A Privacy-Preserving Prediction Method for Human Travel Routes

Wen-Chen HU
Department of Computer Science, University of North Dakota
Grand Forks, ND 58202-9015, USA

Naima KAABOUCH
Department of Electrical Engineering, University of North Dakota
Grand Forks, ND 58202-7165, USA

and

Hung-Jen YANG
Department of Industrial Technology Education, National Kaohsiung Normal University
Kaohsiung City 80201, Taiwan

ABSTRACT

This paper proposes a kind of location-based research, human travel route prediction, which is to predict the track of a subject's future movements. The proposed method works as follows. The mobile user sends his/her current route along with several dummy routes to the server by using a 3D route matrix, which encodes a set of routes. The server restores the routes from the 3D matrix and matches the restored routes to the saved routes. The predicted route is found as the trunk of the tree, which is built by superimposing the matching results. The server then sends the predicted routes back to the user, who will apply the predicted route to a real-world problem such as traffic control and planning. Preliminary experimental results show the proposed method successfully predicts human travel routes based on current and previous routes. User privacy is also rigorously protected by using a simple method of dummy routes.

Keywords: Location-Based Services, Human Travel Routes, Route Prediction, Matrices and Privacy.

1. INTRODUCTION

A human travel route is a sequence of locations (including latitudes and longitudes) taken by a human via walking or driving. Furthermore, human travel route prediction is to predict the forthcoming locations of the human travel routes, which usually follow certain patterns; e.g., a route leads to a landmark such as a park or mall, follows a highway or street, or brings to an event like a ball game or concert. Human travel route prediction is very useful and can be applied to many areas such as location-based advertisements, traffic planning and control, and travel recommendations.

The proposed research predicts the routes based on the current and previous routes. Its accuracy is high because of the consistence of user traveling patterns. In addition, a route is a list of locations. The proposed research is made easy by using a matrix representation, which facilitates the route storage, indexing, transmission, and matching. Route processing then becomes matrix processing, which is simpler and faster than list-of-location processing. Additionally, instead of sending route matrices between the clients and server, this research uses a 3D route matrix, which stores a set of encoded routes, for transmission, so processing time is shortened. User privacy is usually a big headache for location-based services. It becomes simple and effective by using our method. This research sends the user's current route along with several dummy routes to the server, so the server is not able to tell the correct user route among the routes it receives. The server then tries to extend the routes based on the current and previous routes. The predicted routes, including the target predicted route, are sent back to the user for the intended application. Preliminary experiment results show the proposed methods are effective and user-privacy is rigorously preserved.

The rest of this paper is organized as follows. Section 2 gives the background information of this research including two themes: (i) route prediction, and (ii) privacy protection of routes. Section 3 introduces the proposed system. The data structure, route matrices, used in this project is explained in Section 4. Section 5 gives algorithms of the proposed methods. The last section summarizes this research and gives few future research directions.

2. BACKGROUND AND LITERATURE REVIEW

A location-based service is a service based on the geographical position of a mobile handheld device [1]. Two of the LBS examples are finding a nearby ethnic restaurants and locating a nearby store with the best price of a product. Popular LBS include mapping and navigation, search and information, social networking, entertainment, and tracking [2]. A nice introduction of LBS technologies and standards is given in the articles [3][4]. Two themes related to this research are given as follows:

1) *Route prediction*: Human travel route prediction is useful and has many potentials, but at the same time, accurate, efficient, and simple prediction has yet to be found. Common prediction usually uses either probability-based or learning-based approach. The former approach predicts routes based on probabilistic information and the latter characterizes route features and adopts machine-learning algorithms for route prediction. Related approach can be found in the articles [5][6][7][8].
2) *Privacy preservation of route computation*: Various methods of user privacy preservation of route computation have been proposed [9], but most of them do not meet our need,

consistent user identities. There are two methods, spatial cloaking and dummy locations/routes, has the feature of consistent user identities. This former approach is to blur a user's exact location into a cloaked area, so there is low possibility of associating users to locations [10][11]. On the other hand, the latter approach, users send their true location data along with several dummy/false location data to the service providers [12][13].

3. STRUCTURE OF THE PROPOSED SYSTEM

One of the location-based services, route prediction, receives a great attention these days because of its usefulness and popularity. For example, route prediction can be used to send location-based advertisements/announcements or find better network routes. However, it also suffers the privacy problem of most location-based services. This research proposes a privacy-preserving route prediction, so users can use this function without concerns. This section explains the proposed system by giving (i) the structure of the proposed system, (ii) the route matrices for a route representation, and (iii) the method of privacy-preserving route prediction.

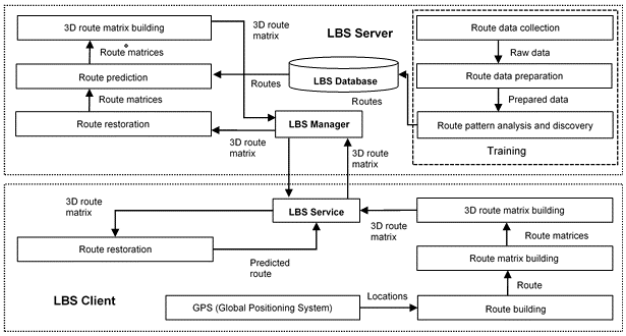

Figure 1. The system structure of the proposed system.

This research predicts routes based on current and previous routes. A 2D route matrix is used to represent a route, a sequence of locations. A mobile user sends his/her current route along with several dummy routes to the LBS server by using a 3D route matrix, which is built by superimposing 2D route matrices. Routes are then predicted by the server, and are sent back to the user for an LBS application. Figure 1 shows the system structure of the proposed system consisting of two parts: server-side and client-side subsystems. It also includes two phases: training and testing, where the training phase is used to fill up the LBS Database with route data and the testing phase is used by the application, predicting routes. The server performs most tasks of this method including collecting and saving sets of routes and the later predicting routes, which will be detailed in the next section. It includes the following components, of which the first three components are used during the training phase:

- *Route data collection*: The first step is to collect route data. This step can be done before the app is put to use or it continues collecting and updating the routes while the app is used.
- *Route data preparation*: Not all collected routes are usable for this research and the routes may need to be processed before being put to use [14]. In addition, the GPS data is usually not stable or consistent. This step may include noise removal, deletion of rarely-used routes, or route completion.
- *Route pattern analysis and discovery*: Before the routes are entered to the database, the system may try to find useful route patterns such as the most popular routes or likely destinations of the routes.
- *LBS Database:* The routes, lists of locations, are then saved in the LBS Database including two tables: `Routes` and `Locations`, whose schema are given in Figure 2.

Routes

TID	Start	End
T1		...
...	...	...
...	...	
...	...	...

Locations

LID	Time	Location	No.
L1	Nov 30 16:13:24 CST 2013	47° 55' 31" N, 97° 01' 57" W	1
...	...	...	...
L42	Dec 03 06:10:31 CST 2013	47° 55' 37" N, 97° 01' 63" W	38
...	...	...	...
...	...	...	...

Figure 2. The schema of the two tables `Routes` and `Locations` in the LBS Database.

- *LBS Manager:* It receives routes from users via 3D route matrices, which will be discussed in the next section. The LBS Manager sends the result routes back to the users after prediction discussed in the Section 5 Route Prediction.

On the other hand, the client-side subsystem is made simple on purpose. It includes two major tasks: (i) sending and receiving routes to and from the server, and (ii) applying the predicted route to a location-based services. By using this approach, user privacy is preserved at the same time because the target routes are sent along with several dummy routes, so it is not possible to associate the users to specific routes.

4. ROUTE MATRICES

Routes, lists of locations, are the major data to be processed in this research. Route/list management is normally not easy and complicated. Some of route indexing and retrieval methods can be found from the article [15]. This research adapts the data structure, matrices, for the route representation. The representation, called the route matrices, facilitates the route storage, indexing, transmission, and processing. This section will introduce the route matrices.

Map Tile System

A map is usually rendered from a number of map tiles for convenience and speed. A map tile system includes many tiles of different zoom levels. By using map tiles, a map can be panned and zoomed easily and quickly. For panning, some of the map tiles instead of the whole map are replaced. For zooming, different zoom-level tiles can be retrieved more efficiently because lengthy tile searches can be avoided if map tiles are used. A map is usually rendered from a number of map tiles for convenience and speed. A map tile system includes many tiles of different zoom levels. By using map tiles, a map can be panned and zoomed easily and quickly. For panning, some of the map tiles instead of the whole map are replaced. For zooming, different zoom-level tiles can be retrieved more efficiently because lengthy tile searches can be avoided if map tiles are used. Several methods are used to find and display the correct maps.

For example, Figure 3 shows a map using pixel coordinates [16]. The pixel at the upper-left corner of the map has pixel coordinates (0,0) and the pixel at the lower-right corner of the map has pixel coordinates (2047,2047). One of the advantages of pixel coordinates is that the specific locations can be pinpointed quickly. At the same time, using an image to represent a world map of a specific zoom level is not realistic because the image size would be huge. Therefore, a map usually consists of a number of map tiles. For example, Figure 3 shows a world map consisting of 64 map tiles from $(0,0)_t$ to $(7,7)_t$ and each tile contains $\left(\frac{2048}{8}\right) \times \left(\frac{2048}{8}\right) = 256 \times 256$ pixels. Given a pair of pixel $(x, y)_p$ coordinates, the corresponding tile coordinates $(a, b)_t$ containing that pixel is given by Equation 1:

$$a = \left\lfloor \frac{x}{256} \right\rfloor \quad and \quad b = \left\lfloor \frac{y}{256} \right\rfloor \tag{1}$$

On the other hand, most locations use latitude and longitude. Examples of how to convert latitude and longitude to tile numbers can be found from the CloudMade web site [17].

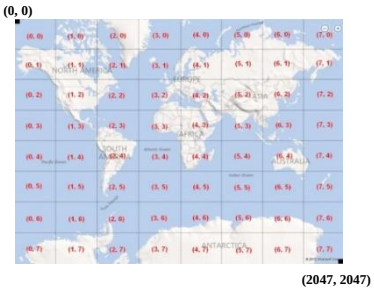

Figure 3. A world map consists of 64 map tiles.

Traditional map panning and zooming is complicated and slow because they need an algorithm to find the correct map locations. Map panning and zooming are made easy by map tiles. Figure 4 shows an example of map tiles of three zoom levels. For panning and zooming, the following methods are used:

- *Zooming*: Zooming in the tile x will result in four tiles $x0$, $x1$, $x2$, and $x3$ and zooming out will do the reverse. For example, zooming in the tile 32 in Figure 4 will bring in the tiles 320, 321, 322, and 323. On the other hand, zooming out the tiles 320, 321, 322, and 323 will result in the tile 32.
- *Panning*: There is no need to replace the whole maps; instead, only certain tiles are replaced. For example, panning the map including the tiles 320, 321, 322, 323 right may drop the tiles 320 and 322 and bring the tiles 330 and 332 in.

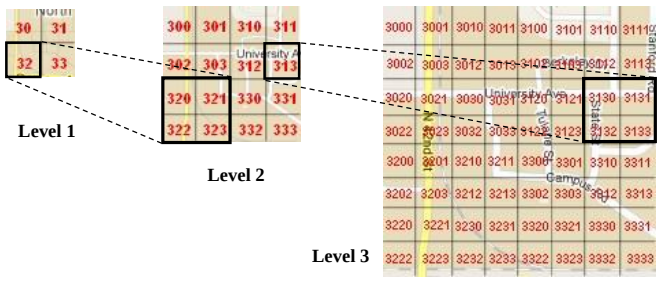

Figure 4. Map tiles of three zoom levels.

Route Matrices

Today's map services could offer great details of the Earth. For example, the Google has a tile of 256×256 pixels. For the highest zoom level 24, each pixel size at equator is 9.3mm, so the tile covers an area of $256 \times 9.33mm \times 256 \times 9.33mm = 2.39 \times 2.39m^2$, which is detailed enough for our purpose.

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \dots & \dots & \dots & \dots \\ a_{m,1} & a_{m,2} & \dots & a_{m,n} \end{bmatrix} \tag{2}$$

A route is a list of locations and lists are normally not easy to manage. This research uses a matrix of map tiles to represent a route. Matrices are a well-known mathematical subject and plenty of algorithms and methods of matrix processing are available. The notation of an $m \times n$ matrix A is given as above. The $(i, j)^{th}$ entry of the matrix A is written as $a_{i,j}$. Figure 5 shows a typical route and its corresponding map tiles and 10×10 route matrix.

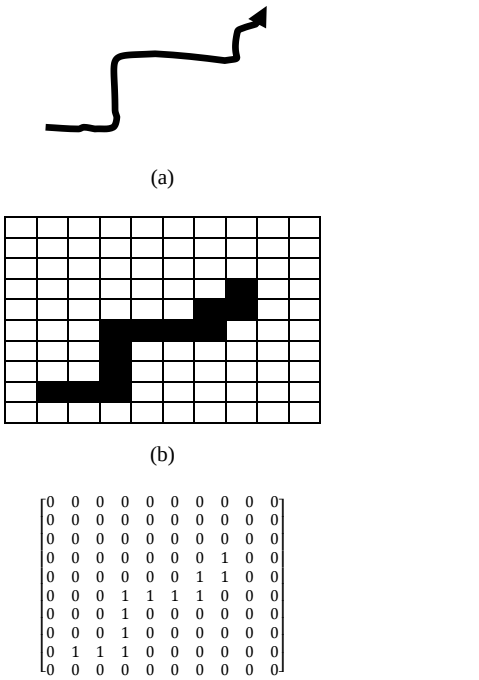

Figure 5. An example of route matrix representation: (a) a route, (b) the corresponding map tiles, and (c) the corresponding 10×10 route matrix.

If the route passes through a map tile, the tile is colored black as shown in Figure 5.b. Figure 5.c is a sample route matrix, whose entry values are either 1 (on) if the corresponding map tile is black or 0 (off) if the corresponding tile is blank. However, the values do not have to be 1 or 0. They could be any values/weights. Equation 3 shows a generic matrix notation for the route p:

$$T_p = \begin{bmatrix} t_{1,1} & t_{1,2} & \dots & t_{1,n} \\ t_{2,1} & t_{2,2} & \dots & t_{2,n} \\ \dots & \dots & \dots & \dots \\ t_{m,1} & t_{m,2} & \dots & t_{m,n} \end{bmatrix}, \qquad where\ t_{i,j} = w_{i,j,p} \tag{3}$$

where $w_{i,j,p}$ is the weight of the route p on the map tile coordinated at $[f_x(i,l), f_y(j,l)]$. The functions $f_x(i,l)$ and $f_y(i,l)$ gives the x and y coordinates of the map tile at the entry $t_{i,j}$, respectively. The argument l is the physical location of the entry $t_{1,1}$ and is used to find the corresponding map tile at $[f_x(1,l), f_y(1,l)]$. For example, the map tile coordinates are $[f_x(1,l)+i-1, f_y(1,l)+j-1]$ for the entry $t_{i,j}$. Most routes can be represented by sparse matrices and many matrix optimization methods and representations can be found in the literature [18].

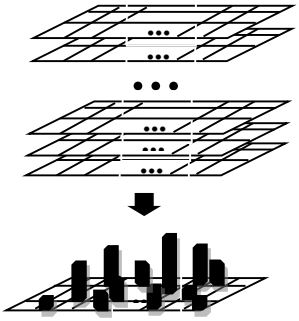

Figure 6. An example of a 3D route matrix including a set of routes and the corresponding 3D route matrix.

3D Route Matrices

A set of routes are collected and saved in the LBS Database. A route is a list of locations and managing lists is always a difficult task. This research requires transmissions of routes between the users and server. A 3D matrix representation is therefore proposed to facilitate the route storage, transmission, and processing. Figure 6 shows an example of 3D route matrix storing a set of routes. A generic 3D route matrix notation is given as follows:

$$D = \begin{bmatrix} d_{1,1} & d_{1,2} & \dots & d_{1,n} \\ d_{2,1} & d_{2,2} & \dots & d_{2,n} \\ \dots & \dots & \dots & \dots \\ d_{m,1} & d_{m,2} & \dots & d_{m,n} \end{bmatrix}, \quad (4)$$

$$where \;\; d_{i,j} = [d_{i,j,1}, d_{i,j,2}, \dots, d_{i,j,h}] \;\; and \;\; d_{i,j,p} = w_{i,j,p}$$

where each entry $d_{i,j}$ is an array of weights of the routes, h is the number of routes, and $w_{i,j,p}$ is the weight of the route p on the map tile coordinated at $[f_x(i,l), f_y(j,l)]$. Again, each entry $d_{i,j}$ is likely a sparse one-dimensional matrix. Other data structures, such as linked lists, can be used to store the weights and thus save the space. Each 3D matrix encodes many routes and several 3D matrices are sent, so the server is not able to associate routes to the user and privacy is preserved.

Route Matching

Route matching is difficult because it usually requires location-by-location matching. The route matching is made easy by using our route matrices. Various definitions of similarity are defined. This research uses the following formula to measure the similarity between two matrices. For example, $R = C\Delta S$ is the result $m \times n$ matrix R of matching the current $m \times n$ route matrix C to the stored $m \times n$ route matrix S:

$$r_{i,j} = c_{i,j}\Delta s_{i,j}$$

$$= \begin{cases} \frac{|C \cap S|}{|C|}, & if\ (c_{i,j} = 0\ and\ s_{i,j} \neq 0)\ or\ (i = i_e\ and\ j = j_e) \\ 0, & otherwise \end{cases} \quad (5)$$

where $i \leq m$, $j \leq n$, $|C|$ is the number of entries of the matrix C being on, $|C \cap S|$ is the number of both corresponding entries of C and S being on, and (i_e, j_e) is the entry for the map tile of the current location (or the final location of the current route). Figure 7 gives an example of the route matching, where $|C| = 9$ and $|C \cap S| = 6$.

5. ROUTE PREDICTION

This research proposes a method of privacy-preserving route prediction based on current and previous routes because a generalization of the past user travel routes usually indicates a route trend. A discussion of the proposed algorithm is given in this section.

Privacy-Preserving Route Prediction

1. *(Client) Send a 3D route matrix encoding the current route along with several dummy routes to the LBS server.*
2. *(Server) Restore routes from the 3D route matrices.*
3. *(Server) For each of the restored routes, perform the Steps 4-6.*
4. *(Server) Match the restored route to the stored routes to generate a weighted matrix.*
5. *(Server) Superimpose the matching result matrices.*
6. *(Server) Find the trunk of the tree, the predicted route, in the superimposed matrix.*
7. *(Server) Send a 3D route matrix encoding the predicted routes to the client.*
8. *(Client) Restore the routes from the 3D matrix.*
9. *(Client) Start using the predicted route.*

Figure 7. An algorithm of the proposed route prediction.

Proposed Method

An algorithm of the proposed method is given in Figure 7. The LBS Service records the current route and sends it along with several dummy routes to the server by using a 3D route matrix. The server then initiates the following steps: (i) restoring the routes from the 3D matrix, (ii) matching the restored route to the saved routes, (iii) superimposing the result matrices after the matching, (iv) the superimposed matrix containing a weighted tree and the trunk of the tree being treated as the predicted route, and (v) the predicted routes sent back to the user for a location-based application. User privacy may be violated at two places in this method: (i) user sending his/her route to the server and (ii) the server sending the predicted route to the user. User privacy is preserved by using this simple approach because the server is not able to tell the user route from the dummy routes submitted together. On the other hand, the target predicted route is retrieved from several predicted routes on the client side, so it is not possible to associate routes to the user other than the user himself/herself.

Algorithm of the Route Prediction

The proposed method builds a tree (rooted at the end location of the current route) from the previous routes. It then takes the trunk of the tree as the predicted route. A path is a sequence of vertices connected by edges and a simple path is a path without repeated

vertices. For the tree $T = (V, E)$, a trunk is a simple path $P = (V', E')$ minimizing $\delta(P) + w(P)$ where

$$\delta(P) = \sum_{v \in V} d(v, P) = \sum_{v \in V} \min_{u \in V'} d(u, v), \quad (6)$$

$$w(P) = \sum_{e \in E'} w(e), \quad (7)$$

V is a set of vertices, E is a set of edges, $V' \subseteq V$, $E' \subseteq E$, $w(e)$ is the weight of the edge e, and $d(u, v)$ is the distance between the nodes u and v [19][20]. Since the matrix does not include edges, this research uses $d(e) = w_v$, the weight of the end node or tile v. A predicted route is usually a path among a set of paths that users like to take or is the most popular one and the trunk is the core of a tree and has a minimal weight of total edges. It is the main reason that the trunk of the tree (superimposed routes) is treated as the predicted route. Experiments or proofs may be needed to prove that this approach correctly predicts the route we want.

6. CONCLUSIONS

This research proposes a location-based service, human travel route prediction, which tries to predict the forthcoming locations of the current travel routes. Two of the many contributions made by this project are

- Travel routes, lists of locations, are normally difficult to manage. Routes in this research are represented by an innovative matrix representation, the route matrices, which facilitate route storage, indexing, transmission, and processing.
- The proposed method makes user privacy preservation simple and robust by sending the target route with several dummy (false) routes. Therefore, the server is not able to associate the routes with the users.

In addition, human travel behavior is useful and has been applied in many applications such as city and street design and planning. This research studies various issues related to human travel behavior; e.g., what are the frequent routes from location A to location B and which part of city has the highest probability of travel anomalies? The experimental results show the proposed method is satisfactory, but more tests and analyses are needed to prove its effectiveness.

The high popularity of smartphones makes human travel route prediction become more useful. For example, mobile users could use the prediction to find other mobile users. However, this research still has rooms for improvements, three of which are suggested as follows:

- Traffic flows compared to previous routes are easier to collect, but they also suffer the problem of low accuracy. Finding another kind of data meeting the requirements of easy collection and high accuracy is our next mission.
- Evaluating a location-based service is always difficult because the subjects are mobile. A simulation model can be developed to facilitate the evaluations.
- User privacy is preserved by checking the routes on the client side, but sending a set of map matrices between clients and servers may consume a great deal of resources such as time and bandwidth. Instead, sending the user routes to the servers may reduce the consumption.

7. REFERENCES

[1] K. Kolodziej and J. Hjelm. *Local Positioning Systems: LBS Applications and Services*, CRC Taylor & Francis, 2006.

[2] S. J. Vaughan-Nichols. Will mobile computing's future be location, location, location? *IEEE Computer*, 4(2), 14-17, February 6, 2009.

[3] S. Wang, J. Min, and B. K. Yi. Location based services for mobiles: technologies and standards. *Proceedings of the IEEE International Conference on Communication (ICC)*, Beijing, China, May 19-23, 2008.

[4] S. Steiniger, M. Neun, and A. Edwardes. *Foundations of Location-Based Services*. 2006. [Online]. Available: http://www.geo.unizh.ch/publications/cartouche/lbs_lecturenotes_steinigeretal2006.pdf

[5] J. Eisner, S. Funke, A. Herbst, A. Spillner, and S. Storandt. Algorithms for matching and predicting trajectories. In *Proceedings of the 13th Workshop on Algorithm Engineering and Experiments (ALENEX11)*, pages 84-95, San Francisco, California, USA, January 22, 2011.

[6] G. Ferrer and A. Sanfeliu. Comparative analysis of human motion trajectory prediction using minimum variance curvature. In *Proceedings of the 6th ACM/IEEE International Conference on Human-Robot Interaction (HRI 2011)*, pages 135-136, Lausanne, Switzerland, March 6-9, 2011.

[7] J. Krumm and E. Horvitz. Predestination: inferring destinations from partial trajectories. In *Proceedings of the 8th International Conference on Ubiquitous Computing (UbiComp 2006)*, Orange County, California, USA September 17-21, 2006.

[8] S.-C. Liou and Y.-M Huang. Trajectory predictions in mobile networks. *International Journal of Information Technology*, 11(11), 109-122, 2005.

[9] C.-Y. Chow and M. F. Mokbel. Privacy of spatial trajectories. In Y. Zheng & X. Zhou (Eds.), *Computing with Spatial Trajectories* (pp. 109-141). New York: Springer, 2011.

[10] A. Deutsch, R. Hull, A. Vyas, and K. K. Zhao. Policy aware sender anonymity in location based services. In *Proceedings of the 26th International Conference Data Engineering (ICDE 2010)*, Long Beach, California, USA, March 1-6, 2010.

[11] T. Xu and Y. Cai. Location anonymity in continuous location-based services. In *Proceedings of the 15th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL GIS 2007)*, Seattle, Washington, USA, November 7-9, 2007.

[12] T. H. You, W. C. Peng, and W. C. Lee. Protecting moving trajectories with dummies. In *Proceedings of the International Workshop on Privacy-Aware Location-Based Mobile Services (PALMS)*, Mannheim, Germany, May 11, 2007.

[13] H. Kido, Y. Yanagisawa, and T. Satoh. An anonymous communication technique using dummies for location-based services. In *Proceedings of IEEE International Conference on Pervasive Services (ICPS'05)*, pages 88-97, Santorini, Greece, 2005.

[14] W.-C. Lee and J. Krumm. Trajectory preprocessing. In Y. Zheng and X. Zhou (eds.), *Computing with Spatial Trajectories* (pp. 3-32). New York: Springer, 2011.

[15] K. Deng, K., Xie, K, Zheng, and X. Zhou. Trajectory indexing and retrieval. In Y. Zheng and X. Zhou (eds.), *Computing with Spatial Trajectories* (pp. 35-59). New York: Springer, 2011.

[16] Microsoft. *Bing Maps Tile System*. 2014. [Online]. Available: http://msdn.microsoft.com/en-us/library/bb259689.aspx

[17] CloudMade. *The CloudMade Developer Zone*. 2014. [Online]. Available: http://developers.cloudmade.com/

[18] S. Pissanetzky. *Sparse Matrix Technology*. London: Academic Press, 1984.

[19] Y. Li, S. Peng, and W. Chu. Optimal Algorithms for Finding a Trunk on a Tree Network and its Applications. *The Computer Journal*, 52(4), 268-275, 2009.

[20] S. Peng and W.-t Lo. Efficient algorithms for finding a core of a tree with a specified length. *Journal of Algorithms*, 20(3), 445-458, 1996.