Sensors and Materials, Vol. 27, No. 8 (2015) 599–604
MYU Tokyo

S & M 1095

Incremental Clustering Method for Recognizing User Destinations and Routes Using Smartphone Global Positioning System Sensor

Je-Min Kim*,†, Hae-Jung Beak[1,†] and Young-Tack Park[2]

Myongji University, 116 Myongjiro, Cheoingu, Yongin, Gyeonggi 449-728, Korea
[1]Jeonju Kijeon University, Jeonjucheon Seoro 267, Wansangu, Jeonju 560-701, Korea
[2]School of Computing, Soongsil University, Sangdo-dong, Dongjakgu, Seoul 156-743, Korea

(Received July 2, 2014; accepted May 11, 2015)

Key words: user route, user destination, clustering, GPS, smartphone

An effective user route identification system is a major technological capability that mobile intelligent systems can exploit. To develop such a system, data that include information about user destinations and routes are required. In this paper, we propose a method that incrementally recognizes user destinations and routes by means of logs collected from a global positioning system (GPS) sensor. Applying clustering methods to detect destinations incrementally requires defined thresholds and a strategy of cluster creation. In addition, to identify routes, a procedure for dividing GPS logs into trajectories and a function for calculating the similarity between trajectories are necessary. In this paper, we address these requirements.

1. Introduction

Personal route information is critical for mobile intelligent systems. Many studies have been conducted on user route prediction and tracking based on mobile sensor data.[1–4] Sensor data presented as simple numerical values cannot be adequately represented semantically, as is often required by mobile intelligent systems. Therefore, systems that identify the personal routes of mobile users based on sensor data are necessary. In order to develop such systems, sample data from destinations and personal routes of mobile users must be included in the design process.

In this paper, we propose several practical approaches to identifying user destinations and travel routes based on global positioning system (GPS) data from smartphones. To enable our system to recognize destinations and routes in real time, we adopt density-based clustering and an algorithm for calculating the similarity between trajectories presented by GPS coordinates. Destinations differ among individuals. Therefore, the thresholds used for density-based clustering (minimum radius of the neighborhood and minimum number of points) are adopted dynamically. In addition, we define an

*Corresponding author: e-mail: kimjemins@hotmail.com
†Both authors contributed equally to this work.

ISSN 0914-4935

incremental creation and merge methods for updating new destinations and routes. Because GPS signal reception indoors is impossible and because most GPS coordinates are inaccurate, misidentifying user movements often occurs. In the process of identifying and updating user routes, we employ a function that calculates the distance between two streets.

2. Materials and Methods

2.1 *Recognizing initial user destinations*

First, we must find the coordinates at which a mobile user remains for more than 10 min according to an initial GPS log because a person stays for more than 10 min in a given destination. Coordinates differ slightly when a user revisits a location. Therefore, in our study, we merged the coordinates representing a revisited location using the density-based spatial clustering of applications with noise (DBSCAN). DBSCAN refines the destinations when a minimum radius of the neighborhood, or *Eps*, and a minimum number of points in an *Eps*, or *MinPts*, are set as thresholds to determine the density of clusters. When a large *Eps* is set, several destinations are clustered into a single destination. However, when a small *Eps* is set, the actual destinations are clustered separately. When we analyze the GPS log collected from participants in our experiments, the distances between the destinations differ according to the travel patterns of the participants. Thus, having the system automatically determine an appropriate *Eps* for each user is necessary. Table 1 provides a means of determining *Eps* automatically, which is accomplished by confirming the number of clusters and reducing the *Eps* based on the general *MinPts*. If this process is repeated, the number of clusters increases. However, when *Eps* falls below the clustering criterion, the number of clusters decreases. Therefore, *Eps* is determined on the basis of this process in order to reduce the number of clusters.

In order to determine the general *MinPts*, we first assume that most people visit common places such as their homes or offices every day. In addition, the fault probability for our system to detect GPS is reflected in a certainty factor, because a smartphone's GPS signal is not always detected. The following eq. (1) is given:

$$\text{General } MinPts = \alpha \times n, \tag{1}$$

where α denotes the certainty factor and n denotes the total number of days in which data was collected. A destination is represented as the center point of each cluster. We

Table 1
Algorithms for recognizing initial destinations.

Algorithm 1 — *Eps* determination	Algorithm 2 — Clustering destinations
Input: Stay-point array *sp*, General *MinPts*	Input: *sp*, General *MinPts*, *Eps*
Output: *Eps*	Output: destination info *dst_info*
Eps ← initial *Eps*	level ← 1
While	While level < 4
clusters[] ← call dbscan(*sp*, *Eps*, *MinPts*)	clusters[] ← call dbscan(*sp*, *Eps*, *MinPts*)
If length of clusters < pre_clusters break	*dst_info* ← assign level to clustered destinations
pre_clusters ← clusters	reduce *MinPts* by 75%
reduce *Eps* 10%	level++

divide all clusters into three-level destinations on the basis of the frequency of visits. Level 1 destinations are visits of more than five days a week, such as to a home or office. Level 2 destinations are visits of one or two days each week, such as to a church. Level 3 destinations are visits of one or two days each month to any destination. Algorithm 2 details the procedure for recognizing each level destination.

2.2 *Recognizing initial user routes*

The algorithmic procedure for recognizing initial routes contains five steps. The first step is detecting candidate trip switch points (*TSPs*). *TSPs* are coordinates that change from natural modes of transportation or travel behavior to artificial (automatic) modes of transportation, or the reverse. The detected candidate *TSPs* are refined into *TSPs* by DBSCAN. The second step involves dividing the coordinate sequences of the initial GPS log into trips based on user destinations and *TSPs*. Each trip includes the sequence of streets on which the user traveled while using the same mode of transportation. The third step involves classifying trips with the same starting and finishing points. The fourth step involves generating trip clusters (i.e., a set of similar trips) after calculating the similarity of trips within each class. The final step involves identifying routes by arranging trip sequences based on the initial GPS log and trip clusters. Trip clustering is the most critical procedure used to recognize routes. Table 2 describes the steps involved in trip clustering.

Function 1 is used to calculate the similarity between a particular cluster C_n and a trip TR_j. The similarity is computed based on the distance between streets of C_n and TR_j. To calculate the distance between two streets, we adopted a method proposed by Froehlich and Krumm.[5] Given s_i (a street with a nonclustered trip TR_j) and s_j (a street in a cluster C_n), the following eq. (2) determines the distance between the two streets (Fig. 1).

$$\begin{aligned} dist(s_i, s_j) &= d_{\text{center}}(s_i, s_j) + d_\theta(s_i, s_j) + d_\parallel(s_i, s_j) \\ d_{\text{center}}(s_i, s_j) &= \left| center_{s_i} - center_{s_j} \right|, \quad d_\parallel(s_i, s_j) = \min(s_{\parallel 1}, s_{\parallel 2}) \\ d_\theta(s_i, s_j) &= \begin{cases} \left| s_j \right| \times \sin(\theta) & 0^\circ \le \theta \le 90^\circ \\ \left| s_j \right| & 90^\circ \le \theta \le 180^\circ \end{cases} \end{aligned} \tag{2}$$

Table 2
Algorithm and function for recognizing initial routes.

Algorithm 3 — trip clustering	Function 1 — sim(C_n, TR_j): Computing similarity of the particular C_n with TR_j
Input: trip set $\{TR_1, TR_2, ..., TR_j\}$, Output: trip-clusters $\{C_1, C_2, ..., C_n\}$	$\text{sim}(C_n, TR_j) = \sum_{n=1} \delta(s^n_{TR_j}, S_{C_n})/n$

$C_1 \leftarrow TR_1$
For every TR_j do
$s \leftarrow$ call sim(C_n, TR_j)
If sim(C_n, TR_j) $\le S_{\max}$ then
$C_n \cup TR_j$
Else
Create a new trip cluster for TR_j
	$\delta(s^n_{TR_j}, S_{C_n})$: The closest distance between $s^n_{TR_j}$ and S_{C_n}
	$s^n_{TR_j}$: The nth street of TR_j
	S_{C_n} : The street set in C_n

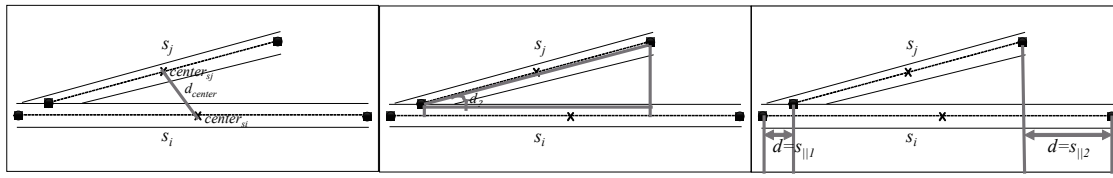

Fig. 1. Factors for computing distance between two streets.

2.3 *Updating user destinations and routes*

Because a user may not always travel through recognized routes to destinations, updating user destinations and routes based on sequences of detected GPS coordinates in real time is necessary. Therefore, the current destination must be merged with existing clusters of destinations, or a new cluster should be created. Figure 2 shows procedures for updating clusters of destinations.

If any coordinates in existing clusters are not included in *Eps* for the current destination p, a new cluster including p is created and the cluster level is set to 3. If the coordinates p_i are included in *Eps* for p, the number of coordinates n and p in *Eps* of p_i is confirmed. When n satisfies *MinPts*, p is merged to the cluster C_i that includes p_i. By contrast, a new cluster is created by p and the cluster level is set to 3 in the event that *MinPts* are not satisfied. Finally, the level of clusters is adjusted by n in each cluster. (If n is more than General *MinPts*, the level is 1; if n is less than 25% of General *MinPts*, the level is 2; and if n is greater than 1, the level is 3.) Regarding the updating of routes, after selecting a representative trip t_i among trips in each trip cluster T_n, we compute the similarity of t_i and trip t_n, which is refined from the sequence of newly collected GPS data. Because a trip consists of a series of streets, the variable t_i is the trip that shares the greatest number of similar streets to all other trips in T_n. If the similar value of t_i and t_n is less than a given threshold, a new trip cluster is created by t_n and a new route is then composed of the sequence of newly collected GPS coordinates.

3. Results and Discussion

GPS logs used in our experiments were collected from the Android smartphones of five participants. The time, GPS coordinates, *travel behaviors*[6] (In ref. 6, travel behaviors are identified from a method using accelerometers. We use only the results of this paper), and streets were recorded in the log. First, we evaluated the efficiency of the automatically calculated *Eps* and the accuracy of the incrementally recognized destinations. To evaluate the accuracy of the recognized destinations, we visualized the clusters of destinations using Google Maps and participants verified the data for their own destination coordinates. Table 3 shows the data for collected logs and numbers of destinations correctly recognized based on *Eps*, which was calculated using the proposed algorithm and two fixed *Eps* values according to general GPS errors (25 and 50 m).

Second, we evaluated the accuracy of the recognized routes by considering their robustness and completeness. The robustness was based on routes in a particular route cluster all having the same travel trajectory. The completeness was based on the same

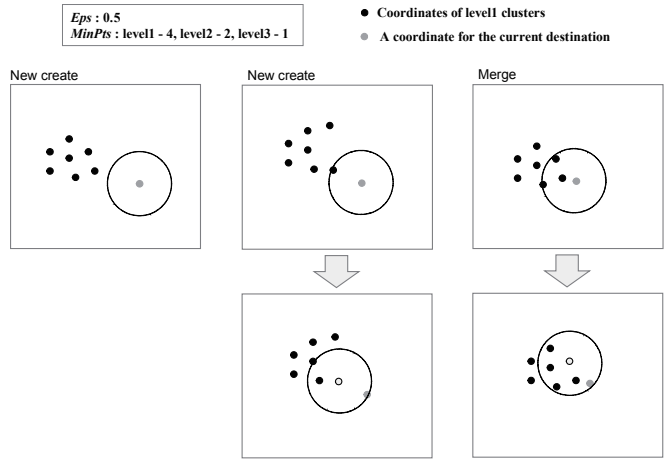

Fig. 2. Procedure for updating a cluster of destinations.

Table 3
Results of incrementally recognized destinations by *Eps*.

Participant	Days for collecting GPS log	Days for collecting GPS logs for recognizing initial destination	Number of initial destinations				After 7 d, number of destination				After 14 d, number of destination			
			Real	*Eps*	25 m	50 m	Real	*Eps*	25 m	50 m	Real	*Eps*	25 m	50 m
1	44 d	30 d	9	9	9	9	9	9	9	9	9	9	9	9
2	44 d	30 d	12	11	10	8	13	12	11	8	13	12	11	8
3	58 d	30 d	17	17	13	17	19	19	14	17	19	19	14	17
4	58 d	30 d	20	19	16	18	20	19	16	18	21	20	17	19
5	65 d	30 d	15	15	12	14	15	15	12	14	15	15	12	14
Participant	Days for collecting GPS log	Days for collecting GPS logs for recognizing initial destination	After 21 d, number of destination				After 28 d, number of destination				After 35 d, number of destination			
			Real	*Eps*	25 m	50 m	Real	*Eps*	25 m	50 m	Real	*Eps*	25 m	50 m
1	44 d	30 d	10	10	10	9	10	10	10	9	11	11	11	10
2	44 d	30 d	15	15	11	8	15	14	11	8	16	14	11	8
3	58 d	30 d	19	19	14	17	19	19	14	17	20	20	15	18
4	58 d	30 d	21	20	17	19	22	21	18	19	24	22	20	20
5	65 d	30 d	17	17	12	15	19	18	12	15	20	19	13	16

travel trajectories in the GPS logs forming a single route. Table 4 shows the robustness and completeness of the routes incrementally identified from the GPS logs of each participant.

From experiments conducted with five smartphone users, the average accuracy of incrementally recognized user destinations was 91.4%, the robustness of identified routes was 94.8%, and the completeness was 89.8%. These results show that the suggested approach is efficient and practical.

Table 4
Robustness (r) and completeness (c) of recognized routes.

Participant	Initial recognition			After 7 d			After 14 d		
	Number	r	c	Number	r	c	Number	r	c
1	14	1.00	0.85	14	1.00	0.85	14	1.00	0.85
2	18	0.94	0.89	20	0.95	0.90	20	0.95	0.90
3	25	0.96	0.92	31	0.97	0.94	31	0.97	0.94
4	31	0.94	0.94	31	0.94	0.94	36	0.94	0.94
5	24	0.96	0.92	24	0.96	0.92	24	0.96	0.92
Participant	After 21 d			After 28 d			After 35 d		
	Number	r	c	Number	r	c	Number	r	c
1	16	1.00	0.88	16	1.00	0.88	19	0.95	0.89
2	24	0.96	0.92	24	0.96	0.92	27	0.93	0.93
3	31	0.97	0.94	31	0.97	0.94	33	0.97	0.88
4	36	0.94	0.94	36	0.94	0.94	39	0.92	0.90
5	30	0.97	0.87	34	0.97	0.88	38	0.97	0.89

4. Conclusions

We proposed a practical approach for recognizing personal destinations and routes using a smartphone GPS sensor. To improve the efficiency of the system, we focused on three major areas of concern: (1) recognition of user destinations by clustering using an algorithm for identifying a suitable *Eps*, (2) identification of user routes by trip clustering by computing the distance between two streets, and (3) an incremental updater of destinations and routes based on sequences of GPS coordinates obtained in real time. The experimental results showed that the suggested approach is extremely efficient and practical and can be effectively adopted for use in constructing a personal route model for location-based services.

Acknowledgements

This work was supported by the ICT R&D program of MSIP/IITP. (10035348, Development of a Cognitive Planning and Learning Model for Mobile Platforms)

References

1 Z. Chen, H. T. Shen, X. Zhou, Y. Zheng and X. Xie: Proceedings of the 2010 International Conference on Management of Data (ACM, New York, 2010) pp. 255–266.
2 L. Liao, D. J. Patterson, D. Fox and H. Kautz: Proceedings of the AAAI Conference on Artificial Intelligence (The AAAI Press, Menlo Park, 2007) pp. 311–331.
3 D. Ashbrook and T. Starner: PUC **7** (2003) 275.
4 K. Tanaka, Y. Kishino, T. Terada and S. Nishio: Proceedings of the 2009 ACM Symposium on Applied Computing (SAC) (ACM, New York, 2009) pp. 190–195.
5 J. Froehlich and J. Krumm: Proceedings of the SAE. 2008 World Congress.
6 M. Han, L. T. Vinh, Y. K. Lee and S.Y. Lee: Sensors **12** (2012) 12588.