
<tr>
<td>1.0</td>
<td>89.2</td>
<td>97.9</td>
<td>100</td>
<td>188.3</td>
</tr>
</tbody>
</table>

In general, the proposed approach cannot guarantee 100% optimality of the obtained approximation solution, but with the analysis of the extended state space, we can get optimal or very close to the optimal solution in relatively short time.

References


About the Author—ALEXANDER KOLESNIKOV received the M.Sc. degree in Physics in 1976 from the Novosibirsk State University, USSR, and the Ph.D. degree in Computer Science in 2003 from the University of Joensuu, Finland. From 1976 to 2003 he was a Senior Research Fellow with the Institute of Automation and Electrometry, Russian Academy of Sciences, Novosibirsk, Russia. In 2003, he joined the Department of Computer Science at the University of Joensuu, Joensuu, Finland. His main research areas are in signal and image processing, vector map processing and compression.

About the Author—PASI FRÄNTI received his M.Sc. and Ph.D. degrees in Computer Science in 1991 and 1994, respectively, from the University of Turku, Finland. From 1996 to 1999 he was a postdoctoral researcher of the Academy of Finland. Since 2000, he has been a professor in the University of Joensuu, Finland. His primary research interests are in image compression, clustering, and speech technology.