IPSJ Transactions on Databases Vol.6 No.3 40–49 (June 2013)

Regular Paper

An Online Method for Trajectory Simplification Under Uncertainty of GPS

Guangwen Liu[1,a)] Masayuki Iwai[2,b)] Kaoru Sezaki[3,c)]

Received: December 20, 2012, Accepted: April 7, 2013

Abstract: A novel simplification method for GPS trajectory is presented in this paper. Trajectory simplification can greatly improve the efficiency of data analysis (e.g., querying, clustering). 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, the more the points should be stored. We applied these two assumptions in our method to simplify trajectory online. Two main components of this method (i.e., divide/merge principle and selection strategy), are elaborated in the paper. Moreover, we define a new error metric — enclosed area metric — to evaluate the accuracy of simplified trajectories, which is proven more robust against the uncertainty of GPS. To implement this measure, we devise a practical algorithm of area calculation for self-intersecting polygons. Through comparing with other methods in a series of experiments over huge dataset, our method is proven effective and efficient.

Keywords: trajectory simplification, enclosed area metric, GPS uncertainty, data reduction, data compression

1. Introduction

Nowadays, GPS-enabled devices, ranging from smart phones to vehicles, are drastically increasing. 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 [1]. In addition, recently there has been a promising application called opportunistic or participatory sensing by smartphones [2], [3]. We can collect many rich and useful data, e.g., noise, illumination and temperature, just by normal users who travel with smartphone in daily life. These sensor data is usually generated along with trajectory. Although data generated from GPS devices are commonly used in a variety of businesses, these efforts will be hindered by the massive volumes of data, which creates the problem of storing, transmitting, and processing [4]. 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 will 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 [5]. The foremost issue is that the enormous volume of data can easily overwhelm human analysis and further computing. For example, towards querying and clustering trajectories, the performance will exponentially decrease due to the number of position data [6], [7].

The above restrictions motivate the need to reduce data volume. 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 [8], [9]. However, these methods often either lose some contextual information or are computation-expensive. Besides, conventional compression methods, such as *LZ* (used in zip) or *DCT* (Discrete Cosine Transformation) can compress the data volume, whereas it does not improve the data processing efficiency (e.g., querying, clustering) as data should be uncompressed to original volume before processing. 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 MBR of information content, 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) Through simulation, enclosed area metric is proven more robust against GPS uncertainty comparing to distance metric.
(4) Evaluate the accuracy with other typical simplification methods in terms of perpendicular distance, synchronized Euclidean distance and enclosed area. In addition, we estimate the effect of parameters over the performance of our method.

The next section describes related work about compressing tra-

[1] Institute of Industrial Science, The University of Tokyo, Meguro, Tokyo 153–8505, Japan
[2] School of Science and Technology for Future, Tokyo Denki University, Adachi, Tokyo 120–8551, Japan
[3] Center for Spatial Information Science, The University of Tokyo, Kashiwa, Chiba 277–8568, Japan
[a)] liugw198209@mcl.iis.u-tokyo.ac.jp
[b)] iwai@im.dendai.ac.jp
[c)] sezaki@iis.u-tokyo.ac.jp

jectories. 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 simplification methods exist [10], [11], [12] and Lawson et al. conducted a very comprehensive survey for them [4]. 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 a naive method which sparsely selects the point to store by 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 spatio-temporal characteristics of the trajectory as well as to its sequential nature. Hence, we can't expect the consistent quality just since it is too sensitive to the case, though the result is satisfactory in some case.

2.2 Dead Reckoning Method

It is a localized processing routine which make use of the characteristics of the immediate neighbouring coordinate points in deciding whether to retain the current point. As shown in **Fig. 1**, P_3 and P_4 are in the same trend with the line segment consisted of P_1 and P_2. However, since P_5 exceeds the threshold of Euclidean distance predefined, the prior point of P_5 will be retained in the simplified trajectory so that the maximum distance displacement does not go beyond the predefined ϵ. 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 accumulates the error in bad case. There are also some variants of this method [8], [13].

2.3 Douglas Peucker Method

DP method was proposed by Douglas and Peucker [14], 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" [15]. 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 **Fig. 2**). Firstly, it attempts to simplify the trajectory with $\overrightarrow{P_aP_b}$, but it discards this attempt when calculate perpendicular distance from every point to line $\overrightarrow{P_aP_b}$ and find P_c is out of the predefined threshold ϵ. Then, it chooses P_c as new anchor point and repeats the attempts with $\overrightarrow{P_aP_c}$ and $\overrightarrow{P_cP_b}$ respectively.

As described above, the algorithm of Douglas Peucker is simple and easy to program. It is extremely efficient because it globally exhibits the least distance error under a given tolerance. Thus, it is recognized as the one that delivers the best perceptual representations of the original lines [16]. Moreover, Douglas Peucker Method is applicable to both 2D line and 3D line in terms of any shape of line. 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 [17]. 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 $O(n^2)$.

2.4 Other Methods

Aside from these conventional methods, recently researchers also presented other interesting methods. Yukun Chen et al. [18] 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.

Besides, the STTrace algorithm [8] is designed to preserve spatiotemporal heading and speed information in a trace. A hybrid between an online and offline approach, STTrace defines a safe

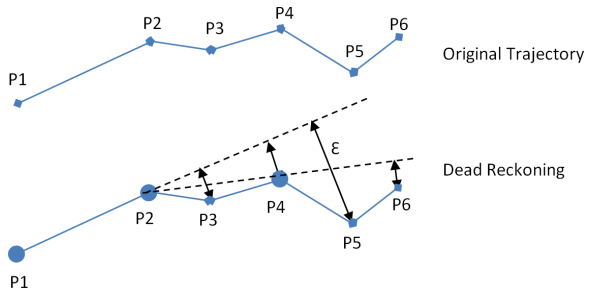

Fig. 1 Dead reckoning simplification.

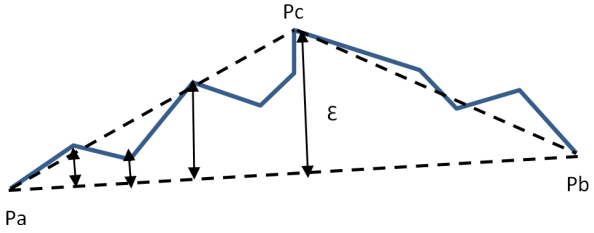

Fig. 2 Douglas Peucker simplification.

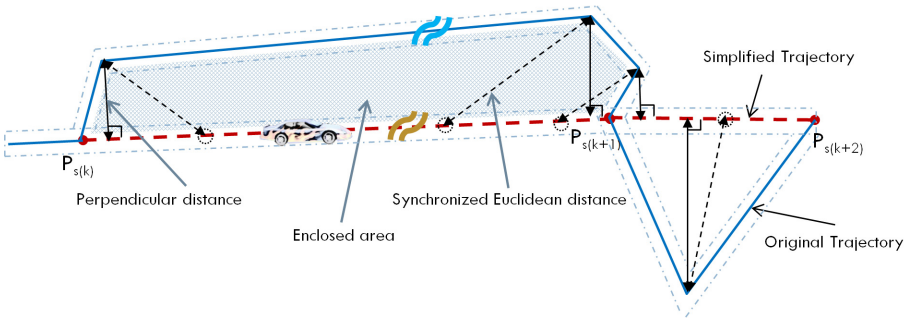

Fig. 4 Error metrics.

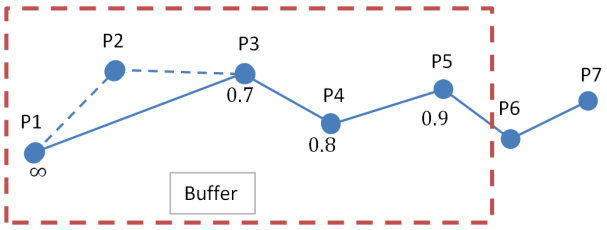

Fig. 3 SQUISH simplification.

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.

In addition, Jonathan Muckell et al. [9] proposed a method called SQUISH to simplify trajectory using a priority queue. As shown in **Fig. 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, P_2 with the lowest weight is moved out from the buffer, but the weight of P_3 cannot be simply obtained by adding the weight of P_2. 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 [19]. 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.

However, all existing methods try to approximate trajectory based on distance measure, including *perpendicular distance* and *synchronized Euclidean distance* (see **Fig. 4**, formal definitions are stated in Section 4). As it can be seen in Fig. 4, in some cases, e.g., between $P_{s(k)}$ and $P_{s(k+1)}$ there are two long parallel lines, even when the distance displacement is slight, the enclosed area displacement will be quite remarkable. Moreover, occasionally *GPS* position is quite inaccurate so that certain points have large displacement, and discrete metric is extremely sensitive to these contingent errors but continuous metric is more resistant to them. It motivates us to develop our method based on enclosed area. In the later section, we will fully prove that enclosed area is a more accurate error metric while considering GPS uncertainty.

3. Proposed Method

There is a wide variety of sensors built in smartphone, and *Android SDK* support tens of sensors including accelerometer, ambient temperature, gravity, gyroscope, light, magnetic field, orientation, pressure, proximity, relative humidity, etc. Aside from them, *GPS*, *Bluetooth* and *Wi-Fi* are also available. By making use of these sensors, we launched a project called *trajectory sensing* to sense the physical environment (including ambient noise, light via microphone and light sensor) and human activities (via accelerometer, orientation sensor). We developed a tool based on *Android* operating system (see **Fig. 5**).

3.1 Problem Statement

Participants take smartphone installing this tool during their commuting routes by walking, bike or car. Finally the data is transmitted to back-end server via cellular network. The ultimate goal of our project is to learn human activities under particular physical environment by sensing and then to discover knowledge. However, the problem we plan to solve in this work is trajectory simplification (i.e., data reduction) in real-time before data analysis.

Trajectory is obtained by recording the successive positions of which a moving object takes across time. Recently, researchers try to enrich trajectory (called *semantic trajectory*) by adding background geographic information to discover meaningful patterns [20]. To extend this concept further, we redefine *semantic trajectory* T as:

$$T = \{p_t = [X(t), Y(t), Z(t), Sm(t)] \mid t \in R\} \quad (1)$$

Here, p_t is a tuple from one data point recorded as time t, in-

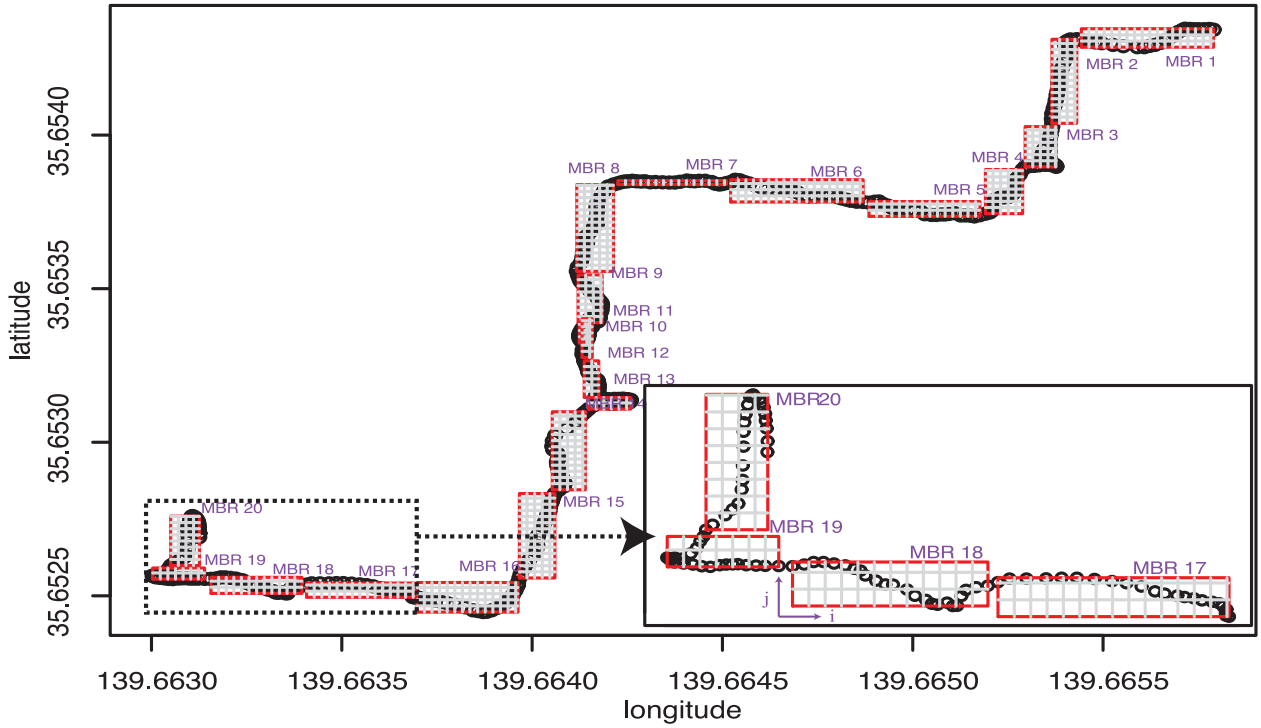

Fig. 6 A sample data of trajectory with MBR.

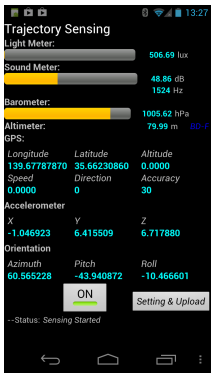

Fig. 5 Trajectory sensing tool.

cluding spatial position $(X(t), Y(t), Z(t))$ and semantic attributes $(Sm(t))$. Spatial position usually refers to latitude, longitude and altitude, and semantic attributes refers to light, noise, temperature and so on. On the other hand, after data reduction a simplified trajectory T' can be defined as:

$$T' = \{p_t = [X(t), Y(t), Z(t), Sm(t))] \mid t \in R \cap \Delta(t) > \varepsilon)\} \quad (2)$$

In fact, we can describe the nature of data reduction problem as constructing a threshold function Δ(t) to achieve the least data loss within favourable or given compression ratio. Our method, just like the existing method is also expected to reach this goal.

After data reduction, the amount of data points decrease but data value and the order are never changed, that is, reduced data meets Eqs. (3) and (4). We call these two properties *value-invariance* and *order-invariance* respectively.

$$\forall p' \in T' \Rightarrow \exists! p \in T : p' = p \quad (3)$$

$$\forall p', q' \in T' \wedge t(p') < t(q') \Leftrightarrow \exists! p, q \in T : (p' = p, q' = q) \wedge t(p) < t(q) \quad (4)$$
where $t(x)$ is the temporal order of x

3.2 Observation

Our Method is derived from such an observation. In **Fig. 6**, there is a sample of GPS trajectory from our trajectory sensing project, and data points are divided into 20 groups with same points - **N**. We can draw minimum bounding rectangle (blue rectangle) called *MBR* on each group. In addition, we divide each MBR into cells of equal size (grey grid) to calculate the information content of MBR. Sum of information content of one MBR is calculated as:

$$\sum_{i=1}^{xn} \sum_{j=1}^{yn} -\log(p_{ij}) \quad (5)$$

where p_{ij} is the frequency of data points within the cell [i,j] (see bottom-right part of Fig. 6).

Finally we calculated the area of each MBR and the sum of information content of each MBR as described in **Table 1**. From Table 1, we can find a strong positive correlation between MBR area and sum of information content. In fact, we calculate the correlation coefficient for them over huge real data and the value is up to about 0.8. Return to our goal of data reduction, it is to achieve the minimal data loss, which means keep information content as same as possible comparing to original data. Therefore, we claim this assumption: **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 8 in Fig.6). Otherwise, we can omit more sampling points (see MBR 7).**

Based on the analysis above, we developed a trajectory sim-

Table 1 Information content and MBR area.

MBR No.	MBR Area	Sum of Information Content
1	208.4	49.2
2	177.8	45.3
3	107.5	27.9
4	140.3	31.4
5	140.3	41.7
6	252.0	53.0
7	36.5	41.7
8	271.9	53.0
9	98.2	28.1

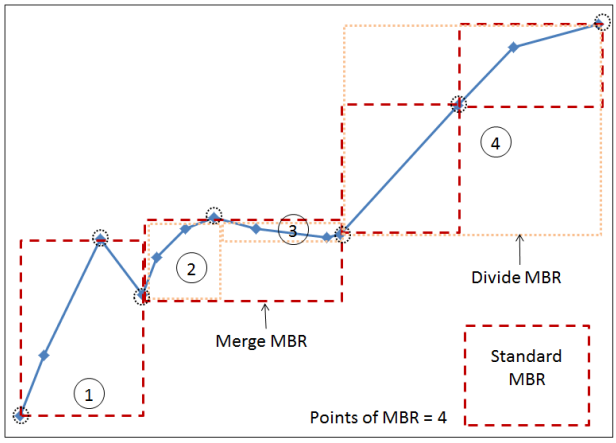

Fig. 7 The illustration of *IC MBR* method.

plification method called *IC MBR* which consists of **divide and merge principle** and **selection strategy** described in detail in Section 3.3 and Section 3.4 respectively.

3.3 Divide and Merge Principle

Based on the assumption stated in Section 3.2, the following principle is employed: we divide the bigger MBRs while merge the smaller MBRs so as to keep the nearly uniform size of MBR. There is an example shown in **Fig. 7**. Initially, 4 MBRs are drawn on it by every 4 points (the number is an input parameter given by user), and then merge MBR 2 and MBR 3 because their area is far less than the standard MBR, while 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 users 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 (e.g., take the moving average 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 yet 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.

Table 2 Points selection strategy.

No.	Condition	Selection Criteria
1	$MBR(N) \subset [St\_MBR * 2, \infty)$	Divide MBR
2	$MBR(N) \subset [St\_MBR, St\_MBR * 2)$	4 points: $x(min)$, $y(min)$, $x(max)$, $y(max)$
3	$MBR(N) \subset [St\_MBR * 0.5, St\_MBR)$	2 points: $x(0)$, $x(N-1)$
4	$MBR(N) \subset [St\_MBR * 0.25, St\_MBR * 0.5)$	1 point: $x(median)$
5	$MBR(N) \subset [0, St\_MBR * 0.25)$	0.5 point: Merge MBR or $x(median)$

Algorithm 1 *IC MBR* method

```
1: function DvideMerge(St_MBR_Pts_Num, St_MBR_Area, Traj)
2:     Num ← St_MBR_Pts_Num
3:     for all point in Traj do
4:         if Num = Buf.Count then
5:             rlt ← SelectPoints(Buf)
6:             if rlt is false then                    ▷ Merge MBR
7:                 Num ← Num * 2
8:             end if
9:         else
10:            Buf.Add(point)
11:        end if
12:    end for
13: end function
14: function SelectPoints(Buf)
15:    Area ← CalcArea(Buf)                  ▷ MBR or Polygon area
16:    Learn(Area, St_MBR_Area)              ▷ Adjust standard MBR area
17:    if Area > St_MBR_Area * 2 then        ▷ divide MBR
18:        SelectPoints(Buf/2)               ▷ first half of Buf
19:        SelectPoints(Buf/2)               ▷ second half of Buf
20:    else if Area < St_MBR_Area/4 then return false
21:    else                                  ▷ by selection strategy
22:        SavePoints()
23:    end if
24: end function
```

3.4 Selection Strategy

Through dividing or merging MBRs, every resulted MBR will contain the comparatively uniform information content. Hence, we take such a strategy to extract points based on such an assumption that points on the boundary of MBR contain more information content and it is advantageous to choose boundary points for the sake of lowering area error. 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 and rule 1 of Table 2).

After integrating the **divide/merge principle** with the **selection strategy**, algorithm of *IC MBR* method can be described as Algorithm 1. This method adapts bottom-up and top-down strategy simultaneously, which recursively approximate line segments within a rectangle. Incidentally, in the 15th line, area is calculated by minimum bounding rectangle or polygon, and in the later section we will discuss the performance of both of them. Besides, our method's time complexity is $O(n/\beta \cdot \beta \log(\beta)) = O(n \log(\beta))$

Table 3 Time complexity comparison.

Method Name	Normal Case	Worst Case
Uniform sampling	n/β	n/β
Dead reckoning	n	$2n$
Douglas-Peucker	$n\log(\beta)$	$n\beta$
IC MBR (Ours)	$n\log(\beta)$	$2n\log(n)$

where β is the point number of buffer. In **Table 3**, we compare our method with conventional methods (adjusted to online process) in terms of time complexity, and in normal case our method is the same as *Douglas-Peucker* method which is the favorable method in many fields.

4. Evaluation Methods

To evaluate the accuracy of each method in terms of perpendicular distance (*PD*), synchronized Euclidean distance (*SED*) and enclosed area (*EA*), we implement *IC MBR* method, *uniform sampling* method, *dead reckoning* method and *Douglas-Peucker* method which is slightly modified to adapt to online processing. 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. Obviously, this additional parameter may affect the initial performance to some extent while improve the time cost.

4.1 Two Conventional Error Metrics

Two conventional error metrics: average perpendicular distance and average synchronized distance are uniformly defined as:

$$m_{1,2}(T^o, T^s) = \frac{1}{N}\sum_{i=1}^{N} d_i^2 \tag{6}$$

Here, N is the total points of original trajectory. *PD* refers to perpendicular distance between the point of original trajectory and the line segment of simplified trajectory (see solid arrows in Fig. 4), which is obtained by the following equation:

$$d_i = \frac{|(x_{s(k+1)} - x_{s(k)})(y_{s(k)} - y_i) - (y_{s(k+1)} - y_{s(k)})(x_{s(k)} - x_i)|}{\sqrt{(x_{s(k+1)} - x_{s(k)})^2 + (y_{s(k+1)} - y_{s(k)})^2}} \tag{7}$$

where $P_{s(k+1)} = (x_{s(k+1)}, y_{s(k+1)}) \in T^s$, $P_{s(k)} = (x_{s(k)}, y_{s(k)}) \in T^s$ and $P_i = (x_i, y_i) \in T^o$.

SED calculates the distance from original point to virtual simplified point which is at identical timestamp (see dot circles in Fig. 4). It considers the temporal attribute of the point sequence, thus it is thought as a better error metric. The virtual simplified point is missing in simplified trajectory, whereas it can be obtained as follows:

$$P_i = P_{s(k)} + \frac{P_{s(k+1)} - P_{s(k)}}{t_{s(k+1)} - t_{s(k)}} t_i \tag{8}$$

4.2 New Error Metric

Although *PD* and *SED* are widely used in the literature, there are two drawbacks as follows:

(1) As stated in Section 2.2, in some cases, the distance-based error is small while the area error is huge. Hence, it will mismatch the real trajectory and simplified trajectory.

(2) In addition, the point of trajectory is not an accurate position since GPS accuracy is uncertain. If one point has an unexceptional change due to GPS uncertainty, distance-based error is subject to this sort of burst error while area metric is robust from the viewpoint of uncertainty tolerance.

Therefore, we introduce a continuous 2-dimensional error metric — *enclosed area*.

4.2.1 GPS Errors Analysis

We will discuss the GPS error sources to justify the necessity of our new error metric. The accuracy of GPS depends on a complicated interaction of various factors.

To analyze the effect of errors on accuracy, a fundamental assumption is usually made that the error sources can be allocated to individual satellite pseudo-ranges. The effective accuracy of the pseudo-range value is termed the user-equivalent range error (UERE). There are the major error sources in GPS which develop error budgets for UERE: satellite clock error, ephemeris error (position of satellites), atmospheric effects (ionospheric and tropospheric delay), receiver noise and resolution, and multipath/shadowing effects [21].

The position error that results from UERE depends on the user/satellites relative geometry, which is called geometric dilution of precision (GDOP). If the satellites viewed from user location is close together (or in a line), the GDOP factor will be higher. Loosely speaking, error in the GPS solution is estimated by the formula [21]:

$$\sigma_p = GDOP \cdot \sigma_{UERE} \tag{9}$$

where σ_p is the standard deviation of the positioning accuracy, σ_{UERE} is the standard deviation of the satellite pseudo-range measurement error and GDOP is the geometry factor of the satellites for a specific location and time of day.

Usually, the error components are considered independent, and the composite UERE for a satellite is approximated as a zero mean Gaussian random variable where its variance is determined as the sum of the variance of each of its components. UERE is usually assumed to be independent and identically distributed from satellite to satellite. However, the position error in a consecutive trajectory is not fully independent. As we see from the error factors described above, GDOP and multipath depend on the specific location, which are the major errors in a short time period while other factors don't fluctuate wildly in a small space-time and be compensated by calibration to some extent. If a user moves in a street, then the GDOP factor and multipath effect caused by blocking by surrounding buildings will be almost constant unless the surrounding is changed abruptly. That is why we assume the GPS error follows Gaussian distribution in later simulation, whereas we claim the case of Fig. 4 is necessary to be solved yet not be properly answered by existing error metrics. In fact, we found there are considerable cases (i.e., trajectory with large area displacement yet small distance displacement) from our collected trajectories. **Figure 8** shows a GPS trajectory in which the blue trajectory is obtained by GPS while the light-yellow one is the real moving trajectory. As the buildings along the road is homogeneous, the GDOP and multipath effect are identical, which resulted in the same distance error. Of course,

Fig. 8 A real GPS trajectory with large area displacement.

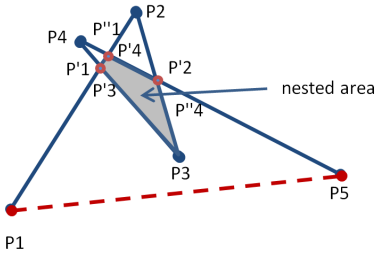

Fig. 9 A self-intersecting polygon.

Algorithm 2 Area Calculation of Arbitrary Polygon

```
1: function CalcAreaOfPolygon(Polygon, PointsNum)
2:     for all point p_i in Polygon do
3:         List.Add(p_i, 0)                          ▷ 0: original point
4:         for j = 0; j < i − 1&&i > 1; j++ do
5:             P ← CrossPoint(p_i, p_{i+1}, p_j, p_{j+1})
6:             if !List.Find(p'_i) then               ▷ after p_i
7:                 List.Insert(p'_i ← P, 1)           ▷ 1: cross point
8:             end if
9:             if !List.Find(p'_j) then               ▷ after p_j
10:                List.Insert(p'_j ← P, 1)           ▷ 1: cross point
11:            end if
12:        end for
13:    end for
14:    anchor ← 0
15:    for i := 0 → List.Count do
16:        if List[i].flag == 1 then
17:            CalcAreaOfSimplePolygon(List.Range(anchor, i))
18:            anchor ← i
19:        end if
20:    end for
21:    CalcAreaOfSimplePolygon(List.Rage(anchor, i))
22: end function
```

this sort of case is common in urban area while the situation is different in other area.

4.2.2 Enclosed Area

Enclosed area which is a polygon confined by original trajectory and simplified trajectory (the dashed area of Fig. 4). Although EA is obviously advantageous, its calculation is quite troublesome [22], [23] provided that the polygon contains self-intersection (see Fig. 8). Nevertheless, we devised such an algorithm to solve this problem (see Algorithm 2):

(1) The intersection point is obtained by geometry formula.

(2) If exist cross point, it will 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. Take the **Fig. 9** as an example, finally the list in Algorithm 2 will be $(P1, P1', P2, P'2, P3, P'3, P4, P'4, P5, P1)$.

(3) The sub sequence of the list split by intersection point is a line segment or a simple polygon whose area can be easily obtained as follows - by *outer product of vector*:

$$Q_j = \frac{1}{2} \mid \sum_{i=1}^{K} \overrightarrow{p_i} \times \overrightarrow{p_{i-1}} \mid \tag{10}$$

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

$$m_3(T^o, T^s) = \frac{1}{N} \sum_{i=1}^{J} area(Q_j) \tag{11}$$

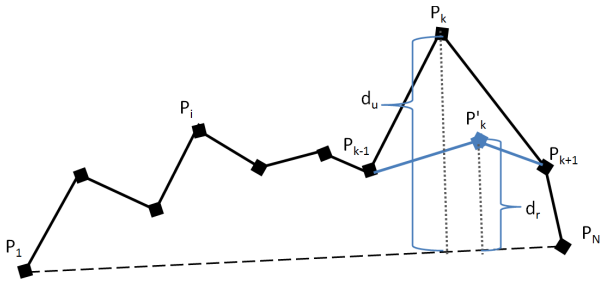

Fig. 10 Error metric under GPS uncertainty.

4.2.3 Uncertainty Tolerance

GPS system has intrinsic error of approximately from several meters to tens of meters. For a specific location, the inaccuracy is not constant and it can be seen as normal distribution, which means some particular points reach large displacement while most points are within slight error range (but in commercial GPS system these slight errors may be eliminated by smoothing techniques). Note that the errors in a series of consecutive locations within a short time are dependent to a large extent as analysed in Section 4.2.1. We want to explore the different effect between distance-based metric and area-based metric while taking the GPS uncertainty into account.

As shown in **Fig. 10**, $P_1 \cdots P_i \cdots P_N$ is the GPS points of original trajectory, and the dot line between P_1 and P_N is the simplified trajectory. In addition, we assume P_k has a large error whose corresponding real position is P'_k, while other points have only slight error. We define uncertainty tolerance which indicates how significant the uncertain (wrong) position affects the simplification process. For distance-based simplification, the uncertainty tolerance is $T_d = \frac{|d_u - d_r|}{d_r}$, where d_u is the distance from

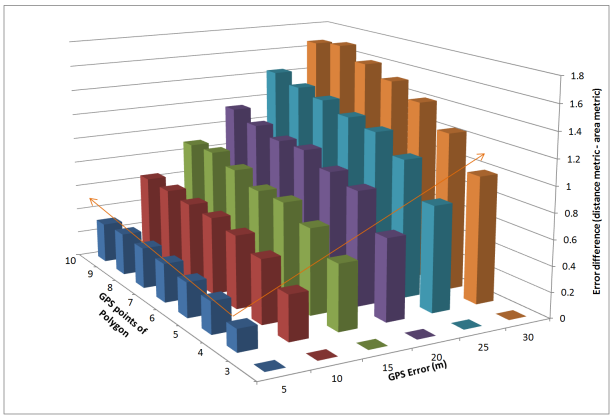

Fig. 11 Difference of uncertainty tolerance.

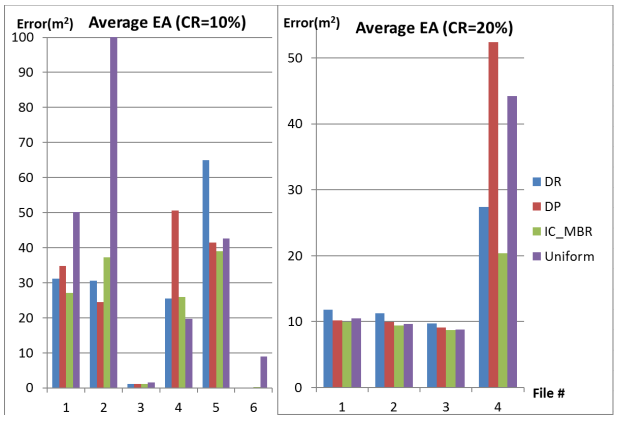

Fig. 12 Average *EA* with 10% and 20% compression ratio.

GPS point to simplified trajectory, and d_r is the distance from real position to simplified trajectory. On the other hand, the uncertainty tolerance of area-based simplification is $T_Q = \frac{|Q_u - Q_r|}{Q_r}$, where Q_u is the area enclosed by GPS points, and Q_r is the area enclosed by points of real position. In Fig. 10, Q_u is the area of polygon $(P_1, \cdots, P_i, \cdots, P_{k-1}, P_k, P_{k+1}, P_N)$, and Q_r is the area of polygon $(P_1, \cdots, P_i, \cdots, P_{k-1}, P'_k, P_{k+1}, P_N)$. For simplifying calculation, here we assume GPS points with slight errors as real position while distinguish the point (i.e., P_k) with the largest error from real position (i.e., P'_k). Note that, the higher the value of uncertainty tolerance, the weaker it is against uncertainty. In Fig. 10, suppose there are only 3 points (P_1, P_k, P_N), then $Q_u = \frac{1}{2} d_u |P_1 P_N|$ and $Q_r = \frac{1}{2} d_r |P_1 P_N|$. The uncertainty tolerance T_Q is $\frac{|Q_u - Q_r|}{Q_r} = \frac{|d_u - d_r|}{d_r} = T_d$, which indicates distance-based metric performance as same as area-based metric in the case of 3 points.

We conduct the experiment by simulating 10,000 times for each pattern (combination of GPS points and GPS error). **Figure 11** shows the result. The vertical axis (Z) is the value of $(T_d - T_Q)$. We can find in the case of 3 points the difference is zero which is consistent with our theory. In generally, distance-based metric performs worse in terms of uncertainty tolerance, and as the GPS error increases, the performances of distance-based metric deteriorates. The same trend can be seen in terms of GPS points of polygon.

5. Experimental Results and Discussion

In this work a series of experiments are conducted by using the data collected through our *trajectory sensing* project. Aside from it, we also make use of Microsoft GeoLife [24], [25] dataset that consists of 178 users in a period of over four years (from April 2007 to October 2011). Various transportation modes are included in the data set, including walking, driving, train travel and etc. 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.

5.1 Performance Comparison

Trajectories are compressed with 3%, 10%, 20% and 50% and are measured by 3 error metrics (namely *EA*, *PD* and *SED*). Here, compression ratio (*CR*) is defined as the number of original points divided by the number of compressed points. From **Fig. 12** (the results of compression ratio 10% and 20% are shown while other compression ratio results are skipped, but a complete comparison is described later), we can find that (1) accuracy gets worse as compression ratio decreases; (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 will earn better accuracy but high compression ratio.

Figure 13 shows the normalized error (the value is scaled to [0,1]) of *EA*, *PD* and *SED* in different compression ratio. As a result, our method holds an absolute advantage in terms of *EA* metric and competitive performance in *SED* metric but poor performance in *PD*. The reason is that we measure displacement directly by enclosed area, and there is an inherent relation between area and distance. Our method filters point based on information contents that is measured by area in 2-dimensional plane, which resulted in a good performance in *EA*.

5.2 Parameters Analysis

In our method there are two important user-specified parameters which are the points number of standard MBR and the area of standard MBR. Besides the area of standard MBR is also subject to the adaptive MBR or fixed MBR. Since the performance is significantly affected by these factors, we will explore the relationship between them and *EA* accuracy.

In the left part of **Fig. 14**, it employs the moving average to adjust standard MBR area. We can find: as the points of standard MBR increases, the *EA* error increases and the compression ratio decreases simultaneously. In fact, there is a positive linear relationship between the points of standard MBR and *EA* error but a negative linear relationship between the points of standard MBR and compression ratio. (Note that although the figure shows a power relationship, actually it is linear relationship since the horizontal axis is also power-scaled.) Hence, we need a trade-off between compression ratio and *EA* error, and in most cases the

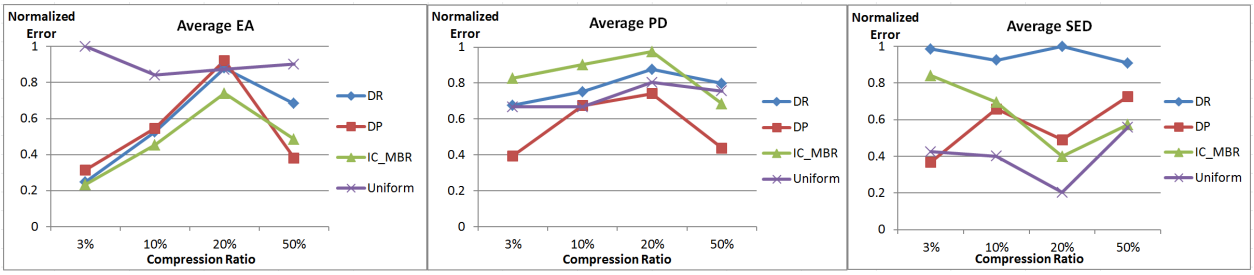

Fig. 13 Normalized error of *EA, PD, SED*.

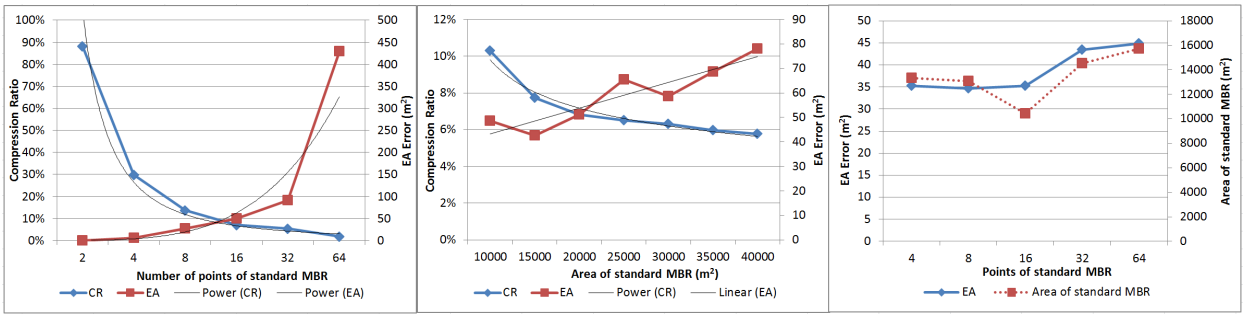

Fig. 14 Parameters effect over accuracy.

number of points of standard MBR ranging from 8 to 20 is advisable.

In the middle part of Fig. 14, we fix the points of standard MBR as 16 but adjust standard MBR area manually. It shows that the compression ratio slightly decreases as the area of standard MBR increases. Besides, the *EA* fluctuates along with area of standard MBR. However, in a real-time mode it is hard to determine the optimal area of standard MBR to get the smallest error.

In the right part of Fig. 14, we observe the change of *EA* with parameters in adaptive adjustment mode when the user specifies the compression ratio. From the figure (here *CR* is fixed at 10%), we find that along with the increase of points of standard MBR, the *EA* error increases. At the same time, the area of standard MBR increases too.

Our method is based on area metric, whereas we calculate not the area of polygon (the actual enclosed area) but the area of MBR instead during divide/merge MBR. The reason is that there is a strong correlation between polygon area and its corresponding MBR area. In addition, to calculate polygon area is much more computation-intensive than to calculate MBR area. In fact, the time complexity of our method with polygon calculation is: $O(n/\beta * \sum_{j=1}^{\beta} \frac{j(j+1)}{2}) = O(n/\beta * \frac{1}{2}(\frac{\beta(\beta+1)}{2} + \frac{\beta(\beta+1)(2\beta+1)}{6})) = O(n * \beta^2)$ where β is the point number of buffer. Comparing to the method with MBR calculation ($O(nlog\beta)$), the scale of time complexity is different. We actually compared these two ways, and find that by calculating polygon area the accuracy can be improved by **2.7 times** while the computation time may increase by **34.3 times**. Due to the requirement of real-time simplification, we choose the simple way — MBR calculation — but sacrifice the accuracy.

In addition, we calculated the correlation of MBR area and information content of MBR, to explore its relation with the *EA* accuracy. As a result, there is no strong correlation between them, namely the strong correlation of MBR area and information content of MBR does not necessarily indicate good accuracy. Note that this result does not go against our method's efficiency since our method is finally evaluated by error metrics as above.

6. Conclusion and Future Work

In this paper, we proposed a novel scheme: **divide/merge principle** and **selection strategy** to reduce data for spatial trajectory. To measure displacement correctly, we newly introduced enclosed area metric which is proven more robust against GPS uncertainty. 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 back-end server. On the other hand, the transformed online DP method needs a big enough buffer to guarantee the accuracy and compression ratio.

However, our method as well as existing methods merely takes geometric feature (linear or areal displacement) into account, which may lead to the loss of other information (e.g., speed). Besides, in the second component of our method (i.e., selection strategy), we just intuitively save boundary points for achieving the least areal displacement. Consequently, this selection can't exhibit the optimal simplification and it is sensitive to the shape of trajectory.

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 (points of standard MBR and area of standard MBR) and better selection strategy.

References

[1] Kupper, A.: *Location-based services: Fundamentals and Operation*, John Wiley & Sons Ltd (2005).
[2] Lane, N.D., Miluzzo, E., Lu, H., et al.: A survey of mobile phone sensing, *IEEE Communications Magazine*, Vol.48, pp.140–150 (2010).
[3] Das, T., Mohan, P. and Padmanabhan, V.N.: PRISM: Platform for remote sensing using smartphones, *MobiSys 2010*, pp.63–70 (2010).
[4] Lawson, C.T., Ravi, S.S. and Hwang, J.-H.: Compression and Mining of GPS Trace Data: New Techniques and Applications, Technical report, State University of New York at Albany (2011).
[5] Jones, M.P.: Satellite Communications Systems Buyer's Guide, *British Antarctic Survey* (2008).
[6] Yanagisawa, Y., Akahani, J. and Satoh, T.: Shape-Based Similarity Query for Trajectory of Mobile Objects, *MDM2003*, pp.63–71 (2003).
[7] Wirz, M., Schläpfer, P., Kjærgaard, M., et al.: Towards an online detection of pedestrian flocks in urban canyons by smoothed spatio-temporal clustering of GPS trajectories, *Proc. 3rd ACM SIGSPATIAL International Workshop on LBSN*, pp.17–24 (2011).
[8] Potamias, M., Patroumpas, K. and Sellis, T.: Sampling trajectory streams with spatio- temporal criteria, *SSDBM 2006*, pp.275–284 (2006).
[9] Muckell, J., Patil, V. and Ping, F.: SQUISH: An Online Approach for GPS Trajectory Compression, *Com. Geo 2011* (2011).
[10] Gudmundsson, J., Katajainen, J., Merrick, D., et al.: Compressing spatio-temporal trajectories, *ISAAC 2007*, pp.763–795 (2007).
[11] Taylor, G.: LINE SIMPLIFICATION ALGORITHMS, available from ⟨http://www.comp.glam.ac.uk/pages/staff/getaylor/papers/lcwin.pdf⟩ (2005).
[12] Cao, H., Wolfson, O. and Trajcevski, G.: Spatio-temporal Data Reduction with Deterministic Error Bounds, *VLDB Journal*, Vol.15, No.3, pp.211–228 (2006).
[13] Lange, R., Dürr, F. and Rothermel, K.: Online Trajectory Data Reduction using Connection-preserving Dead Reckoning, *Mobiquitous 2008* (2008).
[14] Douglas, D. and Peucker, T.: 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).
[15] Jenks, G.: Geographic logic in line generalisation, *Cartographica*, Vol.26, pp.27–42 (1989).
[16] Wu, S.-T. and Márquez, M.G.: A non-self-intersection Douglas-Peucker algorithm, *Computer Graphics and Image Processing 2003*, pp.60–66 (2003).
[17] Hershberger, J. and Snoeyink, J.: Speeding Up the Douglas-Peucker Line-Simplification Algorithm, *Proc. 5th Intl. Symp. Spatial Data Handling*, pp.134–143 (1992).
[18] Chen, Y., Jiang, K., Zheng, Y., et al.: Trajectory Simplification Method for Location-Based Social Networking Services, *LBSN 2009*, pp.33–41 (2009).
[19] Baraniuk and Richard: Compressive Sensing, *IEEE Signal Processing Magazine*, Vol.24, No.4, pp.118–121 (2007).
[20] Ying, J., Lee, W., Weng, T., et al.: Semantic Trajectory Mining for Location Prediction, *ACM GIS2011*, pp.34–43 (2011).
[21] Kaplan, E.D. and Hegarty, C.J.: *Understanding GPS: principles and applications*, Artech House Inc. (2006).
[22] Nievergelt, J. and Preparata, F.P.: Plane-sweep algorithms for intersecting geometric figures, *Comm. ACM*, Vol.25, pp.739–747 (1982).
[23] Parka, S.C. and Shinb, H.: Polygonal chain intersection, *Computers & Graphics*, Vol.26, pp.341–350 (2006).
[24] Zheng, Y., Li, Q., Chen, Y., et al.: Understanding mobility based on gps data, *UbiComp 2008*, pp.312–321 (2008).
[25] Zheng, Y., Zhang, L., Xie, X., et al.: Mining interesting locations and travel sequences from gps trajectories, *WWW 2009*, pp.791–800 (2009).

Guangwen Liu is a Ph.D. candidate in the Graduate School of Frontier Sciences, the University of Tokyo. He received his bachelor's degree in software engineering from Northeastern University, China in 2005, and his master's degree from Beihang University, China in 2008 respectively. He has accumulated vast experience in IT industry as system engineer since 2008. His research interests include ubiquitous computing, mobile sensing, location based services and data mining. He is a student member of ACM.

Masayuki Iwai received his Ph.D. from Keio University in 2004. He was a lecturer at the Graduate School of Media and Governance, Keio University until 2008. From 2008, he joined JST/CREST project at Tokyo Denki University. From 2009–2012, he has been an Assistant Professor at Institute of Industrial Science, the University of Tokyo. He works as an associate professor of Tokyo Denki Univeristy from 2013. His interesting research fields are Trajectory Sensing, SNS Context Analysing, Sensor Data Mining, Wireless Sensor Networks, Distributed/Mobile Computing, Ubiquitous Applications, Human Probe System, Media Art. He is a member of IPSJ, IEICE and IEEE.

Kaoru Sezaki received his B.Eng., M.Eng., and Ph.D. degrees from the University of Tokyo, Tokyo, Japan, in 1983, 1985, and 1988, respectively, all in Electrical. Since 1988, he has been with Institute of Industrial Science, the University of Tokyo and currently is a Professor. He was also co-appointed as associate professor at Center for Spatial Information and Science since 1999 to 2003. He was a Visiting Researcher at University of California at San Diego in 1996. His research interests include ad-hoc networks, sensor networks especially on body sensor networks, application of RFIDs and urban computing. He is a member of IEEE and served as Treasurer of IEEE Tokyo Section as well as that of Japan Council from 2003 to 2004.

(Editor in Charge: *Tomoki Yoshihisa*)