A Method for Online Trajectory Simplification by Enclosed Area Metric

Guangwen Liu [*], Masayuki Iwai [**] and Kaoru Sezaki [***]

[*] Institute of Industrial Science, University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo, 153-8505, Japan, liugw198209@mcl.iis.u-tokyo.ac.jp
[**] Institute of Industrial Science, University of Tokyo, masa@iis.u-tokyo.ac.jp
[***] Center for Spatial Information Science, University of Tokyo, sezaki@iis.u-tokyo.ac.jp

ABSTRACT

A novel simplification method for GPS trajectory is presented in this paper. Based on the observation of information content contained by sampling data, we assume that (1) the sampling points on the boundary of MBR (Minimum Bounding Rectangle) contain more information content, (2) the bigger the area of MBR is, and the more the points should be stored. We applied these two assumptions in our method to simplify trajectory online. Two integral parts of this method – divide/merge principle and selection strategy, are elaborated in the paper. Moreover, we define a more convincing error metric – enclosed area metric – to evaluate the accuracy of simplified trajectories. For this measure, we devise a practical algorithm of area calculation for self-intersecting polygons.

Through comparing with other methods in a series of experiments, our method is proven highly effective and efficient. In addition, it has an extensive application prospect in many fields especially in participatory sensing.

Keywords: Trajectory Simplification, Enclosed Area Metric, Self-intersecting Polygon, Information Content, Trajectory Model, Compression, Participatory Sensing.

1. INTRODUCTION

Nowadays, GPS-enabled devices, ranging from smart phones to vehicles, are drastically increasing. Market analyst firm Canalys reported 41 million standalone global positioning system (GPS) devices that were shipped worldwide in 2008, and it also estimated global smartphone-based GPS usage at about 27 million in 2009 [1]. Moreover, Location-based services and applications built from GPS-equipped mobile devices is a rapidly expanding consumer market, e.g., fleet management, traffic analysis and scientific investigations [2]. In addition, recently there has been a promising application called opportunistic or participatory sensing by smartphones [22, 23]. We can collect many rich and useful data, i.e. noise, illumination and temperature, just by normal users who travel with smartphone in daily life. These sensor data is usually generated along with trajectory. However, due to the limited memory of smartphone, a good trajectory compression method is indispensable for this application.

Although data generated from GPS devices are commonly used in a variety of businesses, these efforts would be hindered by the massive volumes of data, which creates the problem of storing, transmitting, and processing this data [3]. Storing the data is difficult because the sheer volume of data can rapidly overwhelm available data storage. For instance, if data is collected at 30 seconds intervals for 400 users with GPS-equipped smartphone, the volume would be up to 1.1 GB in a month. In addition, these trajectories will cause a heavy load for network transferring which costs highly in view of money and time. The cost of sending large volume of data over remote networks can be prohibitively expensive, normally ranging from $5 to $7 per megabyte [4]. In addition, the enormous volume of data can easily overwhelm human analysis and further computing. For example, towards querying and clustering trajectories, the performance would exponentially decrease due to the number of position data [5, 6].

The aforementioned restrictions motivate the need for automated methods to compress and analyze the data. Moreover, trajectory data is usually collected in a random manner; consequently a part of information is redundant and reducible. Hence, numerous compression methods have been proposed to reduce the size of trajectory data sets [7, 8]; however, these methods often either lose some contextual information or are computation-expensive. In this paper, we present a novel compression method to quickly simplify the trajectory before the position data is transmitted to the server from GPS terminals. Our contributions can be summarized as follows:

(1) We propose a simplification method based on information content MBR which can largely keep as same information content as counterpart of original trajectory.

(2) We also introduce a new error metric based on enclosed area to measure the displacement between original trajectory and simplified one.

(3) Last but not least, evaluate the accuracy or data loss with other typical simplification methods in terms of perpendicular distance, synchronized Euclidean distance and enclosed area.

The next section describes previous work for compressing trajectories. In section 3 our method is described in detail. The evaluation of our method and other algorithms with 3 error metrics including newly introduced metric is described in section 4. Finally, discuss experimental results and future work.

2. RELATED WORK

In the literature various methods exist [9, 10, 11] and Lawson et.al conducted a very comprehensive survey for them [3]. Most widely used methods are uniform sampling, dead reckoning method and Douglas Peucker method, which also are the comparison targets in our paper. We will introduce these existing methods and analyze their advantages and disadvantages here.

2.1 Uniform Sampling

Uniform sampling is the most native method which sparsely selects the point to store every given time interval or distance interval but discards remained points. In some applications, this method is modified by storing the average value of all points within given interval, which is called piece-wise aggregate approximation. Even though uniform sampling may provide a simple and cost-effective solution, it is distinctively insensitive to the spatiotemporal characteristics of the trajectory as well as to its sequential nature. Thus, even though the result is satisfactory in some case, we cannot expect the consistent quality just since it is too sensitive to the case.

2.2 Dead Reckoning Method

It is a localized processing routine which make use of the characteristics of the immediate neighboring coordinate points in deciding whether to retain the current point. As shown in figure 1, P3 and P4 are in the same trend with the line segment consisted of P1 and P2. However, since P5 exceeds the threshold of Euclidean distance predefined, the prior point of P5 would be retained in the simplified trajectory so that the maximum distance displacement does not go beyond the predefined Ɛ. This method has two advantages: (1) it can process the data at local client (mobile terminals) (2) its time complexity is O(n), namely linear. Therefore, it is popular in car navigation though it would accumulate the error in bad case. There are also some variants of this method [7, 21].

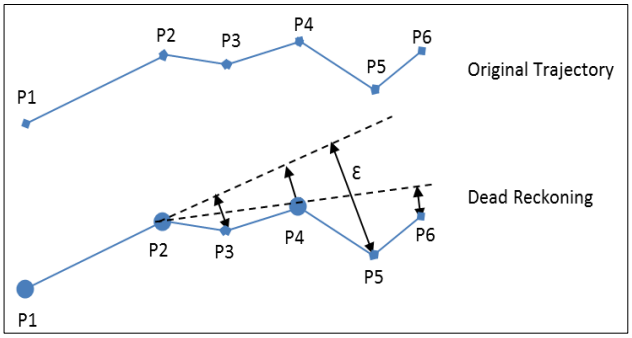

Figure 1: Dead Reckoning Simplification

2.3 Douglas Peucker Method

DP method was proposed by Douglas and Peucker [13], which is widely used in cartography related software like AUTOCAD. "Many cartographers consider it to be the most accurate simplification algorithm available, while others think that it is too slow and costly in terms of computer processing time" [14]. In any case, it is the most famous line simplification algorithm till now. The method recursively selects two points to represent the line segment within a specified tolerance value (see figure 2). Firstly, it attempts to simplify the trajectory with Pa and Pb, but it discards this attempt when calculate perpendicular distance from every point to line Pa-Pb and find Pc is out of the predefined threshold Ɛ. Then, it chooses Pc as new anchor point and repeats the attempts with Pa –Pc and Pc-Pb respectively.

Douglas Peucker Method is easy to program and extremely efficient comparing with other methods. Besides, as its basic idea is universal, it has been independently proposed in other contexts, i.e. image processing, computational geometry, and there is also many work that try to improve this method [15]. Nevertheless, this method loses its power when the trajectory includes self-intersection points. In addition, this method is very computing-expensive in some cases. A straightforward implementation requires O(n) time to find the furthest point from line. Since the iteration depth is linear, the worst-case running time is O(n^2).

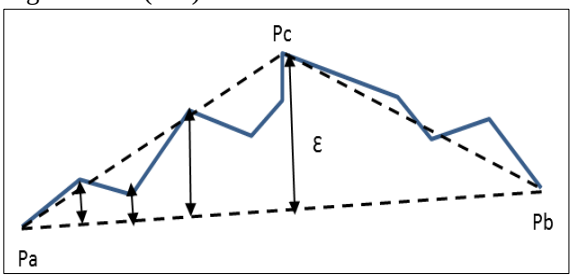

Figure 2: Douglas Peucker Simplification

2.4 Other Methods

Aside from these conventional methods, recently researchers also presented other interesting methods. Yukun Chen et.al [12] proposed such a method to simplify trajectory for LBS networking services. The method focuses on keeping speed and direction change information as much as possible. Hence, they defined the attributes of line segments, including heading direction, neighbor heading change, accumulated heading change, heading change which is the sum of the neighbor heading change and accumulate heading change, and neighbor distance. Then, the method assigns the weight on point in terms of the product of the average heading change and the neighbor distance. Lastly, the method selects the points with high

weight to represent all the sampling points. Their approach is said to outperform Douglas-Peucker method in walking mode trajectories with some constraints. In any case, it is a global process routine so we will not compare it with ours since the purpose of our method is to process data online at the mobile terminal.

The STTrace algorithm [7] is designed to preserve spatiotemporal heading and speed information in a trace. A hybrid between an online and offline approach, STTrace defines a safe area by using the previous two points in the trajectory. A vector which defines the speed and heading between the two locations is used to predict the position of the next point. It uses two input parameters to make this prediction. One of these parameters is the speed threshold which defines how much the speed can vary while still remaining in the predicted range. The other input parameter is the heading threshold that defines how much the heading can vary while still remaining in the predicted range. STTrace is similar to dead reckoning method while it uses the predicted area to filter not the fixed point as in DR method. Although it can overcome complications caused by error propagation, which improved the demerit of DR method, its processing time is far worse than the threshold-based methods.

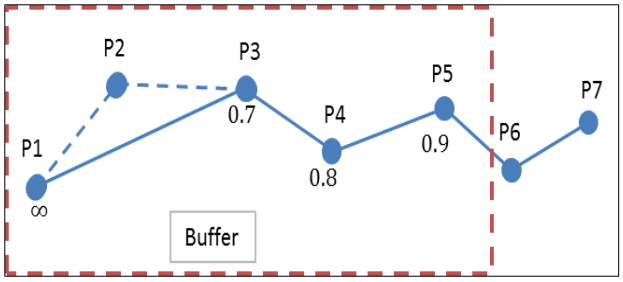

Figure 3: SQUISH Simplification

In addition, Jonathan Muckell et.al [8] proposed a method called SQUISH to simplify trajectory using a priority queue. As shown in Figure 3, the method sets a buffer for processing points in which all points will be assigned a weight. The weight value is determined based on estimating the amount of synchronized Euclidean distance introduced into the compression if that point was removed from the trajectory. It is straightforward to delete the points with the lowest weight in order to keep the shape of trajectory as same as possible, whereas it is a little confusing to add the deleted point's weight on the neighbor point. For example, P2 with the lowest weight is moved out from the buffer, but the weight of P3 cannot be simply obtained by adding the weight of P2. The experiment of this method demonstrates a good result comparing with other methods when the compression rate is not large, otherwise it lost the lead.

All these methods are data-loss methods, though there are data-lossless compression methods in other fields [18]. Usually data-lossless compression is computing-expensive, and for trajectory to some extent, data-loss is acceptable in most cases. Hence, the point is to diminish the data-loss under arbitrary compression ratio. That is, the data-loss measures or error metrics are also extremely important. Generally the error in the domain of trajectory compression is measured by distance, including the perpendicular distance and synchronized Euclidean distance (see figure 4, sampling interval is assumed as constant time interval). As we can see, in some case even the distance displacement is trivial but the enclosed area displacement would be quite remarkable. Since distance is just a discrete measure while area is a continuous measure, we can claim that enclosed area is a more accurate error metric. Thus, in our work, we evaluate methods by all these 3 metrics, namely perpendicular distance, synchronized Euclidean distance and enclosed area.

3. PROPOSED METHOD (IC_MBR)

3.1 Problem Statement

Trajectory is obtained by recording the successive positions of which a moving object takes across time. More formally, an original trajectory T can be defined as:

$$T = \{(X(t), Y(t), Z_*(t), Sm(t), t) | t \in I\}$$

Where $(X, Y, Z) \subset R^3$, $Sm \subset R^n$ and Sm(t) represents other sensing data (i.e. noise, illumination) which vary along with trajectory over time. This is a general definition of multidimensional trajectory, whereas we do not consider Sm data in this study.

On the other hand, a simplified trajectory T′ can be

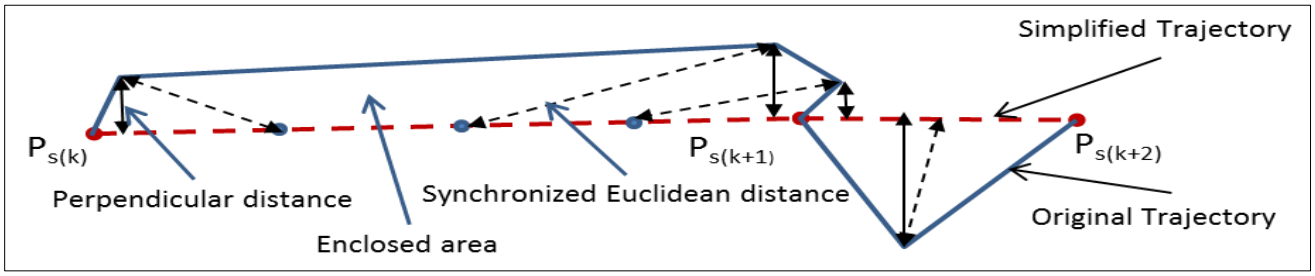

Figure 4: Error Metrics

defined as:

$$T' = \{(X(t), Y(t), Z_*(t), Sm(t), t) | t \in I \cap \Delta(t) > \varepsilon\}$$

In fact, we can describe the nature of the problem (trajectory simplification) as constructing a threshold function $\Delta(t)$ in order to achieve the least data loss within favorable or given compression ratio. Our method, just like the existing method, is also expected to reach the goal.

3.2 Observation

Our Method is derived from such an observation. In figure 5, there is a sample of sound data on which 3 MBRs are drawn. In addition, the histogram of this sample is shown in figure 6, and we can conclude such assumptions by analyzing the information content of each range of sample data (see table 1):

(1) The bigger the area size of MBR of IC (Minimum Bounding Rectangle of Information Content) is, the more the sampling points should be stored (see MBR 2 in figure 5). Otherwise, we can omit more sampling points (see MBR 3).

(2) The sampling points on the boundary of MBR contain more information content (see the circled points in figure 5 and the corresponding IC value (with gray) in table 1).

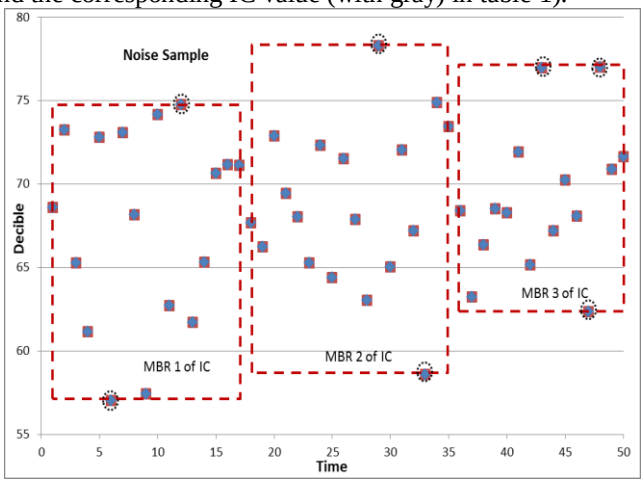

Figure 5: IC MBR on Noise Sample

Table 1: Information Content of Noise Sample

Data Range	Information Content
55~60 db	-log(3/50) = 2.81
60~65 db	-log(7/50) = 1.97
65~70 db	-log(19/50) = 0.98
70~75 db	-log(18/50) = 1.02
75~80 db	-log(3/50) = 2.81
Sum	$\sum_{i=1}^{N} -\log(p_i)$ = 9.59

Based on this observation, our purpose is to achieve the least data loss. In other words, our objective is to keep the sum of information content as same as the original trajectory's counterpart, can be implemented. In a spatial trajectory, tuples of (x, y) can be seen as same as the data in that sample of sound data, since information content is reflected by the area of every group of sampling points and points on the boundary are more important. Thus, we propose the following method (call IC_MBR method) to simplify trajectory.

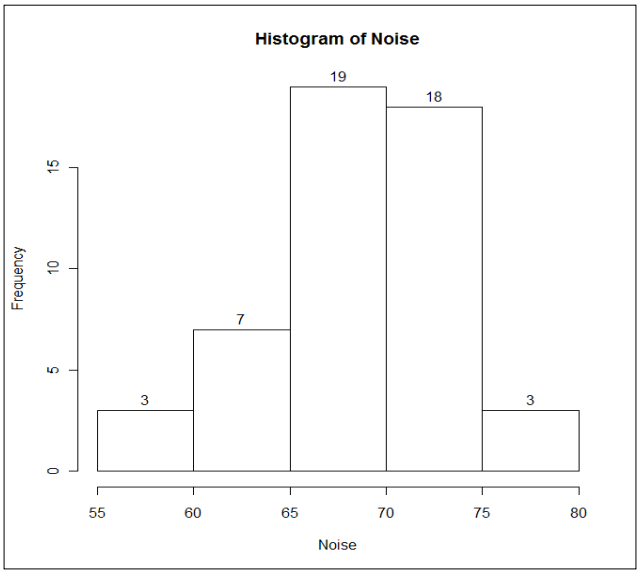

Figure 6: Histogram of Noise Sample

3.3 Divide and Merge Principle

Based on the first assumption stated in section 3.2, such a principle is applied, that we divide the bigger MBRs but merge the smaller MBRs so as to keep the nearly uniform size of MBR. There is an example shown in figure 7. Initially, 4 MBRs are drawn on it by every 4 points (the number is an input parameter specified by user), and then merge MBR 2 and MBR 3 because their area is far less than the standard MBR but split MBR 4 because it is far bigger than the standard MBR (standard MBR is an input parameter which is referred to comparing individual MBR). The size of the standard MBR can be obtained by user's experience or specific requirements, which as well as its points number directly affect accuracy and compression ratio of trajectory that will be discussed in later section. Another technique to determine an appropriate standard MBR (called adaptive MBR) is to dynamically adjust the value by calculating area size within a tuning period (i.e. take the median value of all MBRs). To keep the consistent accuracy, an adaptive MBR is more effective in the case of multi-transportation mode, since the area is subject to variations depending on walk mode or driving mode. Assuming that the original sampling interval is a fixed time interval, standard MBR area is supposed to be assigned as a greater value in driving mode but a less value in walking mode. Hence, if the area or points number of standard MBR is dynamically programmed with the consideration of both

transportation mode and sampling interval, the result may be more satisfactory.

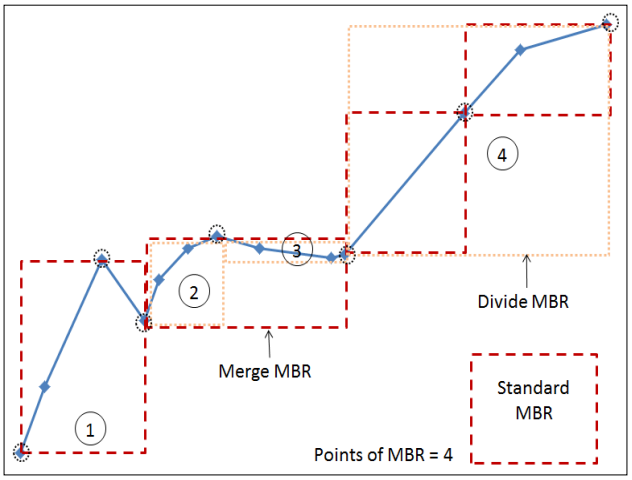

Figure 7: The Illustration of IC_MBR Method

3.4 Selection Strategy

Through dividing or merging MBRs, every resulted MBR will contain the comparatively uniform information content. Hence, we apply such a strategy to extract points based on the second assumption stated in section 3.2. As shown in table 2 (call 4-2-1-0.5 rule), the points to be stored are determined by comparison with the standard MBR area. For instance, select 4 points on boundary of MBR when the MBR meets condition 2 (see table 2), select the first point and the last point when meets condition 3, and select the median point when meets condition 4. In the case of the condition 5, if the MBR is a divided MBR then select the median point; or merge the MBR (which is explained in section 3.3 as well as rule 1 of table 2).

Table 2: Points Selection Strategy

No	Condition	Selection Criteria
1	MBR(N) ⊂ [St_MBR * 2, ∞)	Divide MBR
2	MBR(N) ⊂[St_MBR, St_MBR * 2)	4 points; x(min), y(min), x(max), y(max)
3	MBR(N) ⊂ [St_MBR * 0.5, St_MBR)	2 points; x(0), x(N-1)
4	MBR(N) ⊂ [St_MBR * 0.25, St_MBR * 0.5)	1 point; x(median)
5	MBR(N) ⊂ [0, St_MBR * 0.25)	0.5 point; Merge MBR or x(median)

After integrating the divide/merge principle with the selection strategy, algorithm of IC MBR method can be described as figure 8. This method adapts bottom-up and top-down strategy simultaneously, which recursively approximate line segments within a rectangle.

```
IC_MBR_Algorithm(St_MBR_Points_Num,
                 St_MBR_Area, Traj){
  Num = St_MBR_Points_Num;
  Foreach point in Traj{
   If Num = Buf.Count {
     rlt = SelectPoints(Buf);
     if rlt = false then Num = Num * 2; //Merge MBR
   }else{ Buf.Add(point);}
  }
}
SelectPoints(Buf){
  Area = (Max_X(Buf) – Min_X(Buf)) * (Max_Y(Buf) –
Min_Y(Buf));
  If Area > St_MBR_Area * 2{ //divide MBR
    SelectPoints(Buf/2); //first half of Buf
    SelectPoints(Buf/2); //second half of Buf
  }else if Area < St_MBR_Area / 4{
    Return false; //need to merge
  }else SavePoints(); //by selection strategy
}
```

Figure 8: The Algorithm of IC_MBR Method

4. EVALUATION METHODS

We developed a tool to implement IC_MBR method, uniform sampling method, dead reckoning method and Douglas-Peucker method which is slightly modified to adapt online processing, and to evaluate the accuracy of each method in terms of perpendicular distance (PD), synchronized Euclidean distance (SED) and enclosed area (EA). It is important to note that even though DP method is an offline method, we add a buffer for local processing so that we can compare these methods under fair conditions. Of course, this additional parameter may affect the initial performance to some extent but improve the time cost.

4.1 Two Conventional Error Metrics

Average perpendicular distance and average synchronized distance are uniformly defined as:

$$m_{1,2}(T^o, T^s) = \frac{1}{N}\sum_{i=1}^{N} d_i^2$$

Here, N is the total points of original trajectory.

PD refers to the distance between the point of original trajectory and the line segment of simplified trajectory (see figure 4), which is defined as:

$$d_i = \frac{\left|(x_{s(k+1)} - x_{s(k)})(y_{s(k)} - y_i) - (y_{s(k+1)} - y_{s(k)})(x_{s(k)} - x_i)\right|}{\sqrt{(x_{s(k+1)} - x_{s(k)})^2 + (y_{s(k+1)} - y_{s(k)})^2}}$$

SED considers the temporal attribute of the point sequence, thus it is considered as a better error metric. It calculates the distance from original point to virtual

simplified point which is at identical timestamp (see figure 4). The virtual simplified point is missing in simplified trajectory, whereas it can be obtained as follows:

$$P_i = P_{s(k)} + \frac{P_{s(k+1)} - P_{s(k)}}{t_{s(k+1)} - t_{s(k)}} t_i$$

4.2 New Error Metric

However, both PD and SED are discrete error metric, consequently there is a considerable flaw just as described in section 2. For that reason, we introduce a continuous error metric – enclosed area which is a polygon confined by original trajectory and simplified trajectory (see figure 4). Although EA is obviously advantageous, its calculation is quite troublesome [19, 20] provided that the polygon contains self-intersection (see figure 9). Anyway, we devised such an algorithm to solve this problem (see figure 10): (1) the intersection point can be obtained by geometry formula. (2) If exist cross point, it would be sequentially inserted into a list which consists of original points, but do not insert two or more intersection points on the same line segment. For example, the list L of figure 9 would be (P1, P'1, P2, P'2, P3, P'3, P4, P'4, P5, P1). (3) The sub sequence of L split by intersection point is a line segment or a simple polygon whose area can be easily obtained as follows – by the outer product of vector:

$$Q_j = \frac{1}{2} \left| \sum_{i=1}^{K} \overrightarrow{p_i} \times \overrightarrow{p_{i-1}} \right|$$

According to this algorithm, the nested area (see figure 9) would be accumulated two or more times, whereas it is reasonable. Then, we formally define the third error metric – average enclosed area as:

$$m_3(T^o, T^s) = \frac{1}{N} \sum_{j=1}^{J} \text{area}(Q_j)$$

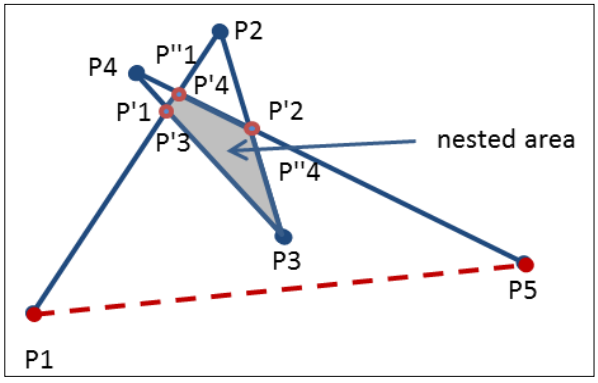

Figure 9: A Self-intersecting Polygon

```
CalcAreaOfPolygon (Polygon, PointsNum){
  Foreach point pi in Polygon{
  List.Add(pi, 0);//0: original point
  For(j=0; j<i-1 && i>1; j++){
    P = CrossPoint(pi, pi+1, pj, pj+1);
    If (!List.Find(p'i)) List.Insert(p'i =P, 1); //after pi
    If (!List.Find(p'j)) List.Insert(p'j =P, 1); //after pj
  }
  Anchor = 0;
  For(i=0; i<List.Count; i++){
    If List[i].flag = 1 {
      CalcAreaOfSimplePolygon(List.Range(anchor,i);
      Anchor = i;
    }
  }
  CalcAreaOfSimplePolygon(List.Range(anchor,i);
}
```

Figure 10: The Algorithm of Area Calculation of Arbitrary Polygon

5. EXPERIMENTAL RESULTS AND DISCUSSION

In this study a series of experiments are conducted by using the Microsoft GeoLife [16, 17] dataset that consists of 178 users in a period of over four years (from April 2007 to October 2011). A GPS trajectory of this dataset is represented by a sequence of time-stamped points, each of which contains the information of latitude, longitude and altitude. Various transportation modes are included in the data set, including walking, driving, train travel and etc. Aside from GeoLife data, we also use other trajectories collected by ourselves which encompass more data items. Experimental data files are selected by different file size, different transportation modes and different trajectory shapes so that we can compare the performance to draw a general conclusion.

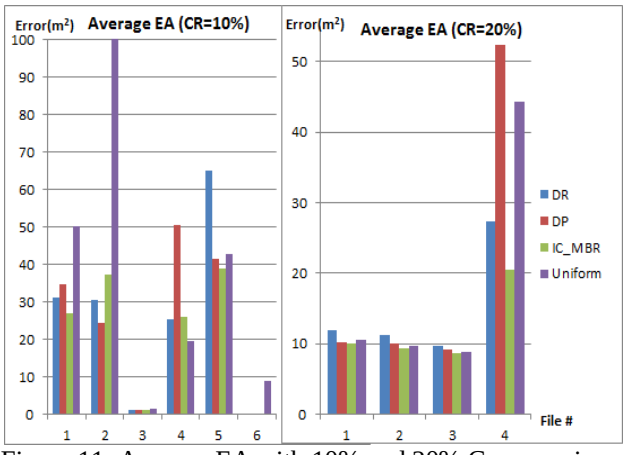

Figure 11: Average EA with 10% and 20% Compression Ratio

Trajectories are compressed with 3%, 10%, 20% and 50% and are measured by 3 error metrics (namely EA, PD and SED). Here, compression ratio is defined as the number of original points divided by the number of compressed points. From figure 11 (the results of compression ratio 10% and 20% are shown and other compression ratio results are skipped here, but a complete comparison is described later), we can find that (1) accuracy gets worse as compression ratio decrease; (2) accuracy also greatly varies due to different trajectory files which mean different shape of trajectories; (3) uniform sampling produces an uncertain output, in other words, its performance drastically fluctuates. In addition, our method can meet different compression ratios and error tolerances by adjust the parameters (points of standard MBR and its area). Generally speaking, these two parameters are similar to distance threshold in DP or DR method, that is, lessening the value of them would earn better accuracy but high compression ratio.

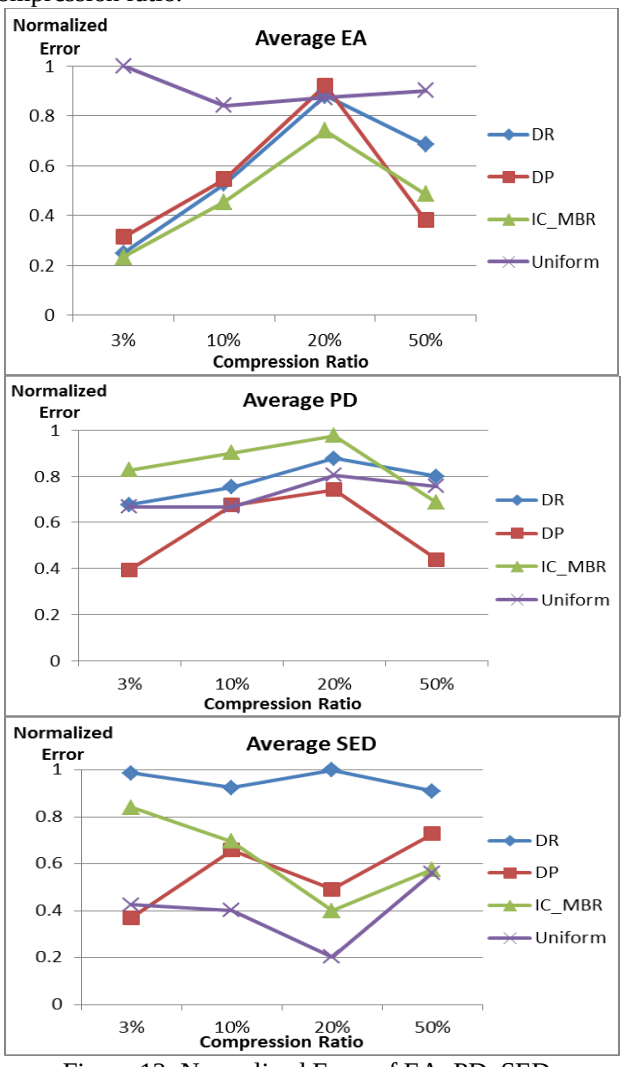

Figure 12: Normalized Error of EA, PD, SED

Figure 12 shows the normalized error results of EA, PD and SED in different compression ratio. As a result, our method holds an absolute advantage in EA metric and competitive performance in SED metric but poor performance in PD. Our method filters point based on information contents which is measured by area in 2-dimensional plane, which resulted in a good performance in EA. Besides, our method runs relatively fast because its time complexity is $O\left(\frac{n}{\beta} * \beta \log \beta\right) = O(n \log \beta)$ where $\boldsymbol{\beta}$ is the number of MBR points.

6. CONCLUSION AND FUTURE WORK

Although DP method still outperforms other methods from the perspective of whole performance (by comparing the average value of all 3 metrics), our method is the most efficient method in terms of EA metric – the most convincing measure. Furthermore, our method is a pure online procedure which can be readily installed at the mobile terminal to preprocess trajectory before sending it to remote server. On the other hand, the transformed online DP method needs a big enough buffer to guarantee the accuracy and compression ratio.

In the future, we consider extending the MBR of IC idea to Minimum Bounding n-dimensional Cube so as to compress multidimensional trajectory. Furthermore, it would be extremely challenging and meaningful to explore the relationship between the accuracy and the features of trajectory, eventually to seek optimal input parameters (number of standard MBR points and area of standard MBR) and better selection strategy.

REFERENCES

[1] Canalys, http://www.canalys.com/static/press_release/2009/r2009031.pdf, Canalys research release, 2009

[2] Axel Kupper, Location-based services: Fundamentals and Operation, England: John Wiley & Sons Ltd, pp.3 -10(2005).

[3] Catherine T. Lawson, S. S. Ravi, Jeong-Hyon Hwang, Compression and Mining of GPS Trace Data: New Techniques and Applications, Technical Report, State University of New York at Albany, 2011.

[4] M. Prior-Jones. Satellite Communications Systems Buyer's Guide. British Antarctic Survey, 2008.

[5] Yanagisawa, Yutaka, Shape-Based Similarity Query for Trajectory of Mobile Objects, in 4th international conference on mobile data management (MDM'03), pp. 63-71(2003).

[6] M. Wirz, P. Schläpfer, M.B. Kjærgaard, et.al, Towards an online detection of pedestrian flocks in urban canyons by smoothed spatio-temporal clustering of GPS trajectories, Proceedings of the 3rd ACM

SIGSPATIAL International Workshop on LBSN, 2011.
[7] M. Potamias, K. Patroumpas, and T. Sellis, Sampling trajectory streams with spatio- temporal criteria, in 18th International Conference on Scientific and Statistical Database Management (SSDBM'06), pp. 275-284(2006).
[8] Jonathan Muckell, Vikram Patil, Fan Ping, SQUISH: An Online Approach for GPS Trajectory Compression, Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications (Com.Geo'11), 2011.
[9] J. Gudmundsson, J. Katajainen, D. Merrick, C. Ong, T. Wolle, Compressing spatio-temporal trajectories, 18th International Symposium, ISAAC 2007, pp.763-795(2007).
[10] George Taylor, LINE SIMPLIFICATION ALGORITHMS,http://www.comp.glam.ac.uk/pages/staff/getaylor/papers/lcwin.pdf, 2005.
[11] Hu Cao, Ouri Wolfson, Goce Trajcevski, Spatio-temporal Data Reduction with Deterministic Error Bounds, VLDB Journal, vol. 15, no. 3, pp.211–228(2006).
[12] Yukun Chen, Kai Jiang, Yu Zheng, et.al, Trajectory Simplification Method for Location-Based Social Networking Services, Proceedings of the 2009 International Workshop on LBSN(LBSN'09), pp.33-41(2009).
[13] Douglas, D. and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitised line or its caricature, The Canadian Cartographer, Vol. 10, pp. 112-122(1973).
[14] Jenks, G.F., Geographic logic in line generalisation, Cartographica, Vol. 26, No. 1, pp. 27-42 (1989).
[15] John Hershberger, Jack Snoeyink, Speeding Up the Douglas-Peucker Line-Simplification Algorithm, http://www.bowdoin.edu/~ltoma/teaching/cs350/spring06/Lecture-Handouts/hershberger92speeding.pdf, 1992.
[16] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma, Understanding mobility based on gps data, In Proceedings of the 10th international conference on Ubiquitous computing(UbiComp'08), pp.312–321(2008).
[17] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, Mining interesting locations and travel sequences from gps trajectories, In Proceedings of the 18th international conference on World wide web(WWW'09), pp.791–800(2009).
[18] Richard Baraniuk, Compressive Sensing, IEEE signal processing magazine, vol. 1053, 2009.
[19] J. Nievergelt, F. P. Preparata, Plane-sweep algorithms for intersecting geometric figures, Communications of the ACM, vol. 25, pp.739-747(1982).
[20] Sang C. Parka, Hayong Shinb, Polygonal chain intersection, Computers & graphics, vol. 26, pp.341-350(2002).
[21] Ralph Lange, Frank Dürr, Kurt Rothermel, Online Trajectory Data Reduction using Connection-preserving Dead Reckoning, In Proceedings of the 5th International Conference on Mobile and Ubiquitous Systems, 2008.
[22] Nicholas D. Lane, Emiliano Miluzzo, Hong Lu, et.al, A survey of mobile phone sensing, IEEE communications magazine, vol. 48, pp.140-150(2010).
[23] Tathagata Das, Prashanth Mohan, Venkata N. Padmanabhan, PRISM: platform for remote sensing using smartphones, Proceedings of the 8th international conference on Mobile systems, applications, and services (MobiSys'10), pp.63-70(2010).