Compression of GPS trajectories

Minjie Chen[1], Mantao Xu[2] and Pasi Franti[1]

[1] *School of Computing, Univ. of Eastern Finland, Finland*

{mchen, franti }@cs.joensuu.fi

[2] *Shanghai Dianji University, China*

xumt@sdju.edu.cn

***Abstract**:* Enormous amounts of GPS trajectories, which record users' spatial and temporal information, are collected by geo-positioning mobile phones in recent years. The massive volumes of trajectory data bring about heavy burdens for both network transmission and data storage. To overcome these difficulties, a number of compression algorithms have been proposed by reducing the number of points in the trajectory data. But these algorithms lack a rigorous investigation on how to encode the reduced trajectories. In this paper, we propose an algorithm that optimizes both the trajectory simplification and the coding procedure using the quantized data. The underlying algorithm is also compared with the existing methods across 640 trajectories from *Microsoft Geolife dataset* using s*ynchronous Euclidean distance* (SED) as the error metrics. Experimental results show that the proposed method saves 60% of compression cost against the current state of the art compression algorithms.

1. Introduction

Location-acquisition technologies have generated a great deal of enthusiasms in the global mobile market in recent years. For example, the Location Based Services (LBS) enables the end-users of GPS mobile phones to obtain their locations, and therefore record travel experiences by a number of time-stamped trajectories. However, most of LBS applications bring about an enormous volume of data to the end-users as well as incur a large amount of redundant storage and a longer uploading/downloading time to mobile service providers. For example, if data is collected at 10 second intervals, a calculation in [1] shows that without any compression, 100 Mb of storage capacity is required to store the GPS trajectories of 400 users for a single day in server side.

To overcome this difficulty, a number of GPS trajectory compression algorithms have been studied in literature. In these algorithms, an approximation technique called *line simplification* has been treated as an active research topic in data reduction of the GPS trajectories, amongst which the Douglas-Peucker algorithm [2] is the most popular one. The algorithm divides the line segment with biggest deviation at each step until the approximated error is smaller than a given error tolerance.

Later, Meratnia [1] indicated that such algorithms were not suitable for GPS trajectory since both spatial and temporal information should be considered. Thus, the so-called TD-TR algorithm was developed, where *synchronous Euclidean distance* was used instead of the perpendicular distance in the Douglas-Peucker algorithm. Similarly,

[1] The research is partially supported by Tekniikan edistämissäätiö (TES), Nokia Scholarship, East Finland Graduate School in Computer Science (ECSE), National Natural Science Foundation of China (Grant No 61072146) and Shanghai Committee of Science and Technology, China (Grant No 10PJ1404400).

Opening Window [1] algorithm was also proposed to solve the line simplification problem sequentially in a greedy manner. Another solution [3] was the *threshold-guided algorithm* via estimating the safe area of the next point using the position, speed and orientation information. *STTrace* sampling algorithm was also implemented using a bottom-up strategy such that the synchronous Euclidean distance was minimized in each step. In [4], a new distance-function called *spatial join* was proposed, which was bounded for spatial-temporal queries.

Recently, a new simplification algorithm SQUISH [11] has been proposed based on the priority queue data structure that preserves the speed information at a much higher accuracy. A multi-resolution simplification algorithm MRPA has also been designed with linear time complexity in [12]. Semantic meanings of the GPS trajectories are also considered during the compression process in urban area in [15] whereas trajectory compression algorithm with network constraint has been developed in [14]. Performance evaluations are also made for several traditional trajectory simplification algorithms [10]. However, there is not one algorithm that always outperforms other compression approaches in all situations [11].

In these algorithms, the performance is measured only on the reduction rate by the line simplification process. The reduced data points are saved directly, which is useful to support the effective trajectory queues in database. However, they lack of a rigorous analytical approach on the encoding procedures of the reduced trajectories. Namely, without further compression after line simplification, 12 bytes are needed at minimum for encoding each point including latitude, longitude and timestamp information. On the other hand, when data compression techniques are used, we can achieve a better compression ratio for the spatial trajectory data, which is appropriate for data storage.

To circumvent the above problem, a quantization technique can be applied to further improve the encoding procedure of these reduced GPS trajectories. The quantization approach has been analytically investigated in the so-called *vector map compression* problem [5-8]. In these algorithms, differential coordinates of adjacent data points are used as the prediction error and the residual vectors are then quantized using a variety of quantization strategies, including uniform quantization, product scalar quantization [5] and vector quantization with fixed-size codebook [6]. Amongst them, a pioneer solution in [7] combines both the advantage of line simplification and quantization via dynamic programming and hence the terminology of *dynamic quantization* (DQ). Likewise, the quantized vectors can be further compressed by arithmetic coding based on the assumption that the resulting quantized differential coordinates obeys a geometric distribution [8]. In all these methods, timestamp information is not considered.

In this paper, we consider the problem of lossy compression for GPS trajectories with latitude, longitude and timestamp information, under a given error tolerance, i.e., synchronous Euclidean distance. In contrast to the existing algorithms, we achieve two significant improvements described below. Firstly, speed and direction changes are incorporated in the encoding process instead of using the differential coordinates in the previous methods. Secondly, line simplification and quantization are combined in the encoding procedures in order to seek the approximated trajectory for compression.

The rest of the paper is organized as follows. The proposed GPS trajectory compression (GTC) algorithm is introduced in Section II, experimental results are reported in Section III, and finally, conclusions and discussions are drawn in Section IV.

2. Proposed GPS Trajectory Compression Algorithm

2.1. Synchronized Euclidean Distance

In this paper, synchronized Euclidean distance [1] is used as the error metrics for evaluating the distortion of the approximated (compressed) GPS trajectories. Here, the error is measured through distances between pairs of temporally synchronized positions, one on the original and one on the approximated trajectories, which can be formulated as follows:

Suppose $P = \{p_1, p_2, \ldots, p_n\} = \{(x_1, y_1, t_1),\ldots,(x_n, y_n, t_n)\}$ and $P' = \{p_{i1}', p_{i2}', \ldots, p_{im}'\} = \{(x_{i1}', y_{i1}', t_{i1}'), \ldots, (x_{im}', y_{im}', t_{im}')\}$ are the original and the corresponding approximated GPS trajectories with $i_1 < i_2 < \ldots < i_m$, $i_1 = 1$, $i_m = n$ and $m \le n$. For each point $p_j = (x_j, y_j, t_j)$ on the original trajectory, its approximated synchronized position can be calculated as:

$$x_j' = x_{i_k}' + \frac{t_j - t_{i_k}'}{t_{i_{k+1}}' - t_{i_k}'} \cdot (x_{i_{k+1}}' - x_{i_k}'),\ where\ t_{i_k}' \le t_j \le t_{i_{k+1}}' \tag{1}$$

$$y_j' = y_{i_k}' + \frac{t_j - t_{i_k}'}{t_{i_{k+1}}' - t_{i_k}'} \cdot (y_{i_{k+1}}' - y_{i_k}'),\ where\ t_{i_k}' \le t_j \le t_{i_{k+1}}' \tag{2}$$

After p_i' is determined, synchronized Euclidean distance is calculated by:

$$SED(p_i, p_i') = \sqrt{(x_i - x_i')^2 + (y_i - y_i')^2} \tag{3}$$

In order to evaluate the distortion of the whole trajectory, maximum synchronized Euclidean distance is used, which is defined as:

$$SED(P, P') = \max_{1 \le i \le n}(SED(p_i, p_i')) \tag{4}$$

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

2.2. Approximate GPS trajectory with given error bound

We consider both the line simplification and the quantization in the approximation process. In vector map compression, differential coordinates are used in the encoding process. However, for GPS trajectories, because of the inconsistency of the differential coordinates after the approximation process, the coding efficiency may be reduced if differential coordinates are used directly. Meanwhile, speed and direction will be more consistent even if a simplification (approximation) step is applied with different reduction rate in different segments.

An approximation example is demonstrated in Fig. 1, suppose we want to approximate a sub-trajectory $P_{i_k}^{i_{k+1}}$ by line segment $\overline{p_{i_k}' p_{i_{k+1}}'}$, where p_{i_k}' is the approximated position in previous step. If time interval $\Delta t(k)$ is known, the speed of the line segment is:

$$spd(k) = d(p_{i_k}', p_{i_{k+1}}) / \Delta t(k) \tag{5}$$

Given maximum SED tolerance ε, we assume the quantization error of point $p_{i_{k+1}}'$ is $\gamma\varepsilon$ at maximum, thus the quantized level for speed can be set as:

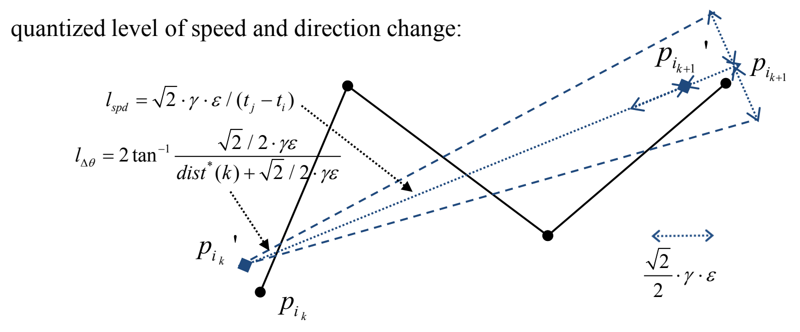
quantized level of speed and direction change:
$l_{spd} = \sqrt{2} \cdot \gamma \cdot \varepsilon / (t_j - t_i)$
$l_{\Delta\theta} = 2\tan^{-1} \frac{\sqrt{2}/2 \cdot \gamma\varepsilon}{dist^*(k) + \sqrt{2}/2 \cdot \gamma\varepsilon}$
$p_{i_{k+1}}'$
$p_{i_{k+1}}$
p_{i_k}'
p_{i_k}
$\frac{\sqrt{2}}{2} \cdot \gamma \cdot \varepsilon$

Fig. 1: An example of the quantization process for the GPS trajectory

$$l_{spd}(k) = \sqrt{2} \cdot \gamma \cdot \varepsilon / \Delta t(k) \tag{6}$$

Here γ is a parameter as the ratio of the quantized error and the total SED on the target point, which is set as γ = 0.5 by our experiment. Thus, the quantized speed for approximated segment can be calculated as:

$$spd^*(k) = [spd(k) / l_{spd}(k)] \cdot l_{spd}(k) \tag{7}$$

Meanwhile, we can get the direction change $\Delta\theta(k)$ with a value between –π and π, where negative value represents the direction change in clockwise.

Given the quantized speed $spd^*(k)$, the quantization level for the direction change can be estimated by:

$$l_{\Delta\theta}(k) = 2\tan^{-1} \frac{\sqrt{2}\gamma\varepsilon / 2}{spd^*(k) \cdot \Delta t(k) + \sqrt{2}\gamma\varepsilon / 2} \tag{8}$$

Thus, the quantized direction change is:

$$\Delta\theta^*(k) = [\Delta\theta(k) / l_{\Delta\theta(k)}] \cdot l_{\Delta\theta(k)} \tag{9}$$

Based on the quantized speed and direction change $spd^*(k)$ and $\Delta\theta^*(k)$, the quantized position $p_{i_{k+1}}' = \{x_{i_{k+1}}', y_{i_{k+1}}', t_{i_{k+1}}'\}$ can be approximated as:

$$\begin{aligned}
x_{i_{k+1}}' &= x_{i_k}' + spd^*(k) \cdot \Delta t(k) \cdot \cos(\Delta\theta^*(k) + \theta^*(k-1)) \\
y_{i_{k+1}}' &= y_{i_k}' + spd^*(k) \cdot \Delta t(k) \cdot \sin(\Delta\theta^*(k) + \theta^*(k-1)) \\
t_{i_{k+1}}' &= t_{i_{k+1}}
\end{aligned} \tag{10}$$

A greedy solution is used for the trajectory approximation in this paper. Start from the first point, the farthest point is found with an approximated SED less than the given error tolerance $SED(P_{i_k}^{i_{k+1}}, \overline{p_{i_k}' p_{i_{k+1}}'}) \le \varepsilon$ for every line segment $\overline{p_{i_k}' p_{i_{k+1}}'}$. The pseudocode can be seen in Fig. 2.

Note that when the input of GPS trajectory is latitude and longitude in WGS84 format, *Mercator projection* is needed as a preprocessing step so that the distance can be calculated directly.

```
ALGORITHM I, APPROXIMATION OF GPS TRAJECTORY
INPUT
$P = \{p_1, p_2, \ldots, p_n\}$: original trajectory
$\varepsilon$: SED error tolerance
OUTPUT
$P' = \{p_{i1}', p_{i2}', \ldots, p_{im}'\}$

$i \leftarrow 1$
$j \leftarrow 2$
$p_{i1}' \leftarrow p_1$
$m \leftarrow 2$
FOR $i = 1$ TO $i = n - 1$ DO
  IF $j > n$
     $p_{im}' \leftarrow p_n'$
     BREAK
  END
  $d \leftarrow SED(P_i^j, \overline{p_i' p_j'})$
  IF $d \le \varepsilon$
     $j \leftarrow j + 1$
  ELSE
     $p_{im}' \leftarrow p_{j-1}'$
     $m \leftarrow m + 1$
     $i \leftarrow j - 1$
     $j \leftarrow i + 1$
  END
END
```

Fig. 2: Pseudocode of proposed GPS trajectory approximation process

2.3. Encoding Process on the Approximated Trajectory

In the encoding process, we need to encode both the differential coordinates and time difference (Δx, Δy and Δt). Suppose p_{i_k}' and $p_{i_{k+1}}'$ are two neighbor points in the approximated trajectory P'. Firstly, the time difference is encoded by the following probability:

$$p(\Delta t(k) = t_{i_{k+1}} - t_{i_k}) = \frac{p + \delta_t / rtsp_{max}}{1 + \delta_t}$$
$$\text{where } p = \mathbf{r_t}(\Delta t(k) / min_{tsp}) / \sum_{s=1}^{rtsp_{max}} \mathbf{r_t}(s), rtsp_{max} = \max(\Delta t(q)) / tsp_{min}, q = 1,2,\ldots,m-1. \tag{11}$$

tsp_{min} is minimum sampling time internal on the GPS trajectory (1s in most cases) and δ_t is a bias factor ($\delta_t = 0.01$), vector $\mathbf{r_t}$ is initialized as a zero vector with size $rtsp_{max} \times 1$.

After $\Delta t(k)$ has been encoded, vector $\mathbf{r_t}$ is updated by:

$$r_t(s) = \begin{cases} 1 + \mu_t r_t(s), & s = \Delta t_k / tsp_{min} \\ \mu_t r_t(s), & \text{else} \end{cases} \tag{12}$$

where μ_t is a forgetting factor[13] which gives higher influence on the recently encoded time intervals with $\mu_t = 0.995$ in this paper. The reason that we use a forgetting factor is that the possible multi-model in the GPS trajectory. A higher reduction rate can possibly be achieved for the segments with slower moving speed (e.g., by walking) comparing with fast moving segments (e.g., by car). Thus, it will be helpful to improve the coding performance if a forgetting factor is used.

The speed value is then predicted by $spd_{pred}(k)$ and $\sigma_{spdpred}^{2}(k)$, which is:

$$\begin{aligned} spd_{pred}(k) &= n_{c1}\sum_i spd^*(i)\cdot(\Delta t(i)\cdot\exp(-\frac{1}{2}\cdot(\frac{t(k)-t(i)}{w_t})^2)), t(i)\ge t(k)-d\cdot\Delta t(k) \\ \sigma_{spdpred}{}^2(k) &= n_{c2}\sum_i \Delta t(i)\cdot((spd^*(i)-spd_{pred}(k))^2+\frac{\gamma^2\varepsilon^2}{6\Delta t^2(i)}+\frac{\sigma_{GPS}^2}{2\Delta t^2(i)} \end{aligned} \tag{13}$$

n_{c1} and n_{c2} are normalized values for the weighting factors, while w_t , d and σ_{GPS} are parameters with $w_t = 20$, $d = 4$ and $\sigma_{GPS} = 5$. The second and third term of $\sigma_{spdpred}^{2}(k)$ are the variance of the quantization procedure and the GPS error correspondingly.

The probability is then estimated by assuming the speed has a *Gaussian* distribution:

$$p(spd^*(k)) = \frac{p+\delta_{spd}/nlv_{spd}(k)}{1+\delta_{spd}} \tag{14}$$

where p has a Gaussian distribution with mean $spd_{pred}(k)$ and variance $\sigma_{spdpred}^{2}(k)$, bias factor δ_{spd} is set as 0.01.

For the direction changes, the encoding process has a similar form with the encoding process for time difference, which is:

$$\begin{aligned} & p(\Delta\theta^*(k)) = \frac{p+\delta_{\Delta\theta}/(2nlv_{\Delta\theta}^0+1)}{1+\delta_{\Delta\theta}}, \\ & \text{where } p = \begin{cases} \sum_{s=s_a(k)}^{s_b(k)} \mathbf{r}_{\Delta\theta}(s) / \sum_{s=-nlv_{\Delta\theta}}^{nlv_{\Delta\theta}} \mathbf{r}_{\Delta\theta}(s), & nlv_{\Delta\theta}^0 < nlv_{\Delta\theta}(k) \\ \frac{\mathbf{r}_{\Delta\theta}([\Delta\theta^*(k)/l_{\Delta\theta}^0])}{(s_d(k)-s_c(k)+1)} / \sum_{s=-nlv_{\Delta\theta}}^{nlv_{\Delta\theta}} \mathbf{r}_{\Delta\theta}(s), & \text{else} \end{cases} \\ & s_a(k) = [(\Delta\theta^*(k)-\frac{1}{2}l_{\Delta\theta}(k))/l_{\Delta\theta}^0],\ s_b(k) = [(\Delta\theta^*(k)+\frac{1}{2}l_{\Delta\theta}(k))/l_{\Delta\theta}^0] \\ & s_c(k) = [(\Delta\theta^*(k)-\frac{1}{2}l_{\Delta\theta}^0)/l_{\Delta\theta}(k)],\ s_d(k) = [(\Delta\theta^*(k)+\frac{1}{2}l_{\Delta\theta}^0)/l_{\Delta\theta}(k)] \\ & l_{\Delta\theta}^0 = \pi/nlv_{\Delta\theta},\ nlv_{\Delta\theta}(k) = \lceil \pi/l_{\Delta\theta}(k) \rceil \end{aligned} \tag{15}$$

where the size of vector $\mathbf{r}_{\Delta\theta}$ is $(1+2lv_{\Delta\theta}) \times 1$ and $nlv_{\Delta\theta}^0$ is set as 180 from our experiment.

After $\Delta\theta(k)$ has been encoded, vector $\mathbf{r}_{\Delta\theta}$ is updated by:

$$r_{\Delta\theta}(s) = \begin{cases} \frac{1}{s_b(k)-s_a(k)+1} + \mu_{\Delta\theta} r_{\Delta\theta}(s), & nlv_{\Delta\theta}^0 < nlv_{\Delta\theta}(k) \text{ and } s_a(k) \le s \le s_b(k) \\ 1+\mu_{\Delta\theta} r_{\Delta\theta}(s), & nlv_{\Delta\theta}^0 \ge nlv_{\Delta\theta}(k) \text{ and } s = [\Delta\theta^*(k)/l_{\Delta\theta}^0] \\ \mu_{\Delta\theta} r_{\Delta\theta}(s) & \text{else} \end{cases} \tag{16}$$

where forgetting factor $\mu_{\Delta\theta}$ is set as 0.995 here.

The probabilities of the time difference, speed and direction change are encoded by arithmetic coding. The pseudocode can be seen in Fig. 3.

Note that in the compressed file, a 192 bits fixed-length head file is used to save the parameters of the trajectory. These values includes start position of x (30 bit), y (30 bit), time(32 bit), tsp_{min} (8 bit), $rtsp_{max}$ (16 bit), m (24 bit), spd_{max} (32 bit) and scaling factor of *Mercator projection* (20 bit).

```
ALGORITHM II, ENCODING PROCESS OF THE APPROXIMATED TRAJECTORY
INPUT
$P = \{p_1, p_2, \ldots, p_n\}$: original trajectory
$\varepsilon$: SED error tolerance
OUTPUT
Encoding file

$P'$ ← Calculate the approximated trajectory by Figure 2.
Calculate parameters $rtsp_{max}$, $tsp_{min}$, $spd_{max}$
Write head file
FOR $k$= 1 TO $k$ = m - 1 DO
    Encode $\Delta t(k)$ by (11)
    Update $\mathbf{r}_t$ by (12)
    Predict $spd_{pred}(k)$ and $\sigma_{spdpred}^2(k)$ by (13)
    Encode $spd^*(k)$ by (14)
    Encode $\Delta\theta^*(k)$ by (15)
    Update $\mathbf{r}_\theta$ by (16)
END
```

Fig. 3: Pseudocode of the encoding process of the proposed GPS trajectory compression algorithm

2.4. Complexity Analysis

In this section, we give the complexity analysis for each step of the proposed algorithm, which is listed the Table I. Note that τ_1, τ_2 and τ_3 are constant values, which are not related to the size of the trajectory data.

TABLE I: Summary of the Expected Time Complexity of the Proposed GPS Trajectory Compression Algorithm

STEP		TIME COMPLEXITY
I. Approximated Trajectory		$O(n^2/m)$
II. Encoding Process	Time Difference	$O(m\cdot\tau_1)$, $\tau_1 = rtsp_{max}$
	Speed	$O(m\cdot\tau_2)$, $\tau_2 = \frac{spd_{\max}}{\varepsilon}\overline{\Delta t}$
	Direction Change	$O(m\cdot\tau_3)$, $\tau_3 = nlv_{\Delta\theta}$
III. Decoding Process		Same as the encoding process

3. Experiments

In order to evaluate the performance of the proposed compression method, we use *Microsoft Geolife dataset* [8] for testing purpose. It includes 640 trajectories with 4,526,030 points, which has a sampling rate between 1s to 5s with different transportation mode such as walking, bus, car, airplane or a multimodal. The code is written in Matlab and all the experiments have been performed on Intel Pentium-4 3.0 GHz CPU running Windows XP with 2G RAM.

The compression performances (KB/hour) are evaluated for different error tolerance: 1m, 3m, 10m, 30m and 100m maximum synchronous Euclidean distance (SED). The proposed GPS trajectory compression algorithm (GTC) is compared with the state-of-art method TD-TR [1][2]. LZMA (7-zip) is used to further compress the reduced trajectory of TD-TR algorithm [3]. We also evaluate the performance of vector map compression algorithm (VMC) [8] for compressing the position information, while the time information is encoded in the same way as in GTC. We can observe in Fig. 4 (left) that the coding cost of the proposed algorithm is only about 35%-40% compared with TD-TR, and it is consistent on different tolerance level. Note that for an uncompressed GPS trajectory with 1s sampling frequency, if twelve bytes were allocated for each point, the total storage cost would be 42.2 KB/hour. Thus, the proposed method achieves a compression ratio more than 100:1 for a 10m max SED. We also evaluate the reduction rate for different approximation algorithms in Fig. 4 (right). Experiments show that although we design a different reduction and approximation algorithm for the encoding purposes, the reduction rate is still similar.

Time costs of the encoding and decoding process are calculated and shown in Fig. 5 (left). The decoding cost reduces when maximum tolerance increases, because more points are reduced already in the approximation step. For the encoding process, since a $O(n^2/m)$ time complexity is needed for the approximation step, the time cost will slightly increase with the high tolerance case.

We evaluate the bit-rate for each component on the reduced trajectory: time difference, speed and direction change. They are compared with the coding cost using differential coordinates [8], which is given in Fig. 5 (right). We observe that the bit-rate of time difference increases when maximum SED increases. This is because the time difference varies more when the trajectory has a higher reduction rate. However, the bit-rate of speed and direction change will not increase even if a higher tolerance is used. This is because we select a higher quantized level when the given tolerance increases. We also notice that a lower bit-rate is needed when speeds and direction changes are considered instead of the differential coordinates, especially for lower error tolerance.

For the given max SED, its corresponding mean or median SED are also evaluated, which is around 50% of the maximum SED for all the tolerance levels in our experiment.

Note that GPS trajectories are never perfectly accurate, due to sensor noise and other factors. Many filtering algorithms are proposed which are summarized in [16]. From our experiment, if a filtering algorithm is performed beforehand, the bit-rate can be reduced around 30%, 20%, 15% for 1m, 3m, 10m maximum SED correspondingly. Meanwhile, if higher tolerance is set, bit-rate will not be changed even if a filtering operation is used.

[2] Various GPS compression algorithms reported in [10] are all based on the line simplification. There are only around 10%-20% differences on the reduction rate in all these methods. Thus, here TD-TR is selected as a typical example in our experiments to evaluate these types of solutions.

[3] We use a similar evaluation method with a commercial software on: http://www.droyd.org/gps-trajectory-compression.

Further information such as proof of the time complexity, details of the experiment result and the matlab code can be seen on http://cs.joensuu.fi/~mchen/GPSTrajComp.htm.

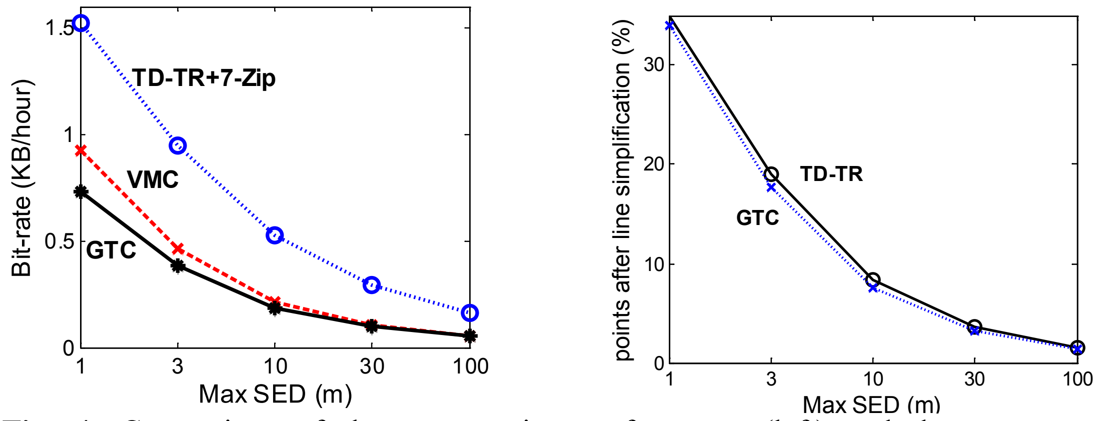

Fig. 4: Comparison of the compression performance (left) and the percentage of remaining points after approximation process.

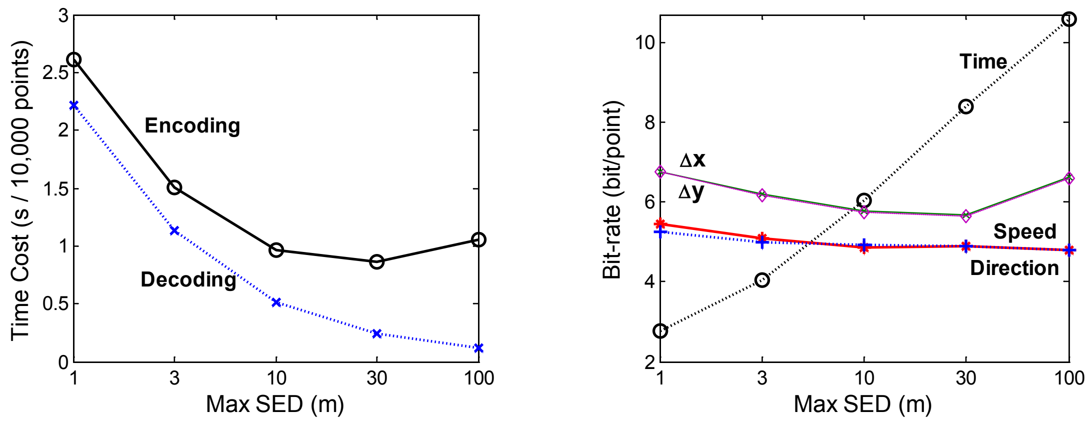

Fig. 5: Time cost of the encoding and decoding process (left), bit-rate of each reduced point for time, differential coordinates (VMC), speed and direction (GTC). (right)

4. Conclusion

In this paper, we have addressed the problem of spatial-temporal data compression, particularly the compression of GPS trajectories with sets of (x, y, t) records. In the proposed algorithm, both data reduction and quantization are considered in the approximation process. Experimental tests demonstrate that the proposed method makes a significant improvement comparing with the state-of-the-art TD-TR algorithm.

There are several immediate extensions of our present work. Firstly, we plan to extend the compression for online application. Also, improvement of approximation and encoding process by dynamic programming will also be considered. Finally, applying a hierarchy of compression stages is an interesting idea for further investigation.

References

[1] N. Meratnia and R. A. de By. "Spatiotemporal Compression Techniques for Moving Point Objects", *Advances in Database Technology*, vol. 2992, 551–562, 2004.

[2] 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.

[3] M. Potamias, K. Patroumpas, T. Sellis, "Sampling Trajectory Streams with Spatiotemporal Criteria", *Scientific and Statistical Database Management* (SSDBM), 275-284, 2006.

[4] H. Cao, O. Wolfson, G. Trajcevski, "Spatio-temporal data reduction with deterministic error bounds", *VLDB Journal*, 15(3), 211-228, 2006.

[5] A. Akimov, A. Kolesnikov and P. Fränti, "Coordinate quantization in vector map compression", *IASTED Conference on Visualization, Imaging and Image Processing* (*VIIP'04*), 748-753, 2004.

[6] S. Shekhar, S. Huang, Y. Djugash, J. Zhou, "Vector map compression: a clustering approach", *ACM Int. Symp. Advances in Geographic Inform*, 74-80, 2002.

[7] A. Kolesnikov, "Optimal encoding of vector data with polygonal approximation and vertex quantization", *SCIA'05*, LNCS, vol. 3540, 1186–1195. 2005.

[8] M. Chen, M. Xu and P. Fränti, "Fast dynamic quantization algorithm for vector map compression", *IEEE Int. Conf. on Image Processing,* 4289-4292, September 2010."

[9] 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.

[10] 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.

[11] 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.

[12] M. Chen, M. Xu and P. Fränti, "A Fast $O(N)$ Multi-resolution Polygonal Approximation Algorithm for GPS Trajectory Simplification", *IEEE Transactions on Image Processing* (in press).

[13] M. D. Reavy and C. G. Boncelet, "BACIC: a new method for lossless bi-level and grayscale image compression", *IEEE Int. Conf. on Image Processing*, vol.2, pp. 282-285, 1997.

[14] G. Kellaris, N. Pelekis and Y. Theodoridis, "Trajectory Compression under Network Constraints", *Lecture Notes in Computer Science*, Vol. 5644, pp.392-398, 2009.

[15] F. Schmid, K. F. Richter and P. Laube, "Semantic Trajectory Compression", *Lecture Notes in Computer Science*, Vol. 5644, pp.411-416, 2009.

[16] W. Lee, J. Krumm, "Chapter 1: Trajectory Preprocessing", in Book "Computing with Spatial Trajectories", Springer, 2011.