Path Planning Algorithms for Robotic Underwater Sensing in a Network of Sensors

Sreeja Banerjee
Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, NE 68588, USA
sreeja@cse.unl.edu

Carrick Detweiler
Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, NE 68588, USA
carrick@cse.unl.edu

ABSTRACT

Monitoring lakes, rivers, and oceans is critical to improving our understanding of complex large-scale ecosystems. In this work, we develop and analyze three path planning algorithms for underwater robots to optimize sensing in conjunction with networks of underwater sensors. The algorithms require different levels of knowledge about the environment: global, local, and decentralized control of the robot by the sensor network. We find our global Voronoi approach produces paths that are typically best for sensing, but are longer, which can be problematic if the robot has limited endurance. The local algorithm, inspired by Tangent Bug, produces paths that are usually shorter while still having good sensing. The decentralized controller also has good sensing and short paths and has the advantage that it can also adapt the depths of the underwater sensors to jointly optimize the sensor network and robot sensing and the robot path length. The drawback is the somewhat higher communication and processing requirements. For each of these algorithms we perform a detailed analysis and comparison in simulation. We identify limitations of each and provide framework for future improvements.

1. INTRODUCTION

Water is crucial for supporting life on earth, so it is important to develop tools to monitor the water bodies. New technologies have enabled the exploration of the vast unexplored aquatic environment. This includes underwater sensors nodes that can be deployed for long periods to study single points in high detail over time and underwater gliders, ROVs, and AUVs that can be deployed to monitor larger regions. By enabling close interaction of these two types of systems there is the potential to enable long-term monitoring of select locations with an underwater sensor network, while periodically filling in the sensing gaps and providing high-resolution sensing with an underwater vehicle.

In this work, we develop and analyze the cost and benefit trade off of three algorithms that optimize the path for sensing of underwater robots moving through networks of underwater sensors, such as those shown in Figure 1. The first algorithm, VORONOIPATH, is a global path planning algorithm using the *Voronoi Tessellation* method. The algorithm needs a priori knowledge of positions of all sensors before the mobile robot enters the water column, but no additional communication is needed once traveling through the water. The second approach, TANBUGPATH, is a local path planning algorithm inspired from the traditional robot path planning *Tangent Bug* algorithm. When the robot enters the water column it knows the position of only the nearest sensor. The robot moves towards this sensor maintaining a minimum distance from it. This sensor then communicates information on the position of the next sensor to the robot and the process continues in this manner. The third algorithm, ADAPTIVEPATH, is based on our prior work in developing adaptive decentralized control algorithms that optimizes the depths of underwater sensors for sensing [5] and for determining the path of an underwater robot in networks of underwater sensors while also constraining the length of the path [4]. In this work we develop and analyze these algorithms in simulation.

Figure 1: AquaNode underwater sensor nodes with the underwater robot Amour.

The research can be broadly divided into three main categories: path planning, sensor placement and underwater sensing. Takahashi *et al.* [19], introduced a path planning algorithm based on *Generalized Voronoi Diagram (GVD)*. Garrido *et al.* [9], Bhattacharya *et al.* [2] used it for global path planning between a source and destination. So and Ye [17] show that *GVD* can be used to solve coverage problems. Kamon *et al.* first introduced *Tangent Bug* algorithm in [12]. Ge *et al.* introduced the local target approach [10]. Buniyamin *et al.* used it in *PointBug* algorithm which results in shorter path lengths [3].

Detweiler *et al.* presented a decentralized controller optimization by adjusting depths of underwater sensors in [5]. In [14], Liu *et al.* introduced a method of joint optimization for minimizing communication cost and maximizing information gainand Stranders *et al.* [18] presented an on-line,

WUWNet '14, November 12-14, 2014 Rome, Italy
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3277-4/14/11:.$15.00
http://dx.doi.org/10.1145/2671490.2674569

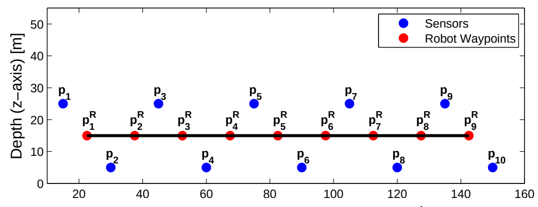


Figure 2: Sensors and robot waypoints ($N = 10, M = 9$)

decentralized coordination algorithm for a team of mobile sensors. Leonard *et al.* [13] designed a mobile sampling network to take measurements and Smith *et al.* [16] performed angle optimization to control path of an autonomous underwater glider. In all these, the focus is on avoiding obstacles or planning the shortest distance path. However, in this research we introduce a robot to a region deployed with semi-mobile sensors to improve sensing quality.

2. PROBLEM FORMULATION

We now formalize the problem statement and discuss our assumptions. While this work presents simulated results, it is motivated by our prior work on underwater sensor network with depth adjustment capabilities, called AquaNodes [20], and an underwater robot that can communicate and collect data from the sensor nodes [7, 8]. Figure 1 shows a picture of the system. The AquaNodes are anchored at the bottom of the water column and can dynamically adjust their depths. Amour can communicate both acoustically and optically with AquaNodes. We assume a 2D configuration and do not examine the impact of communication performance. We leave examining these for future work, but note that in our prior work [4] we implemented similar algorithms on the AquaNodes and found that it worked well in practice. We also do not consider localization as we have shown in our previous work that both sensors and robot are capable of localizing in the water column [6].

Figure 2 shows the setup of our system. There are N sensors at locations $\{p_1 \cdots p_N\}$, and we want to determine the best M locations for the robot, $\{p_1^R \cdots p_M^R\}$, to sense the field. We also want to ensure that the robot sensing locations do not cause the robot to travel excessively far, since the robot has limited energy. For a region we are sensing, Q, we call points we are interested in $q \in Q$. In practice we discretize Q into a fine grid and aim to obtain as much information about all of these points. A covariance function is needed for this algorithm which describes the relationship between sensor positions and all other points in the region of interest. This covariance function is modeled as a multivariate Gaussian, as is often used in objective analysis in underwater environments [15].

To evaluate the performance of our algorithms we use the posterior error or entropy measure [11]. The posterior error of a point can be calculated as:

$$\sigma^2_{q|P} = Cov(q, q) - \Sigma_{q,P} \cdot \Sigma^{-1}_{P,P} \cdot \Sigma_{P,q} \tag{1}$$

The vector$\Sigma_{\ q,P}$ is vector of covariances between q and sensors $P = \{p_1, ..., p_N\}$. The vector$\Sigma_{\ P,q}$ is$\Sigma_{\ q,P}$ transposed. The matrix$\Sigma_{\ P,P}$ is covariance matrix for sensors. The values of$\Sigma_{\ P,P}$ are$\Sigma_{\ p_i,p_j} = Cov(p_i, p_j)$ for each entry (i, j). In our algorithms we do not directly aim to minimize entropy since this requires the inversion of the covariance matrix, which cannot be easily computed on most memory and processor constrained sensor network systems.

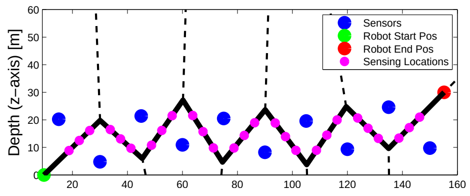


Figure 3: Sensors and Voronoi path

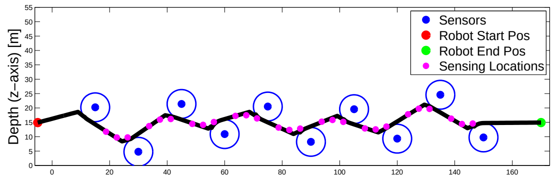


Figure 4: Sensors, sensing radius and Tangent Bug path ($r_S = 5; r_R = 5$)

3. ROBOT SENSING ALGORITHMS

In the following subsections, we discuss three different methods for planning the path of an underwater robot through a sensor network. We assume that movement of sensors is constrained, hence, the sensor network is not able to sense at all possible location in the region. The goal of the path planners is to allow the robot to sense at positions which are under sensed and unreachable by the sensors. In our examples, we use a water column of depth 30m.

3.1 Voronoi Path

Voronoi Tessellation is a method of dividing a given space into number of regions called Voronoi cells and the set of all Voronoi cells for a given set of points is called a Voronoi diagram. Given a set of coplanar points $P = \{p_1, \cdots, p_n\}$ and a distance function $d(x, y)$, it is a subdivision of the space into n different cells, one for each point in P such that a point q lies in the cell corresponding to a point p_i if and only if $d(p_i, q) < d(p_j, q)$ for $i \neq j$.

Let us assume sensors in a plane of water. If we draw Voronoi cells around each sensor, then each boundary of a Voronoi Cell is a straight line which is equidistant from the two sensors on either side of it. Thus, influence of these two sensors is minimum on this line. If we release a robot to increase the information gain between sensors, then it should pass through this straight line. The path obtained this way is the Voronoi Path and can be seen in Figure 3.

3.2 Tangent Bug Path

A global map of environment is often not available before a robot starts moving. Tangent Bug is a local path planning algorithm in which a robot finds endpoints of finite continuous segments by drawing a tangent from its current position to the surface of obstacles [12]. If the intersection points are denoted by a set $O = \{O_1, \cdots, O_n\}$, the algorithm chooses the point O_i such that the distance $d = dist(P_{curr}, O_i) + dist(O_i, P_{end})$ (where P_{curr} is the current position of robot and P_{end} is the destination) is minimized and then moves towards that point. Once it reaches O_i the robot again aims to go towards the destination and repeats the process again if it faces another obstacle.

Tangent Bug algorithm can be applied to Tangent Bug Path algorithm if we assume that each sensor can sense the surrounding region effectively up to a certain distance. This region can be marked by a circular region with the sensor as center and a radius r_S. Any point outside this region is under-sensed by that sensor. A robot does not need to sense any point inside this region and so avoids it. Both sensors and robot communicate acoustically, so their range is limited. A robot has to move towards the sensing boundary of a sensor to be able to communicate with it to get information on the position of the next sensor. The robot also senses locally up to a certain distance which can be denoted by a circular sensing region with radius r_R (r_S and r_R can be same or different). The Tangent Bug Path algorithm proceeds in small intervals and the robot has a local target at each interval. Since the algorithm is local, this local target can be either the next sensor position or the intersection point of the tangent with the circumference of sensing region of the next sensor or the final destination.

Figure 4 shows that the robot moves in the direction of the next target in small intervals. When the robot is closer to the target, the algorithm finds intersection points between the tangent from its current position to the boundary of nearest sensor. The algorithm chooses the intersection point O_i which is closer to the next target, and then moves towards this point. Once O_i is reached, if the straight line that connects the robot to the next target does not pass through the sensing region of the current sensor, then the robot moves along this line. If the line passes through the sensing region then the robot moves along the sensing boundary of the nearest sensor till the algorithm finds a point O_j from which the robot will have an unobstructed path towards the next target. When O_j is found, the robot leaves the boundary and moves towards the next target.

3.3 Adaptive Path

In our prior work we developed a decentralized algorithm for controlling the depth of the underwater sensors to optimize depths for sensing [5]. We extended this work to also consider hybrid systems with underwater sensors and robots that can be used to fill the gaps in sensing while also constraining the length of the robot path [1, 4]. In this section we provide an overview of this algorithm.

We start by defining an objective function, $\mathcal{H}(p_1, ..., p_N)$, that is the cost of sensing at point q given sensors placed at positions $p_1, ..., p_N$. For N sensors, we define our objective function over a region Q as:

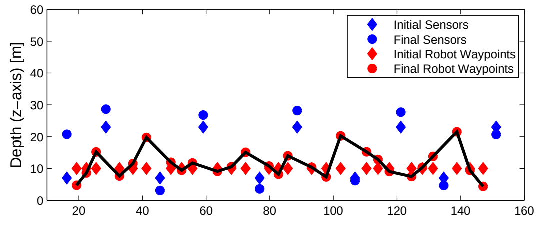
$$\mathcal{H}(p_1, ..., p_N) = \int_Q \left(\sum_{i=1}^{N} f(p_i, q) \right)^{-1} dq \qquad (2)$$

Minimizing this function positions the sensor nodes to cover the region Q for sensing. In practice we use a Gaussian covariance function for $f(p_i, q)$, which relates the location the sensor is placed, p_i, to a point in the region $q \in Q$, but $f(p_i, q)$ can take other forms as well. Equation 2 only serves to position the underwater sensors for sensing. We can extend this to include robot sensing points, $\{p_1^R, ..., p_M^R\}$, by minimizing $\mathcal{H}(p_1, ..., p_N, p_1^R, ..., p_M^R)$ where each point p_i^R is one point along the robot path for M robot waypoints. However, minimizing this function may cause the robot path to be extremely long. So we introduce a constraint on the robot path length:

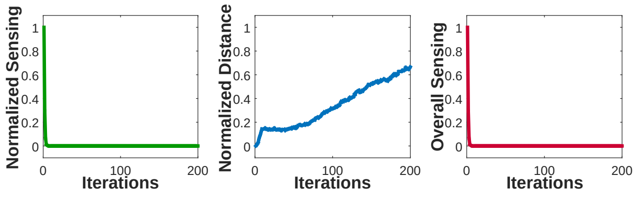
$$\mathcal{P}(p_1^R, ..., p_M^R) \quad = \quad \sum_{i=1}^{M-1} dist(p_i^R, p_{i+1}^R) \qquad (3)$$

where $dist(p_i^R, p_{i+1}^R)$ represents distance between consecutive robot waypoints. We combined the objective functions to balance improvements in sensing with robot path length:

$$\mathcal{H}^R \quad = (1 - \alpha)\mathcal{H} + \alpha\mathcal{P} \qquad (4)$$



(a) Node Layout



(b) Sensing, Distance & Overall Objective function

Figure 5: Start and end layout for AdaptivePath (Sensors at 5m & 25m at start, robot at 10m; $\alpha = 0.01$; $k = 4000$; $V = 3$) with different objective functions

where α is a weight function which varies between [0,1]. The variable α lets us select between longer robot paths that produce better sensing versus shorter paths.

To minimize $\mathcal{H}^R$, we develop a decentralized gradient controller by taking gradient of $\mathcal{H}^R$ with respect to z (depth). Each sensor node communicates with neighbors to compute a motion based on moving in the negative direction of gradient, $-k\frac{\partial \mathcal{H}}{\partial z_i}^R$, scaled by a factor k. In other words we fix x and y positions of all sensors and robot waypoints and then adjust their depths to optimize our $\mathcal{H}^R$. We show in [1, 4] that this decentralized controller converges to a local minimum, using a Lyapunov-style proof. We also show that while we can only guarantee a local minimum, this minimum is typically close to the global minimum and we can adjust α to control the path length the robot travels.

A routine to update depth is run periodically on each sensor. The algorithm computes the change in the desired depth, which is bounded by the maximum speed that the nodes can travel, and changes the node depth. Each sensor node is responsible for computing and updating the robot waypoints p^R that are closest to it. When the robot comes close, it will transmit these locations to the robot. In practice, the sensor network may have a set schedule of times when the robot enters the network, or the robot might inform nearby sensors when it enters the network. These sensors then start running the depth adjustment algorithm to compute the best path for the robot and any corresponding changes in the depths for the sensors. This will propagate through the network, and as the robot moves, it will receive updated waypoints from nearby sensor nodes.

Since this algorithm runs continuously, it can adapt to changing conditions. If an area in the water has different characteristics than the rest of the water column (marked by lower values of covariances, σ_d and σ_s), then AdaptivePath distributes the nodes so that more number of sensors and robot waypoint end in this region irrespective of their starting location. AdaptivePath is more tolerant towards the failure of sensors compared to the other algorithms, as the decision of where the robot waypoints should lie is distributed between each sensor. When a sensor stops working then a robot waypoint can replace its position and thus the overall sensing remains unaltered.

Figure 5(a) shows the initial and final layout of sensors and robot path for AdaptivePath. The sensing objective (Figure 5(b)a) decreases over time, whereas, distance function (Figure 5(b)b increases slightly, but its affect on the overall cost function (Figure 5(b)c) is very low due to the low value of α. The system converges when the overall cost function decreases consecutively for more than 5 iterations.

4. COMPARISON

In this section we compare VoronoiPath, TanBugPath and AdaptivePath algorithms. The three algorithms have different strengths and weaknesses. To make the comparison equivalent for all three algorithms, we specify similar initial conditions for each approach. We specifying $r_S = r_R = 5m$ for TanBugPath and $\sigma_d = 5, \alpha = 0.01$ and $k = 2000$ for AdaptivePath. The low value of α and moderate value of k ensure that sensing objective function has prominent effect on planned path. For three intermediate robot waypoints, the positions of the robot waypoints in the water column start horizontally at 18.75m, 22.5m and 26.25m, and then increase monotonically at gaps of 15m. An overall view of the comparison of the three algorithms is presented in Table 1.

A-Priori Network Knowledge: VoronoiPath is a global algorithm where the location of all sensors must be known at the beginning and it is computed centrally. The algorithm is run before the robot enters water, and the sensing locations are provided to the robot. TanBugPath is a local path planning algorithm. This implies all processes for planning path of a robot through an underwater sensor network is done real-time on-board the robot. AdaptivePath is an adaptive decentralized algorithm. Each of the sensors can independently decide their location in the water column. On introducing the robot waypoints, the sensor closest to the robot waypoints determine the best position for the robot waypoints along with its own position.

Ease of Implementation: In terms of ease of implementation, VoronoiPath planning is centralized and relatively easy to implement as long as the global knowledge of the sensors is available. This is followed by the AdaptivePath where each sensor runs an algorithm to update its depth locally. These sensors optimize locations of nearest robot waypoints on either side. In TanBugPath, the robot needs to communicate with the nearest sensor and needs to maintain a threshold distance with each sensor to plan the path. It also needs a method to localize itself in water.

Communication Requirement: VoronoiPath has the least communication requirement when the robot is inside the water column. However, the robot needs all sensor locations before it enters water, so it has the highest communication requirement when the robot is calculating its path. For TanBugPath the robot communicates with nearest sensors to know positions of the next sensors. In AdaptivePath, the robot is dependent on sensors to know next sensing locations.

Energy Requirement: An important factor in comparing the algorithms is energy efficiency of planned robot paths. A zigzagged path is generally longer and consumes more energy than a relatively straight path. Thus, even in cases where a zigzagged path is sensing efficient, it may not be energy efficient. Often *Voronoi* path is longer compared to *Tangent Bug* and *Adaptive* path for same input sensor locations. For VoronoiPath, the path depends on the location of the sensors only. The robot moves through the Voronoi vertices on its path and hence the total distance covered is **226.35m** for this layout. In TanBugPath, the distance depends on robot start position, sensing radii r_S and r_R, and *interval* size. In the example, distance covered by a robot is **139.77m**. For AdaptivePath, distance depends on starting location of robot waypoints, number of intermediate robot waypoints V, the weight factor α and k value. The average distance for this layout is **140.73m**. We find similar results for different layouts, although poor initial node configurations can cause AdaptivePath to get stuck in a sub-optimal local minima.

Sensing Efficiency: *Posterior error* is a common metric for defining how well an area is covered by sensors as discussed in Section 2. A lower value of posterior error signifies better sensing. AdaptivePath is optimized with respect to an objective function. But VoronoiPath and TanBugPath algorithms do not have any objective function. So we compare the algorithms with respect to the *posterior error* of the final configuration of sensors and robot path. Figure 6 shows the plots of the posterior error values of the final configuration, with and without mobile robot, for three different robot sensing location configurations. It can be observed that the VoronoiPath method performs best for all V and performance of the TanBugPath algorithm is comparable to that of AdaptivePath algorithm. We calculated the mean and standard deviation by considering eleven layouts where the sensor positions could vary within $\pm 2m$ of their location in the horizontal direction.

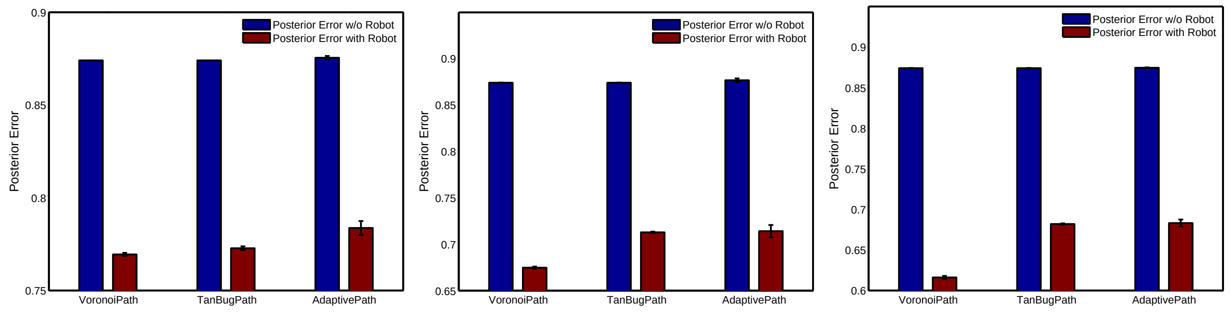
Table 1 summarizes these. The table also shows that since VoronoiPath precomputes the path of the robot before its enters water, any changes in sensor locations is not taken into account after the path has been decided. For TanBugPath and AdaptivePath any immediate changes in sensor locations is considered. However, in case of VoronoiPath and TanBugPath, the traversal of the robot does not have any effect on the positions of sensors already present in the water column. The AdaptivePath algorithm, on the other hand, rearranges the position of the sensors based on the robot path to improve overall sensing. This is an important feature of this algorithm.

5. CONCLUSION AND FUTURE WORK

In this paper we describe an underwater sensor network system that consists of semi-mobile sensors and a mobile robot. We introduced three different path planning algorithms for planning the path of the mobile robot through the network of the semi-mobile sensors. The mobility of the underwater robots enhances the performance of the system and results in better information gain from the area of water column, also large areas can be covered more efficiently with sparse sensor networks. In the future, we plan to investigate path planning in a 3D environments, local approximation algorithms for Voronoi path planning, and adaptations to enable multiple passes of the robot through water column.

6. ACKNOWLEDGMENTS

We would like to thank I. Vasilescu, M. Doniec, and D, Rus for their work on Amour and the original version of the AquaNodes and J. Ostdiek, J. Greenwood, and the NIMBUS Lab for their work on the new AquaNodes. This work was partially supported by NSF IIS-1116221, USDA #2013-67021-2094, and NSF CSR-1217400.



(a) $V = 1$ (b) $V = 2$ (c) $V = 3$

Figure 6: Posterior error comparison for different intermediate robot waypoints, V

Table 1: Comparison of Path Planning Algorithms

Measure	**Voronoi Path**	**Tangent Bug Path**	**Adaptive Path**
A-Priori Network Knowledge	Centralized	Centralized	Decentralized
Ease of Implementation	Easy	Complicated	Moderate
Communication Requirement	Least/Highest	Moderate	High
Energy Requirement	Longer path length	Moderate Path length	Depends on layout, waypoints & α
Sensing Efficiency	Highest	same as ADAPTIVEPATH	same as TANBUGPATH
Change in Sensor Locations	Not taken into account	Taken into account	Taken into account
Sensor Position	Remains Static	Remains Static	Changes with robot path

7. REFERENCES

[1] S. Banerjee. A comparative study of underwater robot path planning algorithms for adaptive sampling in a network of sensors. Master's thesis, University of Nebraska-Lincoln, 2014.

[2] P. Bhattacharya and M. L. Gavrilova. Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path. *Robotics & Automation Magazine, IEEE*, 15(2):58–66, 2008.

[3] N. Buniyamin, W. Wan Ngah, N. Sariff, and Z. Mohamad. A simple local path planning algorithm for autonomous mobile robots. *International journal of systems applications, Engineering & development*, 5(2):151–159, 2011.

[4] C. Detweiler, S. Banerjee, M. Doniec, M. Jiang, F. Peri, R. F. Chen, and D. Rus. Adaptive decentralized control of mobile underwater sensor networks and robots for modeling underwater phenomena. *Journal of Sensor and Actuator Networks*, 3(2):113–149, 2014.

[5] C. Detweiler, M. Doniec, M. Jiang, M. Schwager, R. Chen, and D. Rus. Adaptive decentralized control of underwater sensor networks for modeling underwater phenomena. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 253–266, 2010.

[6] C. Detweiler, J. Leonard, D. Rus, and S. Teller. *Passive mobile robot localization within a fixed beacon field*. Springer, 2008.

[7] C. Detweiller, I. Vasilescu, and D. Rus. An underwater sensor network with dual communications, sensing, and mobility. In *OCEANS 2007-Europe*, pages 1–6.

[8] M. Doniec, C. Detweiler, and D. Rus. Estimation of thruster configurations for reconfigurable modular underwater robots. In *International Symposium on Experimental Robotics*, pages 655–666, 2010.

[9] S. Garrido, L. Moreno, M. Abderrahim, and F. Martin. Path planning for mobile robot navigation using voronoi diagram and fast marching. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006*, pages 2376–2381, Oct 2006.

[10] S. S. Ge, X. Lai, and A. Mamun. Boundary following and globally convergent path planning using instant goals. *Trans on Systems, Man, and Cybernetics, Part B: Cybernetics*, 35(2):240–254, April 2005.

[11] C. Guestrin, A. Krause, and A. P. Singh. Near-optimal sensor placements in gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pages 265–272. ACM, 2005.

[12] I. Kamon, E. Rivlin, and E. Rimon. A new range-sensor based globally convergent navigation algorithm for mobile robots. In *IEEE International Conference on Robotics and Automation, 1996*, volume 1, pages 429–435 vol.1, Apr 1996.

[13] N. E. Leonard, D. A. Paley, F. Lekien, R. Sepulchre, D. M. Fratantoni, and R. E. Davis. Collective motion, sensor networks, and ocean sampling. *Proceedings of the IEEE*, 95(1):48–74, 2007.

[14] J. Liu, F. Zhao, and D. Petrovic. Information-directed routing in ad hoc sensor networks. *IEEE Journal on Selected Areas in Communications*, 23(4):851–861, April 2005.

[15] D. R. Lynch and D. J. McGillicuddy Jr. Objective analysis for coastal regimes. *Continental Shelf Research*, 21(11):1299–1315, 2001.

[16] R. N. Smith, M. Schwager, S. L. Smith, B. H. Jones, D. Rus, and G. S. Sukhatme. Persistent ocean monitoring with underwater gliders: Adapting sampling resolution. *Journal of Field Robotics*, 28(5):714–741, 2011.

[17] A. M.-C. So and Y. Ye. On solving coverage problems in a wireless sensor network using voronoi diagrams. In *Internet and Network Economics*, number 3828 in Lecture Notes in Computer Science, pages 584–593. Jan. 2005.

[18] R. Stranders, A. Farinelli, A. Rogers, and N. R. Jennings. Decentralised coordination of mobile sensors using the max-sum algorithm. In *IJCAI*, volume 9, pages 299–304, 2009.

[19] O. Takahashi and R. Schilling. Motion planning in a plane using generalized voronoi diagrams. *Robotics and Automation, IEEE Transactions on*, 5(2):143–150, Apr 1989.

[20] I. Vasilescu, C. Detweiler, and D. Rus. Aquanodes: an underwater sensor network. In *Proceedings of the second workshop on Underwater networks*, pages 85–88. ACM, 2007.