ENABLING OBSTACLE AVOIDANCE FOR GOOGLE MAPS' NAVIGATION SERVICE

Simeon Nedkov *, Sisi Zlatanova

OTB Research Institute for the Built Environment, TU Delft, Jaffalaan 9, 2628 BX, Delft, The Netherlands – s.b.nedkov@student.tudelft.nl, s.zlatanova@tudelft.nl

Commission IV, WG IV/8

KEY WORDS: Web based, Urban, Planning, Content-based, Hazards

ABSTRACT:

City infrastructures are sensitive to disasters. To aid rescue workers and citizens, a system is needed which determines the shortest route to a certain location, taking the damages of the infrastructure into account. The biggest disadvantage of current navigation systems is that they are "closed" i.e. they are built on top of commercial software packages and as such are only usable by rescue organizations which own licenses for these software packages.

Modern web-technologies provide tools to ease information collection and to facilitate the dissemination of data. Recent successes of crowdsourced platforms such as OpenStreetMap, Ushahidi and Wikipedia, suggest the deployment of the crowdsourcing phenomenon to disaster management. The idea is to let the "crowd" in a disaster area collect information about the state of the infrastructure. People on the street form a highly dispersed network of sensors which is able to provide information in real-time at no cost to the rescue workers.

This paper proposes and implements a method for performing shortest path calculations taking crowdsourced information, in the form of constraints and obstacles, into account. The method is built on top of Google Maps (GM) and uses its routing service to calculate the shortest distance between two locations. Users provide the constraints and obstacles in the form of polygons which identify impassable areas in the real world. The A* pathfinding algorithm is used to guide Google's Directions Service around obstacles.

1. INTRODUCTION

Cities are vulnerable to disasters. Earthquakes, floods and storms can cause severe damage to urban structures. In these kinds of emergency situations, the timely arrival of rescue workers is no longer a function of distance only, but it also depends on the state and accessibility of roads, bridges and tunnels (Cutter et al 2003, Kevany, 2008). To facilitate the navigation of rescue workers and citizens, emergency support systems should provide the shortest route to a certain location, taking the damages of the infrastructure into account (Diehl et al 2005, Zlatanova and Baharin 2008). Such systems are currently available for emergency responders only (Torgt et al 2005, Johnson 2008, Parker et al 2008). These systems are based on commercial or open standards (OGC) technology. The recently developed Dutch system Eagle One (Jacobs et al 2009) is a typical example of commercial solution. The system follows a netcentric approach for information sharing (Boyd et al 2005). It is based on ESRI, Microsoft and Geodan technologies and offers a range of functionalities (drawing, editing, overlays, analysis). Many systems have also been built on basis of OGC web standards (Reichardt and Reed, 2010).

The biggest disadvantage of the above mentioned systems is that they are "closed" i.e. they are built on top of commercial software packages and as such are only usable by rescue organizations which own licenses for these software packages. These solutions are usually complex. Even if made accessible to a larger user community they will require skilled operators to function effectively. As a result, these systems have low momentum: the rescue organization using the system has to take the common GIS steps on its own i.e. they have to gather, analyze and dispatch data/informaion to their users.

Recent successes of crowdsourced platforms such as OpenStreetMap and Wikipedia, suggest the deployment of the crowdsourcing phenomenon to disaster management (Goodchild 2007, Pultar et al 2007). Modern web-technologies provide tools to ease the information collection task and to facilitate the dissemination of data to users. Ushahidi has been actively used for support of several large disasters such as the Pakistan floods of 2011 (http://pakreport.org/ushahidi/) and Japan earthquake 2011 (http://www.sinsai.info/ushahidi/), Google Maps has been used in the Pakistan Earthquake of 2005. Google Maps has been aslo investigated as a tool to assist performing routine works of first responders (e.g. navigation of fire trucks Vozenilek and Zajickova, 2010). However, the functionality of these systems is still very limited. Users can plot information, but they can hardly perform any analysis. Although shortest/fastest route analyses are supported by Google Maps and OpenStreetMap, these are not adapted for emergency situations.

Shortly after a disaster strikes, the most valuable commodity is information. Disaster managers want to know what the severity and extent of the damages is and how quickly should rescue

* Corresponding author.

workers and repair men be sent? Means to collect this information exist and have become quite mature. Once an overview of the damages is created, transportation and logistics become an issue. Which parts of the infrastructure network are still available? What is the fastest route to any given location taking the condition of the infrastructure into account?

Information about the state of the infrastructure can be gathered by rescue workers as they travel through the city. However, a large part of the environment remains unmapped. As a result only a small portion of the road network is used, namely the stretches of road which have been surveyed in the early moments of a disaster.

Information about infrastructure health is vital and, despite its extent, easy to collect: everyone can judge whether a street is accessible for a car or truck. It can therefore be collected by unqualified personnel not directly involved in the rescue efforts e.g. city inhabitants, journalists, idle rescue workers, volunteers, etc. People on the ground can act as a spontaneous and dense sensor network which provides real-time information about the state of road network.

Once this information is gathered rescue workers in possession of powerful GIS software packages as well as users lacking GIS tools should be able to plan their route based on real-time crowdsourced infrastructure health information.

The aim of this project is therefore to investigate a platform, by way of building a prototype, which allows the easy recording of infrastructure blockage information by the "crowd" and is able to use this information to provide routing services. Making this data and the means to collect it available to all through a common and well understood interface will relieve rescue workers from the task of road monitoring, while enabling the public to aid the rescue process.

2. REQUIREMENTS AND OPTIONS

The above vision calls for an open, easy to access, easy to use and light system. An *open* system is here defined as a system which is accessible to a diverse set of people all having different affiliations i.e. a license free system. A system which is *easy to use* is in this context defined as a system which is familiar to users and thus allows them to intuitively work with the system without needing any extensive training or prior knowledge. *Ease of access* is defined as the possibility to use the system regardless of user's location or equipment. The system should work on as many platforms as possible, especially modern smartphones. The best way to achieve this is to build a light web based application. *Light* in this context means that no large software packages are to be installed on the client/rover and that processing and analysis should be performed off the device as much as possible.

The above requirements hint at using a web mapping technology such as Google Maps, OpenStreetMaps, Microsoft Bing Maps, etc. The advent of these web mapping technologies have brought geographical information to the masses teaching them what geographical information is, how to use it and what to expect from it. Building an application on top of these mapping technologies therefore shortens users' learning period. Building the system on top of these web mapping technologies guarantees that users will have access to the most up-to-date information.

In the presented approach, Google Maps (GM) has been chosen because of its wide acceptance. Users are visually accustomed to it, know how to operate and what to expect from it. Furthermore, routing calculations are performed at Google's servers by the so called Directions Service. The Directions Service's input parameters are the starting and ending coordinates of the route and a list of waypoints through which the route must pass. Also, GM's API is optimized for mobile devices. The API is well documented which makes extending the system easy.

3. OVERVIEW AND ALGORITHMS

The system discussed above is implemented using Google Maps built-in functionality which has been extended where needed. Implemented extensions are based on algorithms as described in Worboys 1995.

Shortest route analyses are performed by GM's Directions Service. The Directions Service runs on Google's servers and calculates the shortest route between the provided start and end points. Google Maps does not provide built-in mechanisms to prevent the Directions Service result from intersecting user defined polygons which are representations of real world obstacles. To avoid the obstacle an external pahtfinding algorithm must be implemented. The start and ending points for this algorithms are the intersection points between the route returned from the Directions Service and the obstacle polygon. In this implementation the intersections are found by deploying simple computational geometry algorithms. Ideally, shortest path calculations are graph based. However, Google Maps does not expose its vector data e.g. it is not possible to extract a graph of the streets. Obstacle avoiding shortest path analyses are therefore performed in the raster domain.

3.1 Deployed algorithms

Google Maps does not provide mechanisms to check whether a point is contained by a polygon. It is only possible to perform point-in-bounding-box checks. Point in polygon analyses are used for intersection detection and rasterization of polygons. Checking if a point is contained by a polygon is done using the winding number algorithm. The winding numbers algorithm calculates and sums the angles between a point and all polygon edges. If the summation equals 2 π then the point is said to be in the polygon.

Line intersections are found by applying the side operation. The side operation is used to determine whether a point is located to the left, right or on a line segment. The side test is defined by Eq. 1.

$$side(L, p) = \begin{matrix} 1 \; if \; area(L, p) > 0 \\ 0 \; if \; area(L, p) = 0 \\ -1 \; if \; area(L, p) < 0 \end{matrix} \quad (1)$$

where the area of the triangle formed by the point p and line L is defined as

$$area(L, p) = \frac{L_1 \times L_2 + L_2 \times p + p \times L_1}{2} \quad (2)$$

where L_n = line vertex
p = point

Point p is said to be to the left of line L if the side operation is positive, collinear with L if the side operation is equal to zero and to the right when the side operation is equal to 1. Two lines intersect iff

$$side(L_1, s_2) \neq side(L_1, e_2)$$
$$side(L_2, s_1) \neq side(L_2, e_1) \quad (3)$$

where L_n = is the n-th line
s_n = the start vertex of line n
e_n = the end vertex of line n

So two lines intersect when the start and end vertices of the first line are on both sides of the second line and vice versa.

Shortest path analyses are performed using the A* pathfinding algorithm (Hart et al 1968). Lines are simplified with the Douglas-Peucker algorithm (Douglas and Peucker 1973).

4. IMPLEMENTATION AND TESTS

The process can be split in the following major parts.

1. Obstacle, path and initial route calculation: the user defines the obstacles as well as the start and end points of the route. An initial route is calculated by Google's Directions Service.
2. Route and constraint intersection: intersections between the initial route and obstacles are found using the algorithms described in section 3.1.
3. Shortest path analysis and visualization: the intersections found in step 2 are passed to the A* pathfinding algorithm which finds the shortest path around the obstacle. A new shortest route is requested from Google's Directions Service which is obliged to pass through the points calculated by A*.
4. Result adjustment: the result from step 3 is not perfect and needs minor manual adjustments.

4.1 Obstacle, path and initial route definition

Obstacles are defined by drawing a polygon on the map in a clockwise order. Markers are displayed to identify the polygon vertices. The obstacle creation process is ended by a right mouse click. Next, the user needs to enter the route start and end points. A left click identifies the start point while a right click identifies the end point. The shortest route calculation is performed by the Google Maps' Directions Service.

The Google Maps Directions service takes a begin and end point and calculates the shortest route connecting both points. The Directions Service returns turn-by-turn driving directions. Each turn instruction has a latitude and longitude coordinate. However, the driving instructions' main purpose is navigation. As such, a turn instruction is given only when a turn actually has to be made. It is therefore impossible to predict the number and locations of received latitude/longitude pairs. The route returned from the Directions Service is therefore defined by a list of randomly placed coordinates. For example, long stretches of road will be represented by two coordinates only: one belonging to the instruction stating to get on the road and another to the instruction stating to get off the road, since the driving instruction is of the form 'Turn left on Rotterdamseweg'.

This behaviour makes finding intersections between the route and obstacles difficult as it creates a number of intersection scenarios which need to be treated separately.

4.2 Route and constraint intersection

The aim of the intersection detection procedure is to find the two vertices which lie just outside the obstacle polygon. These will function as start and end point for the A* pathfinding algorithm.

Figure 1 identifies the different intersection possibilities between the route and the obstacle polygon. The polygon is represented by the black area. Its bounding box is also given in the figure. The route segment coordinates returned by the Directions Service are represented by the diamonds and white dots in the polygon.

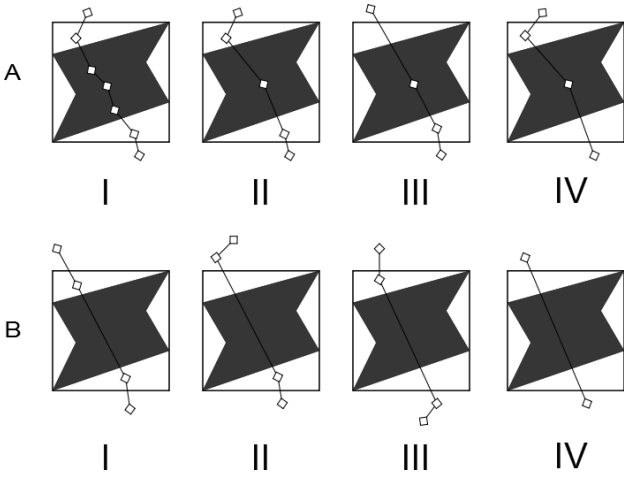

Fig. 1 Intersection modes between the obstacle polygon and route returned from the Directions Service

Category A is characterized by the presence of a route vertex in the polygon. Category B is characterized by the the presence of one or more vertices in the polygon's bounding box but none in the polygon. Case B.IV is special as no vertices are present in the bounding box but the segment does intersect the polygon. Two different intersection techniques are used for both cases.

Category A: The algorithm starts by checking whether any of the returned route vertices (the diamonds in Fig. 1) lie within the bounding box of the obstacle. This is determined using Google Maps' built-in LatLngBounds object's *contains()* function. For all vertices which intersect the bounding box a point-in-polygon test, as described in section 3.1, is performed. Cases A.I-A.III are handled in the same way. First, the first vertex which lies inside the polygon is found. The previous vertex is then set to be the A* starting point. The end point is set to be the first point which is not contained by the polygon. Case A.IV is a a variation on the previous cases since the end point lies outside the bounding box.

Category B: A different approach is needed for category B since no vertices lie inside the polygon but an intersection exists. A point-in-polygon test will not work. Therefore, a line intersection algorithm has been implemented which intersects all route edges with all polygon edges. The line intersection algorithm is based on the side test as described in section 3.1.

In the current implementation cases I-IV are handled in the same way. To optimize the algorithm, only route segments having vertices contained by the bounding box should be intersected with the polygon.

4.3 Shortest path analysis

As mentioned above obstacle avoidance shortest path analyses are performed in the raster domain. Obstacles provided by the user are rasterized. This is done by creating a grid around the polygon and checking which grid cells are contained by the polygon using the aforementioned winding numbers algorithm. Once the intersection points, explained in section 4.2, are found and the obstacle has been rasterized, the A* pathfinding algorithm is used to calculate a path around the obstacle. A result of the A* shortest path algorithm is shown in Fig. 2 (in black).

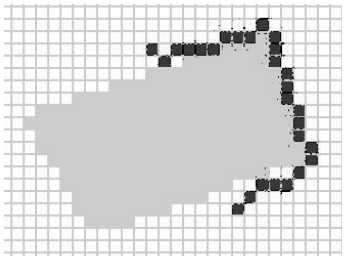

Fig. 2 Result of the A* shortest path algorithm

As can be seen in Fig. 2, the A* algorithm returns many nodes. These are not needed and are done away with using the Douglas-Peucker (DP) simplification algorithm. The sensitivity of the DP algorithm is controlled by a threshold: points which are not significant for the shape of the line are removed. The DP result is used as waypoints for the second Google Maps Directions Service call.

4.4 Visualisation and result adjustment

The A* algorithm has no knowledge about the road network. The returned results will be far from perfect. A certain amount of modification will always be necessary. To facilitate this, the DP result is plotted along side the Directions Service result. Making adjustments to the initial result is done by dragging the DP vertices to appropriate locations (see next section for more details). It is also possible to vary the size of the grid and the DP simplification threshold. Together, these variables control the spacing and amount of waypoints. A larger value for the DP threshold results in less waypoints as only points which are far away from the line connecting the begin and end point. After all modifications have been performed, the user can invoke the Directions Service again to get a new shortest route.

The discussed application is built on top of Google Maps using JavaScript. The GM API is used for defining the obstacles, route begin and end points, thresholds (grid size and DP) and adjusting the initial result.

5. TESTS AND RESULTS

Two examples are discussed in this section. The first example shows the result of a routing request in Delft containing two obstacles. The second example shows the result of a routing request near the bridges of Rotterdam.

Fig. 3 shows the cleaned result of the first example. The DP result is represented by the straight line segments marked by the standard Google markers. In this case these are eight. A Directions Service waypoint is located at every DP vertex. The Directions Service result is the markerless route which snakes through the streets and avoid the two obstacle polygons. The obstacle polygons are represented by the light grey areas.

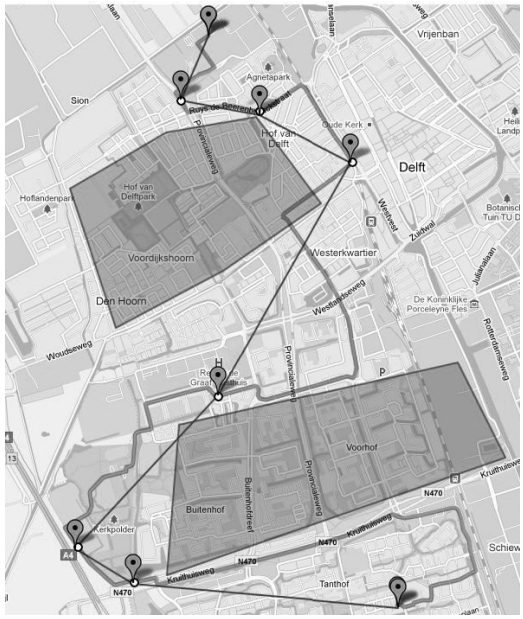

Fig. 3 Cleaned shortest route result

Fig. 4 shows the result presented in the figure above prior to the quick manual adjustment. This initial result is, as explained before, not perfect. A basic understanding of the workings of the system is needed in order to correctly/optimally define the obstacles and improve the initial result (shown below) and obtain a cleaned route (shown above).

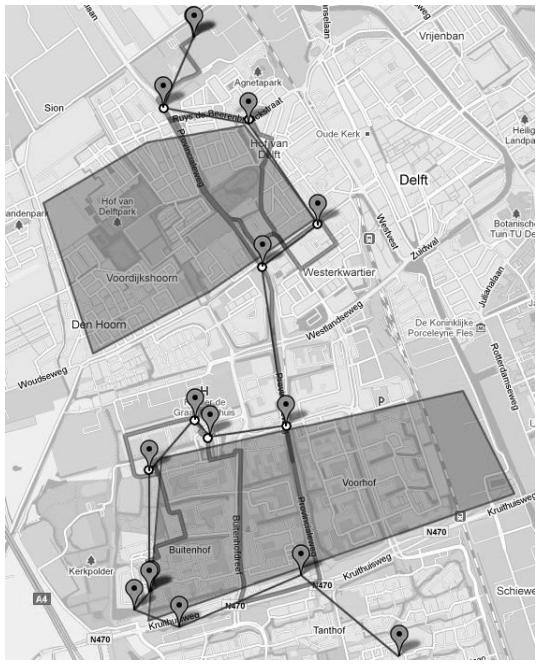

Fig 4. Initial result obtained after step 3 identified in section 4: the route is intersecting both obstacle polygons

When defining obstacles, it should taken into account the way A* works. Obstacle have to be extended to touch (but do not cover) roads which are accessible for travel. The second example illustrates this issue (Fig. 5). The obstacle defined on top of the two bridges on the right is not extended far enough to the left. Since it is not touching the bridge on the left, the DP solution passes over the water to the left of the obstacle. Although the DP result successfully avoids the obstacle, the Directions Service is unable to calculate a path through the supplied waypoints (identified by the markers) as these are located over water. Fig. 6 shows the correct obstacle definition i.e. the obstacle is touching the bridge on the left.

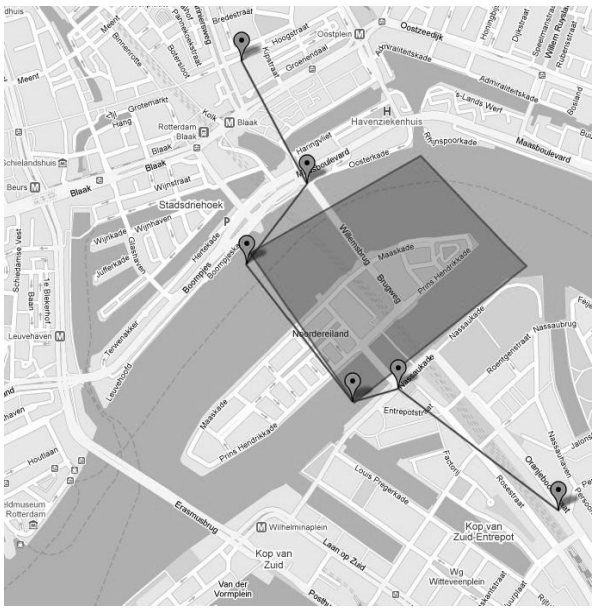

Fig. 5 Demonstration of an improperly defined obstacle

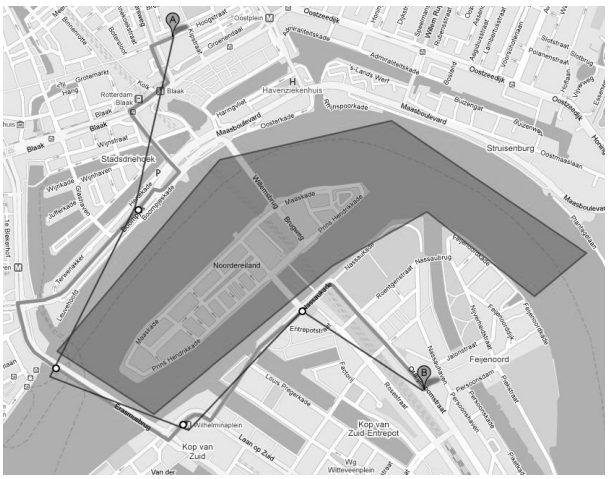

Fig. 6 Properly defined obstacle. The obstacle from Fig. 5 is extended to touch the bridge on the left

The mentioned lack of access to vector data also influences the shortest route result in several different ways. For instance, some DP waypoints may simply fall on the wrong road. This results in a spaghetti like route as shown in Fig. 4. Google Maps automatically snaps waypoints to the closest street. While this is a big advantage (the application would not work otherwise) it tends to cause problems when the DP result snaps to a small one way road instead of the neighbouring high way. Google Maps is aware of street directions and the Directions Service obeys these. Waypoints which happen to be on the wrong side of the road cause the Directions Service to drive twice over the road in order to pass over the given waypoint. Lastly, if the DP threshold is too low i.e. the A* result is not simplified, a lot of waypoints will be returned to the Directions Service. This is troublesome in cities with small streets as the route will be made to go through a large number of them.

6. CONCLUSION

This paper proposes and implements an extension to Google Maps which uses crowdsourced data about the state of the road network to calculate the shortest path between two points. It enables the collection of infrastructure health information by the 'crowd' i.e. rescue workers, journalists, inhabitants, etc by letting them identify blocked roads by drawing polygons on a map. Using this information, the application is able to calculate the shortest path between two points by combining Google Maps' Directions Service with the A* pathfinding algorithm.

Since all pathfinding analyses are performed in the raster domain, the returned solutions will not be perfect i.e. the shortest path around an obstacle will not adhere to the road network as it has no information about it. A certain amount of manual adjustments will always be needed. Some basic understanding of the workings of the system is therefore needed to ensure an optimal definition and positioning of obstacles and modification of the initial result.

The proposed system works best in complex regions. Complexity in this case means a large number of streets, a large area of operations, many spread out obstacles. In these cases it becomes impossible to manually define a route which is optimal in a sense. A* guarantees that its result is the shortest possible path around the obstacle. The Directions Service also finds the shortest route. The obtained route is the result of the stacking of two optimizers. Such a high degree of optimization is difficult to achieve by map observation only.

The system is especially useful when used by people who are not familiar with the layout of the city and the different types of roads. What might look shorter on a map need not be so in reality since, for instance, a shorter route may be slower in terms of time due to a lower speed limit or limited vehicle capacity.

The application is quickly able to find the shortest path given a well constructed obstacles.

7. FUTURE WORK

It is currently not possible to save obstacles and paths for later use. For the system to be accessible to different users it must be possible to save the generated obstacles and paths on a server. Implementing this functionality will open the system to many users which can work together in real-time. Of course, once obstacles have been stored in a database, they can be channelled into any other application.

Rescue workers not familiar with the area will benefit most from the application if they can receive routes created and

checked by someone familiar with the area and system. Therefore, a mechanism needs to be implemented which allows the sharing and sending of calculated routes to specific users.

To ensure the quality of stored obstacles, it would be preferable if all obstacle are checked by an experienced operator prior to publishing, preferably someone who is familiar with the disaster area. Quality checking capabilities can be introduced by way of an obstacle staging area in which all new obstacles are stored and reviewed.

Currently it obstacles do not have any properties. Examples of obstacle properties may be the type of blockage (rubble, mud), inundation depth (in case of flooding, see Mioc et al 2010), vehicle type (blocked for trucks but passable for cars), etc. Introducing obstacle properties will increase the power of the system. Rescue workers will be able to make better decisions.

The interface of the application needs to be polished and extended to allow more thorough control of the objects after they have been created. The current implementation has been constructed as a proof of concept. Little time has been spent on optimizing the interface and making it intuitive to use.

The application can further be optimized by making good use of the Google Maps API as it is specially designed for mobile devices.

REFERENCES

Boyd, C., W. Williams, D. Skinner and S. Wilson, 2005, A Comparison of Approaches to Assessing Network-Centric Warfare (NCW) Concept Implementation, in Proceedings of the international conference 7-9 November Brisbane, Australia, http://www.concepts.aero/system/files/private/NPI-SETE-2005.pdf (last accessed April 2009)

Cutter, S.L., Richardson D. B. and Wilbanks T.J. (eds.) 2003, The Geographical Dimensions of terrorism, Taylor and Francis, New York, ISBN 0-415-94641-7

Diehl, S., Neuvel, J., Zlatanova, S. and Scholten, H. 2006, Investigation of user requirements in the emergency response sector: the Dutch case, in: Second Symposium on Gi4DM, 25-26 September, Goa, India, CD ROM, 6 p.

Douglas, D.H., & Peucker, T.K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization* , *10* (2), 112-122.

Goodchild, M.F., 2007, Citizens as sensors: the world of volunteered geography. GeoJournal 69(4): 211-221.

Hart, P., Nilsson, N., & Raphael, B. 1968, A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* , *4* (2), 100-107.

Jacobs, C. A Riedijk, A Scotta, P. Broojmans and H.J Scholten, 2009, Application of spatial data infrastructure and GIS for disaster management, in Kreek, Rumor, Zlatanova&Fendel (eds.), Urban and Regional Data Management, CRC Press, Taylor & Francis Group, Boca Raton, pp. 277-287

Jafari M, I. Bakhadyrov, A. Maher, 2003, Technological Advances in Evacuation Planning and Emergency Management: Current State of the Art, Technical Report, US Department of Transportation Research and Special Programmes Administration, Available online at : http://www.cait.rutgers.edu/finalreports/EVACRU4474.pdf

Johnson, R. 2008, GIS technology and application for fire services, in Zlatanova&Li (eds.) Geospatial information technology for emergency response, ISPRS book series, Taylor&Francis, London, pp. 351-372,

Kevany, M., 2008, Improving geospatial information in disaster management through action on lessons learned from major events, in: Zlatanova& Li (eds.) Geospatial Information for Emergency Response, Taylor&Francis,London, UK, pp. 3-19

Mioc, D., Nickerson, B., Anton, F., MacGillivray, E., Morton, A., Fraser, D., Tang, P. and Kam, A. 2010 *Early Warning and On-Line Mapping for Flood Events*, Geoscience and Remote Sensing, New Achievements, pp: 147-162 In-Tech, Vukovar

Parker, C.J, R. MacFarlane and C. Phillips, 2008, Integrated emergency management, experiences and challenges of a national geospatial information provider, Ordnance Survey, in Zlatanova&Li (eds.) Geospatial information technology for emergency response, ISPRS book series, Taylor&Francis, London, pp. 275-310,

Reichardt, M. and C. Reed, 2010, Mobolizing ulti-source Geospatial Information for EW and EM: maximize sharing, Enhance flexibility and minimise costs, in Konecny, Zlatanova&Bandrova geographic Information and Cartography for Risk and Crisis Management, Springer, Heidelberg, pp. 191-208

Pultar E., M. Raubal, T.J. Cova, and M.F. Goodchild, 2009, Dynamic GIS case studies: wildfire evacuation and volunteered geographic information. Transactions in GIS 13 (Supplement 1): 85–104

Togt, R., E. Beinat, S. Zlatanova, and H.J. Scholten, 2005, Location interoperability services for medical emergency operations during disasters, in: van Oosterom, Zlatanova & Fendel (Eds.), Geo-information for disaster management, Springer Verlag, Heidelberg, pp. 1127-1141

Vozenilek, V. and L. Zajickova, 2010, Cartographic support of fire engine navigation to operation site, in in Konecny, Zlatanova&Bandrova geographic Information and Cartography for Risk and Crisis Management, Springer, Heidelberg, pp. 114-128

Worboys, M. F. 1995, *GIS : A Computer Science Perspective,* Taylor and Francis, London.

Zlatanova, S. and S. Baharin, 2008, Optimal Navigation for First Responders, in: Van der Walle, Song, Zlatanova&Li (eds.), Information systems for crisis response and management, Joint ISCRAM-CHINA, Gi4DM Conference, 4-6 August, 2008, Harbin, China, pp. 529-542