Hindawi Publishing Corporation
ISRN Robotics
Volume 2013, Article ID 329591, 6 pages
http://dx.doi.org/10.5402/2013/329591

Research Article

Unified Trajectory Planning Algorithms for Autonomous Underwater Vehicle Navigation

Oren Gal

Technion, Israel Institute of Technology, 32000 Haifa, Israel

Correspondence should be addressed to Oren Gal; orengal@tx.technion.ac.il

Received 9 April 2013; Accepted 19 May 2013

Academic Editors: G. C. Gini, J.-S. Liu, and M. Oussalah

Copyright © 2013 Oren Gal. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper presents two efficient methods for obstacle avoidance and path planning for Autonomous Underwater Vehicle (AUV). These methods take into account the dynamic constraints of the vehicle using advanced simulator of AUV considering low level control and stability effects. We present modified visibility graph local avoidance method and a spiral algorithm for obstacle avoidance. The algorithms were tested in challenged scenarios demonstrating safe trajectory planning.

1. Introduction

Path planning and obstacle avoidance are an important issues for Autonomous Underwater Vehicles (AUVs), and as can be noticed lately, these fields are extensively studied.

Currents disturbances can have a big influence in a different water depth and should be considered in underwater environments [1]. For the first time, Soulignac et al. [2] faced with strong current situations. Online replanning in 3D underwater environments with strong, dynamic, and uncertain currents was presented in [3].

AUV also suffers from a limited energy source, which can be minimized by planning optimal trajectories, extending the working time of the vehicle. Optimal path planning using A* algorithm search was presented by Carroll et al. [4]. Garau et al. [5]. Pêtrès et al. [6, 7] used similar concepts using BF (breadth first) search named FM and FM*.

As the presented methods in this paper, other methods do not use grid-based search. The path is presented with a series of points, which are connected one by one. Path planning problem transformed to a constrained optimization problem in terms of the coordinates of these points, generating optimal paths considering AUV's dynamic constraints [1, 8–10].

One of the most known limitations in motion planning algorithms related to real-time computation ability. Planning methods based on local perceptions are computationally less expensive and thus time efficient. Bui and Kim [11] and Kanakakis et al. [12] apply fuzzy logic approaches to AUV path planning. Antonelli et al. [13, 14] integrate virtual force field (VFF); all of these are not optimal planning paths methods.

This paper presents several different AUV path planning algorithms avoiding obstacle based on local perception abilities based on forward looking sonar. The introduced algorithms inherently take into account AUVs dynamic and kinematic constraints. AUV trajectory is simulated as described later. Simulations in typical underwater environments are presented, demonstrating algorithm's capabilities.

2. Vehicle Simulator

AUV platforms are known as underactuated vehicles models; these kinds of models generate dynamic constraints. In such cases, obstacle avoidance algorithms must consider the dynamic envelop of the vehicles not entering unstable states. We developed and implemented AUV simulator. The simulator takes into account the dynamic model of the AUV and the behavior of the sensors. The simulator uses only Forward Looking Sonars (FLS) for obstacle detection. In addition to FLS we have Side Scan Sonars (SSS) for detail

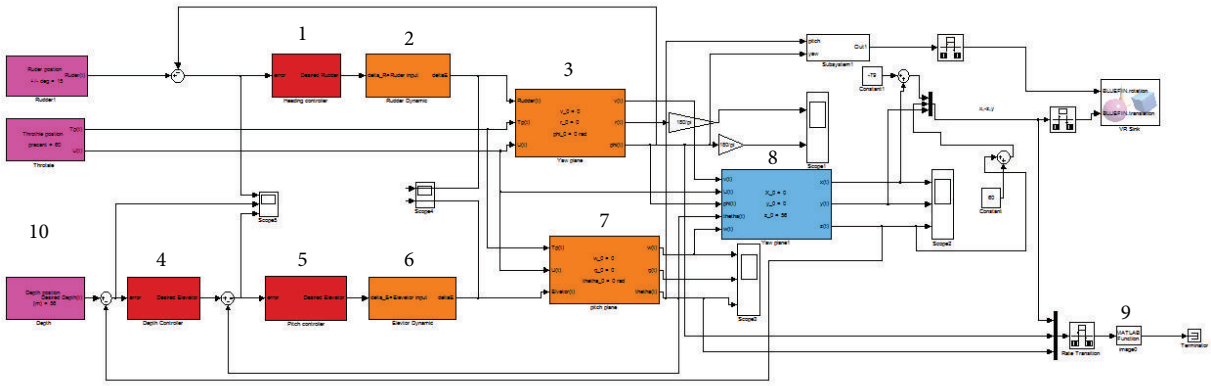


Figure 1: Block diagram of the AUV simulator [Braginsky, BGU].

scanning of the environment. Block diagram of simulator is presented in Figure 1.

3. Path Planning Algorithms

3.1. Visibility Graph Algorithm

3.1.1. General Description. The main idea behind our algorithm is the well-known visibility graph method. However, visibility graph algorithm is an abstract one that cannot be used on-line in real time applications, so many changes and adjustments are needed.

Basically, visibility graph describes the concept of connecting nodes from start point to the goal; nodes are connected to each other if and only if there are no obstacles between them, and the next node is visible from the current node.

Another major issue of the algorithm logic is concerned with node's creation. One of the assumptions of the abstract visibility graph algorithm is that the obstacles are polygonal. On the real world we cannot assume that every obstacle is polygonal. Although, we can trap the obstacle in a polygonal shape but performances are likely to be very conservative. We propose a new concept constructing nodes around obstacles, enabling safety motion of the vehicle.

After constructing a full graph of nodes and arcs, search over the graph can be used for optimal trajectory, such as A* algorithm.

Algorithm completeness is based on the fact that the marine environment is quasi dynamic and a free visible node can be found at each time step, taking into account kinematic and dynamic models.

3.1.2. Algorithm Description. Our algorithm can be divided into three steps: (1) building visibility graph; (2) connecting nodes; and (3) searching over the graph for the best solution.

(1) *Building the graph's nodes*: first, it should be noted that the visibility graph is a global algorithm and thus is not an efficient computation time one. Therefore, we try to minimize node's number to achieve fast calculation times.

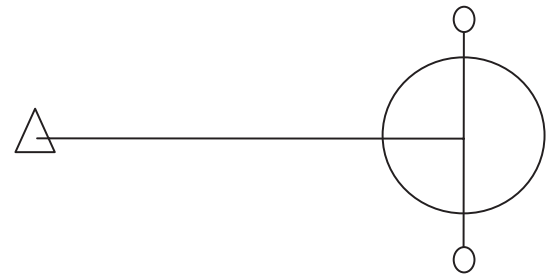


Figure 2: Obstacle's nodes.

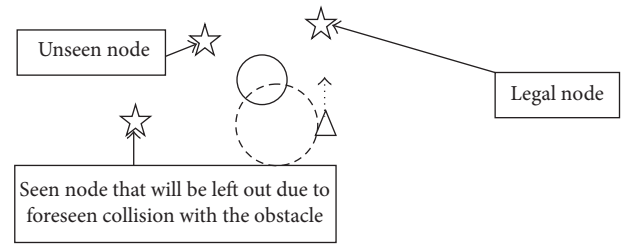


Figure 3: Risky nodes being reduced due to maneuver constraints.

We divide the nodes into two different cases.

(a) Vehicle nodes: we add nodes related to the vehicle. For example, a node straight ahead of the vehicle (that represents straight path without turnings), a node to the right of the vehicle (representing a right turn), and a node to the left. The easiest way is to choose a radius that will allow turning to the node and take nodes on a circle around the start position with this chosen radius. That way we have nodes that represent all turning directions as well as slowing down if needed.

We assume that the obstacles are round by spheres generating smooth trajectories due to dynamic profile and AUVs turning radius. In case of an obstacle which is not round, we bound the obstacle with a circle, considering safety.

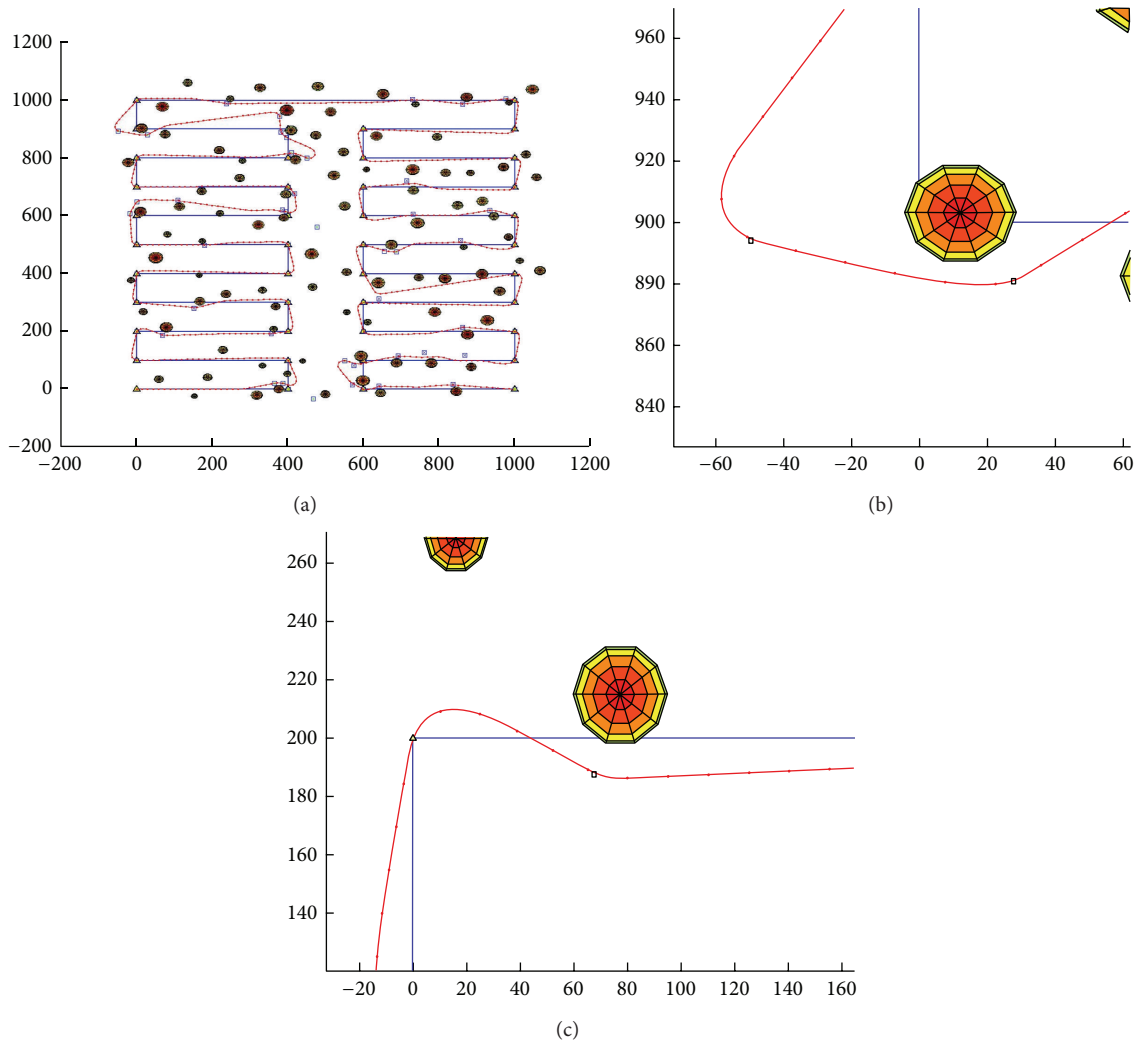


Figure 4: OA with spiral algorithm with typical scenarios (a), (b), and (c).

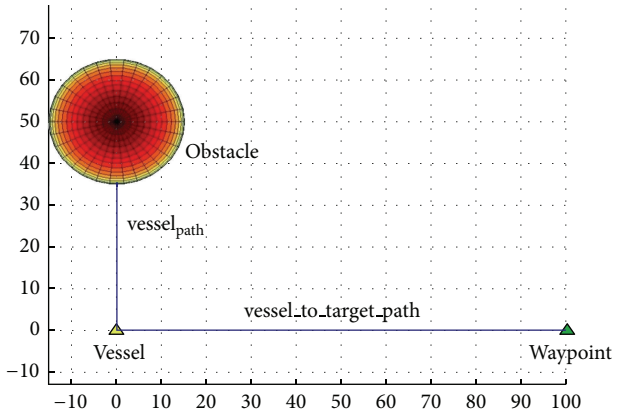


Figure 5: Vehicle moving to a waypoint changing vehicle course.

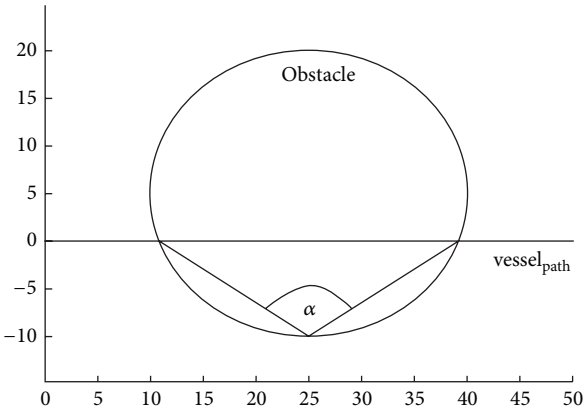


Figure 6: Calculating alpha angle in a collision course.

The next step is to choose the location of the nodes. For each obstacle we calculate the tangent line to the line that connects the obstacle with the robot and add two nodes on this line (the tangent line) right outside of the circle of the obstacle, as can be shown in Figure 2.

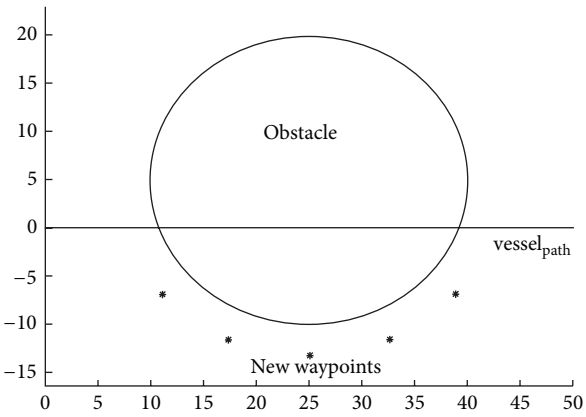


Figure 7: Added waypoints avoiding an obstaces.

(b) Goal nodes:

we add nodes that will help achieve the planned mission. For example, if the vehicle mission is to follow a line, adding nodes on that line will help the vehicle stay close to the original line, defining these nodes's cost as favor one when there are no obstacles.

(2) *Connecting nodes with arcs*: we connect the nodes using the simple idea of vision. If one node can be seen from the other node, these nodes are connected, as can be seen in Figure 3.

We add a simple measure considering the turning radius of the vehicle considering the maneuver constraints based on the turning radius of the vehicle. Otherwise, we do not connect this node.

(3) *Search an optimal path*: the last step is to search the optimal path over the graph from the current position of the vehicle to the goal. As mentioned above, we used an A* algorithm to search the shortest path in the visibility graph.

We define a cost function, taking into account and the shortest distance from the line connecting targets (the line connecting the first node and the last node at each iteration).

For the two nodes n and m, we set the cost of the arc that connects them as

$$F(n,m) = \text{Dist}(n,m). \tag{1}$$

Then we define the price for the node n as:

$$P(n) = \text{Dist From Target Line}\ (n,s,t) + \text{Cost Of Path}(n), \tag{2}$$

where s is the start node, t is the target node, and the Cost Of Path function is the sum of the costs of the arc connecting this node to the previous node and the cost of the previous. Based on that, the shortest path between start to goal is calculated.

3.1.3. Algorithm Simulations. In Figure 4, we present dense environment demonstrating visibility graph algorithm. Start point is located in the bottom left and goal point is located in the bottom right. It can be seen in (a) the nodes that were added during the time the vehicle was advancing its path. Nodes are denoted as blue squares. A square, with an "X" inside it, is a node that was passed by the vehicle. In (b) and (c), we can see a close up of two types of nodes. In (c) an obstacle node and in (b) a vehicle related node are presented.

3.2. Spiral Algorithm

3.2.1. General Description. The spiral algorithm presented for the first time in this paper, as its name may suggest, prefers to stay away from the vehicle's predetermined path and locally avoids an obstacle when there is an imminent danger of a collision. In such a case, circumnavigate of an obstacle is done by adding and placing new waypoints in the vehicle's path, in a semicircle around the center of the original, and in such a radius the vehicle can pass safely by the obstacle. For simplicity, we assume all obstacles are spheres. In the next section, a detailed algorithm stages are described.

3.2.2. Algorithm Stages. In this section, we introduce a more detailed description of the algorithm. We distinguish between two basic situations and the relevant action in a case of an obstacle that may cause collision.

(A) No obstacles were detected: in general, if there are no obstacles, the algorithm output would be diving and moving at nominal velocity at the same direction of the next waypoint in the vehicle's path. The nominal velocity and course can be easily converted to a waypoint.

(B) An obstacle was detected: when an obstacle is detected, several calculations must be done in order to decide whether the obstacle is endangering the vehicle:

(i) First, we check if there is a collision between the course of the vehicle and the obstacle. *CheckIntersect* is a subalgorithm that performs a set of vector calculations to decide whether or not a sphere and a line segment intersect. The intersection points of the sphere and infinite line containing the line segment, if any, are given by (5) based on (3) and (4):

$$\text{discrim} = \left(\left(\text{obs}_{\text{loc}} - \text{vessel}_{\text{loc}}\right) \cdot \text{vessel}_{\text{path}}\right)^2 - \left\|\text{obs}_{\text{loc}} - \text{vessel}_{\text{loc}}\right\|^2 + \left(\text{obs}_{\text{rad}} \times \left\|\text{vessel}_{\text{loc}}\right\|\right)^2 \tag{3}$$

$$\text{sol}_{1/2} = \frac{\left(\text{obs}_{\text{loc}} - \text{vessel}_{\text{loc}}\right) \cdot \text{vessel}_{\text{path}} \pm \text{sqrt}(\text{discrim})}{\left\|\text{vessel}_{\text{loc}}\right\|^2} \tag{4}$$

$$\text{intersect}_{\text{point}_{1/2}} = \text{vessel}_{\text{loc}} + \text{sol}_{1/2} \times \left(\text{vessel}_{\text{path}}\right) \tag{5}$$

CheckIntersect variable returns a true value if an intersection occurs within the finite line segment and false otherwise, where obstacle is the obstacle's center coordinates and radius and vessel$_{path}$ is the line segment starting in the vehicle's location, pointing to vehicle's heading. We limit vessel$_{path}$'s length to be the distance given by

$$\min\left(50, \left\|\text{vessel}_{\text{loc}} - \text{next}_{\text{target}_{\text{loc}}}\right\|\right). \tag{6}$$

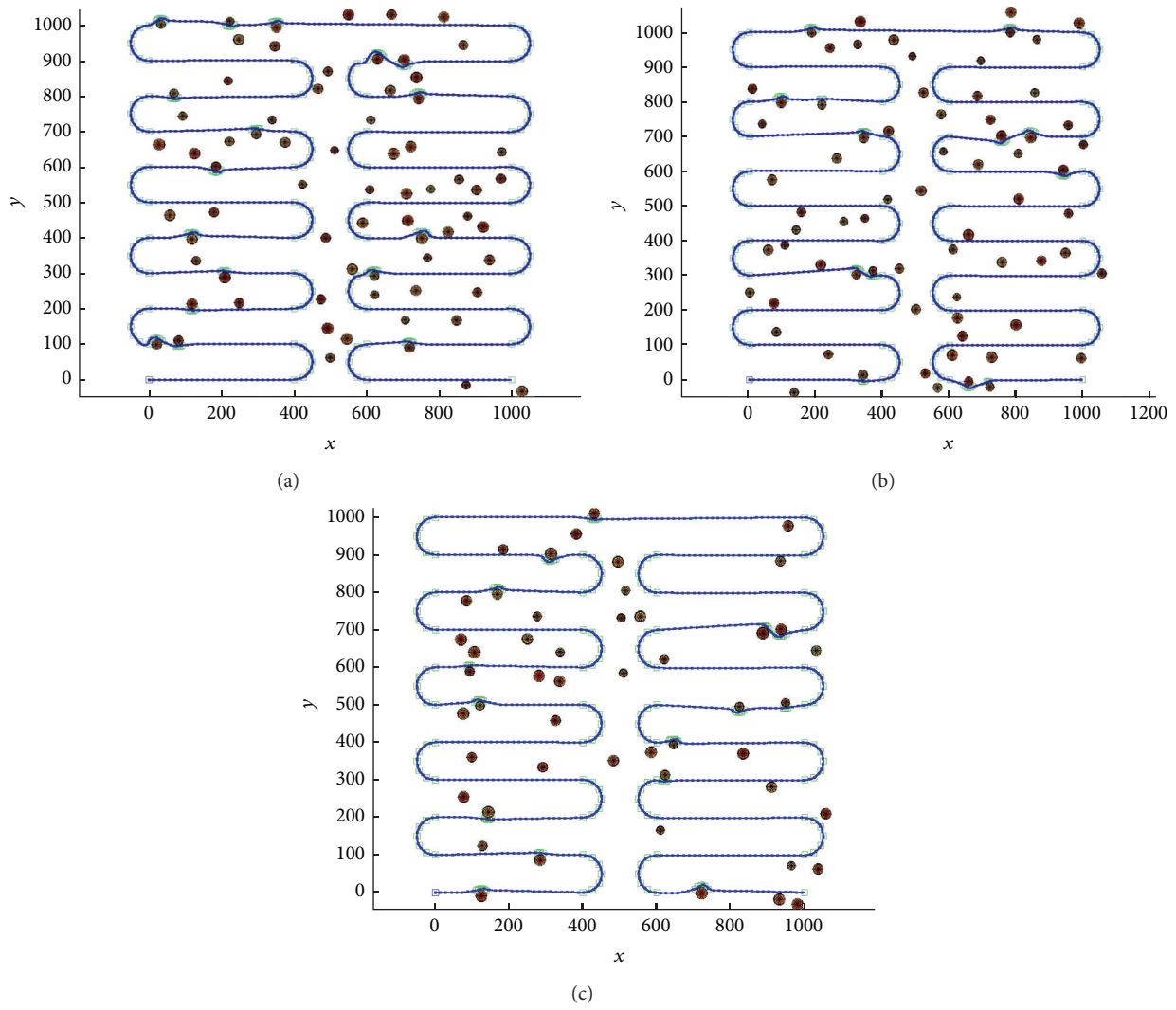


Figure 8: OA with spiral algorithm with typical scenarios (a), (b), and (c).

Obstacles detection ability is related to the vehicle perception abilities, and only local avoidance can be done.

(ii) In the next stage, we calculate

$$\text{Dist}\left(\text{obstacle}_{\text{loc}}, \text{vessel}_{\text{loc}}\right) - \text{obs}_{\text{radius}} < \text{safe}_{\text{marge}}, \quad (7)$$

where $\text{Dist}(\text{obstacle}_{\text{loc}}, \text{vessel}_{\text{loc}})$ is the Euclidean distance between the obstacle and the vehicle, $\text{obs}_{\text{radius}}$ is the obstacle's radius, and $\text{safe}_{\text{marge}}$ is the minimum distance at which we would like to consider the obstacle as a dangerous one.

(iii) Finally, we calculate CheckIntersect (obstacle, vessel_to_target_path) function value, where vessel_to_target_path is the line segment connecting the vehicle to the next waypoint as can be seen in Figure 5. The rational is not to consider obstacles which are not cause collision, in a case of vehicle's turns.

(C) Avoiding an obstacle: in a case of a collision course between the obstacle and the vehicle, we add number of waypoints around the obstacle in a spiral trajectory and locally avoid the obstacle, in the following way.

(i) First, we determine whether the center of the obstacle is to the left or the right of the vehicle's heading. We choose to pass the obstacle on the side that minimizes the distance between the updated and the original path.

(ii) Next, a trigonometric calculation is done, to determine how wide is the "bite" taken out of the vehicle's path by the obstacle. This is done in terms of the angle alpha as can be seen in Figure 6.

Alpha is given by

$$\alpha = 2 * \sin^{-1}\left(\frac{\left\|\text{intersect}_{\text{point}_1} - \text{intersect}_{\text{point}_2}\right\|}{2 \times \text{obs}_{\text{rad}}}\right). \quad (8)$$

If 135 < alpha ≤ 180, five waypoints are added on the boundary of the sphere centered located at the obstacle's center, with the augmented radius mentioned above. The waypoints are added so that the difference in degree between them is equal, as can be seen in Figure 7.

If 90 < alpha ≤ 135, 4 waypoints are added in the same fashion, and if alpha ≤ 90, 3 are added.

3.2.3. Algorithm Simulations. The algorithm has been tested using a simulation of a real vehicle simulator as described above. The scenarios chosen were 1 × 1 km squares, in which were scattered 50 to 70 obstacles of radii ≃ 15 meters. The scenarios are modeled after underwater minefields. The algorithm has shown good results so far, both in obstacle avoidance and in adhering to a course. Simulations can be seen in Figure 8.

4. Conclusions

In this research we presented two different methods for AUV, based on local and global planning methods. One of the major challenges in AUV obstacle avoidance is related to the underactuated model of the vehicle. We challenge these constraints by simulating vehicle trajectory using advanced simulator, modeling the perception abilities of the forward looking sonar, and the obstacles detection in realtime.

We presented a general description for each one of the algorithm and a simulation results with a typical search fields of AUVs.

Further research includes sea test with REMUS 100 AUV, testing the presented algorithms.

References

[1] D. Kruger, R. Stolkin, A. Blum, and J. Briganti, "Optimal AUV path planning for extended missions in complex, fast-flowing estuarine environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '07)*, pp. 4265–4270, April 2007.

[2] M. Soulignac, P. Taillibert, and M. Rueher, "Adapting the wave-front expansion in presence of strong currents," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '08)*, pp. 1352–1358, May 2008.

[3] D. R. Thompson, S. Chien, Y. Chao et al., "Spatiotemporal path planning in strong, dynamic, uncertain currents," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '10)*, pp. 4778–4783, May 2010.

[4] K. P. Carroll, S. R. McClaran, E. L. Nelson, D. M. Barnett, D. K. Friesen, and G. N. Williams, "AUV path planning: an A* approach to path planning with consideration of variable vehicle speeds and multiple, overlapping time-dependent exclusion zones," in *Proceedings of the Symposium on Autonomous Underwater Vehicle Technology (AUV '92)*, pp. 3–8, 1992.

[5] B. Garau, A. Alvarez, and G. Oliver, "Path planning of autonomous underwater vehicles in current fields with complex spatial variability: an A* approach," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 194–198, Barcelona, Spain, April 2005.

[6] C. Pêtrès, Y. Pailhas, Y. Petillot, and D. Lane, "Underwater path planing using fast marching algorithms," in *Proceedings of the Oceans Europe*, vol. 2, pp. 814–819, June 2005.

[7] C. Pêtrès, Y. Pailhas, P. Patrón, Y. Petillot, J. Evans, and D. M. Lane, "Path planning for autonomous underwater vehicles," *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 331–341, 2007.

[8] C. W. Warren, "Technique for autonomous underwater vehicle route planning," *IEEE Journal of Oceanic Engineering*, vol. 15, no. 3, pp. 199–204, 1990.

[9] A. Alvarez and A. Caiti, "A genetic algorithm for autonomous underwater variability," in *Proceedings of International Federation Automatic Control (IFAC) Conference Control Applications Marine Systems*, Glasgow, UK, June 2001.

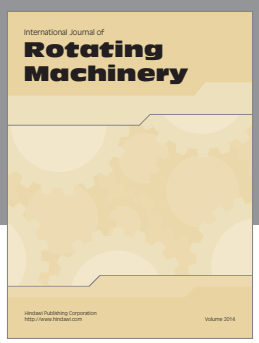
[10] A. Alvarez, A. Caiti, and R. Onken, "Evolutionary path planning for autonomous underwater vehicles in a variable ocean," *IEEE Journal of Oceanic Engineering*, vol. 29, no. 2, pp. 418–429, 2004.

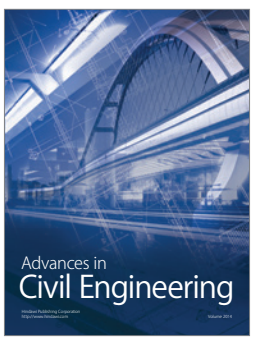
[11] L.-D. Bui and Y.-G. Kim, "An obstacle-avoidance technique for autonomous underwater vehicles based on BK-products of fuzzy relation," *Fuzzy Sets and Systems*, vol. 157, no. 4, pp. 560–577, 2006.

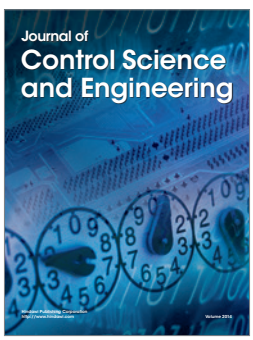
[12] V. Kanakakis, K. P. Valavanis, and N. C. Tsourveloudis, "Fuzzy-logic based navigation of underwater vehicles," *Journal of Intelligent and Robotic Systems*, vol. 40, no. 1, pp. 45–88, 2004.

[13] G. Antonelli, S. Chiaverini, R. Finotello, and E. Morgavi, "Real-time path planning and obstacle avoidance for an autonomous underwater vehicle," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '99)*, pp. 78–83, Detroit, Mich, USA, May 1999.

[14] G. Antonelli, S. Chiaverini, R. Finotello, and R. Schiavon, "Real-time path planning and obstacle avoidance for RAIS: an autonomous underwater vehicle," *IEEE Journal of Oceanic Engineering*, vol. 26, no. 2, pp. 216–227, 2001.







Hindawi
Submit your manuscripts at
http://www.hindawi.com