A Space Time Alarm

Adrian C. Prelipcean*, Falko Schmid**, Takeshi Shirabe*

* KTH Royal Institute of Technology, School of Architecture and Built Environment, Department of Geoinformatics, Stockholm, Sweden
** Universität Bremen, Cognitive Systems Group, Bremen, Germany

Abstract. Many modern mobile communication devices are equipped with a GPS receiver and a navigation tool. These devices are useful when a user seeks to reach a specified destination as soon as possible, but may not be so when he/she only needs to arrive at the destination in time and wants to focus on some activities on the way. To deal with this latter situation, a method and device called “Space Time Alarm” is presented for helping the user reach the destination by a specified deadline. It does so by continuously and efficiently computing how much more time the user may stay at his/her current location without failing to reach the destination by the deadline. Advantage of this approach is that it works completely in the background so that the user's en-route activities will never be interfered with.

Keywords. alarm, space time, deadline, route improvisation, calm technology

Preferred citation
Prelipcean, A.C., Schmid, F., and Shirabe, T. (2014) A Space Time Alarm. *Progress in Location Based Services 2014*: 187-198, DOI: 10.1007/978-3-319-11879-6_13.

1. Introduction

It is almost unthinkable but what would our modern life be like without clocks? Every morning we would wake up unsure if we are going to make the first shift of work. At a station we would wait for a next train without knowing when it comes. Soon after lunch we would already start thinking about when we call it a day to go to pick up children from school. Actually, all these might be unnecessary worries because none of us (including our employers, train companies, and schools) would have the sense of punctuality. Of course, our circadian rhythms or environmental conditions could still tell us approximately where we are in the day. However, it would be impossible to schedule our daily activities as precisely as we do.

Here is a naïve question: will a time come when we wonder how we could live without knowing our exact location (not just time)? Maybe. With the advance of location detection technology such as global positioning systems (GPS) and mobile computing technology such as smartphones, we now experience, in our everyday life, a wide range of services - commonly known as "location-based services" (e.g. Spreitzer & Theimer 1994, Jiang and Xiaobai 2006, Raper et. al 2007 for overviews) - that are customized according to where we are. Location-based services can be even more sophisticated together with the consideration of time. For example, Raubal et. al (2004) has discussed a use of "time geography" (Hägerstrand 1975, Miller 2005) and "affordance theory" (Gibson 1977) to customize a sequence of activities subject to spatial and temporal constraints. While it may be too ambitious to fully optimize a schedule of all daily chores, but existing technology can already do a lot for us to make decisions concerning both location and time (see, e.g., Abdalla (2012) for his "Geo-Temporal Task Planning Application"). Nowadays, friends can exchange their location coordinates (instead of describing to each other where they are) through their smartphones, and use online mapping and spatial query services to find a coffee shop they can meet at the earliest possible time.

No matter how well a schedule is elaborated, it will be useless if it is missed. This is why the alarm function is a useful addition to clocks. Knowing that the alarm will go off when a specified time is passed, we can focus attention on the task at hand.

In the era when highly precise and accurate measurements of location and time are available to (almost) anyone anytime any-

where, what is a spatio-temporal counterpart to the alarm? To answer this, let us put ourselves in a somewhat familiar situation: we need to leave a hotel room for a university auditorium to give a talk at 10:00 am. We can still use the conventional alarm, but in this case, we need to do some calculation, because where the activity is performed is not where the alarm sounds. So we estimate how long it takes from the hotel to the university and back calculate what time we need to leave the hotel, and set the alarm accordingly. Here we can do the travel time estimation mentally or with assistance of mapping and/or navigation software. There are many online services that help us find a route and estimate its travel time.

Suppose further that the talk has been postponed to 2:00 pm, so that we have some time to explore downtown. Then, with the help of the navigation software, can we set the alarm to a right time to stop our exploration? This may not be as easy as in the previous case (where we are sitting in the hotel room) because we are moving around in the city, which means that it takes different amounts of time to get to the destination from different locations, which, in turn, means that the alarm must go off at different times at different locations. Still, if we do not mind to have the navigation software recompute a route and travel time to the destination every time we change our location (perhaps by a specified distance or longer, as employed in Abdalla 2012). Unfortunately, this would lead to a dilemma: if the recomputation is done frequently, it would cost an exceedingly large amount of computing time/resource; otherwise, the travel time might not be computed in a timely manner. In any case, the alarm would not work timely or reliably. It might be an option to use a system proposed by Shirabe (2011), which continuously labels every alternative move at the current location with the estimated arrival time. However, this might impose too much cognitive load on the user as these labels must be updated rapidly and placed on a limited-size screen. As such, the idea of an alarm that takes into account location is interesting but not without limitations.

In view of the problem described above, we have posed a research question: is there a computationally efficient (with respect to running time and storage space) approach to calculating how long one can remain at one's current location without failing to reach a specified destination by a specified deadline, and if so, how it can or should be utilized by a mobile user. This paper offers an answer by proposing a method and device, called "Space

Time Alarm" (STA), for continuously tracking the user in space and time and alarming when the user has to leave the current (spatial) location in order to reach the destination by the deadline. Like the conventional alarm, it does this while running in the background without needing the user's attention. The remainder of the paper is organized as follows. Section 2 describes how the space time alarm works. Section 3 shows how a prototype of the alarm was implemented. Section 4 discusses the alarm's benefits, limitations, and possible extensions. Section 5 concludes the paper.

2. Methods

2.1. Assumptions and data

One crucial step of STA is the estimation of a user's travel time. This is done based on two assumptions: 1) the user moves along streets, not through buildings or open fields and 2) the user moves at a constant speed. The first assumption implies that the user can turn from one street to another at a street intersection and also turn around anywhere on a street.

The street network is represented in digital form by a graph such that nodes represent street intersections and arcs represent street segments connecting two adjacent intersections. Each arc has an associated value representing the time of its traversal, which is obtained by dividing its geometric length by the user's speed. In this representation, if the user moves from node i to node j along arc(i,j), his/her location is specified by two pointers pointing at arc(i,j) and node i, and a numerical value indicating the travel time from node i to that location.

2.2. Components

STA consists of four general components:

- **data input/output means**, which accepts input from a user, and reads data from the storage means and communicate them to the user in various forms (e.g., visual and audio),
- **data storage means**, which stores relevant data (including the digital street network and user input),

- **data processing means**, which reads data from the data storage means, processes them, and writes results to the data storage means, and
- **location and time detection means**, which detects the user's location and time.

2.3. Computational Steps

STA involves five computational steps, as presented in *Figure 1*. The details of each step are described below.

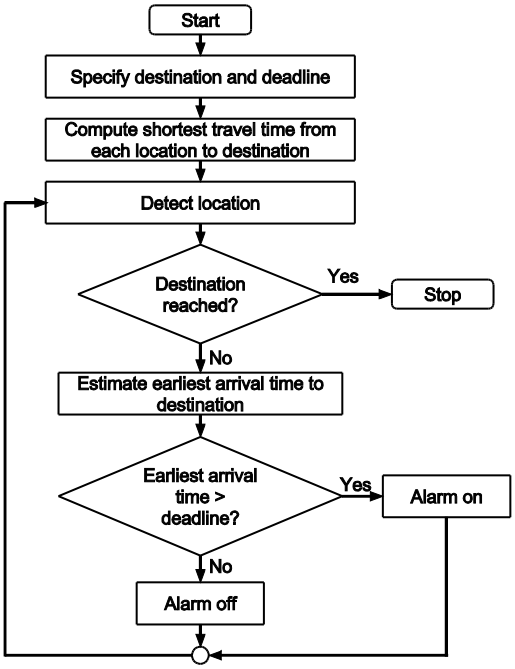

Figure 1. Computational steps of space time alarm

Destination and deadline specification

The data input means accepts input specifying a destination and a deadline through the data input means. Like a conventional alarm, the deadline can be specified in the format of day--hour--minute or the like. Like an existing navigation system, the destination can be specified in the form of a postal address, place name, or geographic coordinate pair or by pointing at its approximate location on a reference map. Any of these inputs is then converted to a node in the digital street network

Shortest travel time computation

The data processing means computes the shortest travel time to the destination from every node in the digital street network. It should be noted that this can be done by running an all-to-one shortest path algorithm (such as Dijkstra's algorithm, which was employed in our implementation as described in the next section) only once. Because relevant nodes are only those from which one can reach the destination by the deadline, the algorithm will be terminated when a node is labeled with a shortest travel time greater than the deadline minus the departure time.

Then the data processing means extracts only a subnetwork that spans the relevant nodes, and stores it in the data storage means. In the subsequent steps, this subnetwork is used instead of the network initially stored.

Location detection

The location and time detection means continuously receives a GPS signal (or any other location sensing data) representing co-ordinate data of the user's location, which is then converted to a point in the digital street network by an algorithm similar to curve-to-curve map matching. The algorithm takes into account an immediate history of the user's locations, as presented by White et. al (2000). From this history, the user's moving direction is determined, too. This process is outlined below with reference to *Figure 2*.

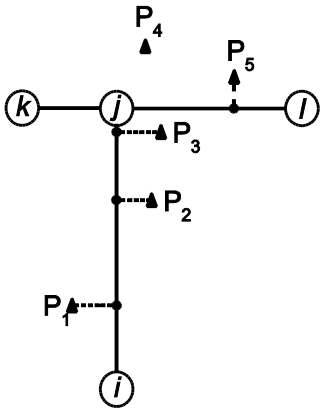

Figure 2. Location detection. The circle represents a node, the line represents an arc, the triangle represents a GPS point, and the dot represents a network point determined by the matching algorithm

A point P_1 is the first point detected by a GPS, and is snapped to arc(i,j), because the arc is closest to P_1 and the distance between them does not exceed a given threshold. Then, a point P_2 is detected by the GPS, and is snapped to arc(i,j) because the arc is closest to P_2 and the distance between them does not exceed the threshold and the movement direction P_1 to P_2 agrees to the arc's direction. Then, a point P_3 is detected by the GPS, and it is not snapped to its nearest arc(j,l) because the movement direction P_2 to P_3 (more or less vertical) would not agree to the arc's direction (more or less horizontal). Instead, P_3 is snapped to arc(i,j) because the distance between P_3 and the arc does not exceed the threshold and the movement directions P_2 to P_3 agrees to the arc's direction. Then, a point P_4 is detected by the GPS but not snapped to its closest arc(i,j) because the distance between them exceeds the threshold. Finally, a point P_5 is detected by the GPS and it is snapped to arc(j,l) because the arc is closest to P_5, the distance between them does not exceed the threshold, and the movement direction P_4 to P_5 agrees to the arc's direction.

This step will be revisited until the user reaches the destination.

Earliest arrival time estimation

At every specified interval, the data processing means estimates the earliest arrival time at the destination by the following simple arithmetic. Suppose that the user is currently at a point p on arc(i,j) as shown in *Figure 3*. Then, obviously, he/she must go through either node i or j to reach the destination. Thus, the earliest arrival time via node j is estimated as t_c + $t(p,j)$ + $d(j)$, where t_c is the current time, $t(p,j)$ is the travel time from p to node j, and $d(i)$ is the shortest travel time from node j to the destination. Similarly, the earliest arrival time via node i is estimated as t_c+$t(p,i)$+$d(i)$. Therefore, the earliest arrival time, $t_e(p)$, from p is given by:

$$t_e(p) = \min(t_c + t(p, j) + d(j), t_c + t(p, i) + d(i))$$

It is important to note that this is a trivial computation, as t_c is given by the location and time detection means at a constant interval, $d(i)$ and $d(j)$ have already been computed and stored in the data storage means, and $t(p,i)$ and $t(p,j)$ are linearly interpolated with p, which has been already detected and stored in the data storage means, along (i,j).

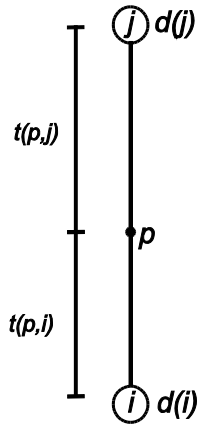

Figure 3. Earliest arrival time estimation. From the current position *p* (represented by a dot) to the destination, it takes *t(p,i)+d(i)* if node *i* (represented by a circle) is visited and *t0 t(p,j)+d(j)* if node *j* (represented by a circle) is visited.

Communication

The data processing means compares the earliest arrival time and the deadline. If the former is greater than the latter, the data input/output means generates an alarm signal (e.g., video, audio, textual, or haptic) and communicates it to the user. Otherwise, it stops generating and communicating the alarm signal.

Optionally, some of the results stored in the data storage means may be also communicated to the user. For example, the earliest arrival time or the time left at the current location may be useful to the user.

3. Implementation

We have implemented STA based on a “full client architecture” (Jing et. al 1999) that connects each of multiple clients with limited storage and computing capability only once to a server with large storage (e.g. for street network data) and powerful computing capability (e.g. for shortest path computation). Having mobile devices as clients, this way allows us to serve multiple users at the same time, *Figure 4* illustrates an overview of the architecture of the current implementation of STA.

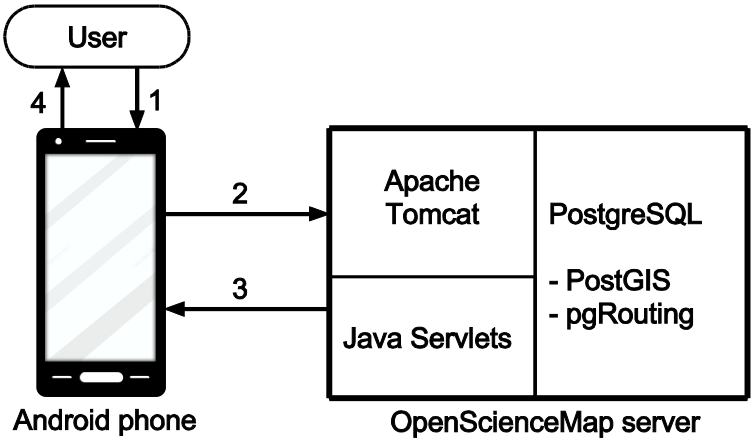

Figure 4. Architecture of an implementation of STA. The OpenScienceMap server (right box) works as the STA server and an Android device as a STA client (phone image), with which a user (top oval) interact.

STA uses the main server of "OpenScienceMap (OSciM)" (Schmid et. al 2013), which plays roles of the data storage means and the data processing means. OSciM is a platform to freely and efficiently offer Android mobiles users mapping and map rendering services. The OSciM server contains a PostgreSQL database management system (DBMS) with PostGIS extension, which stores the complete "OpenStreetMap (OSM)" (Haklay & Weber 2008) data including worldwide street network data and supports the management, query, and analysis of these data. The OSciM server uses an Apache Tomcat web server and implements several Java servlets to efficiently communicate with multiple clients. The PostgreSQL DBMS further contains the pgRouting extension, which specializes in geospatial routing functionality including Dijkstra's shortest path algorithm.

Each client of STA is an Android smartphone or any other Android device, which has a storage medium containing a SQLite DBMS, a central processing unit (CPU), a GPS receiver, and a graphical user interface (GUI), and is able to communicate with the OSciM server. The SQLite DBMS plays a role of the data storage means. The CPU plays the role of the data processing mean. The GPS receiver and the CPU clocks serve as the location and time detection means. The Android GUI serves as the data input/output means.

With reference to *Figure 4*, the workflow of STA is explained below. First, through the Android GUI (*Figure 5*) the user specifies a destination by tapping on the corresponding location on a reference map or typing its address, place name or geographic coordinates in the textbox on the top of the display, which brings the user to the next dialog box in which the date/hour/minute of a deadline is typed (arrow 1). Then, a request for shortest travel time computation together with the specified destination and deadline is sent to OSciM server (arrow 2). In response to the request, the OSciM server runs a servlet requesting the PostgreSQL DBMS to run a shortest path algorithm and extract a subnetwork that spans all nodes from which the destination can be reached by the deadline. The subnetwork is then sent to the Android client using a servlet (arrow 3). The client stores it in the SQLite database and will not contact the server again.

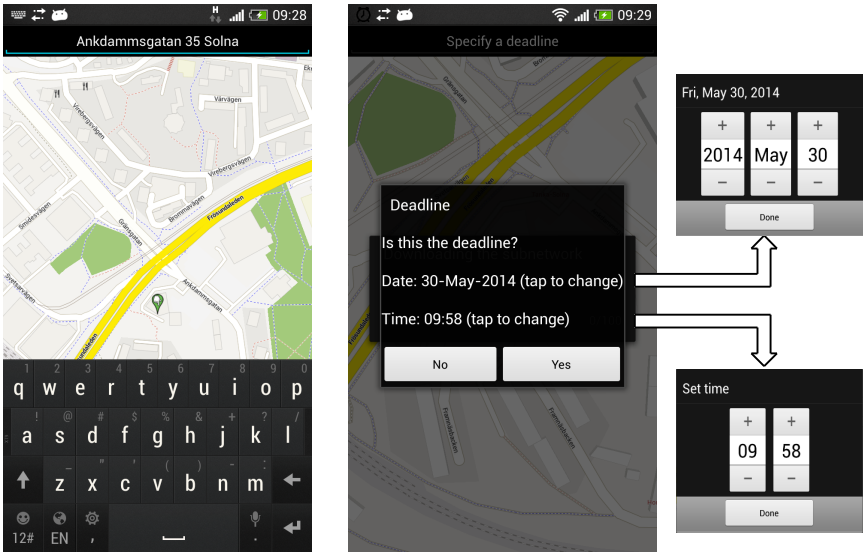

Figure 5. Specification of a destination and a deadline in the STA research prototype. On the left, the map marker represents the location of a specified destination (which, in the present example, corresponds to the address typed in the textbox on top). In the middle, a specified deadline (which is, in the present example, set to 30 minutes from the current time by default) is shown. The date and time of the deadline can be modified through two dialog boxes shown on the right.

The Android client then builds a quadtree indexing structure (Samet 1984) for all the nodes in the subnetwork in order to efficiently perform necessary spatial queries (e.g., retrieval of the node closest to a specified point or retrieval of the node having a specified ID) in the remaining steps. Then, the client's GPS receiver starts tracking the user's location every second. This frequency was chosen as a compromise between the client's battery life and the accuracy in map matching. Each GPS point is then map-matched to a point in the subnetwork. Only the latest map-matched point and the three immediately preceding points are stored in the SQLite database. These points are used to reduce the uncertainty (due to GPS measurement noise) in the user's location and movement direction. Then, the earliest arrival time is estimated using Equation 1 every millisecond (set by default). If the estimated earliest arrival time is later than the deadline, the Android client sounds an alarm sound (arrow 4).

4. Discussion

In this section, we discuss several notable aspects of STA based on our experience with the present implementation of STA.

First of all, STA will not replace existing navigation systems but complement them, as its major purpose is limited to notifying the user of a right time to leave. To see this, imagine a tourist visiting an unfamiliar city. He/she initially uses STA and enjoys sites without distractions, and once STA goes off, he/she switches to a navigation system to take a fastest route and reach his/her eventual destination in time.

Second, although travel time computation relies on the assumption that the user moves at a constant speed (by default 3.6 km/h), this does not mean that STA fails if the user moves at a different speed. If the user slows down (or even stops), he/she loses time, i.e., STA will go off earlier. If the user moves faster (towards the destination), on the other hand, he/she gains time. These have been experienced during our initial test, which is exemplified by the following scenario. In response to our specification of a destination and a deadline, STA calculated that the alarm would go off in 10 minutes. Then we ran for a while and found this number increased to 12 minutes. Then, as we slowed down, the number stopped increasing and eventually started decreasing. Also, we intentionally went to the opposite direction from the destination, and the number decreased. Another inter-

esting observation was that when we went through a building or across a park, we gained time because it had a similar effect to moving faster.

Third, if it is implemented in a full-client architecture, STA works offline as soon as it completes the step of shortest travel time computation (see *Figure 1*). This is because a relevant subset of the digital street network (with each node having its shortest travel time to the destination) is thereafter stored in the client device. A positive consequence is that STA may be used without exposing the client's location (and the user's privacy) to the server.

Fourth, as long as GPS signals are available, STA works without interruptions or delays. Notice that the shortest travel time computation mentioned above is the most computationally expensive step, and it is executed only once when the destination is set. Other steps (such as location detection, estimation of the earliest arrival time and its comparison with the deadline) are done in real time but do not require much computing time or resource.

Finally, the main target users of STA are pedestrians having mobile computing devices such as smartphones. In this mode of transportation (i.e. walking), it is fairly safe to assume that the attribute and topology of the underlying mobility network are static, which means that the shortest travel times initially calculated remain valid until the destination is reached. For STA to accommodate other modes of transportation, however, some computational difficulties must be overcome. In the case of driving, the traffic conditions may change rapidly depending on the time of a day. Every time the digital street network is updated, STA needs to retrieve it from the data server and re-apply the shortest path algorithm to it. If such updates occur too often, STA will fail to alert in a timely manner. In the case of public transportation (e.g., bus and train), STA is expected to work fairly well, as the shortest travel time computation can be done according to published time tables, which are usually reliable. However, the corresponding network data still need to be updated if some unexpected events (e.g., traffic accidents and mechanical troubles) occur. Therefore, the success of the extension of STA depends on how well the shortest travel time (re)computation can be done with real-time network data.

5. Conclusions

We have presented a new location-based service, called Space Time Alarm (STA), for continuously monitoring the location of a user and alerting when the user has to leave the current location in order to arrive at a specified destination by a specified deadline. To do so, assuming that the user moves at a constant speed through a given street network, STA performs a simple logic comprising: 1) accepting a specification of a destination and a deadline and computing the shortest travel time from each node of the network to the destination, 2) tracking the user's location in the network in real time, 3) adding the current time and the travel time from the current location to the destination, and 4) generating an alarm signal if the result exceeds the deadline. Optionally it also shows the amount of time remaining at the current location before it becomes impossible to reach the destination in time. STA can be implemented as a stand-alone mobile application or integrated to an existing navigation system.

STA is expected to benefit any user who has a sufficient amount of time before going to his/her final destination and wants to utilize this spare time to engage in other activities (e.g. shopping and exploration) that require active attention to (and interaction with) the real environment. Once the destination and deadline are set, STA works in the background "calmly" (Weiser & Brown 1996) and lets the user enjoy the en-route activities without the fear of being late. When the alarm goes off, the user is urged to head for the destination, with assistance of a navigation system if necessary.

Although the present implementation of STA is designed for pedestrians only, it is, in principle, possible to extend it to other modes of transportations including driving and public transportation. Care must be taken, however, because this would bring about a greater degree of uncertainty and dynamics in arrival time estimation, which, in turn, would require not just one, but more likely frequent, update of network data and execution of a shortest path algorithm, which would cause a significant amount of computation and thus communication delay especially if STA is embedded in a mobile device.

Finally, an important implication of STA is that while location-based technology may continue to advance in terms of the capability of prescribing spatio-temporal decisions or plans (e.g. route directions) that optimize multiple, possibly conflicting, objectives

and constraints, it can also offer less proactive assistance, which aims to softly influence one's behavior by limiting its intervention to the provision of warnings or suggestions only when undesired outcomes (e.g. being late) are anticipated to happen. The question is not about which of the two capabilities is more important but how to balance them according to the purpose and context and, if necessary, alternate them seamlessly.

Acknowledgments

This project is partially supported by the German Research Foundation (DFG) through the Collaborate Research Center SFB/TR 8, the European Union Seventh Framework Programme --- Marie Curie Actions, Initial Training Network GEOCROWD (grant agreement No. FP7-PEOPLE-2010-ITN-264994).

References

Abdalla, A. (2012) LatYourLife: A Geo-Temporal Task Planning Application. In *Advances in Location-Based Services*. Springer Berlin Heidelberg: 305-325. DOI:10.1007/978-3-642-24198-7_20.

Android. http://www.android.com/. Last accessed on 08/17/2014.

Apache Tomcat. http://tomcat.apache.org/. Last accessed on 08/17/2014.

Gibson, J. J. (1977) The Theory of Affordances. In *Perceiving, acting, and knowing*: 67—82. ISBN 0-470-99014-7.

Hägerstrand, T. (1975) Space, time and human conditions. In *Dynamic allocation of urban space*. Lexington, MA: Lexington Books: 3–14. ISBN 0-347-01052-0.

Haklay M., and Weber P. (2008) Openstreetmap: User-generated street maps. In *Pervasive Computing*, IEEE 7, no. 4: 12-18. DOI: 10.1109/MPRV.2008.80.

Jiang, B., and Xiaobai Y. (2006) Location-based services and GIS in perspective. In *Computers, Environment and Urban Systems* 30, no. 6: 712-725. DOI:10.1016/j.compenvurbsys.2006.02.003.

Jing J., Helal A. S., and Elmagarmid A. (1999) Client-Server Computing in Mobile Environments. In *ACM Computing Surveys*, 31, no. 2: 117-157. DOI:10.1145/319806.319814.

Miller, H. J. (2005) A measurement theory for time geography. In *Geographical analysis* 37, no. 1: 17-45. DOI:10.1.1.65.4624.

PostGIS. http://postgis.net/. Last accessed on 08/17/2014.

PostreSQL. http://www.postgresql.org/. Last accessed on 08/17/2014.

pgRouting project. http://pgrouting.org/. Last accessed on 08/17/2014.

Raper, J., Gartner, G., Karimi H., and Chris Rizos. (2007) A critical evaluation of location based services and their potential. In *Journal of Location Based Services* 1, no. 1: 5-45. DOI:10.1080/17489720701584069.

Raubal, M., Miller, H.J., and Bridwell S. (2004) User-Centered Time Geography for Location-Based Services. In *Geografiska Annaler: Series B*, Human Geography 86, no. 4: 245-265. DOI:10.1.1.196.1642.

Samet, H. (1984) The quadtree and related hierarchical data structures. In *ACM Computing Surveys* (CSUR) 16, no. 2: 187-260. DOI:10.1145/356924.356930.

Schmid, F., Janetzek,H., Wladysiak, M., and Bo H. (2013) OpenScienceMap: open and free vector maps for low bandwidth applications. In *Proceedings of the 3rd ACM Symposium on Computing for Development*. ACM: p. 51. DOI:10.1145/2442882.2442939.

Shirabe, T. (2011) Information on the consequence of a move and its use for route improvisation support. In *Spatial Information Theory*. Springer Berlin Heidelberg: 57-72. DOI:10.1007/978-3-642-23196-4_4.

Spreitzer, M., and Theimer M. (1994) Providing location information in a ubiquitous computing environment (panel session). In *ACM*, vol. 27, no. 5.: 270-283. DOI:10.1145/173668.168641.

SQLite. http://www.sqlite.org/ Last accessed on 08/17/2014.

Weiser, M., and Brown J. S. (1996) Designing calm technology. In *PowerGrid Journal* 1, no. 1: 75-85. DOI:10.1.1.123.8091.

White, C. E., Bernstein, D., and Kornhauser, A. L. (2000) Some map matching algorithms for personal navigation assistants. In *Transportation Research Part C*: Emerging Technologies 8, no.1: 91-108. DOI:10.1016/S0968-090X(00)00026-7.