GBUS - Route GeoTracer

Marta Santos
IST-UTL
Email: marta.santos@ist.utl.pt

Ricardo Lopes Pereira
INESC-ID/IST-UTL
Email: ricardo.pereira@inesc-id.pt

António Brandão Leal
Tecmic
Email: Antonio.Leal@tecmic.pt

***Abstract*—Intelligent Transportation Systems (ITS) are very common nowadays and are adopted by almost every public or private transportation company. To adopt an intelligent transportation system, a company must have a database with all paths travelled by the vehicles and the bus stops' locations of each path. Currently, there is not an automatic solution that allows the user to collect the location of bus stops and its attributes, as well as the bus routes. The system developed, GBUS - Route GeoTracer, enables a single person, equipped with a tablet (or a smartphone) to gather bus routes or variations of existing routes, including bus stop locations, photos and attributes. GBUS consists of a client module and a server side. The client module is a mobile client application, GBUS Mobile Application, which allows the user to collect bus stops and routes from a running vehicle. The server side consists in a server, Route Builder, that receives the information collected and applies a map-matching algorithm to map the collected GPS traces to a digital road map; and a Back-Office Monitoring Application to visualize and edit the information collected. The collected data can then be used to populate a public transportation ITS system, thus reducing the implementation time and cost. GBUS was designed taking into consideration the requirements of Tecmic's XtraN Passenger ITS.**

I. Introduction

Intelligent Transportation Systems (ITS) is the application of Information Technology (IT) to the transportation network to improve safety, productivity, accessibility and mobility. There are several application areas of ITS, one of them being the Advanced Public Transportation Systems (APTS). APTS is the application of ITS in public transit to provide information to its users at home or at stops about vehicles' locations, routes, schedules and prices, and also to improve the efficiency and scheduling of routes. APTS are very common nowadays and are adopted by almost every public or private transportation company. In order to implement these applications, the location of transit stops and the geographic representation of routes are required. Usually this information is not known by the transportation companies and there is no geographic database with this data. In order to adopt an ITS system, the transportation company or the ITS solution provider must collect all the information related to bus stops and routes. This task is usually done manually in the field, in ways which can be very expensive, complex and time consuming. There is not a tool in the market that allows a company to automatically collect the route travelled by each bus as well as locate bus stops and define its attributes.

Tecmic is a company which provides ITS for several types of scenarios. Tecmic's solution for public transportation companies is XtraN Passenger, which enables real-time monitoring of all fleet activity, communication with the driver, arrival time prediction and the provision of information to users in stations, transit stops, and using the Internet or SMS. XtraN Passenger assumes that the location of bus stops, as well as the bus routes are known (have already been collected). However, the transportation companies usually do not keep this information stored in their databases, neither do most ITS providers like Tecmic provide a tool to collect this information.

We propose to expand the XtraN Passenger system by developing GBus - Route Geotracer, a system to automatically collect bus routes and to collect bus stops location and attributes. GBus consists of a client side and a server side. The client side is a mobile application which allows the user to collect the data in the field. The server side consists of a server, the Route Builder, and a back-office monitoring application. The Route Builder provides information and receives data collected with the mobile application. When the user collects a route in a running vehicle, the mobile application periodically captures the vehicle's location, provided by the device's Global Positioning System (GPS) sensor.

GPS sensors are imprecise and the locations obtained usually deviate from the actual locations of the user. The Route Builder server analyses the data received and applies a map matching algorithm to map the collected GPS points onto the road network. The server is also capable of identifying common stretches among the several routes, facilitating route optimization. All the information is then stored in a database.

The server side also includes of a back-office monitoring application to visualize and edit all the information before it is integrated into XtraN Passenger.

This document is organized as follows: Section II describes the previous work in the field, Section III describes the architecture of GBus; Section IV describes the evaluation tests performed and the corresponding results and finally, Section V presents our conclusions.

II. Related Work

The existing Transit Tracking and Transit Information systems assume that the bus stops and routes are known *a priori*. Few are the solutions that automate the process of georeferencing bus stops and collecting and mapping transit routes. The majority of work is divided in two separate areas: data collection, the collection of the geographical location and attributes of bus stops; and map-matching, the process of mapping collected GPS traces into a digital road map.

Bus stop data collection is the process of defining the geographical location of bus stops and its attributes. An attribute is a feature of a bus stop that can be used to represent and to provide additional information about the bus stop (for example, its number, name, photograph, accessibility, etc.). Traditional methods of bus stop collection were entirely manual and involved the use of clipboard, pencil and paper [1]. With the emergence of GPS receivers the field data collection became semi-automated: the coordinates of each bus stop are collected with a GPS unit, while the attributes of each stop are collected in paper. Both kinds of information are then combined in back-office applications.
However, nowadays there are already some automated data collection solutions. Two of these solutions are the Automated Transit Stop Inventory Model (ATSIM)[1], and the ESRI ArcPad[2]. Both applications are intended to be used in the field and allow the collection of bus stops coordinates, photographs, as well as its attributes. ArcPad requires that the user defines all attributes and its corresponding types. On the contrary, ATSIM only allows the customization of 8 of the 64 attributes, providing less flexibility. The main disadvantage of both solutions is that they are intended exclusively for collecting physical locations and attributes, and do not allow the collection of route paths.

A more complete system is EasyTracker, which is designed to provide transit tracking, mapping, and arrival time prediction services with reduced costs and complexity [2]. A smartphone must be installed in all the vehicles and its main function is to periodically send the GPS coordinates of its actual position, creating GPS traces. These traces are then analysed in order to produce route shapes, determine stop locations, and schedules. Single routes are identified based on repetition since it is assumed that buses travel along the same route several times. To extract the bus stops the algorithm estimates the proportion of time spent in any location along the route. Intuitively, buses spend more time in bus stops. However a bus stop may be confused with a traffic signal or a stop sign. In order to correctly identify the bus stops a minimum density threshold is applied. The authors evaluated the performance of the stop extraction algorithm and concluded that the algorithm detects between 85% and 100% of the total number of true bus stops, and only 50% of the total stops identified are true bus stops. According to the authors these results are due to the strong similarity of bus stopping behaviour at bus stops and certain locations, such as intersections, stop signs, traffic signals, and traffic congestion. In addition, some bus stops are rarely used by transit riders which makes them hard to identify. Manual intervention is the solution proposed to solve this problem. Another disadvantage of EasyTracker is the impossibility of defining attributes to bus stops, since the entire process is automated. The collection of bus stops attributes could provide additional information to transit riders and is required by most ITS systems.

Map-matching is the process of correlating positioning data with spatial road network data in order to identify the correct road on which a vehicle is travelling [3] [4]. This process is required due to the error associated with positioning sensors, such as the GPS. Several algorithms using different techniques have been proposed to solve this problem. Existing map-matching algorithms have been classified into four categories [4]:

- Geometric [5]: Geometric map-matching algorithms only make use of geometric information of the road network.
- Topological [6], [7]: Topological algorithms combine geometric information with the topology of the road segments, to match each collected point. These algorithms provide significantly better results than Geometric algorithms.
- Probabilistic [8], [9]: In Probabilistic map-matching algorithms a confidence region is defined around a position obtained from a navigation sensor. This confidence region is defined based on the error of the sensor, and is then superimposed on the road network in order to identify the segments that are inside the region. One of these segments is chosen to match the point usually by using topological information.
- Advanced: Advanced map-matching algorithms are those that use more refined concepts such as Fuzzy Logic [8], [10], a Kalman Filter or an Extended Kalman Filter [11], Dempster-Shafer's theory of evidence [12], particle filters [13], Multiple Hypothesis Technique [14] and an Interacting Multiple Model [15].

In Griffin [16] a different map-matching algorithm is proposed. This algorithm selects some of the collected GPS points and uses online Driving Directions services, such as MapQuest [3] and Google [4], to build the travelled path. The Driving Direction service chosen provides the path by calculating the path to travel between all consecutive GPS points. The selection of points in this algorithm is crucial to its success. If the number of points supplied to the driving directions service is too small, the service will quite possibly return a route that does not match the actual travelled path. On the other hand, if we supply the entire set of points the probability of obtaining an incorrect match is greater since all the points that may cause errors will be introduced. Finally, the success of this matching algorithm is dependent on the correctness and availability of the Driving Direction service selected.

III. Architecture

GBUS consists of a client side and a server side. An overview of its architecture is represented in Figure 1. The client side consists of a tablet application, GBUS Mobile Application. The server side consists of the Route Builder server, the Monitoring Application server and a database.

[1]http://www.ftis.org/atsim.html, last accessed on December 20, 2011
[2]http://www.esri.com/software/arcgis/arcpad/index.html, last accessed on December 20, 2011
[3]http://www.mapquest.com/directions, last accessed on January 7, 2012
[4]http://maps.google.com/, last accessed on January 7, 2012

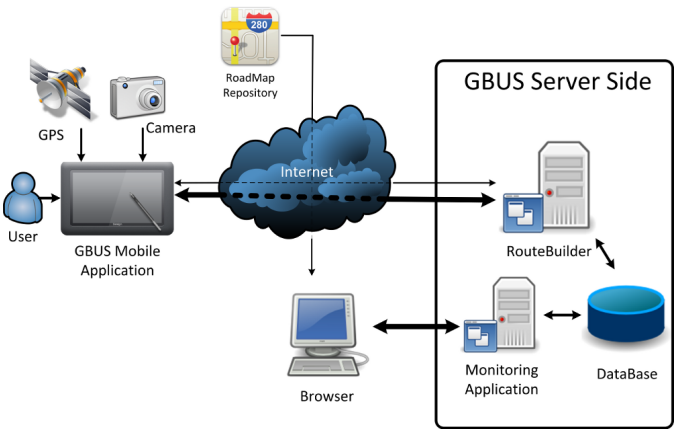

Figure 1. Architectural overview of the GBUS system

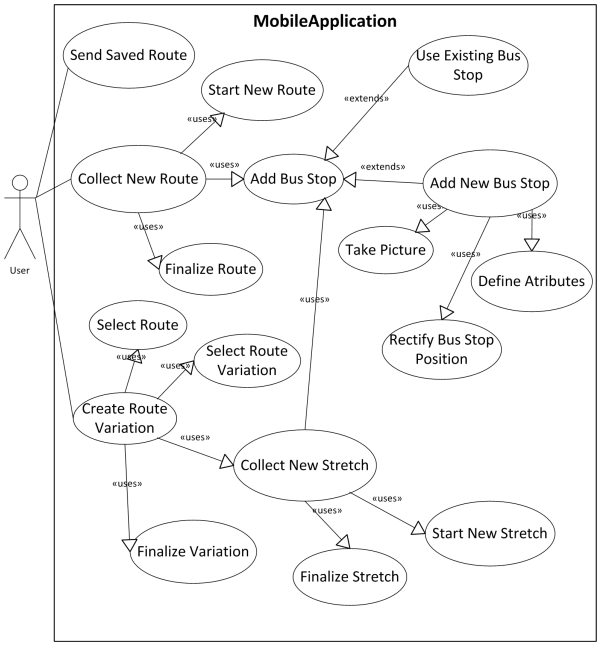

Figure 2. Use case diagram of the Mobile Application

A. Client Side - Mobile Application

The Mobile Application runs in a Android tablet, in a running vehicle, and allows the user to collect new bus stops, routes and variations of existing routes. The actions available to the user are represented in Figure 2. A Bus Route comprises a set of bus stops and a path, which in turn corresponds to a sequence of GPS locations. During the collection of a route, the application accesses the device's GPS to periodically collect the location of the user. The user may also add bus stops to the route being collected. A Bus Stop is a place along the route, where the bus stops allow the passenger to leave or enter the bus. The application allows the user to add an already defined stop or to create a new one at his current location, by introducing its attributes (name, number, etc..). Optionally, the user may take a photograph to attach to the new bus stop, and correct its location on the map to compensate for the GPS error.

The user may also create variations of existing routes. A variation of a route has the same path as the route except in certain parts or stretches. The user creates the new variation by editing the route's path or the path of another variation of the same route. The user may delete parts of the path or add new parts or stretches. When the user terminates the collection of a new route or route variation, the trace file containing the path's GPS points and bus stops is sent to the Route Builder server. To transfer this data we adopted the GPS eXchange Format (GPX)[5]. GPX is a lightweight and open XML data format used to interchange GPS data between applications. GPX is a well documented format, which can be transformed into other file formats, and is recognised by most geographic software. Adopting this format to transfer data in GBUS makes it possible, in the future, to accept data in the Route Builder from other sources or to transfer data from the mobile application to others applications or services. Also, besides being stored in the server database, the data collected is also stored in GPX files at the server, which makes it possible to open or manipulate the data in other applications that recognise the format.

The GPX format has three main different types of data: Waypoints, Routes and Tracks. Waypoints represent points of interest or named features on a map and were used to represent bus stops. Tracks and Routes represent an ordered collection of points. Tracks record where a person has been while Routes are suggestions of paths that the user may follow in the future. Therefore, the routes and route variations in GBUS were represented as Tracks.

B. Server Side

1) Route Builder: The main purpose of the Route Builder is to apply a map matching algorithm to the GPS traces received from the mobile application. As mentioned before, the GPS sensors are imprecise and most of the locations provided are not on a road segment. The algorithms presented in Section II are intended for online map matching (each GPS point is matched as soon as it is collected) and therefore require high performance and correctness. In GBus, the Route Builder performs an offline map-matching, that is, the GPS points are matched after the route is terminated and all points collected. Therefore, in terms of performance and correctness, all the algorithms presented can be used in GBus.

Most algorithms presented required not only GPS locations but also information from Dead Reckoning (DR) sensors. In the GBUS mobile application, the only available information is the user's location obtained from the GPS provider. The map matching algorithm chosen was a topological algorithm proposed by Greenfeld [6] because of its simplicity, correctness, and because it only requires GPS data.

This algorithm consists in two distinct algorithms: Initial Mapping algorithm and Map algorithm. The input to this algorithm is the list of GPS points and bus stops along the

[5]http://www.topografix.com/gpx.asp, last accessed on July 23, 2012

path, in chronological order. The output is a new set of GPS locations describing the path and without errors.

a) Initial Mapping Algorithm: This algorithm is used to locate the user somewhere in the road network and is used in three different situations: for the first GPS point; when the time difference between two points exceeds a certain value; and when the Map algorithm is unable to match a certain point. The algorithm proposed uses geometric point-to-point matching to match the first point, P^0 to the closest node A^0 of the road network (see Figure 3). Then the algorithm finds all arcs connected to A^0 and the next point P^1 is matched to one of these arcs. To determine to which arc P^1 should be matched, the algorithm forms a line segment from P^0 to P^1 and evaluates the similarity of this line segment in terms of orientation and proximity to each arc connected to A^0, using the following formula:

$$W = W_O + W_D$$

Where W is the total score, W_O is the weight for similarity in orientation, and W_D is the weight for proximity of P^1 to the line segment. The weights are calculated using the following formulas:

$$W_O = C_O.cos^{n_O}(\Delta_{AZ})$$

$$W_D = C_D - a.D^{n_D}$$

Where C_O and C_D are constants that constitute the maximum score, Δ_{AZ} is the difference between the azimuths of the line segment from P^0 to P^1 and the arc being considered, and n_O and n_D are power parameters. One can control the amount of weight that is given to proximity and orientation, by selecting different values to these parameters. For example, selecting a higher value for C_O than for C_D means that similarity in orientation is more important than proximity. Selecting a higher value for n_D than for n_O means that the weight will decrease more rapidly as P^1 is further away from the arc, compared to the decrease in the weight as the orientation of the lines diverge away from each other. The parameters used in the map matching algorithm are the parameters recommended in [6], and are represented in Table I. After calculating the total score W for every arc connected to A^0, the arc with the highest score is the arc chosen to match P^1.

In the algorithm presented in [6] the angle between the line segment and each arc was also considered to determine the similarity between the arcs. However this angle was only used if the two arcs intersected. Therefore, we found that it was difficult to relate this parameter with the orientation and proximity parameters, and to define how much it improved the similarity between every pair of arcs. Since the implemented algorithm produced good results considering only the orientation and proximity, we decided not to consider the angle between the segments.

Another difference between the algorithm presented in [6] and the algorithm implemented in GBUS has to do with the fact that the location of a bus stop in GBUS is always correct, since the user may rectify its location when it is created in the mobile application using a Google Maps view. When P^0 is the location of a bus stop, the Initial Mapping algorithm performs a point-to-arc matching to find the closest road segment to P^0. Since P^0 is the location of a bus stop, it is correct and the closest road is definitely the correct match. After the algorithm finds the closest road segment to P^0, the Map algorithm is started. Since a typical route always begin at a bus stop, the correctness of this algorithm is highly improved with the inclusion of bus stops, since it guarantees that the first point of the route is matched to the correct road segment. If the first point was matched to the wrong road segment, the map algorithm could continue to match the subsequent points to the same segment which would result in an incorrect path.

W_O	W_D
C_O = 10	C_D = 10
n_O = 5	a = 0.1
	n_D = 1.2

Table I: Parameters used in the map-matching algorithm

b) Map Algorithm: The Map algorithm runs after the Initial Mapping algorithm, and is used to match the subsequent GPS points. For each GPS point P^t, the algorithm forms a line segment from P^{t-1} to P^t and evaluates its similarity to the currently matched segment A^i in terms of orientation and proximity (as in the Initial Mapping algorithm). Depending on the obtained similarity, the algorithm decides whether P^t maps onto A^i. For that, a similarity threshold is defined. If the obtained similarity is above the threshold, P^t is mapped onto A^i. On the other hand, if P^t does not map onto the segment A^i (the obtained similarity is below the threshold), the algorithm calculates the similarity of the line segment from P^{t-1} to P^t to every segment connected to A^i. The algorithm then selects the segment A^{i+1} whose similarity is the highest and is above the threshold. P^t is then matched to A^{i+1}(see Figure 4). If none of the remaining segments has a similarity value above the threshold, the Initial Mapping algorithm is run.

As in the Initial Map algorithm, when P^t is a bus stop, a point-to-arc matching is performed to find the closest road segment, A^{i+1}. P^t will then be matched to this segment. This process allows the algorithm to determine if the last matched segment A^i was correct. This is done by verifying if A^i is connected to the new road segment A^{i+1} (in case the two segments are different from each other). If the segments are not connected, it was impossible for the user to have travelled from A^i to A^{i+1}, which means that A^i was not the correct match. In this case, each point previously matched to A^i is matched to the closest road point in A^{i+1}.

In both algorithms, Initial Mapping and Map, when a GPS point is matched to a certain road segment it is matched to the road's closest point to the GPS point. When the map matching algorithm terminates, the Route Builder saves all the information in the server database.

Apart from applying the map matching algorithm to the received data, the Route Builder also provides information to the mobile application, such as, existing routes, bus stops, etc. This information is required by the mobile application to

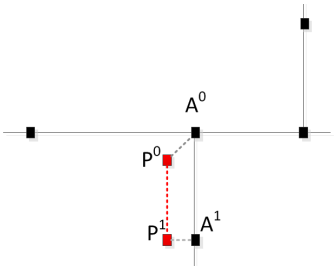

Figure 3. Example of the Initial Mapping algorithm

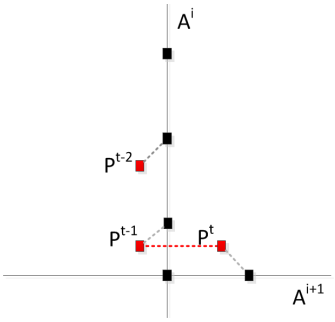

Figure 4. Example of the Map algorithm

perform the actions described in section III-A.

2) Monitoring Application: The Monitoring Application server provides a monitoring web application which allows the user to visualize all collected routes and bus stops. The user is also allowed to change the attributes of bus stops and routes, as well as to correct the routes' paths and bus stops' locations. In order to display this information, the monitoring application accesses the information in the database which was stored by the Route Builder. Also, this application applies the changes done by the user to the database. The actions available to the user are shown in Figure 5.

In XtraN Passenger, the routes are usually described as a sequence of stretches. A stretch is a portion of the route's path, represented by an ordered sequence of bus stops. A common stretch between two or more routes is an ordered sequence of bus stops, common to these routes. Therefore, the stretches that constitute a route may be common to other routes or appear only in this route. Providing information about common stretches between routes may be very useful to improve the efficiency and scheduling of transit routes. For example, it makes it easier to detect whether too many buses travel through the same sequence of bus stops and whether the routes may be re-scheduled to save fuel or time.

The monitoring application allows the user to visualize all stretches in a set of routes. The application starts by finding all common stretches between the selected routes. This is done using string matching. Each bus stop is represented by a character and each route is represented by a sequence of characters, each character being one of the route's bus stops. Each sequence is ordered according to the order in which the bus stops appear in the route. All the sequences are then

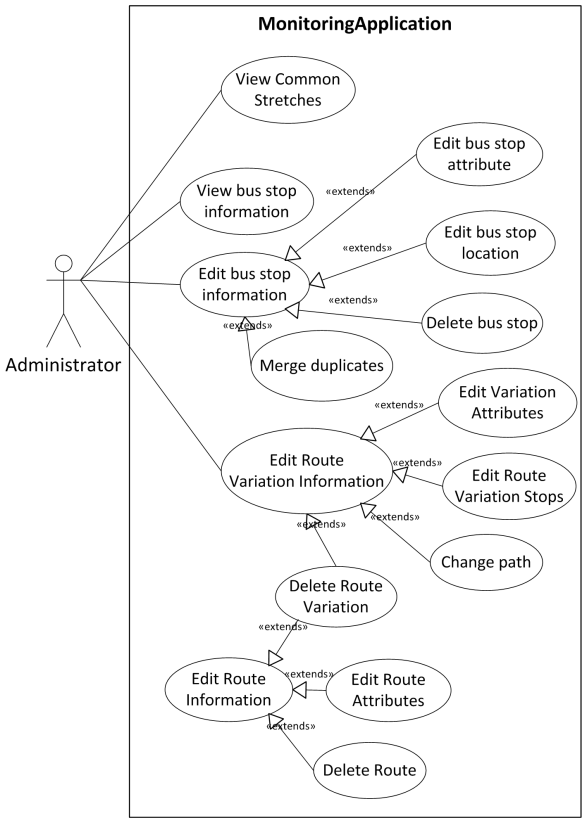

Figure 5. Use case diagram of the Monitoring Application

concatenated in order to create a new string, and the common stretches between the set of routes are found by finding the substrings (sequence of characters) that are repeated in this new string. Since each character represents a bus stop, if the same sequence of characters appears more than once in the new string, it means that more than one route travels through a sequence of bus stops, which is a common stretch between these routes. To find the common substrings, we used Suffix Trees [17]. The suffix tree is built using Ukkonen's method [18] which runs in O(m) time, being m the length of the string. From the suffix tree, we are able to extract the common stretches among the set of variations. The remaining substrings, that is, the sequence of characters (or bus stops) that only appear in one route, are also stretches.

IV. Evaluation

Tests were performed in order to verify the accuracy and precision of the map-matching algorithm. The accuracy, or correct link identification, of the algorithm is the percentage of points matched to the correct road segment. The precision is the degree to which the algorithm provides the same result to repeated collections of the same path.

To execute the map-matching algorithm, geographic road data is required, that is, information about roads and points on the map. We extracted this information from OpenStreetMap[6].

[6]http://www.openstreetmap.org/, last accessed on August 5, 2012

OpenStreetMap allows the download of all underlying map data for small areas, which comes in the form of XML formatted .osm files.

To performed this evaluation, several paths had to be collected in the field using the mobile application. These paths were collected in the small town of Sintra, in Portugal, since it has both areas with high buildings and forestation, where the GPS signals may be blocked. Besides, most paths were collected in residential areas with several roads parallel and close to each other that could lead the algorithm to match the points to the wrong street.

To evaluate the accuracy of the algorithm, several different paths were collected. The collected paths were then sent to the Route Builder which ran the map-matching algorithm for each path. Finally, for each point, it was verified if it was matched to the correct road. The Table II shows the resulting accuracy of the map-matching algorithm. From a total of 772 collected GPS points, 98.4% of these points were matched to the correct road segment. This is a very positive result since the map-matching is done offline, and the user may correct the location of erroneous points with the monitoring application.

Collected points	Accuracy (%)
772	98.4

Table II: Resulting accuracy of the map-matching algorithm

To evaluate the precision of the algorithm, the same path was collected several times. The results of the map-matching algorithm were then analysed to verify if the points from the several paths were matched to the same road segments. The results obtained are shown in Table III. The same path was collected three times and 71.8% of the road segments identified were equal in all resulting paths.

Collected points	Road segments	Common segments	Precision (%)
223	15		
239	16	11	71.80
238	15		

Table III: Resulting precision of the map-matching algorithm

V. Conclusion

In order to adopt an ITS solutions, a transportation company must have a database with all the information about bus stops and routes. We propose a system, GBUS - Route GeoTracer, that enables the collection of bus stops' location and attributes and, at the same time, the automatic collection and map-matching of routes. GBUS consists of a Mobile Application to collect the bus stops and routes; a server, Route Builder, that receives the information collected and applies a map-matching algorithm to map each route to a road map; a database to save the information collected; and a Monitoring Application to visualize and edit all the information. GBUS also identifies common stretches among the collected routes, assisting public transportation companies in increasing the efficiency of their services.

GBUS makes use of a set of existing techniques, such as map-matching and string matching, in order to provide high accuracy and precision, with little human intervention. This system allows the fast and efficient collection of geographical data required to implement an ITS, which can translate into faster and less expensive deployment of ITS.

References

[1] A. Gan and I. Ubaka and F. Cevallos, "An Automated Transit Stop Data Collection System," in *Proceedings of the 2005 Conference on GIS in Transit, National Center for Transit Research*, November 1-3 2005.

[2] J. Biagioni and T. Gerlich and T. Merrifield and J. Eriksson, "EasyTracker: Automatic Transit Tracking, Mapping, and Arrival Time Prediction using Smartphones," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '11. ACM, November 1-4 2011, pp. 68–81.

[3] D. Bernstein and A. Kornhauser, "An introduction to map-matching for personal navigation assistants," Transportation Research Board, Tech. Rep., 1998.

[4] M.A. Quddus and W. Y. Ochieng and R. Noland, "Current map-matching algorithms for transport applications: State-of-the art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 15, no. 5, pp. 312 – 328, 2007.

[5] G. Taylor and G. Blewitt and D. Steup and S. Corbett and A. Car, "Road Reduction Filtering for GPS-GIS Navigation," *Transactions in GIS*, vol. 5, no. 3, pp. 193–207, 2001.

[6] J. S. Greenfeld, "Matching GPS observations to locations on a digital map," in *Proceedings of the 81st Annual Meeting of the Transportation Research Board*, January 13-17 2002.

[7] H. Yin and O. Wolfson, "A weight-based map-matching method in moving objects databases," in *Proceedings of the 16th International Conference on Scientific and Statistical Database Management*, vol. 16. IEEE Computer Society, June 21-23 2004, pp. 437–438.

[8] Y. Zhao, *Vehicle Location and Navigation System*. Artech House, Inc., 1997.

[9] W.Y. Ochieng and M.A. Quddus and R.B. Noland, "Map-matching in complex urban road networks," *Brazilian Journal of Cartography*, vol. 55, no. 2, pp. 1–18, 2004.

[10] S. Syed and M. E. Cannon, "Fuzzy Logic-Based Map-Matching Algorithms for Vehicle Navigation System in Urban Canyons," in *Proceedings of the Institute of Navigation (ION) national technical meeting*, January 26-28 2004.

[11] W. Kim and G. Jee and J. Lee, "Efficient Use of Digital Road Map in Various Positioning for ITS," in *IEEE Symposium on Position Location and Navigation*. IEEE, May 4-6 2000, pp. 170–176.

[12] D. Yang and B. Cai and Y. Yuan, "An Improved Map-Matching Algorithm Used in Vehicle Navigation System," in *IEEE Proceedings on Intelligent Transportation Systems*. IEEE, 2003, pp. 1246–1250.

[13] F. Gustafsson and F. Gunnarsson and N. Bergman and U. Forssell and J. Jansson and R. Karlsson and P. Nordlund, "Particle Filters for Positioning, Navigation and Tracking," in *IEEE Transactions on Signal Processing*, vol. 50, no. 2. IEEE, 2002, pp. 425–435.

[14] J. Pyo and D. Shin and T. Sung, "Development of a map-matching method using the multiple hypothesis technique," in *IEEE Proceedings on Intelligent Transportation Systems*, 2001, pp. 23–27.

[15] Y. Cui and S. Ge, "Autonomous Vehicle Positioning with GPS in Urban Canyon Environments," in *IEEE Transactions on Robotics and Automation*, vol. 19, no. 2. IEEE, 2003, pp. 15–25.

[16] T. Griffin and Y. Huang and S. Seals, "Routing-based map matching for extracting routes from GPS trajectories," in *Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications*, ser. COM.Geo '11, no. 24. ACM, May 23-25 2011, pp. 1–6.

[17] D. Gusfield, *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, 1997.

[18] E. Ukkonen, "On-line construction of suffix-trees," in *Algorithmica*, ser. 14, 1995, pp. 249–260.