Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2012, Article ID 675605, 12 pages
doi:10.1155/2012/675605

Research Article

Route Anomaly Detection Using a Linear Route Representation

Wen-Chen Hu,[1] Naima Kaabouch,[2] Hung-Jen Yang,[3] and S. Hossein Mousavinezhad[4]

[1] *Department of Computer Science, The University of North Dakota, Grand Forks, ND 58202, USA*
[2] *Department of Electrical Engineering, The University of North Dakota, Grand Forks, ND 58202, USA*
[3] *Industrial Technology Educational Department, National Kaohsiung Normal University, Kaohsiung City 80201, Taiwan*
[4] *Department of Electrical Engineering, Idaho State University, Pocatello, ID 83209, USA*

Correspondence should be addressed to Wen-Chen Hu, wenchen@cs.und.edu

Received 18 March 2012; Accepted 2 June 2012

Academic Editor: MoonBae Song

A location-based service is a service based on the geographical position of a mobile handheld device like a smartphone. This research proposes location-based research, which uses location information to find route anomalies, a common problem of daily life. For example, an alert should be generated when a deliveryman does not follow his regular route to make deliveries. Different kinds of route anomalies are discussed and various methods for detecting the anomalies are proposed in this paper. The proposed method based on a linear route representation finds the matched routes from a set of stored routes as the current route is entered location by location. Route matching is made easy by comparing the current location to linear routes. An alert is generated when no matched routes exist. Preliminary experimental results show the proposed methods are effective and easy to use.

1. Introduction

Table 1 shows the worldwide PC and mobile phone sales according to various market research reports [1]. The number of smartphones shipped worldwide has passed the number of PCs and servers shipped in 2011 and the gap between them is expected to keep bigger. The emerging smartphones have created many kinds of applications that are not possible or inconvenient for PCs and servers, even notebooks. One of the best-seller applications is location-based services (LBSs) according to the following market research.

(i) Hillard [2] reports 80% of smartphone owners have location-based services and half of them use services that provide offers, promotions, and sales based on their current locations.

(ii) The most convenient mobile shopping experience is price comparison and product research according to JiWire [3].

(iii) The number of location-based services users was increased from 12.3 million in 2009 to 33.2 million in 2010 (170% increase) in the US based on SNL Kagan [4].

This paper proposes location-based research, which uses location information to find route anomalies. Different kinds of route anomalies are discussed and various methods for detecting the anomalies are proposed in this research. It is divided into five steps: (i) route data collection, (ii) route data preparation, (iii) route pattern discovery, (iv) route pattern analysis and visualization, and (v) route anomaly detection. The major methods use a technique of incremental location search based on a linear route representation, which facilitates the route storage and matching. It begins the searching as soon as the first location of the search route is entered. Location-by-location, one or more possible matches for the route are found and immediately presented. Route matching is made easy by comparing the current location to linear routes. An alert is generated when no matched routes exist. Preliminary experiment results show the proposed methods are effective and easy to use.

The rest of this paper is organized as follows. Section 2 gives the background information of this research, which includes three themes (i) location-based services, (ii) related location-based research, and (iii) route representations and matching. The proposed system is introduced in Section 3, and several simple methods of route anomaly detection are explained too. Section 4 details the two major methods

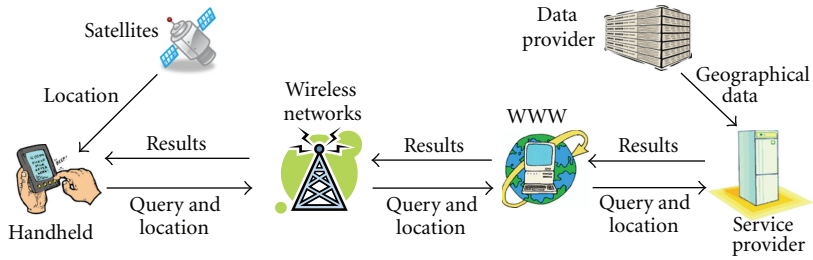


Figure 1: A system structure of generic location-based services.

Table 1: Worldwide PC, cellphone, and tablet PC sales.

Year	Number of units shipped (million)				
	Mobile phones	PCs and servers	Smart phones	PDAs (without phone capability)	Tablet PCs
2002	432	148	—	12.1	—
2003	520	169	—	11.5	—
2004	713	189	—	12.5	—
2005	813	209	—	14.9	—
2006	991	239	64	17.7	—
2007	1153	271	122	—	—
2008	1220	302	139	—	—
2009	1221	306	166	—	1
2010	1609	346	286	—	17
2011	1775	353	486	—	73

using a linear route representation and incremental location searching. Section 5 gives experimental results and evaluations. The last section gives a summary of this research.

2. Background

Three themes are related to this research: (i) location-based services, (ii) related location-based research, and (iii) route representation and matching, which are discussed in this section.

2.1. Location-Based Services (LBSs). A location-based service is a service based on the geographical position of a mobile handheld device [5, 6]. Two of the LBS examples are (i) finding a nearby ethnic restaurants and (ii) locating a nearby store with the best price of a product. A system structure of location-based services, shown in Figure 1, includes the following five major components [7].

(a) Mobile handheld devices, which are small computers that can be held in one hand. For most cases, they are smartphones.

(b) Positioning system, which is a navigation satellite system that provides location and time information to anyone with a receiver.

(c) Mobile and wireless networks, which relay the query and location information from devices to service providers and send the results from the providers to devices.

(d) Service providers, which provide the location-based services.

(e) Geographical data providers, which are databases storing a huge amount of geographical data such as information about restaurants and gas stations.

An example of location-based services using our research is given step by step as follows.

(1) The smartphone (a) includes an application of finding route anomalies.

(2) A mobile user submits a query of finding route anomalies along with the location information from a positioning system (b) to the application program, which runs in background.

(3) The application program calls the server-side programs (d) located at the Aerospace School of the University of North Dakota along with the location information via mobile or wireless networks (c).

(4) The programs at the servers perform the route anomaly detection using an Oracle database (e) which stores route data. Appropriate actions such as sending an alert are taken when an anomaly occurs.

(5) The results such as an acknowledgement of sending an alert are sent back to the smartphone.

A nice introduction of LBS technologies and standards is given by Wang et al. [8].

2.2. Related Location-Based Research. Various kinds of location-based research can be found in journals and conference proceedings. This subsection discusses several important location-based articles involving travel sequences.

(i) Zheng et al. [9] propose a method to mine interesting locations and travel sequences. Three steps are used in this research.

Step 1. Model multiple individuals' location histories with a tree-based hierarchical graph (TBHG).

Step 2. Based on the TBHG, a HITS- (Hypertext Induced Topic Search) based inference model is constructed.

Step 3. Mine the travel sequences among locations considering the interests of these locations and users' travel experiences.

Finally, a large GPS dataset, collected by 107 users over a period of one year in the real world, is used to evaluate their system. The result shows their HITS-based inference model outperformed baseline approaches like rank-by-count and rank-by-frequency.

(ii) In a paper from Zheng et al. [10], an approach based on supervised learning is proposed to automatically infer transportation mode from raw GPS data. The transportation mode, such as walking and driving, implied in a user's GPS data can provide them valuable knowledge to understand the user. Their approach consists of three parts: (i) a change point-based segmentation method, (ii) an inference model, and (iii) a postprocessing algorithm based on conditional probability. They evaluated the approach using the GPS data collected by 45 users over six months period. The result shows the change point-based method achieved a higher degree of accuracy in predicting transportation modes and detecting transitions between them.

(iii) Previous routes can be used to recommend future travel patterns. Yoon et al. [11] propose itinerary recommendation based on multiple user-generated GPS trajectories. Users only need to provide a start point, an end point, and travel duration to receive an itinerary recommendation. Liu and Chang [12] presents a route recommendation system which guides the user through a series of locations. Their system uses the methods of sequential pattern mining to extract popular routes from a set of stored routes from previous users. It then recommends routes by matching the user's current route with the extracted routes.

Some related location-based research can be found from the articles [13–18].

2.3. Route Representations and Matching. Traditionally, a travel route is stored as a series of locations (latitude and longitude) and route matching uses simple comparison. This research saves the routes as sequences of line segments and the route matching becomes finding the distance between the current location and line segments.

2.3.1. Route Representations. Route representations in computer are similar to image representations because each consists of a set of locations/pixels on a two-dimensional plane. Therefore, the representations of images can be applied to route representations and matching.

(i) Chang et al. [4] proposed a 2D string representation. A matching query may specify a 2D string, transforming retrieval into a 2D subsequence matching.

(ii) A 2D C-string for spatial knowledge representation, which employs a cutting mechanism and a set of spatial operators, was proposed by Lee and Hsu in 1990 [19].

(iii) Huang and Jean [20] proposed a 2D C^+-string representation scheme which extended the 2D C-string by including relative metric information about the picture into the strings.

Other representations can be found from the articles by Lee et al. [21] and Wu and Chang [22].

2.3.2. Route Matching. Incremental search is a progressive search, which finds matched text as the search string is entered character by character. Most incremental searches are based on the research of Aho and Corasick [23], who develop an algorithm to locate all occurrences of any of a finite number of keywords in a string of text. The algorithm consists of constructing a finite state pattern matching machine from the keywords and then using the pattern matching machine to process the text string in a single pass. Construction of the pattern matching machine takes time proportional to the sum of the lengths of the keywords. The number of state transitions made by the pattern matching machine in processing the text string is independent of the number of keywords. Several incremental search methods have been proposed as follows.

(i) Meyer [24] gives a problem of searching a given text for occurrences of certain strings, in the particular case where the set of strings may change as the search proceeds. He modified the algorithm of Aho and Corasick to allow incremental diagram construction, so that new keywords may be entered at any time during the search. The incremental algorithm presented essentially retains the time and space complexities of the nonincremental one.

(ii) The machine of Aho and Corasick must be reconstructed all over again when a keyword is appended. Tsuda et al. [25] propose an efficient algorithm to append a keyword for the machine. They present the time efficiency comparison with the original algorithm using the actual simulation results. The simulation results show the speed up factor, by the algorithm proposed, to be between 25- and 270-fold when compared with the original algorithm by Aho and Corasick which requires the reconstruction of the entire machine AC.

(iii) Koenig et al. [26] develop a search algorithm that combines incremental and heuristic search, namely, Lifelong Planning A* (LPA*). It is named "Lifelong Planning" because it reuses information from previous searches. Their method repeatedly finds shortest paths from a given start vertex to a given goal vertex while the edge costs of a graph change.

A survey of incremental heuristic searches can be found from the article by Koenig et al. [27].

3. The Proposed System

The GPS (global positioning system) function of smartphones provides location information of mobile users. Collections of location information are able to depict the mobile users' travel routes such as walking routes between homes and schools or salesman's delivery routes. This research uses the location information to find any route anomalies, for example, a pupil does not take the daily route to school. The proposed system is introduced in this section.

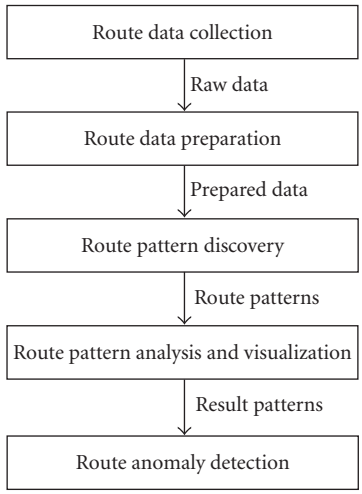


Figure 2: The five steps used by the proposed system.

3.1. The Proposed Steps. This research is to find route anomalies. It is divided into five steps as shown in Figure 2.

(a) *Route data collection*: this step collects route data before the application is used.

(b) *Route data preparation*: raw GPS data is usually not reliable and consistent and includes many noises. It has to be prepared before used.

(c) *Route pattern discovery*: not all routes are valid, for example, a very short route is usually not useful. This step puts valid routes into a database and removes invalid routes.

(d) *Route pattern analysis and visualization*: it analyzes the routes and allows users to view the routes on maps.

(e) *Route anomaly discovery*: this step is used to find any route anomalies, the theme of this research. The method of incremental search is used in this step.

Figure 3(a) shows the application icon, Route Checking, on an Android device. After clicking the icon, it displays the entry interface of this system as in Figure 3(b), which includes three radio buttons.

(i) *Collect route data*, which is for route data collection after this button is submitted. This function can be activated anytime, but most likely it is activated at the beginning of using this application.

(ii) *Check routes*, which redirects to the interface in Figure 9(b) for route checking after this button is submitted.

(iii) *Show stored routes*, which is used to display the information of the stored routes. An example of the basic route information is shown in Table 2, which includes the number and start and end times and locations of each route. The latitude and longitude of a location are represented by r and θ of the polar coordinate system. Routes can be added to the system from time to time and users are able to delete undesirable or never-used routes. Table 2 shows the basic information about routes. Details of each route are stored elsewhere.

Table 2: Basic route information including route numbers, and start and end times and locations.

Route number	Location	Time	Latitude	Longitude
1	Start	05/26/2012, 18:23:42	47.926823°	−97.080351°
	End	05/26/2012, 18:45:23	47.926825°	−97.080351°
2	Start	05/27/2012, 09:31:58	47.926810°	−97.080325°
	End	05/27/2012, 09:50:41	47.926812°	−97.080324°
⋮	⋮	⋮	⋮	⋮
n	Start	05/31/2012, 13:13:37	47.926857°	−97.080357°
	End	05/31/2012, 14:20:37	47.926859°	−97.080358°

3.2. System Implementation. The collected route data is usually raw because GPS data is usually not reliable and consistent and contains many noises [16]. The data needs to be processed before being used effectively. The methods of data preparation include filtering, recovery, restoration, and trajectory. Route data preparation used in this research includes the following.

(i) Two locations with slightly differences are treated the same, for example, the location $(r_1, \theta_1) = (5603, 243.10°)$ is the same as the location $(r_2, \theta_2) = (5609, 243.10°)$, because of low GPS accuracy and the traveler may take the same route but with different locations such as walking on the other side of a street.

(ii) Similar locations in a row (e.g., the traveler is idle) are reduced to one.

(iii) If the distance between two consecutive locations is greater than a threshold value, it may imply a GPS disconnection. Location trajectory may need to be used to insert locations between the two locations [28].

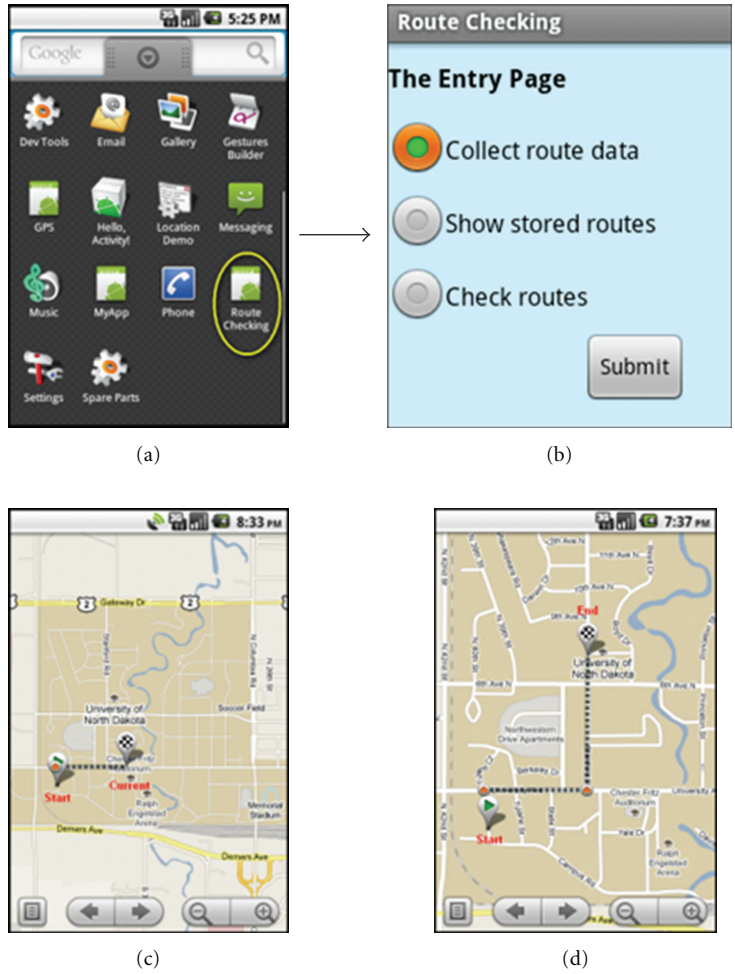
(iv) If a location is substantially different from the surrounding locations, it may imply a GPS noise. The location may be removed and location trajectory is used.

The system then checks the prepared route data and removes invalid routes, which may include the following.

(i) A route is with a very short distance such as few yards long.

(ii) A route is with many broken locations or noises and it cannot be fixed by the method of trajectory.

(iii) A route is with a very long or short duration, for example, less than one minute or longer than two hours.



(a) (b)

(c) (d)

Figure 3: (a) The application icon "Route Checking" on a device, (b) the entry page of the application, (c) the current route map, and (d) an example of a stored route.

This research uses location information to detect route anomalies. Many methods can be used to find route anomalies. Four kinds of detection are introduced in this subsection: (i) time check, (ii) border check, (iii) start and destination check, and (iv) route check. The first three methods are simple and are explained in this subsection. The last method, route check, is the focus of this research and is more complicated. It uses the method of pattern matching to find any route anomalies. The next subsection will be dedicated to it.

(i) *Time check*: for example, the Route number 1 takes about 23 minutes. If a trip follows the route and takes far more than 23 minutes, then an alert may be generated. Also, the start and end times can be used in this check. For example, the schools start at 8 : 30 AM. If the student has not arrived at school by 8 : 30 AM, then an alert may be generated.

(ii) *Border check*: for example, the delivery routes are within a community. If a route reaches out of the community, then an alert may be generated. An easy way to find the border is to box the routes such as the one shown in Figure 4.

(iii) *Start and destination check*: if the traveler does not start from any beginning locations of routes or does not reach any destination by a specific time, it may deserve an alert.

(iv) *Route check*: the above methods are simple, but lack accuracy. This method checks the routes with higher accuracy and uses more complicated algorithms, which will be detailed in the next section.

4. Route Checking Using a Linear Route Representation

Route checking is more complicated. This section discusses the methods used to find route anomalies. Traditional routes are represented by series of locations, which are complex and difficult to use. An algorithm is developed to straighten the routes so the routes can be stored as a set of line segments and route matching becomes a simple task of checking the distance between the current location and line segments. The proposed route checking can be divided into two cases, ordered and unordered routes. There is no alert generated if the traveler does not start at the beginning location or stop

Figure 4: An example of boxing a route.

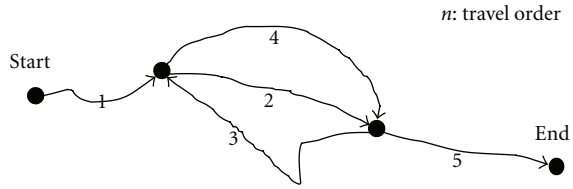


Figure 5: An example of order routes.

at the end of a route since the start and destination check can be easily used to check this condition.

4.1. Linear Approximating a Human Travel Route. The proposed linear approximation algorithm converts a human travel route into line segments. The approximation captures the essence of a route in the fewest possible line segments. A polygonal approximation, based on an error function, is applied to this method [29]. Before applying this algorithm, however, the route has to be smoothed to unit thickness so that branch routes may be located. Let e be the maximum allowable error. For a given location A, through which an approximation line must pass, one can define two points B and C at a distance e from A. The algorithm searches for the longest segment where the curve is contained between two parallel tangents starting B and C (see Algorithm 1).

The function *BEST_PATH* finds the route of the longest line segment when the location p_i has more than one 8-neighbor. This is why the route needs smoothing before applying the algorithm. A nonunit thickness route may mislead the algorithm into calling the *BEST_PATH* function. This algorithm requires quadratic time because the *BEST_PATH* function needs to check as many paths as possible and any location on the route, may invoke it. Figure 5 shows an example of order routes and Figure 6 shows the corresponding linear route after applying the algorithm *LINEAR_APPROX* to the route in Figure 5.

4.2. Ordered Routes. For this kind of routes, the order of the locations is relevant, for example, bus routes. An example of ordered routes is given in Figure 5, where the directional subroutes are the numbers 1, 2, 3, 4, and 5. The numbers

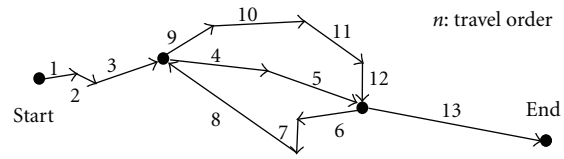


Figure 6: The corresponding linear route after applying the algorithm LINEAR_APPROX to the route in Figure 5.

also imply the travel order, for example, the traveler takes the route: start → 1 → 2 → 3 → 4 → 5 → end. Assume the traveler must travel along the stored routes. A simple algorithm using incremental location searching for finding route anomalies in ordered routes is given below. It checks the traveler's locations one by one. If the distance between the current location and the current line segments is greater than a threshold value, then it reports an anomaly. The traveler's beginning location can start anywhere in a route and the traveler can end the checking anywhere and anytime. (see Algorithm 2).

Assume there are five available routes where the letters represent line segments:

(1) c → a → e → f → n → l → s → p → g → h

(2) c → a → d → e → f → m → n → l → s → t → p → g → h → j

(3) s → t → o → g → h → c → a → d → f → m → n

(4) n → l → s → t → p → g → h → l → s → t → o → p

(5) c → a → d → b → e → m → r → n → s → t → g → h

Two examples are given next to show how this algorithm works.

(a) This example shows a route does not generate alerts by using this algorithm. Assume a traveler takes the route c → c → a → d → d → d → f → l → l. The five stored routes after applying the algorithm are as follows:

(1) c → a → e → f → n → l → s → p → g → h (2 matches)

(2) c → a → d → e → f → m → n → l → s → t → p → g → h → j (perfect matches)

(3) s → t → o → g → h → c → a → d → f → m → n (4 matches)

(4) n → l → s → t → p → g → h → l → s → t → o → p (no match)

(5) c → a → d → b → e → m → r → n → s → t → g → h (3 matches)

where an underlined location means a match. The Route number 2 has a perfect match, so no alert is generated by this algorithm. However, an alert may be generated because no end location of any route is reached. This anomaly can be found by the start and destination check as mentioned in Section 4.1.

// Linear Approximating a Human Travel Route

LINEAR_APPROX (ROUTE, p_i, p_j, e)
// ROUTE: a human travel route in a series of locations (latitude, longitude)
// p_i: the initial location of the route
// p_j: the neighbor of p_i on the route
// e: the maximum allowable error

(1) $b \leftarrow p_i + e$
(2) $c \leftarrow p_i - e$
(3) IS_FIRST ← TRUE
(4) **while** NUMBER_8_NEIGHBOR (p_i) $\neq 0$
(5) **while** NUMBER_8_NEIGHBOR (p_i) > 1
(6) $p_j \leftarrow$ BEST_PATH (ROUTE, p_i, e)
(7) LINEAR_APPROX (ROUTE, p_i, p_j, e)
(8) **print** p_i
(9) ROUTE [p_i] ← CLEAR
(10) $\overline{L_b} \leftarrow \overline{bp_j}$
(11) $\overline{L_c} \leftarrow \overline{cp_j}$
(12) **if** IS_FIRST
(13) $\overline{L_{\text{above}}} \leftarrow \overline{L_b}$
(14) $\overline{L_{\text{below}}} \leftarrow \overline{L_c}$
(15) IS_FIRST ← FALSE
(16) **else**
(17) **if** $\overline{L_b}$ **is above** $\overline{L_{\text{above}}}$
(18) $\overline{L_{\text{above}}} \leftarrow \overline{L_b}$
(19) **if** $\overline{L_c}$ **is below** $\overline{L_{\text{below}}}$
(20) $\overline{L_{\text{below}}} \leftarrow \overline{L_c}$
(21) **if** $\angle(\overline{L_{\text{above}}}, \overline{L_{\text{below}}}) > 0$
(22) **print** p_i
(23) $b \leftarrow p_i + e$
(24) $c \leftarrow p_i - e$
(25) IS_FIRST ← TRUE
(26) $p_i \leftarrow p_j$
(27) $p_j \leftarrow$ 8_NEIGHBOR(p_i)
(28) ROUTE[p_i] ← CLEAR
(29) **print** p_i

Algorithm 1

Route Checking for Ordered Routes

(1) Every stored route is an available route.
(2) Find the start location of the traveler in an available route i and set the current line segment i_c to the first matched segment.
(3) d_0 = the distance between the current location and the current line segment i_c.
d_1 = the distance between the current location and the next line segment i_{c+1}.
if [$d_0 \leq e$ (the allowable error)], then do nothing,
else if [($d_0 > e$) and ($d_1 \leq e$)], then $i_c = i_{c+1}$ and $i_{c+1} = i_{c+2}$ (the next segment after i_{c+1}),
else remove the route from the available routes. If the available routes are empty, report an anomaly and stop.
(4) Repeat the above step for the next location of the traveler until the traveler ends the route checking.

Algorithm 2

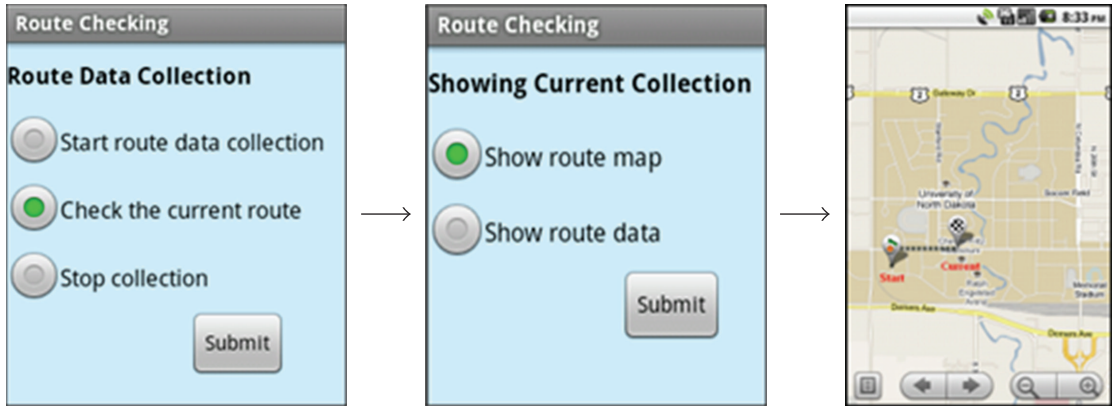


Figure 7: (a) The page of route data collection, (b) the page of checking the current route, and (c) the current route map.

(b) This example shows a route generates an alert by using this algorithm. Assume a traveler takes the route c → a → d → d → f → p → p → p. The five stored routes after applying the algorithm are as follows:

(1) c → a → e → f → n → l → s → p → g → h (2 matches)

(2) c → a → d → e → f → m → n → l → s → t → p → g → h → j (4 matches)

(3) s → t → o → g → h → c → a → d → f → m → n (4 matches)

(4) n → l → s → t → p → g → h → l → s → t → o → p (no match)

(5) c → a → d → b → e → m → r → n → s → t → g → h (3 matches)

The last location p does not have a match in any route so an alert is generated by this algorithm.

4.3. Unordered Routes. For unordered routes, the order of the locations, maybe except the start and end locations, is irrelevant, for example, newspaper delivery. Figure 7 shows an example of unordered routes if the numbers are ignored. In order to find anomalies in unordered routes, the route data collection needs to find and save all line segments connected to intersections, such as the {3, 4, 8, 9} and {5, 6, 12, 13} in Figure 10, and bidirectional subroutes among the start and end segments of each route. An intersection is a road junction where two or more roads either meet or cross at grade. For example, the route in Figure 10 includes the following items:

(i) Start segment: Segment 1,

(ii) End segment: Segment 13,

(iii) Line groups: {3, 4, 8, 9} and {5, 6, 12, 13},

(iv) Bidirectional subroutes: 1↔2↔3, 4↔5, 6↔7↔8, 9↔10↔11↔12, and 13.

The algorithm of finding route anomalies in unordered routes is given below. It is more complicated compared to the one for ordered routes because the traveler's next location could be in several possible subroutes. When an intersection is reached, the algorithm checks all available subroutes from the intersection.

Algorithm 3 does not check whether all subroutes are visited. It could be easily remedied by adding a checker on each subroute. Also, the algorithm generates an alert if a subroute is visited twice because it is unusual to travel a subroute twice in most cases. It could be easily modified if users do not want it to generate an alert for this condition. Again, one example is given next to show how this algorithm works. Assume there are three available routes where the letters represent line segments. The three routes and the items saved for each route are listed in the following:

Route 1. **c → b → d → a → e → f → n → l → t → p → g → i → h**

(I) Start segment: Segment c,

(II) End segment: Segment h,

(III) Line groups: {a, e, p, t} and {g, i, l, n},

(IV) Subroutes: (i) c↔b↔d↔a, (ii) e↔f↔n, (iii) l↔t, (iv) p↔g, and (v) i↔h.

Route 2. **c → b → d → a → p → g → o → r → n → j → l → t → s → u**

(I) Start segment: Segment c,

(II) End segment: Segment u,

(III) Line groups: {a, p, s},

(IV) Subroutes: (i) c↔b↔d↔a, (ii) p↔g↔o↔r↔n↔j↔l↔t, and (iii) s↔u.

Route 3. **c → d → a → e → y → f → z → n → k → i → h**

(i) Start segment: Segment c,

(ii) End segment: Segment h,

(iii) Line groups: None,

(iv) Subroutes: c↔d↔a↔e↔y↔f↔z↔n↔k↔i↔h.

Route Checking for Unordered Routes

(1) Every stored route is an available route and every sub-route of an available route is an available sub-route.
(2) Find the start location of the traveler in an available sub-routes of an available route i and set the current line segment i_c to the first matched segment.
(3) d_0 = the distance between the current location and the current line segment i_c.
d_1 = the shortest distance between the current location and the line segment i_{c+1} among the ones connected to i_c.
if [$d_0 \leq e$ (the allowable error)], then do nothing,
else if [$(d_0 > e)$ and $(d_1 \leq e)$], then mark the i_c unavailable and $i_c = i_{c+1}$,
else remove the route from the available routes. If the available routes are empty, report an anomaly and stop.
(4) Repeat the above step for the next location of the traveler until the traveler ends the route checking.

Algorithm 3

The following example shows a route generates an alert by using this algorithm. Assume a traveler takes the route a → e → f → f → m → n → n → i. The following list shows how the algorithm works location by location:

(1) **a** → e → f → f → m → n → n → i, where the current location is a:

Route 1: c↔b↔d↔a
Route 2: c↔b↔d↔a
Route 3: c↔d↔a↔e↔y↔f↔z↔n↔k↔i↔h

(2) a → **e** → f → f → m → n → n → i, where the current location is e:

Route 1: e↔f↔n
Route 2: none
Route 3: c↔d↔a↔e↔y↔f↔z↔n↔k↔i↔h

(3) a → e → **f** → f → m → n → n → i, where the current location is f:

Route 1: e↔f↔n
Route 2: none
Route 3: c↔d↔a↔e↔y↔f↔z↔n↔k↔i↔h

(4) a → e → f → **f** → m → n → n → i, where the current location is m:

Route 1: e↔f↔n
Route 2: none
Route 3: c↔d↔a↔e↔y↔f↔z↔n↔k↔i↔h

(5) a → e → f → f → **m** → n → n → i, where the current location is m:

Route 1: none
Route 2: none
Route 3: none

Table 3: An example of detail information of a route.

number (minute)	Time	Latitude	Longitude
0	05/26/2010, 18:23:42	47.926823°	−97.080357°
1	05/26/2010, 18:24:42	47.926823°	−97.080357°
⋮	⋮	⋮	⋮
22	05/26/2010, 18:45:23	47.926825°	−97.080357°

where an underlined segment means a match. An alert is generated because there are no available subroutes when the current location is m.

5. Experimental Results

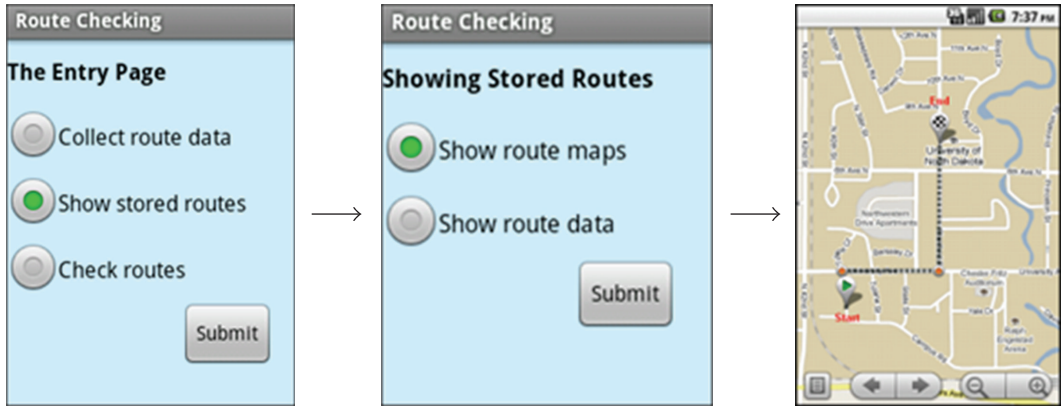
This section gives the experimental results.

5.1. Route Data Collection. The first step of this research is to collect route information. The collection could be activated anytime and anywhere. Figure 3(b) shows the entry page of the application. After a user picks the button "collect route data," the system displays the interface in Figure 7(a) including the following three radio buttons.

(i) *Start collecting route data*, which starts collecting route data. Typical location information provided by a smartphone includes times, the latitude, and longitude of a location. The location information is collected as frequently as possible and the collection frequencies depend on the travelling methods. For example, the frequencies for walking, biking, and driving are different. This process runs in background by using multithreading, so the smartphone can still function as usual.

(ii) *End collection*, which ends the current route data collection.

(iii) *Check the current route*, which is used to check the status of the current data collection including the map as in Figure 7(c) and data as in Table 3.



(a) (b) (c)

Figure 8: (a) The system entry page, (b) the page of showing stored routes, and (c) an example of a stored route.

Table 2 shows the basic information about routes. Details of each route are stored elsewhere. An example of the Route number 1 details is given in Table 3, where the locations and times are collected periodically, for example, every minute for walking and every 10 seconds for driving.

5.2. Route Data Preparation and Route Pattern Discovery, Analysis, and Visualization. The collected route data is usually raw because GPS data is usually not reliable and consistent and contains many noises [13]. The system also allows users to check the stored routes as in Figure 8(b). Other than showing them the route information as in Tables 2 and 3, users will also like to view the stored routes. One of the routes is shown in Figure 8(c).

5.3. Route Anomaly Detection. This research uses location information to detect route anomalies. An alert as in Figure 9(c) is generated when an anomaly is found. Otherwise, the smartphone just functions as usual. The alert can be sent via an email or a phone call. The interface in Figure 9(b) is for checking routes including the following three radio buttons.

(i) *Start checking the route*, which runs in background by using multithreaded programming, so the smartphone can still function as usual.

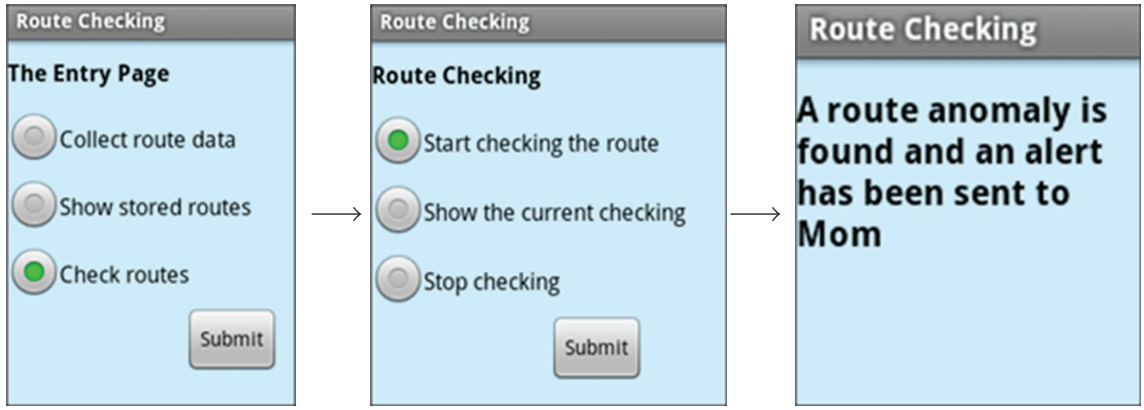
(ii) *End checking*, which stops the current route checking. Also, the check ends when an anomaly is found.

(iii) *Show the current checking*, which is used to check the status of the current route checking. For example, how closely will the current route trigger an alert or how many routes are matched so far? Other than showing the data of the current route as in Table 3 by using the bottom button in Figure 10(b), the system also shows the current position in a possible route as in Figure 10(c) by using the top button in Figure 10(b).

The proposed methods are convenient and effective. The execution is also efficient. The times used for the time check and start and destination check are constant. The times for the border check and route checks are $O(n)$, where n is the number of user locations, because the checks are performed location by location and each location requires a constant time to do the matching. The time for the procedure *Linear_Approx* is also $O(n)$, where n is the number of locations in a route, because the algorithm straightens the route location by location. Additionally, the procedure is only used during route collections, but not route checking, which happens more often.

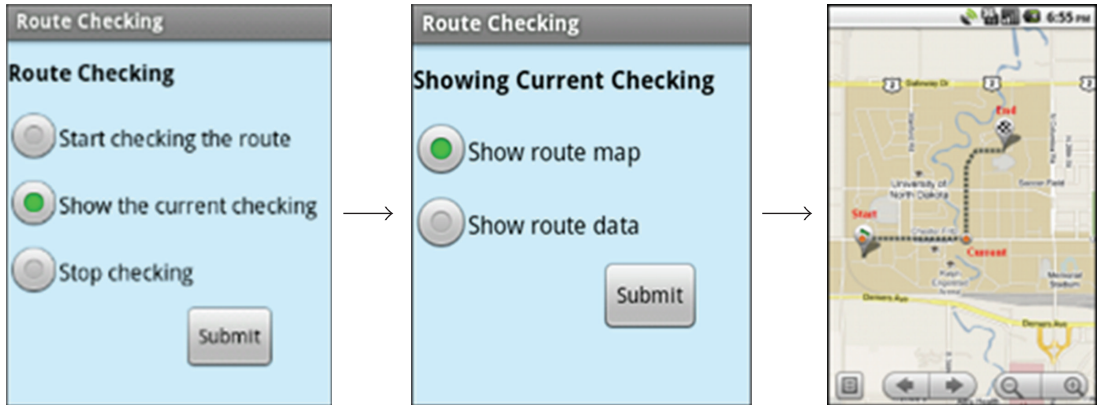
6. Conclusion

Mobile application stores (or app stores) sell or provide mobile applications/services for handheld devices such as smartphones. The applications/services are not necessarily from the storeowners. Many of them are from the third parties such as independent developers. A wide variety of mobile applications is available on the stores. One of the best-selling apps is related to location-based services, which cover a wide range of usages such as finding the lowest-price product in a nearby store and locating the nearest gas stations. This research proposes location-based research, which uses location information to find route anomalies, a common problem of daily life. For example, an alert should be generated when a school bus misses part of a route. Different kinds of route anomalies are discussed and various methods for detecting the anomalies are proposed in this paper. It is divided into five steps: (i) route data collection, (ii) route data preparation, (iii) route pattern discovery, (iv) route pattern analysis and visualization, and (v) route anomaly detection. The major methods use a linear route representation and incrementally search locations, which finds matched routes as the search route is entered location by location. It begins the searching as soon as the first location of the search route is entered. Location by location, one or more possible matches for the route are found and immediately presented. An alert is generated when no matched routes exist. Experimental results show the proposed methods are effective and easy to use.



(a) (b) (c)

Figure 9: (a) The entry page of the application, (b) the page of route checking, and (c) an acknowledgment message after detecting a route anomaly.



(a) (b) (c)

Figure 10: (a) The page of route checking, (b) the page of showing cu rrent checking, and (c) the current position in a stored route.

Other than the linear route representation, the proposed incremental location search is based on string matching, which is simple but effective. A search based on the following methods is worth consideration.

(i) *Finite automata*: the collected routes are used to build a finite automaton, which is then used to check any route anomalies.

(ii) *Matrix multiplication*: similar routes are found by matrix multiplications between the current route and the stored routes. If the product is greater than a threshold value, the stored route is said to be *similar* to the current route. Since the matrices are stored as strings, the multiplications can be done efficiently.

(iii) *Neural networks*: a route is a sequence of locations. Route matching is used to find any route anomalies and a modified Hopfield neural network can be designed to solve this problem.

(iv) *Approximate string matching*: routes are stored as strings or sequences of locations. Approximate string matching is then used to find any route anomalies.

References

[1] W.-C. Hu, "Worldwide: Computer and Device Sales," 2012, http://www.handheldresearch.org/.

[2] F. Hillard, "Digital Influence Study," 2012, http://www.factbrowser.com/facts/4671/.

[3] JiWire, "JiWire Mobile Audience Insights Report Q2 2011," 2011, http://www.factbrowser.com/facts/3196/.

[4] C. C. Chang, E. Jungert, and G. Tortora, *Intelligent Image Database System*, World Scientific, Singapore, 1996.

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

[6] A. Kupper, *Location-Based Services: Fundamentals and Operation 2005*, John Wiley & Sons.

[7] S. Steiniger, M. Neun, and A. Edwardes, "Foundations of Location-Based Services," 2006, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.94.1844.

[8] S. Wang, J. Min, and B. K. Yi, "Location based services for mobiles: technologies and standards," in *Proceedings of the IEEE International Conference on Communication (ICC '08)*, Beijing, China, May 2008.

[9] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining interesting locations and travel sequences from GPS trajectories," in

Proceedings of the 18th International World Wide Web Conference (WWW '09), Madrid, Spain, April 2009.

[10] Y. Zheng, L. Liu, L. Wang, and X. Xie, "Learning transportation mode from raw GPS data for geographic applications on the web," in *Proceedings of the 17th International Conference on World Wide Web (WWW '08)*, pp. 247–256, April 2008.

[11] H. Yoon, Y. Zheng, X. Xie, and W. Woo, "Smart itinerary recommendation based on user-generated GPS trajectories," *Lecture Notes in Computer Science*, vol. 6406, pp. 19–34, 2010.

[12] D. Liu and M. Chang, "Recommend touring routes to travelers according to their sequential wandering behaviours," in *Proceedings of the 10th International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN '09)*, pp. 350–355, Kaohsiung, Taiwan, December 2009.

[13] M. Duckham, S. Winter, and M. Robinson, "Including landmarks in routing instructions," *Journal of Location Based Services*, vol. 4, no. 1, pp. 28–52, 2010.

[14] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi, "Trajectory pattern mining," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '07)*, pp. 330–339, San Jose, Calif, USA, August 2007.

[15] W. Liu, Y. Zheng, S. Chawla, J. Yuan, and X. Xie, "Discovering spatio-temporal cal interactions in traffic data streams," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11)*, San Diego, Calif, USA, August 2011.

[16] G. Myles, A. Friday, and N. Davies, "Preserving privacy in environments with location-based applications," *IEEE Pervasive Computing*, vol. 2, no. 1, pp. 56–64, 2003.

[17] H. Tootell, "Location-based services and the price of security," in *Proceedings of the IEEE Internaional Symposium on Technology and Society (ISTAS '06)*, Queens, NY, USA, June 2006.

[18] L. X. Pang, S. Chawla, W. Liu, and Y. Zheng, "On mining anomalous patterns in road traffic streams," in *Proceedings of the 7th International Conference on Advanced Data Mining and Applications (ADMA '11)*, pp. 237–251, Beijing, China, December 2011.

[19] S. Y. Lee and F. J. Hsu, "2D C-string: a new spatial knowledge representation for image database systems," *Pattern Recognition*, vol. 23, no. 10, pp. 1077–1087, 1990.

[20] P. W. Huang and Y. R. Jean, "Using 2D C^+-strings as spatial knowledge representation for image database systems," *Pattern Recognition*, vol. 27, no. 9, pp. 1249–1257, 1994.

[21] S. Y. Lee, M. K. Shan, and W. P. Yang, "Similarity retrieval of iconic image database," *Pattern Recognition*, vol. 22, no. 6, pp. 675–682, 1989.

[22] T. C. Wu and C. C. Chang, "Application of geometric hashing to iconic database retrieval," *Pattern Recognition Letters*, vol. 15, no. 9, pp. 871–876, 1994.

[23] A. V. Aho and M. J. Corasick, "Efficient string matching: an aid to bibliographic search," *Communications of the ACM*, vol. 18, no. 6, pp. 333–340, 1975.

[24] B. Meyer, "Incremental string matching," *Information Processing Letters*, vol. 21, no. 5, pp. 219–227, 1985.

[25] K. Tsuda, M. Fuketa, and J. I. Aoe, "An incremental algorithm for string pattern matching machines," *International Journal of Computer Mathematics*, vol. 58, no. 1-2, pp. 33–42, 1995.

[26] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning A$*$," *Artificial Intelligence*, vol. 155, no. 1-2, pp. 93–146, 2004.

[27] S. Koenig, M. Likhachev, Y. Liu, and D. Furcy, "Incremental heuristic search in AI," *AI Magazine*, vol. 25, no. 2, pp. 99–112, 2004.

[28] Y. Chen, K. Jiang, Y. Zheng, C. Li, and N. Yu, "Trajectory simplification method for location-based social networking services," in *Proceedings of the International Workshop on Location Based Social Networks (LBSN '09)*, pp. 33–40, 2009.

[29] T. Pavlidis, *Structural Pattern Recognition*, Spring, New York, NY, USA, 1977.