Voice Maps – the system for navigation of blind in urban area`

ANDRZEJ STEPNOWSKI, ŁUKASZ KAMIŃSKI and JERZY DEMKOWICZ
Department of Geoinformatics, Faculty of Electronics, Telecommunications and Informatics Gdańsk University of Technology
11/12 Narutowicza Street, 80-233 Gdańsk
POLAND
astep@pg.gda.pl kamyk@eti.pg.gda.pl demjot@eti.pg.gda.pl

Abstract: - The novel prototype application of the system supporting the street navigation and independent, outdoor movement of the blind is presented. The system is capable of finding the route from the indicated source to chosen destination, using dedicated digital map and a set of various sensors. Subsequently, the system supports the movement of the blind along the found route. The user's position is obtained with the use of DGPS receiver. In order to further improve accuracy, particle filtering method is used. The system operates on a casual smartphone and communicates with the blind by the touch screen and by the voice messages generated by speech synthesizer.

Key-Words: - Voice maps, street navigation of blind, GPS, smartphone, graph, particle filter, speech synthesis

1 State of the art

It is a problem of great importance to improve the capability of independent, outdoor movement of the blind. Usually, blind people are able to move independently only along the routes which they have already learnt together with a sighted guide. That fact limits significantly their everyday life. Blind people have a strong need to walk around by themselves, even when they are in new location or accidentally lost their way. Nowadays asking a passer-by for help is the only solution, but we have to be aware of the fact, that often there is nobody around ready to help.

The usefulness of GPS as well as digital charts applied for navigation support was discovered years ago. Nowadays, the wide variety of different software and hardware solutions is used in order to make the land and street navigation easier. However, they are mainly used for car navigation or tourist support, and only a few companies or institutions have tried to take advantage of the GPS in order to support the self-dependent moving of the blind.

At present there are practically two products accessible on the market which provide the functionality for supporting the independent, outdoor movement of the blind based on the digital chart of the terrain and the GPS receiver. The first is the "Sendero GPS"[1] developed by the Sendero Group LLC, based on the Braille Note platform and the second is the "Trekker" [2] by HumanWare Group, based on palmtop with dedicated set of buttons – Fig. 1. Those systems are capable of supporting the movement of the blind by finding a walk route from the source to the destination point, assisting the user during his walk, and allowing to explore the set of points of interest (POI) in the surroundings. However, it must be pointed out that:

- Those systems are dedicated mainly for the North American market.
- They use only one sensor type – the GPS receiver – which does not ensure very high accuracy of positioning in urban area.
- Those systems are characterized by very high purchase cost – of a few thousands of USD for a single unit.
- No dedicated direction sensors are used – the GPS receiver can be used to obtain azimuth, but that method is highly inaccurate during very slow movement of the pedestrian.

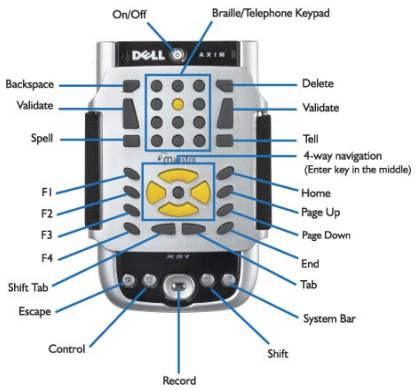

Fig. 1. Image of "Trekker" by the HumanWare Group, showing how to use dedicated buttons [2]

In Poland, the "Nawigator" by MiGRAF is

ISBN: 978-960-474-281-3

used, but its functionality is poor in comparison with "Sendero GPS" or "Trekker". "Nawigator" utilises the GPS receiver data and is able to record in its memory the pedestrian's route, in a form of the coordinates of a set of points defining the path. Then the system assists the blind during his walk along the route, but the prior process of defining the route *in situ* must be performed with help of a sighted person – the guide.

2 System concept and architecture

The system architecture is presented in Fig. 2. The developed system is composed of following parts:

- Street Navigation Supporting System (the main system) responsible for direct, automatic assistance in moving of the blind user,
- Database, responsible for collecting and delivering necessary spatial data.
- Application for "in situ" data acquisition, used by operator that collect precise spatial data,
- OpenStreetMap service, that is used as a data source for area, which haven't been yet visited by operators.

The main system's crucial modules are:

- Spatial Data Cache, that stores the data in the devices persistent memory.
- System Kernel implementing the algorithms for path finding and supporting the user's movement.
- GPS Unit providing the position of the user, using GPS and DGPS receivers, prototype version of inertial unit for places, where signal from GPS satellites is not available and particle filter, which further improves accuracy.
- Compass Unit delivering the user's azimuth, using either built-in or external sensors, mainly magnetometers and electronic compasses (it may depend on platform)
- Dedicated user interface, based on modern smartphones capabilities (touch screen, vibrations, speech synthesis etc.).

The system has a set of data pre-installed in its cache, containing main districts of Gdansk city. As long as the user won't leave that area, the application will work autonomously, and no connection with an external database is required.

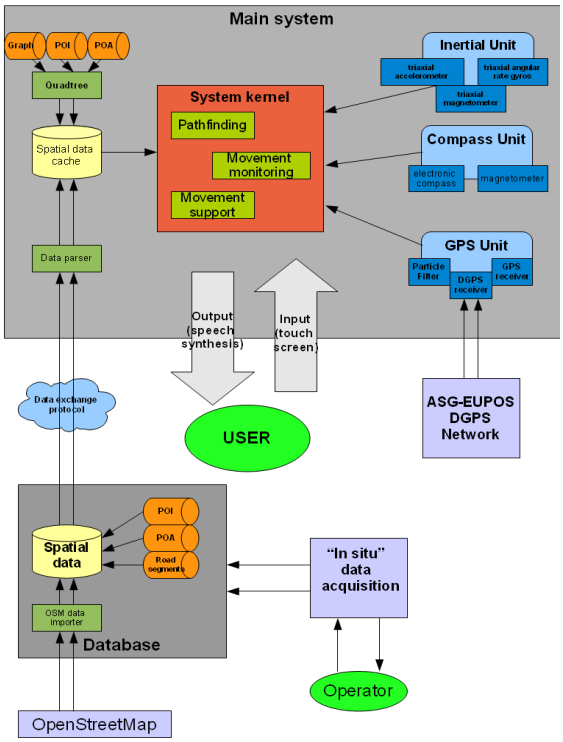

Fig. 2. Architecture of the system

3 Selection of mobile platform

A great impact was put on the portability of the system. The current reports about smartphone market share were precisely investigated and proved, that Java programming language is the most common and wide-spread language used on modern mobile platforms. It's the main programming language of Android and BlackBerry, and can be also used to write applications for all JavaME phones, including Symbian, most of Windows Mobile devices and many less powerful phones with custom operating systems (so-called "feature phones").

ISBN: 978-960-474-281-3

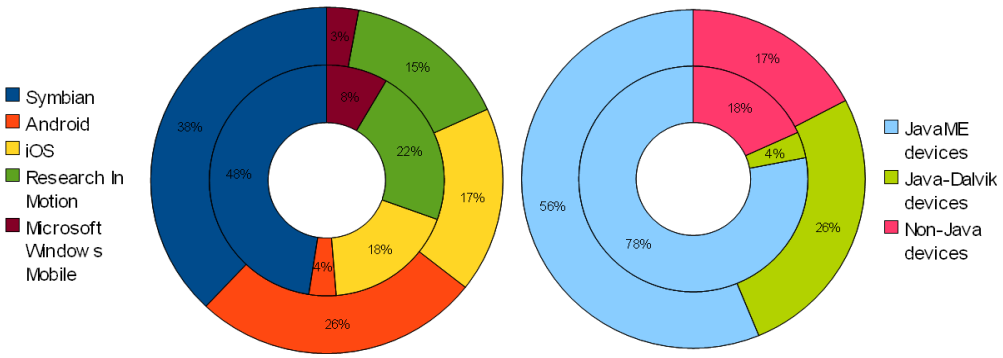

Fig. 3. On the left - smartphone market share between main operating systems. Inner circle shows data from Q3 2009, outer circle data from Q3 2010. On the right – percentage of JavaME, Dalvik and non-Java devices in Q3 2009 and Q3 2010. Other operating systems together have less than 4% of share in global smartphone market[3]

The main problem are the differences between Java-enabled platforms. For example, Android has its own JVM (Java Virtual Machine), called Dalvik, and BlackBerry smartphones use enhanced JavaME platform, with many custom packages and libraries. To solve that issue, we decided two divide the implementation on two parts:

- system kernel, written using constrained set of Java classes, accessible on all smartphone platforms,
- implementation of elements specific to certain platform (user interface, speech synthesis, memory usage, communication with sensors).

After counting lines of code in both parts, we found out, that size of source code of platform specific implementation can be estimated as 10-15% of the size of system kernel's source code, depending on the platform. Therefore, most of the code is written only once, can be tested more precisely and easily and portability of the system is achieved with a relatively low cost.

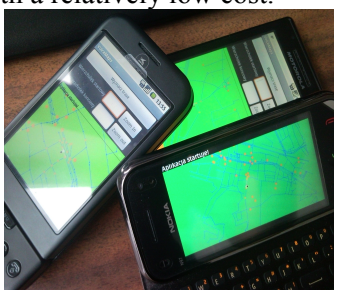

Fig. 4. Prototype application on different smartphones: Nokia n97 (JavaME), Motorola MileStone and T-Mobile G1 (both Android).

4 User interface

The role of user interface is obvious – it helps user to communicate with the system. Of course, the process of communication is bidirectional.

Systems generates instructions using mainly speech synthesis. The implementation of speech synthesis depends greatly on the platform that is used. Android offers the simplest solution – it has build in Text-To-Speech mechanism, which can be easily extended with cheap additional voices (for example, polish voice implemented by SVOX company costs around 3$ on Android Market). TTS API is relatively easy, therefore dynamic speech synthesis was easy to implement.

JavaME platform is more problematic. We were unable to find qualitative mechanism that works without troubles and supports polish voice. Because of that, two alternative methods were prepared. The first one uses remote speech generation. Speech synthesis takes place on the central server, sound files are generated on demand, send using wireless internet connection and then read by the device. That approach was tested with the Ivona Speech Synthesizer and API based on web services (“Software as a Service” technique). The main flaw is the fact, that system looses its autonomy – it needs instant internet connection in order to operate correctly. There is also one more issue – messages are read with additional delay, which in certain conditions may reach a few seconds, what is unacceptable.

The second approach is to generate all the necessary sound files before, keep them in device's memory and read when they are needed. However, it requires more persistent memory and excludes low-end devices. What's more, data files become much bigger, as sound file needs to be generated for every name or attribute. It makes data download or update much more time consuming. The combination of those two methods, where most common sound files

are kept in memory and other generated remotely, behaves better that any of the approaches used separately, but dynamic speech synthesis, available on the Android platform, is without a doubt much better.

Users communicate with the systems using mainly touch screen. The menu is arranged in a form of a tree. The options of the current part of that tree (the part that the user has entered) are placed on the screen, each of them occupies a rectangular area of significant size. User moves his finger on the touch screen, feels vibrations when he leaves one rectangular area and enters another, and hears name of the option that he has just moved to. The last touched option is considered as a selected one. Double taping on the screen will activate selected option, either moving the user to another sub-tree of the menu, or triggering certain action, f.e. activating verbose mode of messages. On the right-bottom corner of the screen there is also "Back" option, which takes the user back to the upper/previous part of the menu. The menu can be also used on devices without touch screens. In that case, arrow keys are used to move around and one of the action keys replaces double tapping.

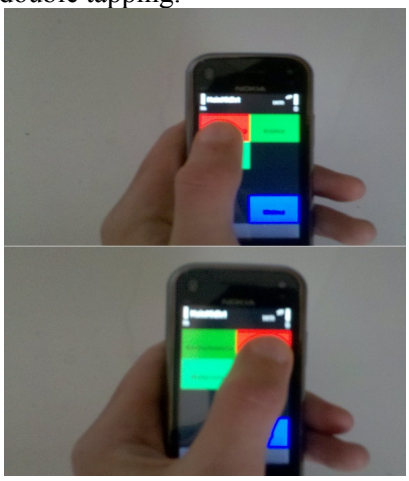

Fig. 5. Touch menu example. User moves from one option to another (selected option is always red). Menu can be easily used with one hand – user do not have to stop using white cane when selecting menu options.

In order to input text, user can use either hardware keyboard, or virtual touch screen keyboard. In both cases, system reads loudly every letter that is written by the user, so that he can identify his own mistakes and correct them quickly. On Android platform there are two additional input mechanism. One utilizes "8pen" [4] approach of fast typing, which is hard to learn, but very efficient if mastered. The second method, implemented by authors, uses Android's mechanism of gesture recognition and prototype version of gesture alphabet, based on the Moon's alphabet.

5 Navigation and user tracking

In our system, provided with spatial data in a form of a graph, we are able to implement the navigation algorithms. The graph of pedestrian paths is sparse, therefore the best way to represent it is to use the adjacency list. The graph is weighted – each edge (representing a path fragment) has some data associated with it, that can be used to calculate the "cost" of the walk from its beginning to its end. The bigger the length of the edge or the narrower path, or the worse the surface – the higher the cost. All the data describing the "difficulty" of the path are used in calculation of the cost of the graph edges in order to prepare the graph for searching the optimal routes.

5.1 Optimal Route Searching

Once the weighted graph is prepared, the shortest path algorithms are used to determine optimal route for the blind user. The most widely known algorithm among the shortest path algorithms family is Dijkstra's algorithm, having the worst case performance of:

$$O(|E|+|V|\log|V|) \quad (1)$$

where $|E|$ is the number of edges, and $|V|$ the number of vertices.

However, the performance of Dijkstra's algorithm is worse than performance of algorithms using heuristics, like A* algorithm [5]. A* calculates the heuristic of the distance between two vertices and prefers the ones which are possibly closer to the goal than others. Therefore, it may omit many vertices that do not need to be examined in order to find optimal route. That is why the performance of the A* is usually much better than performance of Dijkstra's algorithm. What is more, as long as an admissible heuristic is used, which means that the statement (2) below is true, the A* algorithm always computes the optimal route:

$$h(x, y) \leq C(x, y) \quad (2)$$

where x and y are vertices, $h(x, y)$ is the heuristic of the cost of route between them, and $C(x, y)$ is the actual, real cost of the route.

It is very easy to find the admissible heuristic – it may be defined as the straight line distance between two vertices, with the assumption that the route between them has the lowest possible difficulty.

Applying the different approach, it is

ISBN: 978-960-474-281-3

possible to “overestimate” the heuristic, breaking the rule (2). By doing so, we risk that the output of the algorithm will not be optimal, but on the other hand, we achieve further progress in improving the algorithm performance [5].

Some tests were made in order to examine the differences in algorithms speed, along with their correctness. Five algorithms were tested, namely:

- Dijkstra's algorithm (DIJKSTRA),
- A* algorithm with admissible heuristic (ASTAR),
- A* algorithm with heuristic overestimated by 50% (ASTAR_POLT),
- A* algorithm with heuristic overestimated by 100% (ASTAR_2),
- A* algorithm with heuristic overestimated by 300% (ASTAR_4).

The average time required to find the result (Fig. 6) and algorithms' correctness were tested (Table 1. and Table 2.).

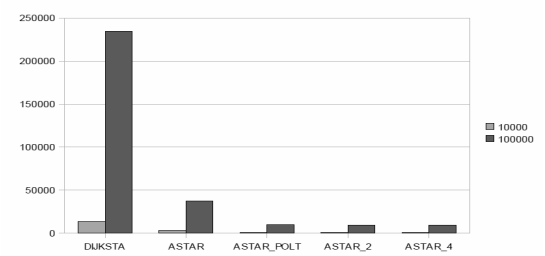

Fig. 6. Average time (in microseconds) required by the tested algorithms to find a path between two points in 10000-vertices (light) and 100000-vertices (dark) graphs

The performed tests have proved that Dijkstra's algorithm has significantly lower performance than the A* algorithms family, and it should not be considered as a base algorithm for finding optimal paths in the system. The non-admissible heuristic causes many wrong answers, so in fact the A* algorithms breaking rule (2) rarely produce an optimal answer. However, as shown in the Table 2, the relative error, may be acceptable, if we consider A* algorithm with the low rate of overestimation.

Table 1. Percentage of non-optimal answers provided by particular algorithms (in %) depending on graph size (in number of vertices)

	100	1000	10000	100000
DIJKSTRA	0	0	0	0
ASTAR	0	0	0	0
ASTAR_POLT	21,92	67,98	85	86,5
ASTAR_2	39,93	81,15	93,2	94
ASTAR_4	44,37	88,2	96,4	97,5

Table 2. Average relative error of particular algorithms (in %) depending on graph size (in number of vertices).

	100	1000	10000	100000
DIJKSTRA	0	0	0	0
ASTAR	0	0	0	0
ASTAR_POLT	0,52	2,89	6,16	7,11
ASTAR_2	1,38	6,58	12,72	15,12
ASTAR_4	3,06	12,68	23,82	29,96

5.2 Supporting the User's Movement

Finding the optimal route from the current or indicated user's position to the destination is an important, but initial step of supporting the blind person's navigation process. The next goal is to help the user walk safely and correctly along the found route. We have applied “the small steps” strategy, guiding the user to the successive route vertices, which usually represent crossings and turns. Movement direction and user's position are monitored continuously, and every significant and mistaken change of user's movement direction or position causes the system alert, and as a consequence, the short voice message is given for the user. It describes the mistake and suggests what to do at the moment to continue walking correctly and safely. Once the next vertex is reached, the following one is chosen to be the current aim, and the system becomes to guide the user to reach it. Main examples of situations occurring during the users’ walk has been shown in Fig. 7.

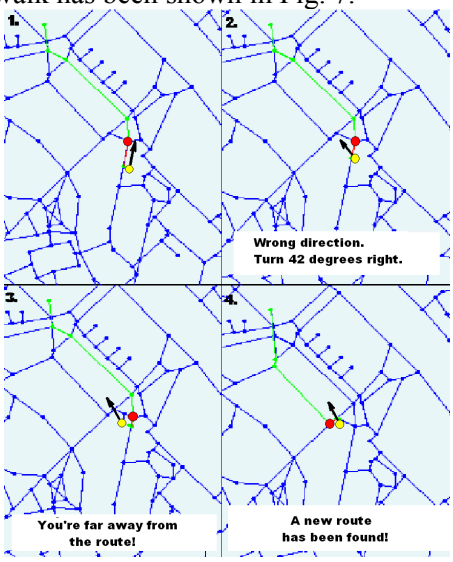

Fig. 7. System's hints depending on the user's behaviour – user moving correctly (1), user with wrong direction (2), user getting far away from the route (3) and user abandoning the route (4), which causes new route search. Yellow dot – the user, red one – the next vertex to reach within the route.

ISBN: 978-960-474-281-3

6 Particle filter

Despite the primary positioning accuracy improvement obtained by the use of connection of DGPS receiver with ASG-EUPOS network, the further accuracy improvement was reached by means of particle filtering of geographic location readouts. Particle filtering is an advanced estimation technique derived from a sequential version of Monte Carlo methods.

This technique generates a large number of candidate solutions (particles) in search for the best solution of a complex problem. The final system state is a weighted average of particles states. With successive iterations of the simulation, some particles assume negligible weights and do not effectively take part in the simulation. Those particles are replaced with the ones that more accurately converge to the optimal solution – this is a so called resampling technique. The algorithm takes into account inaccuracies by introducing measurement errors and DOP information [7].

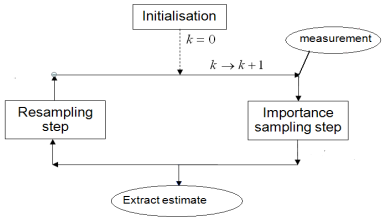

Fig. 8. Particle Filter Diagram

The filter block diagram is presented in Fig. 8. The initialisation, performed only once, sets the initial position and measurement's covariance, which is used in importance sampling step, during weight determination. This step uses DOP information for weight modification.

As a results a better, smoothed position is obtained, as shown in the histogram on Fig. 10. (variance 16.9 m), as compared to the histogram on Fig.9 (variance 27.3 m).

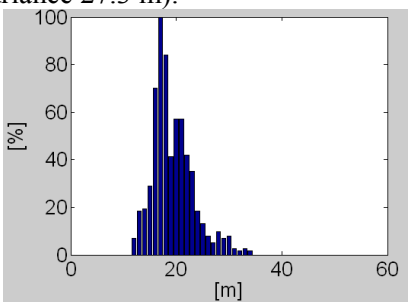

Fig. 9. Sample distance error histogram before particle filtering

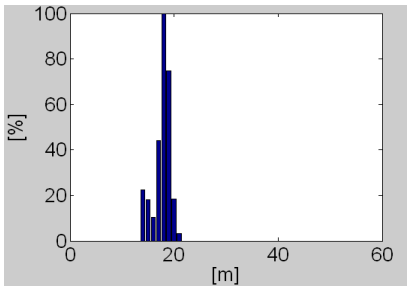

Fig. 10. Sample distance error histogram after particle filtering

7 Conclusion

The prototype application of the system supporting the independent moving of the blind was presented. All the system's modules were described. Proposed solution is under heavy development, but first tests, taken with the help of blind experts and volunteers, have proved that the results are promising.

The plan for the future is to implement next version of the prototype, that can be used by blind people independently, without help and supervision of sighted people, and to collect spatial data of additional polish cities. Once it is achieved, more intensive tests can be launched.

References:

[1] Sendero Group, *http://www.senderogroup.com/support.htm*

[2] HumanWare, *http://www.humanware.com/*

[3] Gartner, *http://www.gartner.com/it/page.jsp?id=1466313*

[4] The 8pen, *http://www.the8pen.com/*

[5] DeLoura, M.A. : *Game programming gems I,* Cengage learning, *2000*

[6] Kamiński, Ł., Łubniewski Z., Kowalik T., Stepnowski A.*:* Voice Maps – portable, dedicated GIS for supporting street navigation and self-dependent movement of the blind, *Lecture Notes of ETI Faculty of GUT - Vol. 18, No 8 (2010), p. 281-286* .

[7] Przemyslaw Baranski, Maciej Polanczyk, Pawel Strumillo, Fusion of Data from Inertial Sensors, Raster Maps and GPS for Estimation of Pedestrain Geographic Location in Urban Terrain, *Metrology and Measurements Systems, 2010*