An A*-Based Search Approach for Navigation Among Moving Obstacles

Zhiyong Wang and Sisi Zlatanova

Abstract Finding an optimal route in a dynamic transportation network affected by disasters is a critical problem for emergency response. Although many routing algorithms have been developed, and some of them show the ability to guide first responders around the static damaged infrastructure, there are few efforts devoted to the efficient routes avoiding moving obstacles. Emergencies caused by natural or man-made disasters can result in both static and moving obstacles in a transportation network, which poses a set of serious challenges for researchers in the navigation field. In this paper, we study the shortest-path problem for one moving object to one destination in a dynamic road network populated with many moving obstacles. Existing approaches, which are developed for stationary networks, are incapable of managing complex circumstances where the status of the road network changes over time. We propose a model to represent the dynamic network and an adapted A* algorithm for shortest path computations in the context of moving obstacles. Moreover, this paper presents a web-based application for route planning. It integrates an agent-based simulation tool for both analysis of the dynamic road network and simulation of first responders' movements, and web technologies for enabling the response community to easily and quickly share their emergency plans and to work collaboratively. We provide an experimental comparison of performance with the standard A* algorithm under different circumstances to illustrate the effectiveness of our approach.

Keywords A* algorithm · Navigation · Moving obstacles · Emergencies

Z. Wang (✉) · S. Zlatanova
Delft University of Technology, Jaffalaan 9, 2628 BX Delft, The Netherlands
e-mail: Z.Wang-1@tudelft.nl

S. Zlatanova et al. (eds.), *Intelligent Systems for Crisis Management*, Lecture Notes in Geoinformation and Cartography, DOI: 10.1007/978-3-642-33218-0_2,

1 Introduction

Disaster relief involves in a number of coordinated activities including searching and rescuing survivors, health and medical assistance, food and water distribution, and transferring injuries. Much of the successful and effective relief work relies on the safe and fast navigation. The complexity and dynamics of transportation network under disasters present considerable challenges to technology innovations related to Location-Based Service (LBS) and vehicle navigation. One challenging issue is that not only the position of the vehicle changes over time, but also the road network and its accessibility vary with the development of physical phenomena (flood, plume, fire, etc.) that cause disasters.

In a transportation network affected by disasters, road conditions could change drastically and many factors—such as plumes, landslides and floods—may cause one or more road segments to be unavailable during specific periods of time. For instance, in the context of a chemical plant explosion that results in many moving contaminant plumes, these moving plumes can be considered as obstacles with changing shapes and positions. Figure 1 presents an example of a moving obstacle and an edge *A_B* connected by two nodes *A* and *B*. As shown, the obstacle moves and intersects the edge *A_B* during the temporal interval $[t_2, t_3]$. During this period of time, the edge is temporarily blocked (i.e. out of service). In this case, the emergency response units should not be guided right through the edge during the time when it is affected by the toxic plume. For this purpose, rescue managers may need to know the movement of plumes and the spatio-temporal information of blocks in the road network in order to allow rescue vehicles to pass for quick response through affected areas. Therefore, how to obtain the safe relief route considering dynamic disaster-related information becomes important for emergency managers.

Route planning during disasters is a practical application that, in recent years, has attracted a lot of attention in the navigation field, but more work still needs to be undertaken. References [1, 2] investigate route planning services taking blocked areas or streets into account. Nevertheless since the status of the road network

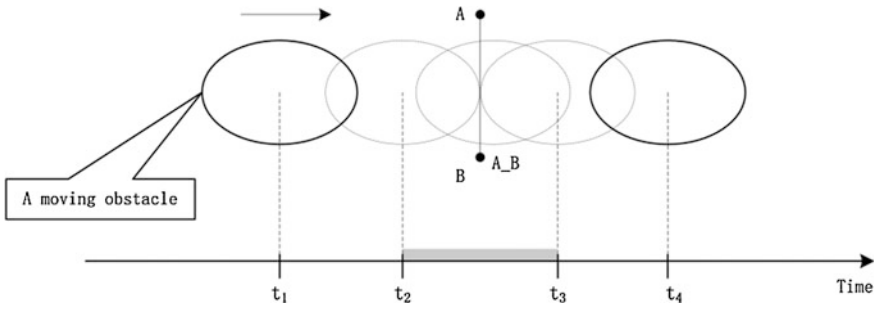

Fig. 1 Example of a block, and its corresponding temporal intervals when the edge *A_B* is blocked by a moving obstacle

varies with the disasters over time, it is necessary to take into consideration the moving obstacles in the path finding process. Similar research on navigation considering moving obstacles has been considerably investigated in the robotic field [3–5], but these related work mostly concern path planning in free space, and do not take into consideration constrains of the real road network. With the advance of disaster modeling and simulation technologies [6–9], some researchers try to incorporate the disaster simulation to improve the routing process. References [10, 11] both study the calculation of evacuation route under the flood disaster, considering vehicle types and the effect of water depth on walking speed respectively. However, they only focus on the routing in the case of flooding, taking its specific characteristics into consideration, which limits their application to other types of disasters, e.g. plumes.

In this research, we examine the problem of finding an optimal path for moving objects (first responders) to avoid moving obstacles. By considering multiplicity of moving objects and destinations, reference [12] tends to classify the navigation problem into 7 categories. With consideration of characteristics of obstacles, this classification can be extended to cases where one/many moving objects have to be navigated to one/many destinations avoiding many static/moving obstacles. In this paper, as the first step of the study of navigation among obstacles, we will focus our discussions on the shortest-path problem for one moving object to one destination in a road network populated with many moving obstacles. Since traditional techniques that are developed for static networks might not be applicable in dynamic scenarios, we propose our algorithm to solve the routing problem with moving obstacles. The proposed algorithm can compute the shortest path between a moving object and its destination in a dynamic network where road segments are blocked by moving obstacles. Because the A* algorithm [13] is generally more efficient than Dijkstras algorithm in terms of the running time and has been widely used in many applications [14–16], we develop the shortest path algorithm based on the classical A* algorithm. We extend the algorithm by incorporating the predicted information of moving obstacles. Besides, we also introduce the waiting option for the rescue vehicle to avoid moving obstacles, minimizing the total traveling time in the meantime. Although waiting has the clear disadvantage of using up precious time, it may be beneficial to allow the vehicle to remain at strategically favorable locations in some circumstances with moving obstacles, which can make their way faster than any other alternative route.

In the following sections, we also present a web-based application for navigation among moving obstacles, providing an effective way to make the shortest path calculation and application accessible to emergency managers and the public. The application can display the affected area on-line and provide the obstacle-avoiding route including waiting options. This application uses an agent simulation tool to compute the shortest path in the road network and to simulate the movement of the responder based on calculated route results. In connection with the simulator, this work also combines web-mapping technology where calculated routes and waiting information are visualized, enabling the response community to easily and quickly share their emergency plans and to work collaboratively.

2 The Model and the Shortest Path Algorithm

In a disaster response, it would be advantageous for the responder to have anticipation of the situation and include that anticipation in the path finding process to respond properly and effectively to challenges presented by the rapidly changing and dangerous environment. To provide a safer and maybe faster route to the destination, another improvement is expected to be made by introducing the possibility to wait in the route determination. For instance, in case of a chemical incident, a toxic plume moves across an edge, blocking the responder's way to the target point E temporarily, as shown in Fig. 2. When the fire truck arrives at grey point of the affected edge, it can either choose to find an alternative route or wait until the edge is available again depending on the time of arrival and changes of road conditions. In some circumstances, waiting at some specific points strategically might be the fastest and safest option. If the moving obstacle can move away from the determined route soon, the fire truck can wait for only a short period of time and continues its way to reach the destination point, saving more time than it would take to follow other alternative routes. To support dealing with changes of environment affected by disasters, we need a model to represent the dynamic information of the road network. The decision of determining whether to wait or not will also be made based on predictions, which can be done by extending the classical A* algorithm.

In following sections, we will first give some definitions and notations, and then discuss the basics of the model that is used to represent dynamics of the road network. According to this model, a modified A* algorithm is presented to incorporate the dynamic data of the road network affected by moving obstacles.

2.1 The Model

Let $G = (N, E)$ be a network consisting of a finite set of nodes N and edges between the nodes in N. For convenience, we denote the edge between two nodes

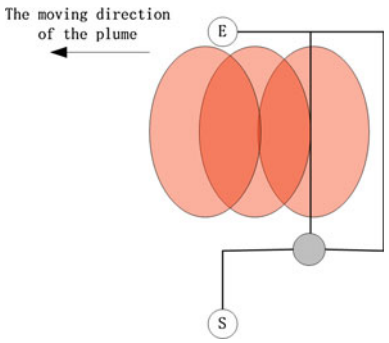

Fig. 2 An example of the waiting option for the responder

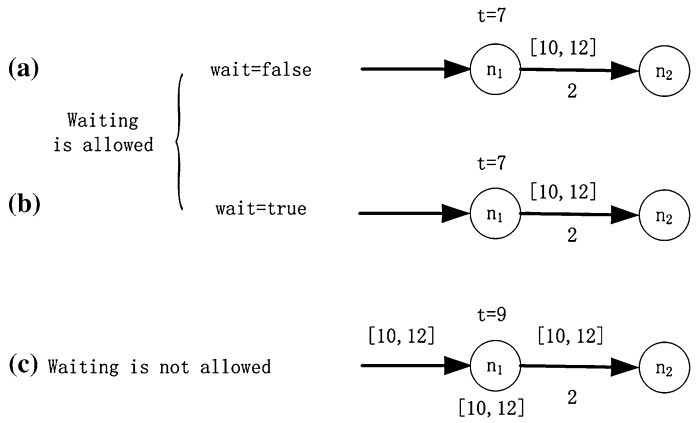

Fig. 3 **a** Not to wait if the object can safely pass through the next edge before the block starts. **b** Wait until the end of the block in the next edge. **c** Waiting is not allowed

u and v by uv. We also assume that each edge has a weight that is a non-negative real number. We denote it by $len(e)$ to represent the length of each edge e. To capture the possible changes of the availability of road segments, additional information is attached to the edges. In our approach, each edge in the road network is assigned a set of temporal intervals which can be represented as follows: $S_{uv} = (ID, b_1, ..., b_k, ..., b_m)$, $uv \in E$, $1 \leq k \leq m$, where ID uniquely identifies the edge uv, $b_k = (t_{ck}, t_{ok}), t_{ck} < t_{ok}$, indicates the time period in which the edge is inaccessible, t_{ck} represents the time when the closing starts, t_{ok} denotes the end time of the temporal block, and m is the total number of blocks in this edge. This allows for the multiple storage of road segment ID with different closing time series. In a similar way, we use $S_u = (ID, b_1, ..., b_k, ..., b_m)$, $u \in N$ to represent and store the state of the node. All these dynamic information of the availability of road segments in a disaster area are obtained through intersection computation between the road network and obstacles in the form of moving polygons. For simplicity, we do not differentiate points along the edge, i.e. once the edge intersects the obstacle, the whole edge is not available.

2.2 The Modified Shortest Path Algorithm

Following above discussions, we extend the standard A* algorithm by introducing waiting options to compute the shortest path from a given starting time in a dynamic network. The pseudocode of the whole algorithm is given in Algorithm 1. We assume the travel starts at time 0. The time instances when the edge is closed or open again are stored as an amount of time relative to the given starting time. The object moves at constant speed *moveRate*, starting from *startPoint* to *targetPoint*. We set a threshold to decide if the re-computation should be executed

to allow for some beneficial waiting that is not considered in the previous computation. Our main adaptation to the A* algorithm is to introduce the waiting time as an additional cost attached to each edge, as shown in Algorithm 2. We propose three methods to compute the waiting time that needs to be added to the travel cost of the road segment. In the following, we will provide a brief description of how this algorithm works.

Algorithm 1: Modified A* algorithm

```
Initialize startPoint, targetPoint, moveRate, wait=false
Initialize closedSet // The set of nodes already evaluated
Initialize openSet // The set of tentative nodes to be evaluated
while openSet is not empty do
    x = the node in openSet having the lowest f_score value.
    if x = targetPoint then
        if totalTravelTime > threshold && iter < maxIteration then
            wait = true
            go to step 2
        end if
        return results: shortest route, total travel time, and total waiting time
    end if
remove x from openSet
add x to closedSet
for each y in neighbor_nodes(x) do
    if y in closedSet then
        continue
    end if
end for
arriveTime = x.g_score, nextEdgeID = get ID of Edge xy
calculateWaitTime(arriveTime, nextEdgeID) // main adaption
if y not in openSet then
    add y to openSet
    tentative_is_better = true
else if tentative_g_score < y.g_score then
    tentative_is_better = true
else
    tentative_is_better = false
end if
if tentative_is_better = true then
    y.came_from = x
    y.g_score = travelCost
    y.h_score = heuristic_estimate_of_distance(y, targetPoint)
    y.f_score = y.g_score + y.h_score
    x.waitingTime = waitingTime
end if
end while
end for
return failure
```

Algorithm 2: calculateWaitTime(arriveTime, nextEdgeID)

```
tentativeTravelCost = arriveTime + len(nextEdge)/moveRate
if the current node is affected then
    if arriveTime > t_om of the current node then
        if wait && waitLineList.ccontains(nextEdge) then
            WaitingTime = calcWTatEdgesInList(arriveTime, nextEdgeID)
        else
            WaitingTime =calcWTwithoutWaitOption(arriveTime, nextEdgeID)
        end if
    else
        WaitingTime = calcWTwithoutWaitOption(arriveTime, nextEdgeID)
    end if
else
    WaitingTime = calcWTwithWaitOption(arriveTime, nextEdgeID)
end if
travelCost = tentativeTravelCost + waitingTime
```

Algorithm 3: calcWTwithWaitOption(arriveTime, nextEdgeID)

```
if the next edge is affected then
    get blocking intervals of the next edge
    for k = 1 to m do
        if arriveTime <t_ck, then
            if tentativeTravelcost < t_ck then
                waitLineList.add(nextEdge); waitingTime = 0; break
            else
                waitingTime = t_ok − arriveTime; break
            end if
        else
            if arriveTime < t_ok then
                waitingTime = t_ok − arriveTime; break
            else
                waitingTime = 0; break
            end if
        end if
    end for
else
    waitingTime = 0
end if
return waitingTime
```

Algorithm 4: calcWTatEdgesInList(arriveTime, nextEdgeID)

```
get blocking intervals of the next edge
waitingTime = 0
for k = 1 to m do
    if arriveTime < t_ok then
        waitingTime = t_ok − arriveTime; break
    end if
end for
return waitingTime
```

Algorithm 5: calcWTwithoutWaitOption(arriveTime, nextEdgeID)

```
get blocking intervals of the next edge
for k = 1 to m do
    if arriveTime < t_ck then
        if tentativeTravelcost < t_ck then
            waitingTime = 0; break
        else
            waitingTime = inf; break
        end if
    else
        if  arriveTime < t_ok then
            waitingTime = inf; break
        else
            waitingTime =0; break
        end if
    end if
end for
return waitingTime
```

In adaptions of the A* algorithm, we assign each edge a variable, called *waitingTime*, to represent the total waiting time associated with the road segment. The *waitingTime* is calculated based on the time of arrival and the timeframe that the road segment is closed. Since waiting is not always beneficial, two waiting policies are considered: waiting is allowed if the current node is not affected or the object arrives after the time the node is blocked by obstacles, and waiting is not allowed if the object arrives before the node is affected. Depending on the re-computation condition indicated by the boolean flag *wait*, two rules are applied in the first waiting policy: not to wait if the object can safely pass through the next edge before the block starts, and wait until the end of the block in the next edge. We also assume that the accessibility of the edge and the node follows a unidirectional relationship:

(i) If the edge is temporarily not accessible, one or both nodes belonging to this edge can still be accessible.

(ii) If the node is not accessible, all edges connected to this node are not accessible.

Following these assumptions, we present there examples in Fig. 3 to illustrate the calculation of the waiting time using three sub-algorithms 3, 4, 5 respectively. As shown in Fig. 3, there are two nodes n_1 and n_2, and it takes 2 time units for the object to move from node n_1 to node n_2. In example (a), waiting is allowed because the node n_1 is not affected by the obstacle. The object arrives at $t = 7$, can pass through the edge and arrive at node n_2 safely at $t = 9$ before the edge is affected by the moving obstacle. By applying the Algorithm 3, we obtain the waiting time 0, which means the object does not need to wait. The edge n_1n_2 should be stored in the array *waitLineList* and will contribute to the next computation step if re-computation is invoked. In example (b), the scenario is the same as the first one but the object is asked to wait as indicated by $wait = true$, which means that the computed result by the first computation without considering beneficial waiting options exceeds the threshold, and the object should wait at this node. The waiting time calculated by algorithm 4 is $12 - 7 = 5$ and should be included to the re-computation process. In the third example, waiting is not allowed. The object arrives at $t = 9$, but it can not safely pass through the edge because the edge will be closed at 10 before the object arrives at node n_2 at time $9 + 2 = 11$. It can not wait at node n_1 either, since when the object is waiting at the interval $[9, 12]$, the closing will start at 10. Therefore the waiting time is set *inf* by the algorithm 5 to trigger the next computation step. As we obtain the waiting time of each edge, it should be added to the total travel cost of the edge to find the optimal route avoiding moving obstacles. After the first computation, the output total travel time is compared with the threshold to determine whether the calculated route is satisfied. If not, the re-computation will be executed taking some beneficial waiting at edges that are stored in Algorithm 3 into consideration.

3 Case Study

In order to assess the performance of our proposed approach as a feasible solution, we construct a web-based application and present an extensive experimental evaluation of the model and the modified A* algorithm. The application consists two main components: an agent-based simulation tool and a JavaScript web mapping library. The proposed model and the algorithm are realized in the multi-agent simulator, called Mason [17, 18], and are evaluated with a real road network. Moving obstacles are synthetically generated to simulate the disasters and visualized in the form of one or more moving polygons crossing the road network. First responder is modeled as a mobile agent who can follow the route calculated by the path finding algorithm. The road network dataset is obtained from the OpenStreetMap (www.openstreetmap.org) database for our testing. All the needed data are imported into a PostGIS database and are fed into the agent simulation

tool via GeoTools (www.geotools.org). The simulation output data is displayed to users through Openlayers (www.openlayers.org), a tool for exhibiting spatial data on web pages.

As the first step of the case study, we apply our model and algorithm to the basic case that one moving object has to be routed to one static destination, avoiding many moving obstacles. The proposed algorithm has been tested with two different datasets: the urban area of Delft and the Rotterdam downtown area. A comparison with the standard A* algorithm without considering predictions is conducted to evaluate the practical application of our approach in path planning, as shown in Table 1.

3.1 Case 1: Delft

In the case of Delft the standard A* algorithm and our algorithm produce very different routes, which is depicted in Fig. 4. The upper line represents the shortest path through the city center, and the lower line is the route calculated by our algorithm, where points along the route indicates the waiting locations for the responder to avoid moving obstacles. Figure 5 shows the snapshot of the simulation of responder's movement. As shown in Table 1, even though the difference in anticipated travel time in this case is relatively small, the main difference here is that the route R3 calculated by our algorithm considering predictions introduces much less additional waiting time than the route R2 provided by the standard A* algorithm without considering predictions. Apart from slowing their movements, following the shortest path could also endanger responders themselves if they do not include predictions into their routing process. In some situations where moving obstacles are moving faster than the relief vehicle, it may be too late for responders to realize that it would be difficult for them to get out of the dangerous area if they continue with their shortest route.

3.2 Case 2: Rotterdam

We also test our approach in the case of Rotterdam with a different dataset of the road network. As can be seen in Fig. 6, the responder has to cross a river to reach the rescue point, and there are a number of tunnels and bridges between the start point and the target point.

Continuing with our analysis, Fig. 6 shows the routes calculated by the two algorithms and Fig. 7 shows the snapshot of simulation of movements of both the responder and obstacles. As can be observed from Table 1, similar to what is reported in the previous section, the generated routes differ significantly. This is caused by the fact that both moving obstacles block the shortest path, and the response unit, who follows the shortest route, has to wait for a long time until it

Table 1 Calculated results

Distance (km)		Anticipated travel time (min)	Waiting time (min)	Total travel (min)
Case 1	(*Delft*)			
R1	2.11	4.4	X	X
R2	2.11	4.4	13.9	18.3
R3	2.43	5.8	5.4	11.2
Case 2	(*Rotterdam*)			
R1	5.22	8.9	X	X
R2	5.22	8.9	18.4	27.3
R3	6.27	0	0	9.0

Notes

[a] R1: The shortest route calculated by the standard A* algorithm without considering predictions of obstacles

[b] R2: The shortest route calculated by the standard A* algorithm without considering predictions and followed by the object considering obstacles along the route (the distance of R2 equals the distance of R1)

[c] R3: Route provided by the modified A* algorithm considering predictions of obstacles

[d] X means no value

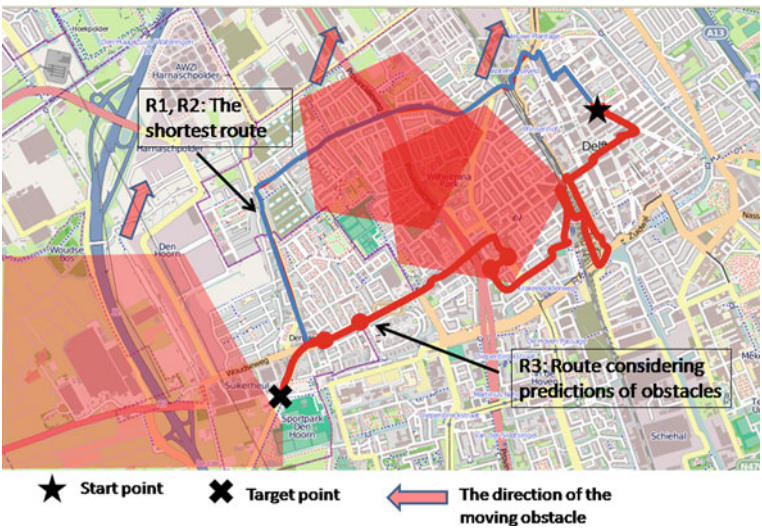

Fig. 4 Route considering predictions of obstacles versus the shortest route

can take the tunnel Maastunnel to reach the destination on the other side of the river. On the other hand, the proposed algorithm considering predictions computes a different and quicker route that crosses the bridge Erasmusbrug, introducing no waiting time and thus achieving a significant improvement of the performance of the response unit.

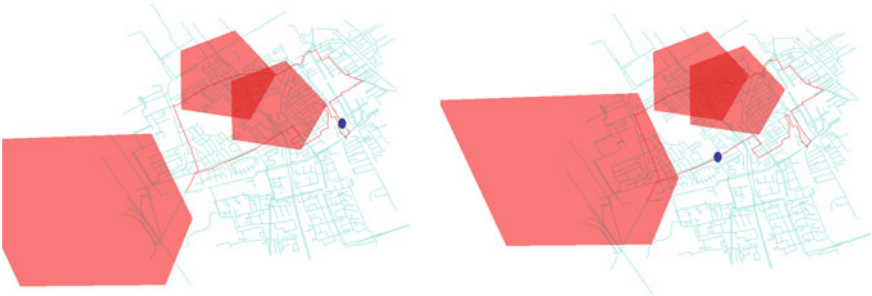

Fig. 5 Snapshot of simulation of movements of both obstacles (in *red*) and the responder (in *blue*)

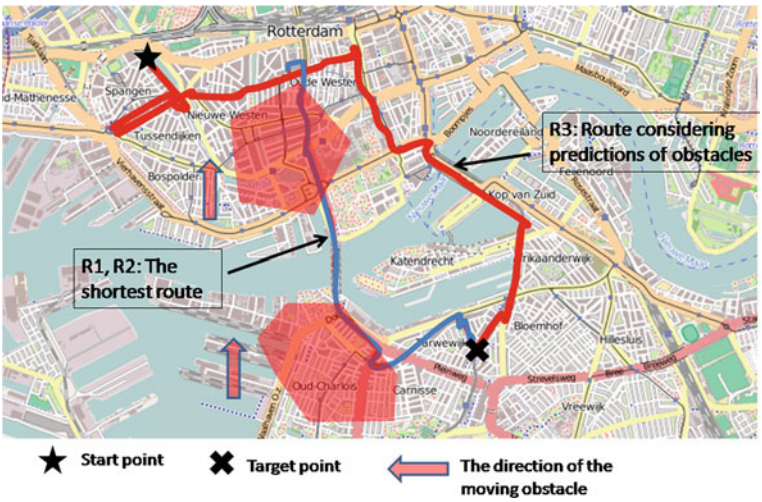

Fig. 6 Route considering predictions of obstacles versus the shortest route

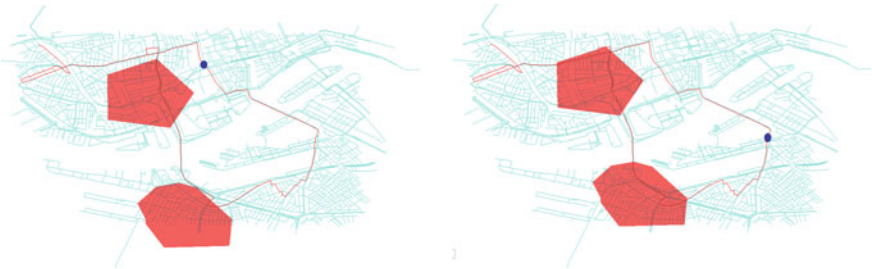

Fig. 7 Snapshot of simulation of movements of both obstacles (in *red*) and the responder (in *blue*)

4 Conclusions and Future Works

Emergency navigation plays a vital role in disaster response and there is a great need for navigation support in the spatio-temporal road network populated by static/moving obstacles. However, despite the considerable amount of route guidance research that has been performed, investigations on navigation among obstacles are still sparse. The paper presents a model to represent a spatio-temporal network, and proposes an algorithm for obstacle-avoiding path computation considering predictions of obstacles' movements. Besides, a web-based navigation system, integrating the agent-based simulation tool and web-mapping technology, has been developed and tested in the real world network. We use both real-life and artificial data sets in our experiments. The real-life datasets are road systems in Delft and Rotterdam, and both are extracted from the OpenStreetMap database. The disaster data sets are artificially generated obstacles that are used to represent the physical phenomena during disasters. We also compare our algorithm with the standard A* algorithm. As demonstrated by experimental results, our approach provides a promising way for navigation among moving obstacles.

In our future work, we plan to adapt the proposed algorithm to deal with changes of the disaster-related information, in particular those that are not predicted for the environment. Due to the difficulty of collecting disaster-related data, there are always some gaps between the real situations and predictions provided by the disaster model, which creates a need for re-routing. A possible approach is to include the range of accuracy of the disaster model in obtaining the dynamic information of the road network to facilitate the re-evaluation of the calculated route. We also would like to introduce variable travel speed into the re-routing process, since the moving speed is an important factor considerably influenced by both traffic conditions and the infrastructure. Another next step would be to explore further some extreme cases (e.g., the obstacle covers the target point during the course of an incident, resulting in no available route until the incident is over) and the problem variants discussed briefly in [19] (e.g., one moving object has to be routed to many static destinations, avoiding many moving obstacles).

References

1. S. Schmitz, A. Zipf, P. Neis, New applications based on collaborative geodata—the case of routing, in *XXVIII INCA International Congress on Collaborative Mapping and Space Technology*, Gandhinagar, Gujarat, India (2008)
2. S. Nedkov, S. Zlatanova, Enabling obstacle avoidance for Google maps' navigation service, in *Proceedings of the 7th Geoinformation for Disaster Management*, Anltalya, Turkey (2011)
3. F. Kunwar, F. Wong, R.B. Mrad, B. Benhabib, Guidance-based on-line robot motion planning for the interception of mobile targets in dynamic environments. J. Int. Robot. Syst. **47**(4), 341–360 (2006)

4. H. Li, S.X. Yang, M.L. Seto, Neural-network-based path planning for a multirobot system with moving obstacles. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **39**(4), 410–419 (2009)
5. E. Masehian, Y. Katebi, Robot motion planning in dynamic environments with moving obstacles and target. Int. J. Mech. Syst. Sci. Eng. **1**(1), 20–25 (2007)
6. F. Darema, Dynamic data driven applications systems: new capabilities for application simulations and measurements, in *Proceedings of the 5th international conference on Computational Science*, Atlanta, GA, USA (2005)
7. X. Hu, Dynamic data driven simulation. SCS M&S Mag. 1 16–22 (2011)
8. P. Pecha, R. Hofman, V. Šmídl, Bayesian tracking of the toxic plume spreading in the early stage of radiation accident, in *Proceeding of European Simulation and Modelling Conference*, Leicester, UK (2009)
9. G. Lu, Z. Wu, L. Wen, C. Lin, J. Zhang, Y. Yang, Real-time flood forecast and flood alert map over the Huaihe River Basin in China using a coupled hydro-meteorological modeling system. Sci. China Ser. E: Technol. Sci. **51**(7), 1049–1063 (2008)
10. D. Mioc, F. Anton, G. Liang, On-line street network analysis for flood evacuation planning, in *Remote Sensing and GIS Technologies for Monitoring and Prediction of Disasters*, ed. by S. Nayak, S. Zlatanova (Springer, Berlin, 2008), pp. 219–242
11. Y. Liu, M. Hatayama, N. Okada, Development of an adaptive evacuation route algorithm under flood disaster. Annuals of Disaster Prevention Research Institute, Kyoto University, vol. **49**, 189–195 (2006)
12. S. Zlatanova, S.S.K. Baharin, Optimal navigation of first responders using DBMS, in *Joint Conference of the 3rd International Conference on Information Systems for Crisis Response and Management/4th International Symposium on Geo-Information for Disaster Management* (2008)
13. P.E. Hart, N.J. Nilsson, B. Raphael, Correction to a formal basis for the heuristic determination of minimum cost paths. SIGART Newslett. **37**, 28–29 (1972)
14. B. Huang, Q. Wu, F.B. Zhan, A shortest path algorithm with novel heuristics for dynamic transportation networks. Int. J. Geogr. Inf. Sci. **21**(6), 625–644 (2007)
15. G. Nannicini, D. Delling, D. Schultes, L. Liberti, Bidirectional A* search on time-dependent road networks. Networks **59**(2), 240–251 (2012)
16. T. Ohshima, P. Eumthurapojn, L. Zhao, H. Nagamochi, An A* algorithm framework for the point-to-point time-dependent shortest path problem, in *Computational Geometry, Graphs and Applications*, ed. by J. Akiyama, et al. (Springer, Berlin, 2011), pp. 154–163
17. S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, Mason: a new multi-agent simulation toolkit, in *Proceedings of the 2004 SwarmFest, Workshop* (2004)
18. S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, G. Balan, Mason: a multiagent simulation environment. Simulation **81**(7), 517–527 (2005)
19. Z. Wang, S. Zlatanova, Taxonomy of navigation for first responders, in *Proceedings of the 9th International Symposium on Location-Based Services* Munich, Germany (2012)

http://www.springer.com/978-3-642-33217-3

Intelligent Systems for Crisis Management
Geo-information for Disaster Management (Gi4DM)
2012
Zlatanova, S.; Peters, R.; Dilo, A.; Scholten, H. (Eds.)
2013, XV, 386 p., Hardcover
ISBN: 978-3-642-33217-3