A Linear Route Representation for Route Anomaly Discovery

Wen-Chen Hu and Liang Cheng
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

Lei Chen
Department of Computer Science, Sam Houston State University
Huntsville, TX 77341, USA

ABSTRACT

Mobile computing research evolves constantly and quickly. New mobile devices, technologies, methods, or applications are introduced every day. One of the mobile applications, location-based services (LBS), has attracted great attention recently. 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 pupil does not follow his/her regular route to school. 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. An alert is generated when no matched routes exist. Preliminary experimental results show the proposed methods are effective and easy-to-use.

Keywords: Location-Based Services, Mobile Security, Linear Route Representation, Linear Approximation, and Smartphones.

1. INTRODUCTION

The number of smartphones shipped worldwide has passed the number of PCs and servers shipped worldwide 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. 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. 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 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 AND LITERATURE REVIEW

Traditionally, a travel route is stored as a series of locations (latitude, 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. Related research can be found from the articles [3,5,8,9,10,11,12].

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. Chang, Jungert, & Tortora [2] proposed a 2-D string representation. A matching query may specify a 2-D string, transforming retrieval into a 2-D subsequence matching. A 2D C-string for spatial knowledge representation, which employs a cutting mechanism and a set of spatial operators, was proposed by Lee & Hsu in 1990 [6].

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 [1], 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. The number of state transitions made by the pattern matching machine in processing the text string is independent of the number of keywords.

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 a saleman's delivery routes. This research uses the location information to find any route anomalies, e.g., a pupil does not take the daily route to school. The proposed system is introduced in this section.

The Proposed Steps

This research is to find route anomalies. It is divided into five steps as shown in Figure 1:

i. *Route data collection*: This step collects route data before the application is used,
ii. *Route data preparation*: Raw GPS data is usually not reliable and consistent and includes many noises. It has to be prepared before used.
iii. *Route pattern discovery*: Not all routes are valid, e.g., a very short route is usually not useful. This step puts valid routes into a database and removes invalid routes,
iv. *Route pattern analysis and visualization*: It analyzes the routes and allows users to view the routes on maps, and
v. *Route anomaly detection*: This step is used to find any route anomalies, the theme of this research.

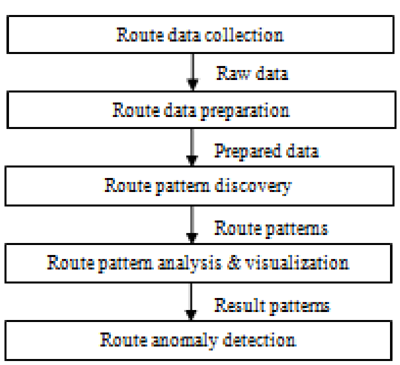
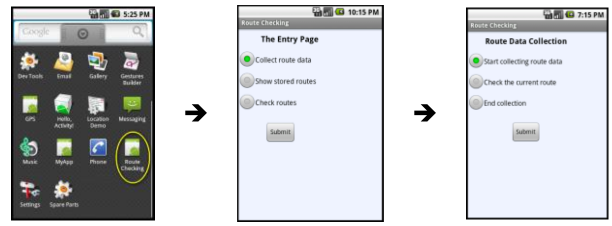


Figure 1. The five steps used by the proposed system



(a) (b) (c)

Figure 2. (a) The application icon "Route Checking" on a device, (b) the entry page of the application, and (c) the page of route data collection

Figure 2.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 7.b, which includes three radio buttons:

- *Collect route data*, which redirects to the interface in Figure 2.c 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.
- *Check routes*, which redirects to the interface in Figure 6.b for route checking after this button is submitted.
- *Show stored routes*, which is used to display the information of the stored routes as shown in Table II, 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. Table 1 shows the basic information about routes. Details of each route are stored elsewhere.

Table 1. Basic route information including route numbers, and start and end times and locations

Route #	Start			End		
	Time	*r* (meter)	θ	Time	*r* (meter)	θ
1	05/26/2010, 18:23:42	8326	236.20°	05/26/2010, 18:45:23	9397	236.22°
2	05/27/2010, 09:31:58	8594	235.34°	05 27/2010, 09:50:41	8526	235.33°
…	…	…	…	…	…	…
n	05/31/2010, 13:13:37	8038	236.86°	05/31/2010, 14:20:37	8976	236.90°

System Implementation

The collected route data is usually raw because GPS data is usually not reliable and consistent and contains many noises [4]. The data needs to be processed before being used effectively. The methods of data preparation include filtering, recovery, restoration, and trajectory. The system then checks the prepared route data and removes invalid routes. This research uses location information to detect route anomalies.

Many methods can be used to find route anomalies. Four kinds of detection are introduced next:

- *Time check*: For example, the Route #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 8:30am. If the student has not arrived at school by 8:30am, then an alert may be generated.
- *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 3.

Figure 3. An example of boxing a route

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

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 [7]. 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*:

```
// Linear Approximating a Human Travel Route

LINEAR_APPROX( ROUTE, p_i, p_j, e )
    // ROUTE: a 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

b ← p_i + e
c ← p_i – e
IS_FIRST ← TRUE
while NUMBER_8_NEIGHBOR(p_j) ≠ 0
    while NUMBER_8_NEIGHBOR(p_j) > 1
        p_j ← BEST_PATH(ROUTE, p_i, e)
        LINEAR_APPROX(ROUTE, p_i, p_j, e)
        print p_i
    ROUTE[p_j] ← CLEAR
    \overline{L_b} ← \overline{bp_j}
    \overline{L_c} ← \overline{cp_j}
    if IS_FIRST
        \overline{L_above} ← \overline{L_b}
        \overline{L_below} ← \overline{L_c}
        IS_FIRST ← FALSE
    else
        if \overline{L_b} is above \overline{L_above}
            \overline{L_above} ← \overline{L_b}
        if \overline{L_c} is below \overline{L_below}
            \overline{L_below} ← \overline{L_c}
        if ∠( \overline{L_above} , \overline{L_below} ) > 0
            print p_i
            b ← p_i + e
            c ← p_i – e
            IS_FIRST ← TRUE
    p_i ← p_j
    p_j ← 8_NEIGHBOR(p_j)
ROUTE[p_j] ← CLEAR
print p_j
```

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 non-unit 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 4 shows an example of order routes and Figure 5 shows the corresponding linear route after applying the algorithm *LINEAR_APPROX* to the route in Figure 4.

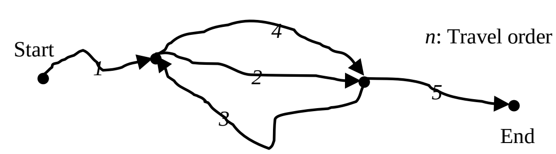


Figure 4. An example of order routes

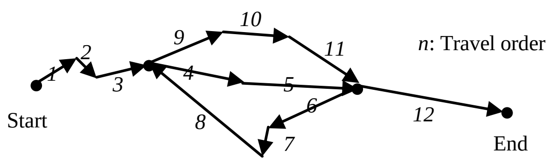


Figure 5. The corresponding linear route after applying the algorithm LINEAR_APPROX to the route in Figure 4

Ordered Routes

For this kind of routes, the order of the locations is relevant, e.g., bus routes. An example of ordered routes is given in Figure 4, where the directional sub-routes are the numbers, 1, 2, 3, 4, and 5. The numbers also imply the travel order, e.g., 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.

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 [$d_0 \le e$ (the allowable error)], then do nothing,
   else if [($d_0 > e$) and ($d_1 \le e$)], 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.

Unordered Routes

For unordered routes, the order of the locations, maybe except the start and end locations, is irrelevant, e.g., newspaper delivery. Figure 4 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, 11, 12} in Figure 5, and bidirectional sub-routes 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 4 includes the following items:

- Start segment: Segment 1,
- End segment: Segment 12,
- Line groups: {3, 4, 8, 9} and {5, 6, 11, 12}, and
- Bidirectional sub-routes: 1↔2↔3, 4↔5, 6↔7↔8, 9↔10↔11, and 12.

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 sub-routes. When an intersection is reached, the algorithm checks all available sub-routes from the intersection.

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 [$d_0 \le e$ (the allowable error)], then do nothing,
   else if [($d_0 > e$) and ($d_1 \le e$)], 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.

The above algorithm does not check whether all sub-routes are visited. It could be easily remedied by adding a checker on each sub-route. Also, the algorithm generates an alert if a sub-route is visited twice because it is unusual to travel a sub-route twice in most cases. It could be easily modified if users do not want it to generate an alert for this condition.

5. EXPERIMENTAL RESULTS

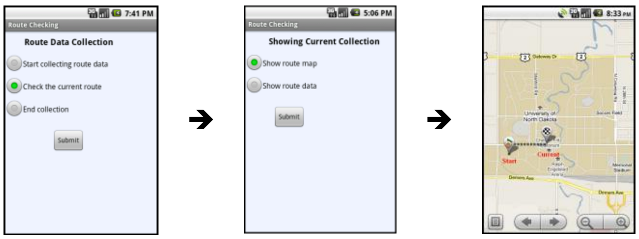
This section gives the experimental results.

Route Data Collection

The first step of this research is to collect route information. The collection could be activated anytime and anywhere. Figure 2.b shows the entry page of the application. After a user picks the button "Collect route data," the system displays the interface in Figure 6.a including three radio buttons:

- *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 multi-threading, so the smartphone can still function as usual.

- *End collection*, which ends the current route data collection.
- *Check the current route*, which is used to check the status of the current data collection including the map as in Figure 6.c and data as in Table 2.



(a) (b) (c)

Figure 6. (a) The page of route data collection, (b) the page of checking the current route, and (c) the current route map

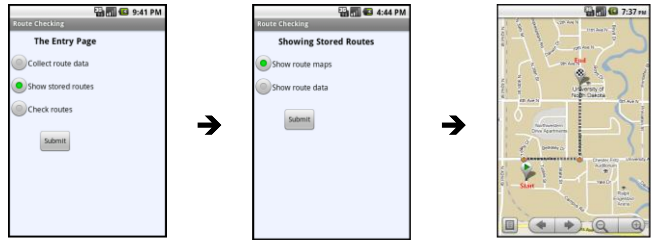
Table 1 shows the basic information about routes. Details of each route are stored elsewhere. An example of the Route #1 details is given in Table 2, where the locations and times are collected periodically, e.g., every minute for walking and every 10 seconds for driving.

Table 2. An example of detail information of a route

# (minute)	Time	*r* (meter)	θ
0	05/26/2010, 18:23:42	8326	236.20°
1	05/26/2010, 18:24:42	8394	236.20°
…	…	…	…
22	05/26/2010, 18:45:23	9397	236.22°

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 [4]. The system also allows users to check the stored routes as in Figure 7.b. Other than showing them the route information as in Tables II and II, users will also like to view the stored routes. One of the routes is shown in the Figure 7.c.

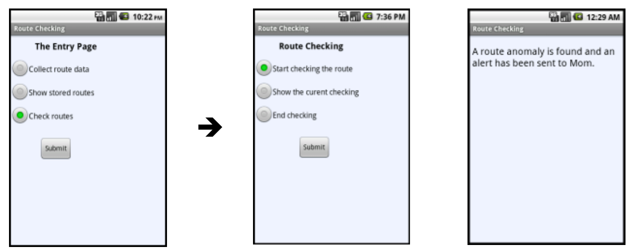


(a) (b) (c)

Figure 7. (a) The system entry page, (b) the page of showing stored routes, and (c) an example of a stored route

Route Anomaly Detection

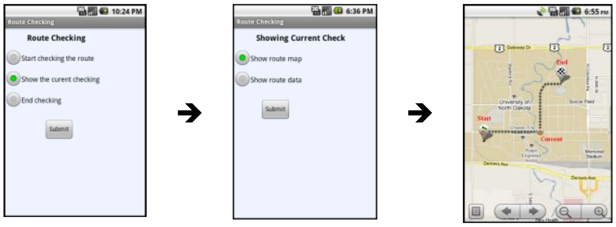
This research uses location information to detect route anomalies. An alert as in Figure 8.c is generated when an anomaly is found. Otherwise, the smartphone just functions as usual. The alert can be sent via an email or a phone call. The interface in Figure 8.b is for checking routes including three radio buttons:



(a) (b) (c)

Figure 8. (a) The entry page of the application, (b) the page of route checking, and (c) an acknowledgment message after detecting a route anomaly

- *Start checking the route*, which runs in background by using multi-threaded programming, so the smartphone can still function as usual.
- *End checking*, which stops the current route checking.
- *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 2 by using the bottom button in Figure 9.b, the system also shows the current position in a possible route as in Figure 9.c by using the top button in Figure 9.b.



(a) (b) (c)

Figure 9. (a) The page of route checking, (b) the page of showing current checking, and (c) the current position in a stored route

The proposed methods are convenient and effective. The execution is also efficient. The times used for the time check and start & 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. Also, the procedure is only used during route collections, but not route checking, which happens more often.

6. CONCLUSION

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. 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:

- *Finite automata*: The collected routes are used to build a finite automaton, which is then used to check any route anomalies.
- *Matrix multiplication*: Similar routes are found by matrix multiplications between the current route and the stored routes.
- *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.
- *Approximate string matching*: Routes are stored as strings or sequences of locations. Approximate string matching is then used to find any route anomalies.

7. REFERENCES

[1] A. V. Aho and M. J. Corasick, "Efficient string matching: An aid to bibliographic search," *Communications of the ACM*, 18(6): 333-340, 1975.

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

[3] Y. Chen, K. Jiang, Y. Zheng, C. Li, and N. Yu, "Trajectory simplification method for location-based social networking services," *Proceedings of the 2009 International Workshop on Location Based Social Networks*, pages 33-40, 2009.

[4] Y. Iijima and Y. Ishikawa, "Finding probabilistic nearest neighbors for query objects with imprecise locations," *Proceedings of the 10th International Conference on Mobile Data Management: Systems, Services and Middleware*, Taipei, Taiwan, pages 52-61, May 18-20, 2009.

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

[6] S. Y. Lee and F. J. Hsu, "2D C-string: A new spatial knowledge representation for image database systems," *Pattern Recognition*, 23(10):1077-1087, 1990.

[7] T. Pavlidis, *Structural pattern recognition*, Spring-Verlag, New York, 1977.

[8] 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 [Accessed: December 12, 2011].

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

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

[11] Y. Zheng, L. Liu, L. Wang, and X. Xie, "Learning transportation mode from raw GPS data for geographic applications on the Web," *Proceedings of the 17th International World Wide Web Conference (WWW 2008)*, pages 247-254, Beijing, China, April 21-25, 2008.

[12] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining correlation between locations using human location history," *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 472-475, Seattle, Washington, November 4-6, 2009.