A Fast $O(N)$ Multi-resolution Polygonal Approximation Algorithm for GPS Trajectory Simplification

Minjie Chen *Student Member, IEEE*, Mantao Xu and Pasi Fränti *Senior Member, IEEE*

Abstract— **Recent advances in geo-positioning mobile phones have made it possible for users to collect a large number of GPS trajectories by recording their location information. However, these mobile phones with built-in GPS devices usually record far more data than needed, which brings about both heavy data storage and a computationally expensive burden in the rendering process for a web browser. To address this practical problem, we present a fast polygonal approximation algorithm in 2-D space for the GPS trajectory simplification under the so-called integral square synchronous distance error criterion in a linear time complexity. The underlying algorithm is designed and implemented using a bottom-up multi-resolution method, where the input of polygonal approximation in the coarser resolution is the polygonal curve achieved in the finer resolution. For each resolution (map scale), priority queue structure is exploited in graph construction to construct the initialized approximated curve. Once the polygonal curve is initialized, two fine-tune algorithms are employed in order to achieve the desirable quality level. Experimental results validated that the proposed algorithm is fast and achieves a better approximation result than the existing competitive methods.**

Index Terms—**GIS, Polygonal Approximation, Priority Queue, Reduced Search Dynamic Programming, GPS Trajectory Simplification.**

I. INTRODUCTION

Location-acquisition technologies, such as geo-positioning mobile devices, enable users to obtain their locations and record travel experiences by a number of time-stamped trajectories. In the location-based web services, users can record, then upload, visualize and share those trajectories [34]. Therefore, people are more likely to find the travel routes that interest them and acquire reference knowledge facilitating their travel from other's trajectories. However, these GPS devices usually record far more data points than necessary and these redundant data points will decrease the performance of the data collection. For example, if data is collected at 10 second intervals, a calculation in [32] shows that without any compression, 100 Mb is required to store just 400 objects for a single day. Moreover, these redundant GPS trajectories will also cause a longer uploading/ downloading time to the mobile service providers. The dense representation will also bring about a heavy burden for a web browser when rendering these trajectories on client-side. In some cases, web browsers may even get out of memory and crashed. From our experiment, it takes approximately one second for rendering 1,000 points on the map. Therefore, a fast polygonal approximation algorithm is needed for the trajectory simplification task, i.e. multiple GPS trajectory simplifications are conducted corresponding to different map scale beforehand such that the trajectories can be efficiently visualized.

In recent years, polygonal approximation in 2-dimensional space has attracted a considerable interest with a great deal of applications such as geographic information systems (GIS), computer graphics and data compression. Given a polygonal curve $P = (p_1, \ldots, p_n)$, the problem of polygonal approximation is to seek a set of ordered points, P' (a subset of P):

$$P' = (p_{i_1}, p_{i_2}, \ldots, p_{i_m}) \tag{1.1}$$

as an approximation of P, where $1 = i_1 < \ldots < i_m = N$. Polygonal approximation can be categorized into two classes of sub-problems:

a) *min-ε problem*: given N-vertices polygonal curve P and integer M, approximate a polygonal curve P' with minimum approximation error with at most M vertices.

b) *min-# problem*: given N-vertices polygonal curve P and error tolerance ε, approximate a polygonal curve P' with minimum number of vertices within the error tolerance ε.

For polygonal approximation, there exist different solutions, which vary in reduction efficiency and computational overhead. For example, an optimal algorithm provides the best reduction efficiency but causes the highest overhead $O(N^2)$ - $O(N^2\log N)$ [1-5, 10-13, 15], while solutions based on heuristics lower the computational overhead at the cost of reduced reduction rates $O(N\log N)$ [7-9]. A compromise between the optimal and heuristic solutions is the *reduced search dynamic programming* (RSDP) [17, 18, 23]. The algorithm uses a *bounding corridor* surrounding a reference curve to limit the search space during the minimizing process. In different application, different error criteria have been defined [1-5].

For the GPS trajectory simplification, since both spatial and temporal information should be considered, a number of

Manuscript received on May 11, 2011, revised on Nov 10, 2011 and Jan 2, 2012. The research was supported by Tekniikan edistämissäätiö (TES), Nokia Scholarship, MOPSI project EU (EAKR), National Natural Science Foundation of China (Grant No 61072146) and Shanghai Committee of Science and Technology, China (Grant No 10PJ1404400).

Minjie Chen was with School of Computing, University of Eastern Finland (e-mail: mchen@cs,joensuu.fi). Mantao Xu was with School of Electrical Engineering, Shanghai Dianji University (e-mail: xumt@sdju.edu.cn). Pasi Fränti was with School of Computing, University of Eastern Finland (e-mail: franti@cs,joensuu.fi)

heuristic methods have also been proposed with different error measures, such as *Trajectory simplification* (TS) [31], *top-down time-ratio* (TD-TR) [32], *Open Window* (OW) [32], *threshold-guided algorithm* [33], *STTrace* [33], *spatial join* [35], SQUISH [37] and *generic remote trajectory simplification* (GRTS) [38]. Performance evaluations are made for several traditional trajectory simplification algorithms in [36]. In these algorithms, the performance is measured on the reduction rate by the line simplification process. It is noted in [37] that there is not one algorithm that always outperforms other approaches in all situations. In the GPS trajectory simplification, the reduced data points are mostly saved directly with a fixed bit length, which is required to support both the rendering process and the effective trajectory queues in database. On the other hand, when data compression techniques are used, a better compression ratio is achieved for the GPS trajectory data [41], which is appropriate for data storage.

In this paper, we present a fast $O(N)$ time polygonal approximation algorithm for the GPS trajectory simplification. The proposed method applies a joint optimization for both *min-# approximation* using *local integral square synchronous Euclidean distance* (LSSD) criterion and *min-ε approximation* using *integral square synchronous Euclidean distance* (ISSD) criterion.

The proposed GPS trajectory simplification algorithm is implemented in a real-time application for the rendering process of the GPS trajectories on the map[1].

II. RELATED WORK

In this section, we will review the related work in the GPS trajectory simplification in several aspects, such as error measures, approximation of the polygonal curves, fine-tune solutions by reduced search and multi-resolution polygonal approximation. The contributions of the paper are also summarized at the end of each sub-section.

A. Error Measures

The primary goal of the GPS trajectory simplification techniques is to reduce the data size without compromising much of its precision. Thus, there is a need to find appropriate error measures in algorithms and performance evaluation.

In polygonal approximation, different error criteria have been defined, such as *tolerance zone*, *parallel-strip*, *uniform measure, minimum height* and *minimum width* [1-5]. Later, Meratnia [32] indicated that such algorithms were not suitable for GPS trajectory since both spatial and temporal information should be considered. Therefore, the errors were measured through distances between pairs of temporally synchronized positions, called *synchronous Euclidean distance* (SED).

The definition can be formulated as follows:

$P_i^j = (p_i, \ldots, p_j)$ is the sub-curve of P and $\overline{p_i p_j}$ is the line segment between p_i and p_j (an approximated edge in P'). For each point $p_k = (x_k, y_k)$ with time t_k ($i < k < j$) on the original GPS trajectory, its approximated temporally synchronized position $p_k' = (x_k', y_k')$ can be calculated as:

$$x_k' = x_i + \frac{t_k - t_i}{t_j - t_i} \cdot (x_j - x_i) \tag{2.1}$$

$$y_k' = y_i + \frac{t_k - t_i}{t_j - t_i} \cdot (y_j - y_i) \tag{2.2}$$

After the approximated position p_k' is determined, synchronized Euclidean distance is calculated by:

$$SED(p_k, p_k') = \sqrt{(x_k - x_k')^2 + (y_k - y_k')^2} \tag{2.3}$$

In synchronized Euclidean distance, the continuous nature of moving objects necessitates the inclusion of temporal as well as spatial properties.

Except for the above error measures, other error functions were also considered in some literatures. For example, position, speed and orientation information were all used in the *threshold-guided algorithm* [33]. In [35], a new distance-function called *spatial join* was proposed, which was bounded for spatial-temporal queries. In the area of shape matching, Fréchet distance [39] also took the continuity of shapes into account with a time complexity $O(MN)$, where M and N are the number of points correspondingly [40].

However, in most algorithms, in order to calculate the approximated error of the line segment $\overline{p_i p_j}$, at least $j - i$ distance calculations are needed. In [15], the calculation process was solved in dual space by a priority queue structure, which achieved the best processing time $O(\log N)$ with a preprocessing time $O(N\log N)$.

In this work, we further study the cost-effective spatio-temporal error measures, which can be computed in constant time. Namely, we extend *local integral square error* criterion (LISE) and *integral square error* criterion (ISE) [4-6] and derive two new error measures for the GPS trajectory simplification problems, called *local integral square synchronous Euclidean distance* (LSSD) and *integral square synchronous Euclidean distance* (ISSD). LSSD and ISSD have the same properties with LISE and ISE, i.e., they can be computed efficiently in $O(1)$ time after pre-calculating all the accumulative terms within $O(N)$ time, whereas temporal information is also considered meanwhile. The further discussion of the error measures will be made in Section III.

B. Polygonal Approximation: Optimal and Heuristic Methods

Optimal polygonal approximation algorithms are mostly implemented by incrementally constructing a *directed acyclic graph* (DAG), and therefore inevitably suffer a computational cost limitation of $O(N^2)$ at minimum[1-5, 10,11,13,30]. An advance achieved by Agarwal [12] is to combine an iterative graph algorithm and a divide-and-conquer approach, which offers the best time and space complexity of $O(N^{4/3+\delta})$ by using the L_1 *metric*, where $\delta > 0$ is an arbitrarily small constant. Later, the graph-based framework has been significantly reorganized and optimized by using two *priority queues* dynamically [15]. Albeit this approach was not proven to reduce the time complexity in theory, it provided remarkable improvement in the processing time in practice.

In real-time application, quadratic time complexity maybe too high and, therefore, most applications utilized a class of heuristic methods in order to achieve near-linear time

[1] Two datasets are considered, which are MOPSI dataset (http://cs.joensuu.fi/mopsi) and geolife dataset [34]

complexity. A set of well known heuristic algorithms are *split* and *merge* approaches [7-9]. The split algorithms divide the segment causing biggest deviation, whereas the merge algorithms combine the pair of segments with the least deviation. The classic *Douglas–Peucker* split algorithm [7] can be implemented in $O(N\log N)$ time on average, while its worst-case time complexity is $O(N^2)$. Later Hershberger et al. [8] showed that it can be implemented in $O(N\log^* N)$ time, where log* denotes the iterated logarithm function. Respectively, Pikaz et al. [9] proposed a merging algorithm with $O(N\log N)$ time complexity. These heuristic methods are of low time complexity but may lead to an undesirable approximation result. Note that topological and geometric properties are also considered as an important constraint in the simplification process in GIS applications. In [44], *simple Detours (SD) heuristic* was proposed, where no new vertices would be introduced after the approximation process.

In the GPS trajectory simplification, a number of algorithms have also been well studied and developed and most of them are heuristic methods. In [32], a trajectory simplification algorithm is implemented greedily by a so-called *Opening window* approach. *Synchronous Euclidean distance* is also defined and applied by incorporating the time dimension, instead of the original perpendicular distance. In [33], the parameters including coordinates, speed and orientation are all considered in calculating the safe area of the next point, which is called as *threshold-guided algorithm*. Indeed, all these algorithms solve the min-# problem in a greedy manner, of which the time complexity is $O(N^2)$. *STTrace sampling algorithm* [33] is also implemented using a bottom-up strategy where the synchronous Euclidean distance is minimized in each step. In [38], *generic GRTS protocol* combines optimal and heuristic algorithms [1, 32], which allows a trade-off between the computational complexity and the reduction efficiency. Recently, a new simplification algorithm SQUISH [37] is proposed based on the priority queue data structure which preserves speed information at a much higher accuracy. In [31], *trajectory simplification* algorithm is proposed, where different point headcounts are assigned in terms of the product of the average heading change and the distance of each segment. After that, min-ε problem is solved in each segment by using a local weighting process in $O(N\log M)$ time. However, as the distances of neighborhood points are used instead of perpendicular distance in the simplification procedures, the algorithm is not robust when the sampling frequency is not uniform.

Graph-based methods can achieve better approximation result than those heuristic ones but at a higher computational cost. Therefore, in the initialization process of the proposed solution, graph-based methods are used and further speeded up by both a novel priority queue structure and a stopping search criterion, which leads to $O(N^2/M)$ time complexity and $O(N)$ space complexity. Here, N and M are the number of the points for the input and output GPS trajectories respectively. However, using a stopping search criterion will cause a trade-off of the optimality. This will be introduced in Section IV.

C. Fine-tune by Reduced Search

For the GPS trajectory simplification, optimal algorithms provide the best reduction efficiency but cause the highest overhead, while solutions based on heuristics lower the computational overhead at the cost of worse reduction rates. A compromise between the optimal and heuristic solutions is the *reduced search dynamic programming* [17, 18, 23]. The algorithm uses a *bounding corridor* surrounding a reference curve or initialized curve in the state space, followed by a limited search for the minimum cost path. This idea is presented and known as Sakoe-Chiba band [42], which has been extensively used in *Dynamic Time Wrapping* (DTW) approaches dealing with the similarity calculation of time-series [43].

If the initialized curve is evenly distributed in the state space, the time complexity for RSDP is ideally $O(W^2N^2/M^2)$, where W is the width of bounding corridor. We will also prove that the expected time complexity for RSDP is still achievable as $O(W^2N^2/M^2)$ even if the precondition of even distribution is not satisfied. In particular, if the number of vertices for the approximated curve is proportional to that of the input curve, namely, $M = N/c$, a linear time complexity can be achievable for RSDP. This will be later shown to be an important property when selecting *bottom-up* approaches for the multi-resolution case. However, the main difficulty of RSDP is that a large corridor bound and many iterations are needed in order to achieve a desirable solution when the approximated curve is poorly initialized, which causes a high computational cost.

In this paper, we extend the reduced search dynamic programming and employ two fine-tune algorithms to minimize both the number of output points M and the approximated error ε, which leads to a time complexity $O(WN^2/M)$ and $O(N^2/M)$ correspondingly. The fine-tune algorithms are speeded up by lifting the vertex position in the tree structure as well as solve the e*quivalent solution problem*. This will be discussed in Section V.

In Section III-V, the UK map with 10911 point (See Fig. 7.1) will be selected as an example to demonstrate the proposed algorithm.

D. Multi-resolution Polygonal Approximation

Multi-resolution polygonal approximation can be applied for scalable representation and compression of vector maps in GIS system [19, 20]. For solving min-ε problem, two heuristic approaches split (top-down) and merge (bottom-up) are known with a time complexity of $O(N\log N)$. Split and merge are applied locally and can often result in undesirable approximation results in the later hierarchy process.

Optimal split algorithm (OSA) is proposed in [21], where the optimal approximation at the higher resolution level is achieved using the result of lower (previous) resolution level. This provides resolution hierarchy in sequential order ($1\rightarrow2\rightarrow4\rightarrow\ldots$) but at a cost of $O(N^2)$ time complexity.

In [22], a bottom-up multi-resolution algorithm for min-ε problem is proposed with near-linear time complexity. Min-ε problem is solved using the fine resolution as input for approximating the corresponding coarser resolution iteratively ($N\rightarrow N/2\rightarrow N/4\rightarrow\ldots$). For each scale, simplified reduced search dynamic programming (RSDP) is also incorporated. As integral square error criterion is used, the approximation error between two vertices in any resolution level can be calculated in a constant time according to the pre-calculating cumulative summations in the original curve, see Section III.

Although the bottom-up approach [22] is computationally efficient, this approach can only solve min-ε problem. In practice, in order to progressively display the GPS trajectory data, we need to approximate a number of approximated results with corresponding error tolerance for each resolution, which is considered as a min-# problem. Moreover, the reduced search algorithm is a fine-tune method which needs an initial curve beforehand. If the curve is not well initialized, a number of iterations are needed to obtain the near-optimal result.

In this work, a *bottom-up multi-resolution* approach is proposed with linear time and space complexity, which implements the algorithms in Section III-V for each intermediate resolution. This will be discussed in Section VI.

III. ERROR MEASURE: FROM LISE TO LSSD

In order to improve the computational efficiency, two error measures, which is called as *integral square error* (ISE) and *local integral square error* (LISE) [4- 6], are jointly used for approximating polygonal curves:

$$f_{ISE}(P') = \sum_{j=1}^{M-1} \delta(P_{i_j}^{i_{j+1}}) \tag{3.1}$$

$$f_{LISE}(P') = \max_{1 \le j < M} \delta(P_{i_j}^{i_{j+1}}) \tag{3.2}$$

where the error δ can be calculated by:

$$\begin{aligned}
&\delta(P_i^j) = \sum_{i<k<j} d^2(p_k, \overline{p_i p_j}) \\
&= \frac{1}{a_{ij}^2 + b_{ij}^2} \sum_{i<k<j} (a_{ij} \cdot x_k + b_{ij} \cdot y_k + c_{ij})^2 \\
&= ((j-i-1) \cdot c_{ij}^2 + a_{ij}^2 \cdot (S_{x^2}^{j-1} - S_{x^2}^i) + b_{ij}^2 \cdot (S_{y^2}^{j-1} - S_{y^2}^i) \\
&+2 \cdot a_{ij} b_{ij} \cdot (S_{xy}^{j-1} - S_{xy}^i) + 2 \cdot a_{ij} \cdot c_{ij} \cdot (S_x^{j-1} - S_x^i) \\
&+2 \cdot b_{ij} \cdot c_{ij} \cdot (S_y^{j-1} - S_y^i)) / (a_{ij}^2 + b_{ij}^2)
\end{aligned} \tag{3.3}$$

here d is the perpendicular distance from p_k to $\overline{p_i p_j}$. $a_{ij} = (y_j - y_i)$, $b_{ij} = (x_i - x_j)$, $c_{ij} = y_i x_j - x_i y_j$ and S_x, S_y, S_{x2}, S_{y2}, S_{xy} are the accumulated sums of the x, y coordinates on the curve P respectively:

$$\begin{aligned}
&S_x^i = \sum_{j=1}^i x_j,\ S_y^i = \sum_{j=1}^i y_j,\ S_{x^2}^i = \sum_{j=1}^i x_j^2, \\
&S_{y^2}^i = \sum_{j=1}^i y_j^2,\ S_{xy}^i = \sum_{j=1}^i x_j y_j,\ i = 1, \ldots, N
\end{aligned} \tag{3.4}$$

The main advantage of the integral square error criterion is that the approximation error $\delta(P_i^j)$ can be obtained efficiently in $O(1)$ time after pre-calculating all the accumulative terms within $O(N)$ time (see equation 3.3) [4, 16], An example of calculating ISE and LISE is illustrated in Fig. 3.1.

Although LISE and ISE criterion are computationally efficient, time information is not considered. For the simplification of the GPS trajectories, we extend LISE and ISE criteria and derive two new error measures, called *local integral square synchronous Euclidean distance* (LSSD) and *integral square synchronous Euclidean distance* (ISSD), which have the same properties with LISE and ISE:

$$f_{ISSD}(P') = \sum_{j=1}^{M-1} \delta_{SED2}(P_{i_j}^{i_{j+1}}) \tag{3.5}$$

$$f_{LSSD}(P') = \max_{1 \le j < M} \delta_{SED2}(P_{i_j}^{i_{j+1}}) \tag{3.6}$$

$$\begin{aligned}
&\text{Here } \delta_{\mathrm{SED2}}(P_i^j) = \sum_{i<k<j} SED^2(p_k, p_k') \\
&= (c_1^2 + c_3^2)(j-i-1) + (c_2^2 + c_4^2)(S_{t^2}^{j-1} - S_{t^2}^i) \\
&+2(c_1 c_2 + c_3 c_4)(S_t^{j-1} - S_t^i) + (S_{x^2}^{j-1} - S_{x^2}^i) \\
&+(S_{y^2}^{j-1} - S_{y^2}^i) - 2c_1(S_x^{j-1} - S_x^i) - 2c_3(S_y^{j-1} - S_y^i) \\
&-2c_2(S_{xt}^{j-1} - S_{xt}^i) - 2c_4(S_{yt}^{j-1} - S_{yt}^i)
\end{aligned} \tag{3.7}$$

Here $c_1 = \dfrac{x_i t_j - x_j t_i}{t_j - t_i}$, $c_2 = \dfrac{x_j - x_i}{t_j - t_i}$, $c_3 = \dfrac{y_i t_j - y_j t_i}{t_j - t_i}$, $c_4 = \dfrac{y_j - y_i}{t_j - t_i}$

S_x, S_y, S_t, S_{x2}, S_{y2}, S_{t2}, S_{tx} and S_{ty} are the accumulated sums of the x, y and t on the GPS trajectory respectively:

$$\begin{aligned}
&S_x^i = \sum_{j=1}^i x_j,\ S_y^i = \sum_{j=1}^i y_j,\ S_t^i = \sum_{j=1}^i t_j,\ S_{x^2}^i = \sum_{j=1}^i x_j^2, \\
&S_{y^2}^i = \sum_{j=1}^i y_j^2,\ S_{t^2}^i = \sum_{j=1}^i t_j^2,\ S_{tx}^i = \sum_{j=1}^i t_j x_j,\ S_{ty}^i = \sum_{j=1}^i t_j y_j,
\end{aligned} \tag{3.8}$$

Computation of the above approximation errors $\delta_{\mathrm{SED2}}(P_i^j)$ also takes $O(1)$ time with an $O(N)$ time accumulated sum pre-calculation. The proof of the LSSD and ISSD calculation can be seen in Appendix.

In the following sections, ISE and LISE criteria will be used for the approximation of the polygonal curves, while LSSD and ISSD criteria will be used for the GPS trajectory simplification.

IV. MIN-# INITIALIZATION FOR GPS TRAJECTORY SIMPLIFICATION

For the *min-# problem*, Imai and Iri's graph-based approach [1] comprises two essential steps: constructing DAG and shortest path search by *breadth-first traversal* (BFT). In order to construct a DAG, $N(N - 1)/2$ approximation errors are calculated for every pairs of vertices, and thus the time complexity for initializing the solution for min-# problem is $O(N^2)$ if LISE or LSSD criterion is applied.

In this work, we revisit two computationally efficient improvements for min-# problem. The first improvement is to reduce the computational cost of DAG construction by maintaining two *priority queue* structures [15, 29]. The reason is that there is no need to construct the graph G explicitly and only edges visited by the BFT are included. For simplicity, we define a term *number of links* $L(p_i)$ to denote the minimum number of line segments to connect the starting vertex p_1 to p_i under given error tolerance ε:

$$L(p_i) = \min(L(p_k)) + 1,\ s.t.\ \delta(P_k^i) < \varepsilon,\ 1 \le k < i \tag{4.1}$$

where the initial condition is set as $L(p_1) = 0$. Suppose all the vertices with k links are firstly identified by shortest path search, which is maintained by a priority queue $V_{L(k)}$ in descending

Fig. 3.1. An example of calculating ISE, LISE, LSSD and ISSD. *Given* $P = (p_1, p_2, p_3, p_4, p_5, p_6)$, and the approximated curve $P' = (p_1, p_4, p_6)$, where p_2', p_3' *and* p_5' *are the approximated temporally synchronized position.* ISE is estimated as $d_1^2+d_2^2+d_3^2$ and LISE is estimated as $d_1^2+d_2^2$ (left). Meanwhile, ISSD is estimated as $d_1'^2+d_2'^2+d_3'^2$ and LSSD is estimated as $d_1'^2+d_2'^2$(right).

order. The next search will be performed on the remaining unvisited vertices set S_u by testing if they have an edge connecting with vertices in $V_{L(k)}$ (i.e. approximation error lower than given tolerance ε), which is called as *edge tests* here. These connected vertices will be removed from unvisited vertices set S_u and enqueued in the priority queue $V_{L(k+1)}$. Suppose two vertices $p_a, p_b \in V_{L(k)}$ with $a > b$, if $\exists\, p_c \in S_u$ and $\delta(P_b^c) < \varepsilon$, then p_c will be removed from S_u such that the edge test between p_a and p_c can be avoided. Moreover, edge tests are also avoided for the vertices with the same number of links. After all the unvisited points have been tested between $V_{L(k)}$ and S_u, in next step, the vertices in $V_{L(k+1)}$ will be used as the starting points for edge tests. The shortest path search will be terminated when the last vertex p_n is connected to p_1. Albeit the priority-queue based search is not able to mitigate the worst case time complexity, it turns out that a number of edge tests are greatly saved.

The second improvement is to apply a stopping criterion in the shortest path search, which is efficient in the case of low error tolerance. For example, a good stopping criterion has been proposed for *tolerance zone criterion* [11] by maintaining the intersection of two cones. An alternative solution has also been proposed in [15, 28] by verification in dual space. Both of the implementations hold the optimality for solving the min-# problem. To pursue the best computational cost savings as possible for LISE/LSSD criteria, a simple stopping criterion is applied in edge tests by utilizing a preset *high threshold,* e.g. two times of given tolerance [17]. Edge tests for the subsequent vertices in the unvisited vertices set will be omitted once the approximation error becomes larger than a given high threshold. Applying a stopping criterion leads to a significant improvement to a time complexity of $O(N^2/M)$ but the optimality is not guaranteed. To overcome this difficulty, we extend our effort in improving the robustness of the stop search criterion. Instead of using a fixed high threshold, we adopt the error tolerance of the next coarser resolution as a high threshold in the multi-resolution implementation, of which the robustness has been validated by experiments; see Section VI for more discussion.

We combine both the advantage of the priority queue structure and the stopping criterion to achieve the most computationally efficient implementation in the initialization of min-# problem. Accordingly, the output is a tree structure; see Fig. 5.2(left). The pseudo code is given in Fig. 4.2. Both the theoretical proof and the experiments are given for the complexity analysis of the proposed initialization algorithm.

Theorem 1. The proposed initialization algorithm for solving the min-# problem under LISE/LSSD criterion leads to an expected time complexity of $O(N^2/M)$ and a space complexity of $O(N)$ respectively.

Proof. See Appendix.

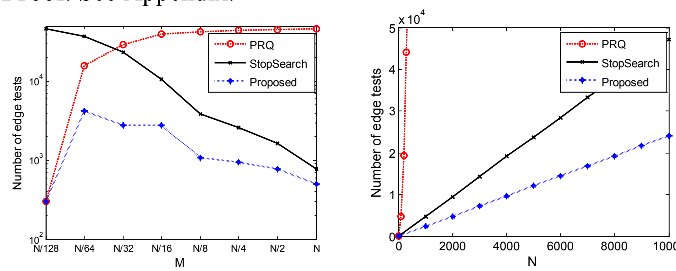

Fig. 4.1. Number of *edges tests* for solving *min-# problem* under different error tolerance (left) and with different number of input vertices (right) for UK map (Curve II). In the left figure, the resulting number of output vertices M is shown in x axis instead of the given error tolerance.

ALGORITHM I, MIN-# INITIALIZATION

```
INPUT
P={p1, p2,…pN}:  original polygonal curve
th: LISE/LSSD error tolerance
hth: high threshold of error tolerance
OUTPUT
T : Tree structure

V1 ← {1}
V2 ← Ø
  Su ← {2,…,N}
  REPEAT
     REPEAT
        maintainPRQ()
     UNTIL V1 = Ø
     V1 ← V2
     V2 ← Ø
  UNTIL pN ∈ V1

  Procedure maintainPRQ()
  ind1 ← dequeue(V1)
  FOR ind2 = Su (1) TO Su (end)
     dist ← δ(P_ind1^ind2)
     IF dist ≤ th
        Su ← { Su \ ind2 }
        V2 ← enqueue(ind2)
        Update T by ind1.child ← ind2, ind2.father ←
ind1
     ELSE IF (dist> hth)
        break
     ENDIF
  ENDFOR
  RETURN V1, V2, Su, TT
```

Fig. 4.2. Pseudo code of min-# initialization

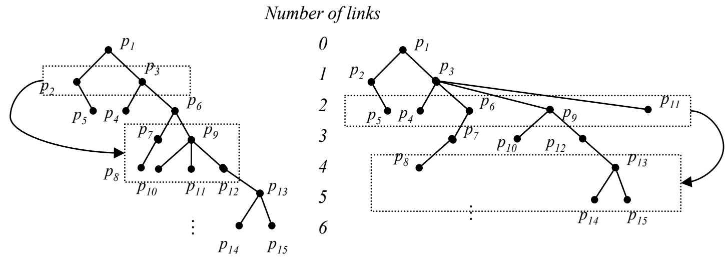

Fig. 5.2. An example of reducing the number of output vertices with a width of bounding corridor $W = 2$: the target vertices with 1 links (left) and target vertices with 2 links after tree structure updated given $\delta(P_3^9) < \varepsilon$, $\delta(P_3^{11}) < \varepsilon$ (right). Left figure is a typical example after the initialization step for Algorithm I.

In the graph-based initialization algorithm, the main bottleneck is the cost of edge tests (calculating the edge approximation errors, line 22 of Algorithm I) during graph construction. In order to evaluate the computational improvement achieved by the proposed algorithm, the number of edge tests is calculated and treated as an indicator of the computational efficiency in Fig. 4.1. Here "PRQ" represents the previous graph-based polygonal approximation algorithm using priority queues structure [15, 29], "StopSearch" is the stopping criterion using a predefined high threshold [17]. It can be observed that the proposed algorithm is able to combine the computational advantages of both two algorithms.

V. FINE-TUNE THE INITIAL APPROXIMATED RESULT

As a stopping criterion is incorporated in Algorithm I (line 27) to reduce the computational cost in the initial approximation process, the optimality is not guaranteed. Thus, two fine-tune algorithms are introduced in this section in order to improve the approximation performance. Both the number of vertices and the ISE/ISSD are minimized.

A. Minimizing Number of Vertices

To the benefit of best computational efficiency, the initialization in Algorithm I for min-# problem is a compromise of the optimality for minimizing the number of vertices. In order to mitigate the limited optimality, we need to minimize the number of vertices based on the initialized curve so that a better result can be achieved. The reduced search algorithm (RSDP) can be utilized for minimizing the number of vertices but it leads to $O(W^2N^2/M)$ time complexity. To speed up the procedure, we exploit a new fine-tune method at a time complexity of $O(WN^2/M)$ instead, which is achieved by lifting the vertex position in the output tree structure after the initialization step in Algorithm I. The pseudo code is given in Fig. 5.1.

A graphical illustration is demonstrated in Fig. 5.2 of lifting vertex position: starting from vertex p_1 with 0 links, at each iteration (line 11-30 in Algorithm II), edge tests are performed to verify if the approximation error is less than the given tolerance between the currently processed vertices with k links and those target vertices with $\{k + 2, \dots, k + W + 1\}$ links. An example is given in Fig. 5.2 (left) when the width of the bounding corridor is $W = 2$. Suppose p_2 and p_3 are the vertices with 1 link, all the vertices with 3 links (p_7 and p_9) and 4 links (p_8, p_{10}, p_{11}, p_{12}) are chosen as the target vertices for edge tests. If the connected edge exists, the tree structure is updated by lifting the target vertices (line 22-24). The process of updating the tree structure can be done recursively, see Fig. 5.2 (right).

The proposed fine-tune algorithm provides the following advantages over the original reduced search approach for min-# problem. Firstly, calculation of the approximated errors between any pair of vertices with adjacent number of links is unnecessary and can be omitted. Secondly, once the tree structure is updated by the lifting operations, edge tests for those lifted vertices are also avoided.

ALGORITHM II, MINIMIZING THE NUMBER OF OUTPUT VERTICES

```
1.  INPUT
2.  P={p1,p2,…pN}: original polygonal curve
3.  T: tree structure
4.  W: width of bounding corridor
5.  th: LISE/LSSD error tolerance
6.  OUTPUT
7.  T: updated tree structure
8.
9.  PRQ1 ← {1}
10.   REPEAT
11.     PRQ2 ← child nodes of all vertices in PRQ1
12.     Vtar{k} ← Ø, k ∈ {0, 1, 2, …, W}
13.     Vtar {0} ← PRQ2
14.     FOR k = 1 TO W
15.       Vtar {k} ← child nodes of all vertices in Vtar {k-1}
16.     ENDFOR
17.     FOR k = W TO 1
18.       FOR ind1 = PRQ1(1) TO PRQ1(end)
19.         FOR ind2 = Vtar {k}(1) TO Vtar {k}(end)
20.           dist← δ(P_ind1^ind2)
21.           IF dist ≤ th
22.             Vtar {k} ← { Vtar {k} \ind2 }
23.             PRQ2 ← enqueue(ind2)
24.             Update T by ind1.child ← ind2, ind2.father ← ind1
25.           ENDIF
26.         ENDFOR
27.       ENDFOR
28.     ENDFOR
29.     PRQ1 ← PRQ2
30.     PRQ2 ← Ø
31.   UNTIL pN ∈ PRQ1
```

Fig. 5.1. Pseudo code of minimizing the number of vertices

Theorem 2. The proposed algorithm for the output vertex reduction under LISE/LSSD criterion has an expected time complexity of $O(WN^2/M)$ and a space complexity of $O(N)$ respectively. The original reduced search dynamic programming method has an expected time complexity of $O(W^2N^2/M)$.

Proof. See Appendix.

Intuitively, the fine-tune algorithm can also be done iteratively. However, since the graph-based method has already achieved an ideal initial approximation, according to our experiments, optimal results can be derived in most cases by setting $W = 2$ with one iteration. The main bottleneck here is also the number of edge tests (line 20 in Algorithm II). In Fig.

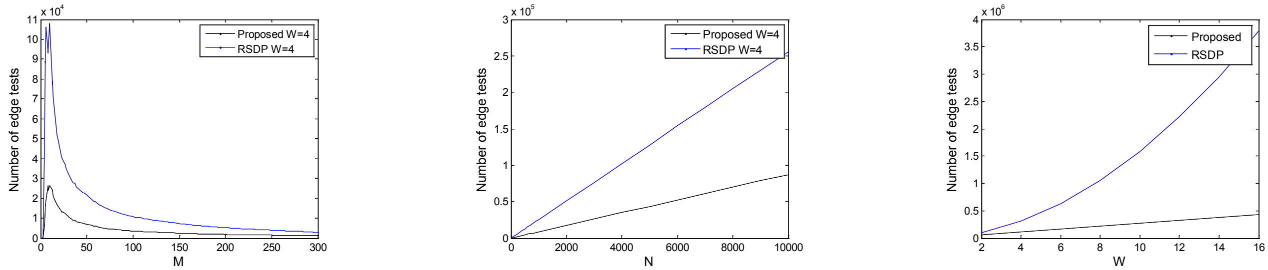

Fig. 5.3. Number of edge tests in minimizing the number of output vertices. Different error tolerance (left), different number of input vertices (middle) and different width of bounding corridor W (right) are tested on UK map (Curve II). In left figure, the resulting number of output vertices M is shown in x axis instead of the given error tolerance. In left and right figure, the input polygonal curve is the UK map with N=10911.

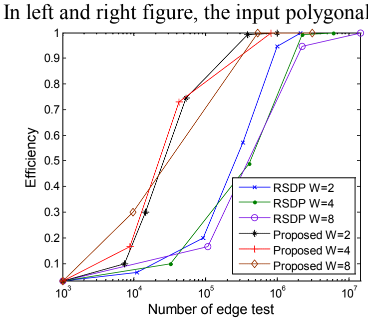

Fig. 5.4. Performance comparisons of the proposed fine-tune algorithm and RSDP when the original curve is selected as the initial curve directly. UK map (Curve II) is tested with $\varepsilon = 0.01$.

5.3, the actual time cost is evaluated by calculating the number of edge tests against the three parameters: width of bounding corridor W, number of output vertices M, and number of input vertices N. To further demonstrate the efficiency of the proposed fine-tune algorithm, we also evaluated the performance when the initialization step is skipped and the original polygonal curve is selected as input directly. We can observe that the optimal result is achieved with less than five iterations by the proposed fine-tune algorithm and the number of edge tests is much less than the reduced search dynamic programming, which is shown in Fig. 5.4.

B. Minimizing global integral square error

After the number of vertices is reduced by LISE/LSSD criterion, a so-called *equivalent solution problem* may still exist. In other words, given an error tolerance ε, a number of solutions for min-# approximation can be achieved with the same number of output vertices M but they lead to distinct approximation performance, see Fig. 5.6. Hence, an additional post-processing step based on ISE/ISSD criterion is needed in order to find the best approximation result among these equivalent solutions, which can also be considered as a min-ε problem. The pseudo code can be seen in Fig. 5.8.

After executing Algorithm II which effectively updates the tree structure, additional post-processing is performed to identify the best possible curve P' with the minimum ISE/ISSD:

$$P' = \arg\min f_{ISE/ISSD}(P'), s.t. f_{LISE/LSSD}(P') < \varepsilon \tag{5.1}$$

This can be solved by dynamic programming in terms of the following recursive expression:

$$\begin{aligned} &D(p_j) = \min(D(p_i) + \delta(P_i^j)), 1 \le i < j \\ &A(p_j) = \arg\min_i (D(p_i) + \delta(P_i^j)), 1 \le i < j \\ &s.t.\ \delta(P_i^j) < \varepsilon, L(p_j) = L(p_i) + 1 \end{aligned} \tag{5.2}$$

where $A(p_j)$ is the parent vertex of p_j and $D(p_j)$ is the accumulated ISE/ISSD.

Theorem 3. Minimization of global integral square error under the constraint of local integral square error has an expected time complexity of $O(N^2/M)$ and a space complexity of $O(N)$.

Proof. See Appendix.

From Theorem 3, the minima can be found in $O(N^2/M)$ time and no iterations are needed. The above minimization offers a significant improvement (theoretically W^2 time faster) over the original RSDP that has a time complexity of $O(W^2N^2/M)$. In Fig. 5.7, the histograms of the approximated LISE are plotted before and after the fine-tune step. As the ISE is the sum of LISE for all the approximated segments, we can observe that ISE is significantly reduced while LISE has not increased after the fine-tune process.

ALGORITHM III, FIND BEST SOLUTION USING INTEGRAL SQUARE ERROR CRITERION

```
INPUT
P={p1,p2,…pN} ← original polygonal curve
T ← tree structure
th ← LISE/LSSD error tolerance
OUTPUT
P' ← approximated curve

E ← {0, ∞, ∞, ….., ∞}, N x 1 vector storing the approximated error
A ← {0, 0, 0, ….., 0}, N x 1 vector for backtracking
  H ← {0, 0, 0, ….., 0}, M x1 vector
  V1 ← {1}
  M ← 1
  REPEAT
    V2 ← child nodes of all the vertices in V1
    FOR ind1 = V1(1) TO V1(end)
      FOR ind2 = V2(1) TO V2(end)
        dist ← δ(P_ind1^ind2)
        IF (E(ind1) + dist < E(ind2)) && (dist ≤ th)
          A(ind2) ← ind1
          E(ind2) ← E(ind1) + dist
        ENDIF
      ENDFOR
    ENDFOR
    V1 ← V2
    M ← M + 1
  UNTIL E(N)!= ∞
  //Backtracking
  H(M) ← N
  FOR m = M TO 2 DO
    H(m-1) ← A(H(m))
  ENDFOR
  P' ← P(H)
```

Fig. 5.8. Pseudo code of minimizing integral square error

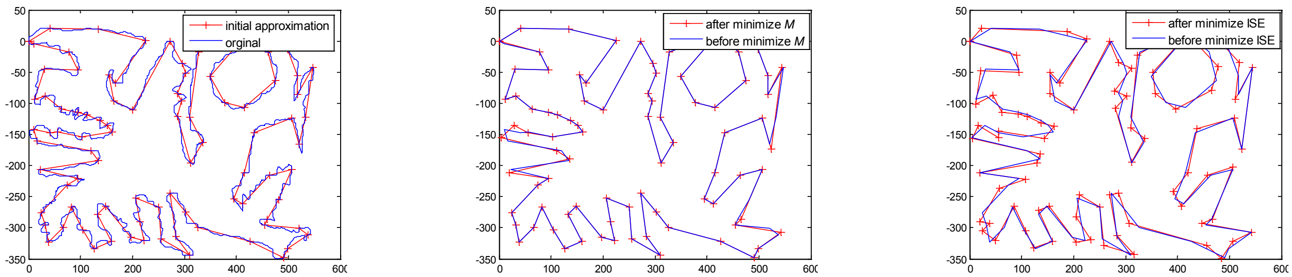

Fig. 5.5. An example of the proposed polygonal approximation. Curve I [25] is used with $\varepsilon = 1500$ and the optimal solution is $M_{opt} = 86$. Initial approximated curve is obtained with $M'' = 91$ (left). Approximated curve ($M = 86$) is obtained after reducing number of output vertices with $f_{ISE}(P') = 1.04{\cdot}10^5$ (middle). The final solution is obtained by minimizing ISE with $f_{ISE}(P') = 4.88{\cdot}10^4$ (right).

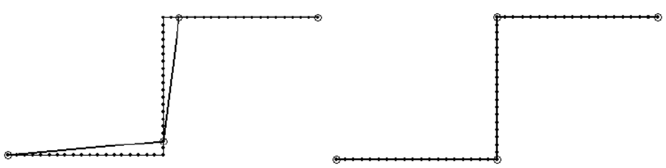

Fig. 5.6. An example of equivalent solutions in *min-# approximation*, where both approximated curves meet the error tolerance $\varepsilon = 2$ and have same output $M = 4$.

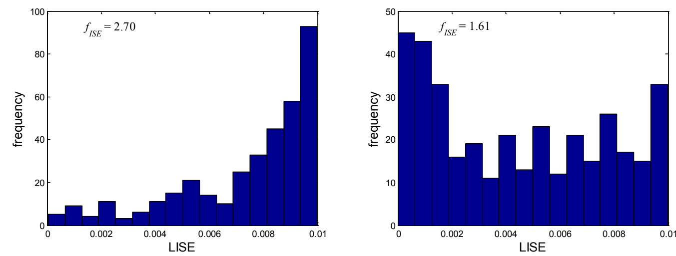

Fig.5.7. LISE distribution of all the approximated edges with $\varepsilon = 0.01$ for Curve II. A best approximation result with $f_{ISE}(P') = 1.61$ (right) is found from all the *equivalent solutions*, which is much lower than result after Algorithm II with $f_{ISE}(P') = 2.70$ (left). Both approximation results have $M = 364$.

TABLE I. SUMMARY OF THE PROPOSED POLYGONAL APPROXIMATION ALGORITHM. * REPRESENTS THE INITIAL CURVE IS EQUALLY PARTITIONED

STEP	TIME COMPLEXITY: RSDP	TIME COMPLEXITY: PROPOSED	IMPROVEMENTS AND CONTRIBUTIONS
I	$O(N^2/M)$	$O(N^2/M)$	Combine priority queue and stopping criterion to reduce the computation cost. Proof is given.
II	$O(W^2N^2/M)$*	$O(WN^2/M)$	Time complexity is reduced. Proof is given.
III	$O(W^2N^2/M)$*	$O(N^2/M)$	Time complexity is reduced. Proof is given.

C. *Summary of the Near-optimal Approximation Algorithm*

Polygonal approximation algorithm for the joint optimization of both min-# approximation using LISE/LSSD criterion and min-ε approximation using ISE/ISSD criterion has been introduced as a three step procedure: initialization of min-# problem, minimizing the number of output vertices, and minimizing integral square error. Proof has been given that the proposed algorithm has expected time complexity of $O(N^2/M)$ and space complexity of $O(N)$, and experiment results have demonstrated that the practice is consistent with the theoretical analysis. An example of the proposed algorithm is shown in Fig. 5.5. The improvement of the time complexity is also summarized in Table I.

VI. LINEAR TIME MULTI-RESOLUTION POLYGONAL APPROXIMATION METHOD

In order to further improve the computational efficiency, in this section, a *bottom-up multi-resolution polygonal approximation* approach is proposed by implementing Algorithm I and Algorithm III in Section III-V in each map scale, which achieves a linear time and space complexity. Given error tolerance ε, a joint optimization for both min-# approximation using LISE/LSSD criterion and min-ε approximation using ISE/ISSD criterion is solved. The underlying algorithm consists of three sequential procedures:

I. Error tolerance initialization. Initialize $\log_c N$ error tolerances $\{e_1^*, e_2^*, e_3^*, \ldots\}(e_1^* < e_2^* < e_3^* \ldots)$.

II. Initial curve approximation. A number of polygonal curves $\{P_1^*, P_2^*, \ldots, P_k^*\}$ are approximated based on bottom-up multi-resolution approach with corresponding error tolerance $\{e_1^*, e_2^*, e_3^*, \ldots\}$. Algorithm I and Algorithm III are used for approximating the curve of each resolution.

III. Final approximation. A polygonal approximation is conducted under the given error tolerance ε by selecting the most suitable input curve among those approximated curves $\{P_1^*, P_2^*, \ldots, P_k^*\}$.

In step I, the error tolerances $e_1^*, e_2^*, e_3^* \ldots$ $(e_1^* < e_2^* < e_3^* \ldots)$ are estimated according to the LISE/LSSD error criterion:

$$e_k^* = \frac{1}{N/c^k - 1} \sum_{1 \le j < N/c^k} \delta(P_{i_j}^{i_{j+1}})$$
$$i_j = \frac{N-1}{N/c^k - 1} \cdot (j-1) + 1 \qquad (6.1)$$

here $c > 1$ is a parameter to control the number of intermediate scale. For example, if $c = 2$, in each scale, the number of points will be around $N \rightarrow N/2 \rightarrow N/4 \rightarrow \ldots$,. The above estimation can be viewed as the average LISE/LSSD error for all approximated segments when the curve is equally partitioned. The approximated curve under the error tolerance e_k^* has the property $M_k \approx N/c^k$, where M_k is the number of output vertices in the kth resolution. Note that there are less intermediate scales when a larger c is selected, thus achieves a better reduction rate

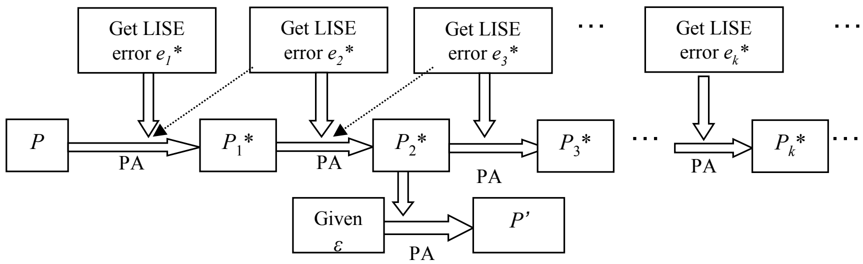

Fig.6.1. Workflow of the proposed bottom-up multi-resolution method. Error tolerance of coarser resolution is selected as high threshold for polygonal approximation, which is labeled by dashed line in the figure. In this example, if $e_2^* < \varepsilon < e_3^*$, then the approximation of P_3^* and P_4^*,… can be skipped.

at the cost of a higher computational cost. When c→∞, there are no intermediate scales and it's exactly the approximation algorithm we described with $O(N^2/M)$ time complexity (Algorithm I – Algorithm III).

In step II, a bottom-up multi-resolution algorithm is applied to estimate the approximated curves P_1^*, P_2^*, P_3^*, … under the corresponding error tolerances e_1^*, e_2^*, e_3^*, …. Here, e_{k+1}^* is used as the high threshold in the approximation procedure of resolution k. The approximated result achieved in the previous finer resolution is used as the input of polygonal approximation in the next coarser resolution ($N_{k+1} = M_k$), where Algorithm I and Algorithm III are applied in each approximation. Since the optimality of these initial approximation results is not significantly compromised, the step of minimizing the number of vertices described in Algorithm II can be omitted.

In step III, given error tolerance ε, a polygonal approximation is conducted to obtain the final approximation result by selecting the most suitable input P_k^* among those approximated curves in step II such that:

$$k = \arg\max_k (e_k^* < \varepsilon) \tag{6.2}$$

The workflow of the proposed algorithm is presented in Fig. 6.1. As the time complexity of the approximation process is $O(N_k^2/M_k)$ on each resolution, we have the following theorem:

Theorem 4. Both the time complexity and the space complexity of the proposed bottom-up multi-resolution algorithm are $O(N)$.

Proof. See appendix

Corollary 4.1. Given $0<\varepsilon_1<\varepsilon_2<_{\dots}<\varepsilon_R$ as R number of error tolerances, its corresponding approximated curves can be also constructed in linear time.

Proof. As the approximated curve for error tolerance ε_i can be used as the input for approximating the curve with error tolerance ε_{i+1}, the total time complexity is $O(N + M_1 + M_2 + \dots) = O(N)$. □

VII. Experiments

In order to evaluate the performance of the proposed *multi-resolution polygonal approximation algorithm* (MRPA), two polygonal curves are used as a test case. Curve I is an artificial curve used in [25] with 5004 vertices, curve II is the UK map contour with 10911 vertices. For the GPS trajectory simplification algorithm, two datasets are used, which are the MOPSI dataset and Geolife dataset [31]. The graphical presentations are shown in Fig. 7.1.

A. Performance for Artificial Polygonal Curve and Vector Map

For min-# problem, the performance of polygonal approximation is evaluated by its *efficiency* [26, 27], which is defined as:

$$efficiency = \frac{M_{opt}}{M} \tag{7.1}$$

Here M_{opt} is the result of the optimal solution.

In Table II, efficiency and computational cost are evaluated under different error tolerance. It can be observed that the proposed approach has a lower time cost and its performance is better than that of the two fast heuristic methods: split [7] and merge [9].

In Table III, we compare the performance when parameter c varies. For larger c, better performance is achieved at higher time cost. We can observe that least time cost is achieved when $c = 2$, which is in accordance with the theoretical analysis.

In Fig. 7.2, time cost is also analyzed in comparison with the split and merge algorithms when the size of input curve N increases. Both the low and high error tolerance cases are tested in the experiment. We can observe that the time cost of the proposed algorithm linearly increases in both cases and it achieves better result than the two comparative heuristic algorithms when the number of input vertices increases.

As the proposed approximation algorithm is a joint optimization for both min-# approximation using LISE criterion and min-ε approximation using ISE criterion, in Fig.7.3, a comparison is made on the integral square error and the efficiency of the approximated curve by using different error tolerances. We can observe that the proposed algorithm has achieved both higher efficiency (less number of output vertices) and equal or less integral square error comparing with the competitive algorithms.

B. Performance Evaluation for GPS Trajectory Simplification

The performance of the proposed GPS trajectory simplification algorithm is tested on two datasets, which are MOPSI dataset with 344 trajectories 744,610 points and Geolife dataset with 640 trajectories 4,526,030 points. The root mean square error (RMSE), average error (MAE), median error (MEDE) and maximum error (MAXE) are all calculated in order to evaluate the efficiency of the proposed algorithm under synchronous Euclidean distance. In Table V, we also compare these error measures for the GPS trajectories with walking and no-walking segments. We can observe that although the same LSSD error tolerance is used, walking trajectories can have less distortion with more details information comparing with no-

CURVE I [25], N=5004 | CURVE II(UK MAP): N = 10911 | MOPSI DATASET: 344 TRAJECTORIES, N= 744,610 | GEOLIFE DATASET[31]: 640 TRAJECTORIES, N= 4,526,030

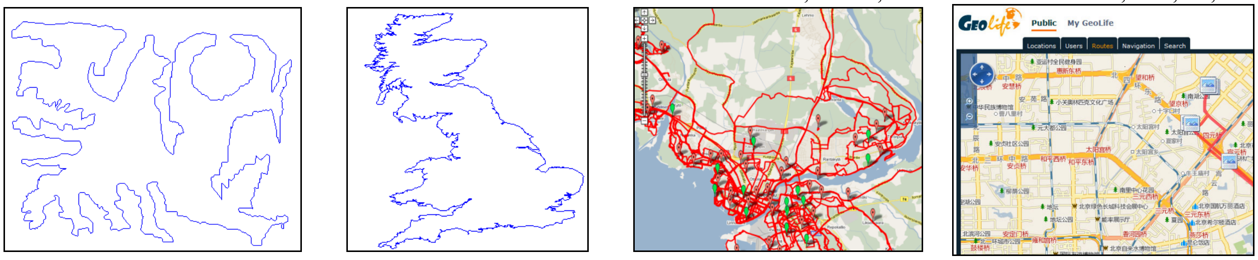

Fig.7.1. Testing data in the experiments

TABLE.II

COMPARISON OF THE EFFICIENCY AND PROCESSING TIME (C=2).

CURVE I	M_{opt}	EFFICIENCY			TIME COST (MS)			
		SPLIT[7]	MERGE[9]	PROPOSED	SPLIT[7]	MERGE[9]	MRPA	OPTIMAL[1]
$\varepsilon_1 = 1$	824	0.68	0.81	**0.85**	7	3	6	847
$\varepsilon_2 = 100$	193	0.66	0.74	**0.78**	6	4	6	791
$\varepsilon_3 = 10^4$	49	0.68	0.71	**0.77**	4	4	7	794
CURVE II	M_{opt}	EFFICIENCY			TIME COST (MS)			
		SPLIT[7]	MERGE[9]	PROPOSED	SPLIT[7]	MERGE[9]	MRPA	OPTIMAL[1]
$\varepsilon_1 = 10^{-4}$	1986	0.71	0.83	**0.86**	18	11	15	3699
$\varepsilon_2 = 10^{-2}$	364	0.66	0.72	**0.75**	14	12	17	3678
$\varepsilon_3 = 1$	72	0.66	0.70	**0.75**	11	13	17	3592

TABLE.III

EFFICIENCY AND PROCESSING TIME FOR CURVE I AND II WHEN DIFFERENT C IS SELECTED

CURVE I	M_{opt}	EFFICIENCY			TIME COST (MS)		
		c = 1.5	c = 2	c = 4	c = 1.5	c = 2	c = 4
$\varepsilon_1 = 1$	824	0.82	0.85	0.87	7	6	9
$\varepsilon_2 = 100$	193	0.74	0.78	0.81	8	6	11
$\varepsilon_3 = 10^4$	49	0.72	0.77	0.79	8	7	11
CURVE II	M_{opt}	EFFICIENCY			TIME COST (MS)		
		c = 1.5	c = 2	c = 4	c = 1.5	c = 2	c = 4
$\varepsilon_1 = 10^{-4}$	1986	0.84	0.86	0.91	18	15	20
$\varepsilon_2 = 10^{-2}$	364	0.75	0.75	0.77	21	17	21
$\varepsilon_3 = 1$	72	0.76	0.75	0.80	22	17	22

TABLE.IV

PERFORMANCE OF GPS TRAJECTORY SIMPLIFICATION BY SYNCHRONOUS EUCLIDEAN DISTANCE

RESOLUTION 1: f_{LSSD} = 50	MOPSI DATASET(744,610 POINTS) AVERAGE N/M = 8.02				GEOLIFE DATASET(4,526,030 POINTS) AVERAGE N/M = 10.1			
METHOD	RMSE	MAE	MEDE	MAXE	RMSE	MAE	MEDE	MAXE
D-P	4.51	2.38	1.32	39.0	10.7	4.13	1.12	134.1
TD-TR	1.82	1.41	1.23	**4.61**	1.89	1.47	1.28	**4.91**
OW	1.89	1.45	1.23	5.33	1.99	1.53	1.30	5.85
STTrace	4.37	2.67	1.60	21.1	4.93	3.16	2.07	26.0
TS	24.0	11.9	3.64	132.8	42.3	16.6	3.58	363.3
MRPA	**1.61**	**1.23**	**1.05**	5.88	**1.46**	**1.06**	**0.83**	6.51
RESOLUTION 2: f_{LSSD} = 2000	MOPSI DATASET AVERAGE N/M = 25.1				GEOLIFE DATASET AVERAGE N/M = 29.5			
METHOD	RMSE	MAE	MEDE	MAXE	RMSE	MAE	MEDE	MAXE
D-P	13.8	8.39	5.08	81.1	52.3	22.2	6.73	416.6
TD-TR	6.85	5.55	4.82	**17.7**	7.48	6.04	5.24	**19.40**
OW	7.40	5.86	4.89	21.1	8.59	6.80	5.73	25.59
STTrace	33.9	19.9	8.67	132.1	39.3	24.4	14.6	169.2
TS	82.7	48.7	20.9	316.2	200.4	98.6	29.0	1090.0
MRPA	**5.96**	**4.76**	**4.07**	23.9	**5.60**	**4.19**	**3.27**	29.0
RESOLUTION 3: $f_{LSSD} = 10^5$	MOPSI DATASET AVERAGE N/M = 79.4				GEOLIFE DATASET AVERAGE N/M = 109.6			
METHOD	RMSE	MAE	MEDE	MAXE	RMSE	MAE	MEDE	MAXE
D-P	42.0	29.0	19.9	173.3	154.8	80.4	32.6	867.1
TD-TR	26.7	21.6	18.3	**70.5**	28.9	23.2	19.4	**84.8**
OW	29.5	23.4	19.2	82.1	33.8	26.7	22.1	103.8
STTrace	198.9	131.6	72.4	559.4	251.2	160.9	96.5	871.4
TS	270.1	181.3	106.8	763.5	691.0	399.6	179.7	2733.5
MRPA	**22.9**	**18.5**	**15.8**	79.2	**21.4**	**15.7**	**11.7**	115.6

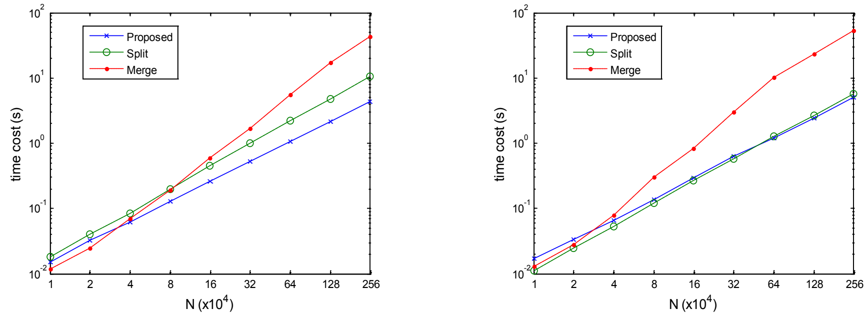

Fig. 7.2. Processing time cost is plotted for different number of input vertices for curve II. Low error tolerance (left) and high error tolerance (right) are both tested.

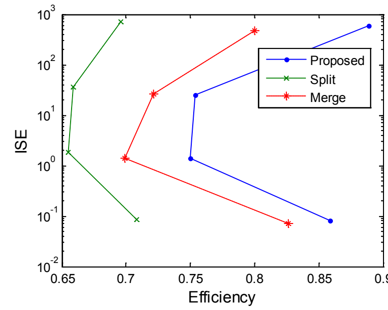

Fig. 7.3. *Efficiency* and integral square error for different error tolerance $\varepsilon = 10^{-4}$, 10^{-2}, 1 and 100.

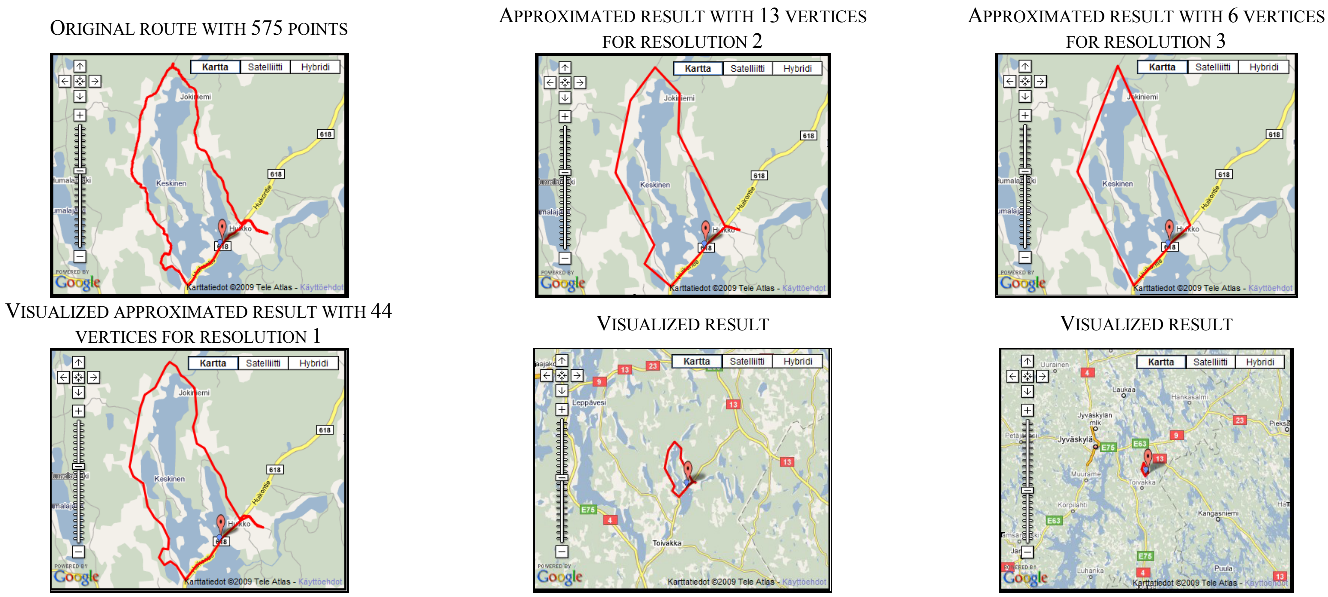

Fig. 7.4. Example of the GPS trajectory simplification by the proposed algorithm.

walking segments.

The proposed polygonal approximation algorithm is also compared with other GPS trajectory simplification algorithms with the same number of approximated points. Theses competitive algorithms are *Douglas–Peucker* algorithm (*D-P*) [7], *TD-TR* [32], *Open Window* (*OW*) [32], *STTrace* [33] and *TS* [31]. The results are shown in Table IV, where synchronous Euclidean distance is considered as the error measure. We can observe that the proposed algorithm yields the minimum distortion than other solutions. The time cost of the trajectory simplification is also summarized in Table VI. It follows from our experiment that the time cost of the proposed algorithm is higher than the Trajectory Simplification (TS) algorithm [31]. This is because the constant factor in the proposed algorithm is larger than other solutions, which comes from the LISE/LSSD calculation and the graph structure maintenance. For example, based on our experiment, in Fig. 7.2, when N >10000, the proposed solution will have less time cost than split or merge algorithm. Note that the proposed solution also achieves a better approximation performance than those fast solutions.

An application of the proposed approximation algorithm for the GPS trajectory simplification is demonstrated in Fig. 7.4 over a sample route with 575 vertices, where the GPS trajectory is visualized in different map scale with 44, 13, 6 vertices correspondingly. As suitable error tolerance is selected for each resolution, the visualization of the GPS trajectory is not compromised by the reduced data whereas the rendering time is

TABLE.V
PERFORMANCE OF PROPOSED GPS TRAJECTORY SIMPLIFICATION ALGORITHM FOR DIFFERENT TRANSPORTATION MODE UNDER SYNCHRONOUS EUCLIDEAN DISTANCE (M) (MOPSI DATASET)

RESOLUTION 1: $f_{LSSD} = 50$,					
	RMSE	MAE	MEDE	MAXE	*N/M*
WALKING	1.54	1.20	1.06	5.79	9.38
NO-WALKING	1.71	1.19	0.88	6.29	4.92
RESOLUTION 2: $f_{LSSD} = 2000$					
	RMSE	MAE	MEDE	MAXE	*N/M*
WALKING	5.25	4.26	3.68	21.1	32.6
NO-WALKING	8.23	6.27	5.07	33.2	12.9
RESOLUTION 3: $f_{LSSD} = 10^5$					
	RMSE	MAE	MEDE	MAXE	*N/M*
WALKING	17.9	14.6	12.5	60.8	119.7
NO-WALKING	34.3	27.4	23.5	128.1	35.9

TABLE.VI
TIME COST OF THE TRAJECTORY SIMPLIFICATION

	TIME COST (S)	
	MOPSI	GEOLIFE
D-P[7]	1.83	12.9
TD-TR[32]	1.95	13.1
OW[32]	25.4	320.8
STTrace[33]	1160.1	20589
TS[31]	0.85	6.8
Proposed	1.48	10.2

greatly reduced. The code and the testing dataset can be seen on http://cs.joensuu.fi/~mchen/GPSTrajSimp.htm.

VIII. Conclusion

We have proposed a fast $O(N)$ time polygonal approximation algorithm for the GPS trajectory simplification by a joint optimization on both *local integral square synchronous Euclidean distance* (LSSD) and *integral square synchronous Euclidean distance* (ISSD) criterion, which is effective and computationally efficient. The proposed method is designed by the bottom-up multi-resolution approach. In each resolution, a near-optimal polygonal approximation algorithm is exploited, which has a time complexity of $O(N^2/M)$. Both the theoretical analysis and the experimental tests have demonstrated that the proposed method had made a significant progress in solving the GPS trajectory simplification problem in a real-time application. Moreover, the proposed polygonal approximation algorithm and fine-tune strategy in Algorithm II and Algorithm III can also be extended and exploited to other error criteria.

There are several potential extensions of our present work. For example, in our future work, topology properties, road network information and the similarity of the multiple GPS trajectories can also be considered in the approximation process.

Appendix

Proof of the local integral square synchronous Euclidean distance (LSSD) in Eq. (3.7).

For the sake of computational efficiency of the synchronous Euclidean distance, we extend the *local integral square error* (LISE) criterion and derive a new error measure, called *local integral square synchronous Euclidean distance* (LSSD), where

$$\delta_{\text{LSSD}}(P_i^j)=\sum_{i<k<j} SED^2(p_k,p_k^{'})$$

p_k' is the approximated position at time t_k if sub-curve P_i^j is approximated by edge $\overline{p_ip_j}$, see the definition in (2.3). Thus:

$$\delta_{\text{LSSD}}(P_i^j)=\sum_{i<k<j}(x_i\cdot\frac{t_j-t_k}{t_j-t_i}+x_j\cdot\frac{t_k-t_i}{t_j-t_i}-x_k)^2$$
$$+\sum_{i<k<j}(y_i\cdot\frac{t_j-t_k}{t_j-t_i}+y_j\cdot\frac{t_k-t_i}{t_j-t_i}-y_k)^2$$
$$=\sum_{i<k<j}(\frac{x_it_j-x_jt_i}{t_j-t_i}+\frac{x_j-x_i}{t_j-t_i}\cdot t_k-x_k)^2+\sum_{i<k<j}(\frac{y_it_j-y_jt_i}{t_j-t_i}+\frac{y_j-y_i}{t_j-t_i}\cdot t_k-y_k)^2$$
$$=c_1^2(j-i-1)+c_2^2\sum_{i<k<j}t_k^2+\sum_{i<k<j}x_k^2+2c_1c_2\sum_{i<k<j}t_k-2c_1\sum_{i<k<j}x_k$$
$$-2c_2\sum_{i<k<j}t_kx_k+c_3^2(j-i-1)+c_4^2\sum_{i<k<j}t_k^2+\sum_{i<k<j}y_k^2+2c_3c_4\sum_{i<k<j}t_k$$
$$-2c_3\sum_{i<k<j}y_k-2c_4\sum_{i<k<j}t_ky_k$$
$$=(c_1^2+c_3^2)(j-i-1)+(c_2^2+c_4^2)(S_{t^2}^{j-1}-S_{t^2}^{i})+2(c_1c_2+c_3c_4)(S_t^{j-1}-S_t^{i})$$
$$+(S_{x^2}^{j-1}-S_{x^2}^{i})+(S_{y^2}^{j-1}-S_{y^2}^{i})-2c_1(S_x^{j-1}-S_x^{i})-2c_3(S_y^{j-1}-S_y^{i})$$
$$-2c_2(S_{xt}^{j-1}-S_{xt}^{i})-2c_4(S_{yt}^{j-1}-S_{yt}^{i})$$

Where

$$c_1=\frac{x_it_j-x_jt_i}{t_j-t_i},\ c_2=\frac{x_j-x_i}{t_j-t_i},\ c_3=\frac{y_it_j-y_jt_i}{t_j-t_i},\ c_4=\frac{y_j-y_i}{t_j-t_i}$$

S_x, S_y, S_t, S_{x2}, S_{y2}, S_{t2}, S_{tx} and S_{ty} are the accumulated sums of the x, y and t on the GPS trajectory respectively.

Computation of the above approximation error $\delta_{\text{LSSD}}(P_i^j)$ takes $O(1)$ time with an $O(N)$ time accumulated sum pre-calculation.

Proof of Theorem 3:

Suppose that under error tolerance ε, a curve P with N vertices can be approximated by a curve P' with M vertices. The number of vertices with k links is n_k, $k=0,1,..,M-1$. In total, $2N$ space is needed to record the accumulated errors and the backtracking vector, thus it has a space complexity $O(N)$.

As every node is only visited once in tree traversal step with $O(N)$ in total, the main bottleneck is the cost on edge tests, which can be calculated as follows:

$$f=\sum_{i=1}^{M-1}n_{i-1}\cdot n_i\quad s.t.\sum_{i=0}^{M-1}n_i=N,n_0=1,n_{M-1}=1,n_i\geq 1,i=0,...,M-1$$

Suppose that M vertices are firstly selected with the number of links from 0 to M - 1 respectively. For the remaining N-M vertices, if the number of links of every vertices is distributed randomly under a *multi-nominal distribution*, then we have:

$$(u_1,u_2...,u_{M-2})\sim Mult(U;(\frac{1}{M-2},\frac{1}{M-2},...,\frac{1}{M-2}))$$

where $u_i=n_i$ - 1, i =1, 2, …, $M-2$ and the corresponding statistical properties of u_i (i =1, 2,…,$M-2$) can be formulated as follows:

$$E(u_i)=\frac{N-M}{M-2}$$
$$\text{var}(u_i)=(N-M)\cdot\frac{1}{M-2}\cdot(1-\frac{1}{M-2})$$
$$\text{cov}(u_i,u_j)=E(u_iu_j)-E(u_i)E(u_j)=-(N-M)\cdot\frac{1}{M-2}\cdot\frac{1}{M-2}$$
$$E(u_iu_j)=\text{cov}(u_i,u_j)+E(u_i)E(u_j)$$
$$=\frac{-N+M+N^2+M^2-2NM}{(M-2)^2}=O(N^2/M^2)$$
$$E(u_i^2)=E^2(u_i)+\text{var}(u_i)$$
$$=\frac{(N-M)^2+(N-M)\cdot(M-3)}{(M-2)^2}=O(N^2/M^2)$$

Thus, the expected time complexity:

$$E(f)=n_1+n_{m-2}+(M-3)E(n_1n_2)$$
$$=1+E(u_1)+1+E(u_{M-2})+(M-3)E((1+u_1)(1+u_2))$$
$$=2+\frac{N-M}{M-2}\cdot 2+(M-3)(1+2\cdot E(u_1)+E(u_1u_2))$$
$$=2+\frac{N-M}{M-2}\cdot 2+(M-3)\cdot\frac{N^2-5N+M+4}{(M-2)^2}=O(N^2/M)$$

To sum up, the expected time complexity is $O(N^2/M)$ and space complexity $O(N)$ □

Proof of Theorem 1:

As the output of the *min-# initialization* is a tree structure, $2N$ space is needed in order to record all the parent and child nodes on the tree and its space complexity is $O(N)$.

The time complexity of min-# initialization mainly consists of two parts: number of edge tests and maintenance cost of two priority queues. The cost of edge tests can be calculated in a similar manner as in Theorem 3:

$$f=\sum_{i=0}^{M-2}n_i\cdot(2\cdot(\sum_{j=0}^{i}n_j)/(i+1)),\ s.t.\sum_{i=0}^{M-1}n_i=N,n_0=1,n_i\geq 1,i=0,...,M-1$$

$$(u_1,u_2...,u_{M-1})\sim Mult(U;(\frac{1}{M-1},\frac{1}{M-1},...,\frac{1}{M-1}))$$

where $u_i = n_i - 1$, $i = 1, 2, \ldots, M-1$

$$E(f) = 2(M-1)\cdot[\frac{1}{i+1}E(n_i^2) + \frac{i}{i+1}E(n_j n_j)]$$

From Theorem 3, we have:

$$E(n_i n_j) = O(N^2/M^2), E(n_i^2) = O(N^2/M^2)$$

Thus $E(f) = O(N^2/M)$

As $E(n_i) = 1 + E(u_i) = 1 + \frac{N-M}{M-2}, i = 1,\ldots,M-2$

The cost of maintaining the priority queues is:

$$E(g) = E(\sum_{i=0}^{M-1} n_i \cdot \log_2(n_i)) = 1 + \sum_{i=1}^{M-1} E(n_i \cdot \ln(n_i))/\ln 2$$

Suppose a linear function is constructed as follows:

$$y_i = \ln(E(n_i)) + \frac{1}{E(n_i)}\cdot(n_i - E(n_i))$$

The constructing function has the property $\ln(n_i) \le y_i$, and thus

$$E(g) \le 1 + \frac{1}{\ln 2}\sum_{i=1}^{M-1} E(n_i \cdot (\ln(E(n_i)) + \frac{1}{E(n_i)}\cdot(n_i - E(n_i)))), i = 1,\ldots,M-2$$

$$\le 1 + \underbrace{\frac{1}{\ln 2}(-1 + \ln\frac{N-1}{M-1})\cdot(N-1)}_{O(N\log(N/M))} + \underbrace{\frac{1}{\ln 2}\frac{(M-1)^2}{N-1}E(n_2^2))}_{O(N)}$$

$$E(g) = O(N\log(N/M))$$

Thus the min-# initialization has an expected time complexity of $O(N^2/M)$ and a space complexity of $O(N)$. □

Proof of Theorem 2:

First, we give the proof of the time complexity for simplified *reduced search dynamic programming* method. Suppose the initial approximated curve $P' = (p_{i_1}, p_{i_2}, \ldots, p_{i_M})$, where i_1, i_2,…, i_M are the indexes on the curve. *s.t.* :

$$n_k = i_{k+1} - i_k, k = 1,\ldots,M-1$$

The number of edges tests of *reduced search dynamic programming* is:

$$f = \sum_{i=1}^{M-1}((1 + \sum_{j=i-W/2}^{i+W/2-1} n_j)\cdot(1 + \sum_{j=i-W/2+1}^{i+W/2} n_j)),$$

$$s.t. \sum_{i=1}^{M-1} n_i = N-1, n_i \ge 1, i = 1,\ldots,M-1$$

Let us define: $u_i = n_i - 1$, $i = 1, 2, \ldots, M-1$ and assume that the curve P' is randomly initialized as in Theorem 3 such that u_i has the property:

$$(u_1, u_2 \ldots, u_{M-1}) \sim Mult(U; (\frac{1}{M-1}, \frac{1}{M-1}, \ldots, \frac{1}{M-1}))$$

The expected time complexity is therefore estimated as:

$$E(f) = (M-1)\cdot((1 + \sum_{j=i-W/2}^{i+W/2-1} n_j)\cdot(1 + \sum_{j=i-W/2+1}^{i+W/2} n_j))$$

$$= (M-1)\cdot(1 + \sum_{j=i-W/2}^{i+W/2-1} n_j + \sum_{j=i-W/2+1}^{i+W/2} n_j + \sum_{j=i-W/2}^{i+W/2-1} n_j \cdot \sum_{j=i-W/2+1}^{i+W/2} n_j)$$

$$= (M-1)\cdot((1 + 2\cdot W\cdot E(n_j) + (W-1)\cdot E(n_j^2) + (W^2 - W + 1)\cdot E(n_i n_j))$$

According to Theorem 3, we have

$E(n_j) = O(N/M), E(n_i n_j) = O(N^2/M^2)$ and $E(E(n_j^2)) = O(N^2/M^2)$

Thus, $E(f) = O(W^2 N^2/M^2)$ □

On the other hand, the proposed reduced search method is achieved by lifting the vertex position in the output tree structure in the initialization. The memory cost of maintaining a tree structure is $O(N)$. Likewise, the cost of number of edges tests is calculated as:

$$f = \sum_{i=0}^{M-3} n_i \cdot (\sum_{j=i+2}^{i+W+1} n_j))$$

$$E(f) = (M-2)W\cdot E(n_i n_j) = O(WN^2/M)$$

As $E(n_i n_j) = O(N^2/M)$, we have $E(f) = O(WN^2/M)$

Thus, it has an expected time complexity of $O(WN^2/M)$ and a space complexity of $O(N)$ □

Proof of Theorem 4:

From Theorem 1-3, space complexity of the near-optimal polygonal approximation algorithm is $O(N)$. An additional cost is the pre-calculated sums, which also takes $O(N)$ space. As we do not need to record all the information of the intermediate scales, the total space complexity is $O(N)$.

The time complexity of the proposed bottom-up multi-resolution algorithm mainly consists of three parts: error tolerance initialization (step I), initial curve approximation (step II) and the final approximation (step III). As the approximation error between two vertices can be calculated in constant time, the time cost of step I can be calculated as follows:

$$\sum_{k=1}^{\log_c N}\frac{N}{c^k} = \frac{\frac{N}{c}(1-(\frac{1}{c})^{\log_c N})}{1-\frac{1}{c}} = \frac{\frac{N}{c}(1-\frac{1}{N})}{1-\frac{1}{c}} = \frac{N-1}{c-1} = O(N)$$

In step II, the time complexity of the proposed polygonal approximation method is $O(N_k^2/M_k)$. As the number of input and output vertices obeys the equation $M_k = N_k/c$ for each resolution, the time complexity can be estimated by:

$$\sum_{k=0}^{\log_c N-1}\frac{(N/c^k)^2}{N/c^{k+1}} = \sum_{i=0}^{\log_c N}(\frac{N}{c^{k-1}}) = N\cdot\frac{c(1-\frac{1}{c^{\log_c N}})}{1-\frac{1}{c}} = (N-1)\cdot\frac{c^2}{c-1} = O(N)$$

Since the proposed polygonal approximation algorithm (Algorithm I-III) has time complexity of $O(N_k^2/M_k)$, the computational cost of step III can be written as $O(cN_k)$, where the value of the parameter is always $c > 1$.

To sum up, the proposed multi-resolution polygonal approximation has a time complexity of $O(N)$ □

ACKNOWLEDGMENTS

The authors would like to thank the reviewers and the editor for their valuable comments and suggestions, which have been very useful in improving the technical content and the presentation of the paper. We would also thank Prof. Juha Alho and Dr. Ville Hautamäki for the useful discussion during this work.

REFERENCES

[1] H. Imai, M. Iri, "Polygonal Approximations of a Curve-Formulations and Algorithms", *Computational Morphology*, 71-86, Amsterdam, 1988.

[2] G. T. Toussaint, "On the complexity of approximating polygonal curves in the plane", *IASTED,* Switzerland, 59-62, 1985.

[3] A. Melkman, J. O'Rourke, "On polygonal chain approximation", *Computational Morphology*, 87-95, Amsterdam, 1988.

[4] J. C. Perez, E. Vidal, "Optimum polygonal approximation of digitized curves", *Pattern Recognition Letter*, vol. 15, 743–750, 1994.

[5] Chung K.-L., Yan W.-M., Chen W.-Y., "Efficient algorithms for 3-D polygonal approximation based on LISE criterion", *Pattern Recognition* vol. 35, 2539-2548, 2002.

[6] B. K. Ray, K. S. Ray, "A non-parametric sequential method for polygonal approximation of digital curves", *Pattern Recognition Letter*, vol. 15, 161–167, 1994.

[7] D. H. Douglas, T. K. Peucker, "Algorithm for the reduction of the number of points required to represent a line or its caricature", *The Canadian Cartographer*, 10 (2), 112-122, 1973.

[8] J. Hershberger, J. Snoeyink, "Cartographic line simplification and

polygon CSG formulae in $O(n \log^* n)$ time", *5th International Workshop on Algorithms and Data Structures*, 93–103, 1997
[9] A. Pikaz, I. Dinstein, "An algorithm for polygonal approximation based on iterative point elimination", *Pattern Recognition Letters*, 16(6): 557-563, 1995.
[10] W. S. Chan, F. Chin, "On approximation of polygonal curves with minimum number of line segments or minimum error", *Lecture Notes in Computer Science,* vol.650, 378-387, 1992.
[11] D. Chen, O. Daescu, "Space-Efficient Algorithms for Approximating Polygonal Curves in Two Dimensional Space", *Computing and Combinatorics*, vol.1449, 45-55, 1998.
[12] P. K. Agarwal, K.R. Varadarajan, "Efficient algorithms for approximating polygonal chains", *Discrete Comput. Geom*. 23, 273–291, 2000.
[13] M. Salotti, "Optimal polygonal approximation of digitized curves using the sum of square deviations criterion", *Pattern Recognition*, 35(2), 435-443, 2002.
[14] D. Eu, G.T. Toussaint, "On approximation polygonal curves in two and three dimensions", *Graphical Models, and Image Processing*, 56(3), 231-246, 1994.
[15] O. Daescu, N. Mi, "Polygonal chain approximation: A query based approach", *Computational Geometry*, 30(1), 41–58, 2005.
[16] K. L. Chung, P. H. Liao, J. M. Chang, "Novel efficient two-pass algorithm for closed polygonal approximation based on LISE and curvature constraint criteria", *Journal of Visual Communication and Image Representation* 19(4), 219-230, May 2008.
[17] A. Kolesnikov, P. Fränti, "A fast near-optimal min-# polygonal approximation of digitized curves", *ACIT'2002*, 418-422, 2002.
[18] A. Kolesnikov, P. Fränti, "Reduced-search dynamic programming for approximation of polygonal curves", *Pattern Recognition Letters*, 24 (14), 2243-2254, October 2003.
[19] B. P. Buttenfield, "Transmitting vector geospatial data across the Internet", *Proc. GIScience, LNCS*, vol. 2478, 51-64, 2002.
[20] C. Le Buhan Jordan, T. Ebrahimi, M. Kunt, "Progressive content-based shape compression for retrieval of binary images", *Computer Vision and Image Understanding*, 71(2), 198-212, 1998.
[21] A. Kolesnikov, P. Fränti, X. Wu, "Multiresolution Polygonal Approximation of Digital Curves", *17th International Conference on Pattern Recognition (ICPR'04*), vol.2, 855-858, 2004.
[22] P. F. Marteau, G. Ménier, "Speeding up simplification of polygonal curves using nested approximations", *Pattern Analysis and Application*, 12(4), 367-375, 2008.
[23] A. Kolesnikov, "Fast Algorithm for ISE-bounded Polygonal Approximation“, *IEEE International Conference on Image Processing,* 1013-1015, 2008.
[24] G. Papakonstantinou, P. Tsanakas, G. Manis, "Parallel approaches to piecewise linear approximation", *Signal Processing*, vol. 37, 415-423, 1994.
[25] M. Salotti, "An efficient algorithm for the optimal polygonal approximation of digitized curves", *Pattern Recognition Letters*, vol. 22, 215-221, 2001.
[26] P. L. Rosin, "Techniques for assessing polygonal approximations of curves", *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(6), 659-666, 1997.
[27] P. L. Rosin, "Assessing the behavior of polygonal approximation algorithms", *Pattern Recognition*, 36(2), 505-518, 2003.
[28] O. Daescu, N. Mi, C. S. Shin, A. Wolff, "Farthest-point queries with geometric and combinatorial constraints", *Computational Geometry*, 33 (3), 174–185, 2006.
[29] O. Daescu, "New results on path approximation", *Algorithmica* 38(2), 131–143, 2004.
[30] A. Gribov, E. Bodansky, "A new method of polyline approximation", *Proc. of International Conference on Structural, Syntactic and Pattern Recognition, LNCS*, vol. 3138, 504-511, 2004.
[31] Y. Chen, K. Jiang, Y. Zheng, C. Li, N. Yu, "Trajectory Simplification Method for Location-Based Social Networking Services", *ACM GIS workshop on Location-based social networking services*, 33-40, 2009.
[32] N. Meratnia, R. A. de By, "Spatiotemporal Compression Techniques for Moving Point Objects", *Proceedings of the Extending Database Technology*, 765-782, 2004.
[33] M. Potamias, K. Patroumpas, T. Sellis, "Sampling Trajectory Streams with Spatiotemporal Criteria", *Proceedings of the Scientific and Statistical Database Management* (SSDBM), 275-284, 2006.
[34] Z. Yu, X. Zhou, "Computing with Spatial Trajectories", *Springer*, 2011.
[35] H. Cao, O. Wolfson, G. Trajcevski, "Spatio-temporal data reduction with deterministic error bounds", *VLDB Journal*, 15(3), 211-228, 2006.
[36] J. Muckell, J. H. Hwang, C. T. Lawson, S. S. Ravi, "Algorithms for compressing GPS trajectory data: an empirical evaluation", *SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 402-405, 2010.
[37] J. Muckell, J. H. Hwang, V. Patil, C. T. Lawson, F. Ping , S. S. Ravi, "SQUISH: an online approach for GPS trajectory compression", *International Conference on Computing for Geospatial Research & Applications*, 1-8, 2011.
[38] R. Lange, T. Farrel, F. Dürr, K. Rothermel, "Remote real-time trajectory simplification", *IEEE International Conference on Pervasive Computing and Communications*, 1-10, 2009.
[39] H. Alt, L. J. Guibas, "Handbook of Computational Geometry", pp. 121-153, 1999.
[40] H. Alt, C. Knauer, C. Wenk, "Matching polygonal curves with respect to the Fréchet distance", STACS, LNCS , 63-74, 2001.
[41] M. Chen, M. Xu and P. Fränti, "Compression of GPS Trajectories", *Data Compression Conference*, 62-71, Snowbird, USA, 2012. (http://cs.joensuu.fi/ ~mchen/GPSTrajComp.htm)
[42] H. Sakoe, S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition", *IEEE Trans. on Acoustics, Speech and Signal Processing*, 26(1), 43-39, 1978.
[43] H. Ding, G. Trajcevski, P. Scheuermann. X. Wang, E. J. Keogh, "Querying and mining of time series data: experimental comparison of representations and distance measures", *Proceedings of the VLDB*, 1(2), 1542-1552, 2008.
[44] R. Estkowski, J. Mitchell, "Simplifying a polygonal subdivision while keeping it simple", *Symposium on Computational Geometry*, 40-49, 2001.

Minjie Chen received his B.Sc. and M.Sc degrees in biomedical engineering from Shanghai Jiaotong University, Shanghai, China, in 2003 and 2007. Since 2008, he is a PhD student in Computer Science at the University of Eastern Finland. His research interests include image denoising and compression, spatial-temporal data compression and medial image analysis.

Mantao Xu received the B.Sc. degree in mathematics from Nankai University, Tianjin, China, in 1991, the M.Sc. degree in applied mathematics from Harbin Institute of Technology, Harbin, China, in 1997, and the Ph.D. degree in computer science from the University of Joensuu, Joensuu, Finland, in 2005 repsectively. He served as a Research Lab Manager with Kodak Health Group and Carestream Health Inc, Global R&D Center, Shanghai, China, from 2005 to 2010. He is now a Research Professor at School of Electric Engineering, Shanghai Dianji University, Shanghai, China. His research interests include medical image analysis, multimedia technology and pattern recognition.

Pasi Fränti received his MSc and PhD degrees from the University of Turku, 1991 and 1994 in Science. Since 2000, he has been a professor of Computer Science at the University of Eastern Finland. He has published 56 journals and 128 peer review conference papers, including 10 IEEE transaction papers. His research interests include clustering algorithms, vector quantization, lossless image compression, voice biometrics and location-based systems. He has supervised 14 PhDs and is currently the head of the East Finland doctoral program in Computer Science & Engineering (ECSE). He serves as an associate editor for Pattern Recognition Letters.