Autonomous Cooperation of Heterogeneous Platforms for Sea-based Search Tasks

by

Andrew J. Shafer

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2008

Author . . . .
Department of Electrical Engineering and Computer Science
May 23, 2008

Certified by . . . .
John J. Leonard
Professor
Thesis Co-Supervisor

Certified by . . . .
Michael R. Benjamin
Visiting Scientist
Thesis Co-Supervisor

Accepted by . . . .
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Autonomous Cooperation of Heterogeneous Platforms for Sea-based Search Tasks

by

Andrew J. Shafer

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2008, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Many current methods of search using autonomous marine vehicles do not adapt to changes in mission objectives or the environment. A cellular-decomposition-based framework for cooperative, adaptive search is proposed that allows multiple search platforms to adapt to changes in both mission objectives and environmental parameters. Software modules for the autonomy framework MOOS-IvP are described that implement this framework. Simulated and experimental results show that it is feasible to combine both pre-planned and adaptive behaviors to effectively search a target area.

Thesis Co-Supervisor: John J. Leonard
Title: Professor

Thesis Co-Supervisor: Michael R. Benjamin
Title: Visiting Scientist

Acknowledgments

I would like to thank Professor John Leonard and Dr. Michael Benjamin for taking me on as a UROP student and guiding my research. I also want to thank Professor Henrik Schmidt for providing financial support for my M.Eng.

The Office of Naval Research supported my participation in a research cruise to Dabob Bay, WA where I learned more about marine vehicle operations. I am also indebted to the researchers at NSWC–Panama City, FL for teaching me the basics of mine warfare and mine countermeasures. Special thanks go to Rafael Rodriguez for his “MCM 101” presentation and Gerald Dobeck for his talk about CAD/CAC.

Joe Curcio taught me many things about the kayaks and real-world engineering trade-offs. Kevin Cockrell was invaluable as a person to bounce ideas off. Toby Schneider knew all of the little tricks for conducting successful water operations.

Finally, I want to thank all of my family and friends for their support throughout the years. Without them I never would have thought it possible to graduate.

Contents

List of Figures

List of Tables

Chapter 1

Introduction to Autonomous Mine Countermeasures

1.1 Project Goals

The goal of this project is to design and implement a software system to enable multiple marine vehicles to search a given area. Specifically, tools are needed to 1) model sensor performance, 2) maintain and synchronize a map of the area, and 3) implement a search algorithm.

1.2 Limitations of Current Search Methods

Previous research in the searching domain has focused on "lawnmower" style patterns of mapping an area [4, 15, 26, 42–44]. However, this process deteriorates as more vehicles are added. If a vehicle is assigned to a portion of the overall area and then breaks down, the other vehicles must be manually reconfigured to scan that portion. Also, this process generally assumes the platforms are identical and does not take advantage of the capabilities that may be unique to each vehicle (e.g. one may have side-scan sonar, one forward-looking sonar, another optical or magnetic sensors, etc.). Finally, this process does not allow for cooperation with other objectives of the vehicle, such as collision avoidance or periodic surfacing.

Current search algorithms also generally assume a constant probability of detection for the targets they seek. This assumption may not be realistic for real-world environments where the motion of the water and the composition of the sea-bottom can vary drastically.

Previous research in AUV behavior development has primarily focused on implementing behaviors for a single vehicle [10–12], such as a single AUV taking temperature readings during ocean trials. With access to larger numbers of autonomous craft, research is now needed to explore possible ways for these vehicles to interact [6, 34], especially if the vehicles have different capabilities (such as an autonomous surface vehicle and an autonomous underwater vehicle, see [47]).

1.3 Background

1.3.1 A Brief Overview of Mine Countermeasures (MCM)

The ultimate goal of mine countermeasures (MCM) is to neutralize mines (called targets). There are two basic ways to go about this, minesweeping and mine hunting. Minesweeping refers to the process of neutralizing all mines in an area without specifically locating them first. This might involve, for example, using a helicopter to tow a large ferrous sled through mined waters, hoping to activate the mines so that they no longer pose a threat to area shipping traffic. Mine hunting, on the other hand, involves locating, detecting, and classifying mines for later neutralization. This neutralization might be done by the mine hunting vehicle itself, or it might be done by Navy divers or specialized vehicles. (For a more thorough introduction, see [25].)

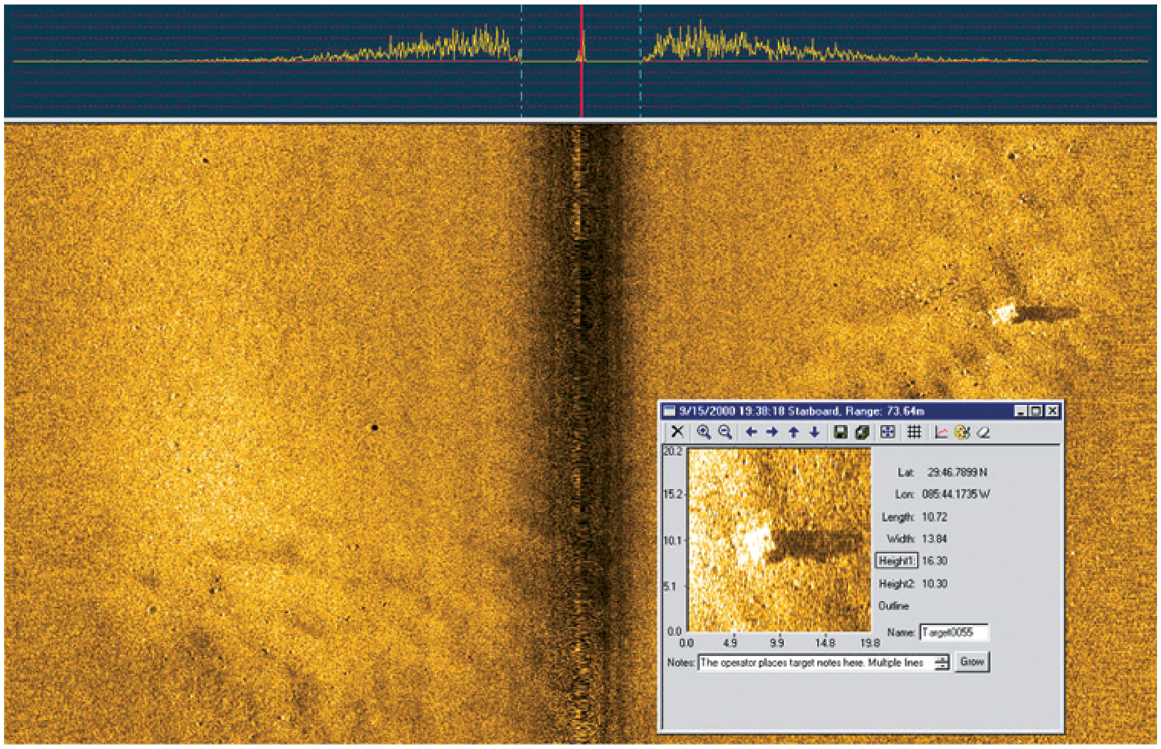
Minesweeping is typically done after the conflict is over. Its objective is to clear an entire area to allow commercial and military shipping to safely move through the area. Mine hunting, however, might be done in hostile waters, such as clearing the waters near an enemy beach for an amphibious assault. This means that minehunters (either vessels or divers) often need to be discrete and difficult to detect.

The earliest iterations of mine hunting involved Navy divers locating underwater mines, marking their locations, and setting explosive charges for later detonation. This process is very accurate with regards to the detected targets but is extremely time consuming, expensive, and also dangerous for covering large areas.

This process was improved with the invention of sonar which allowed single ships to map the seafloor. This sonar data was then processed into a graphical form that a trained operator would look at to mark the mine-like objects (see Figure 1-1). At this point it is beneficial to explain the terminology for mine hunting. Mine hunting involves two steps; the first step is *detection*. For the detection phase, an oil drum, a large rock, and a mine should all be "detected". The second step is *classification*. Classification is determining which objects are mine-like and which are not. For this example, the oil drum and mine could both be considered mine-like. These two phases are defined as mine-hunting. (To complete the description of mine countermeasures: the third phase, *identification*, seperates mines (the targets) from non-mines (the clutter). The first three phases are sometimes referred to as DCI. The fourth phase, *neutralization*, eliminates the threat from the mines.) [36]

The maturing of autonomous underwater vehicles (AUVs, or unmanned undersea vehicles, UUVs) initially automated the data-collection phase of mine hunting. That is, AUVs were programmed to follow a lawnmower style path over a search region, record data, and then return home. The AUVs were retreived and the data was downloaded to a computer where trained operators would look at the images to detect and classify mine-like objects. This automated data collection allowed a large search area to be covered efficiently, but still required operators to look over hundreds of images to identify mine-like objects. This procedure is tiring and demanding on the human operators.

The next advancement in mine hunting detection and classification was using computer image processing to pre-screen the images, assisting the operator by pointing out obvious targets. The automated detection and classification of mine-like objects is called CAD/CAC (computer-aided detection/computer-aided classification). Recent advances in image processing, combined with space and energy efficient computers (see Figure 1-2), have now allowed CAD/CAC to be performed onboard the AUV.


9/15/2000 19:38:18 Starboard, Range: 73.64m

Lat: 29:46.7899 N

Lon: 085:44.1735 W

Length: 10.72

Width: 13.84

Height1: 16.30

Height2: 10.30

Outline

Name: Target0055

Notes: The operator places target notes here. Multiple lines

Figure 1-1: MCM operators look for proud mine-like objects in side-scan sonar images. Detection and classification can be easy in simple terrains like this but increases in difficulty as the amount of clutter (such as rocks and other large debris) increase. Image courtesy Bluefin Robotics.

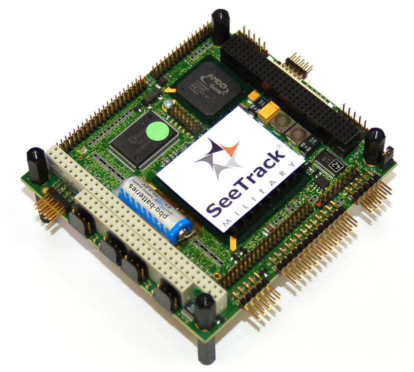


Figure 1-2: This small, power-efficient CAD/CAC module by SeeByte allows for the onboard detection and classification of mine-like objects on an AUV.

Currently, the only advantage that onboard CAD/CAC provides is that the AUV can communicate a “top-ten” list of targets *while still searching*, allowing vessels to begin identification and neutralization before the entire area has been scanned. The AUVs, however, do not use the information they have gained to alter their search patterns. A active area of research in this field is to try to close the loop and allow the AUV to autonomously make adjusts to its plan based on the information it is gathering.

1.3.2 Autonomous Underwater/Surface Vehicles (AUVs/ASV)

Autonomous underwater vehicles come in a variety of sizes, from 9-inch diameter man-portable units like Hydroid’s REMUS[1] all the way up to Bluefin’s 21-inch diameter BPAUV[2] (see Figure 1-3). The larger units are equipped with acoustic modems, more sophisticated sensors, larger batteries, and the capability of towing acoustic sensors. The smaller units are generally more energy efficient, but they often lack acoustic modems for underwater communication. All kinds of AUVs are capable of carrying some form of sonar for bottom sensing, including mine-like object detection.

Autonomous surface vehicles are now cheap and simple enough to operate that it is possible for a research group to have three to twelve of these vehicles for testing. The SCOUT platform (see Figure 1-4) of robotic kayaks cost around $30,000/unit, are equipped with GPS, a compass, a thruster capable of nearly four knots, and an off-the-shelf PC for autonomy. [18] Variants of the SCOUT have been fitted with such equipment as long-wave radios [34], acoustic modems [19] for communicating with underwater vehicles, and CTDs for taking water sound-velocity measurements. This inexpensive platform has allowed for the testing of some novel multi-vehicle experiments in collision avoidance. [10, 12]

[1] `http://www.hydroidinc.com/pdfs/remus100web.pdf`
[2] `http://www.bluefinrobotics.com/bluefin_21bpauv.htm`

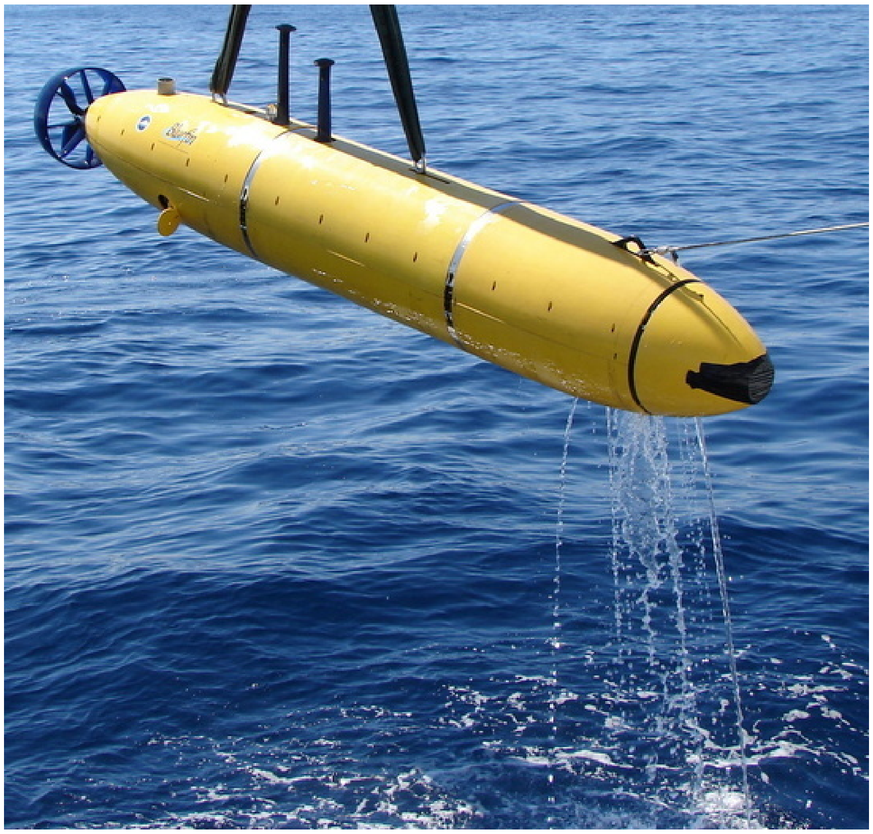


Figure 1-3: Bluefin's 21-inch diameter AUV is capable of diving exceptionally deep for long periods of time to take readings near the ocean floor. Advanced autonomy software (see Section 1.3.3) makes the independent operation of these vehicles possible. Image courtesy Matt Grund.

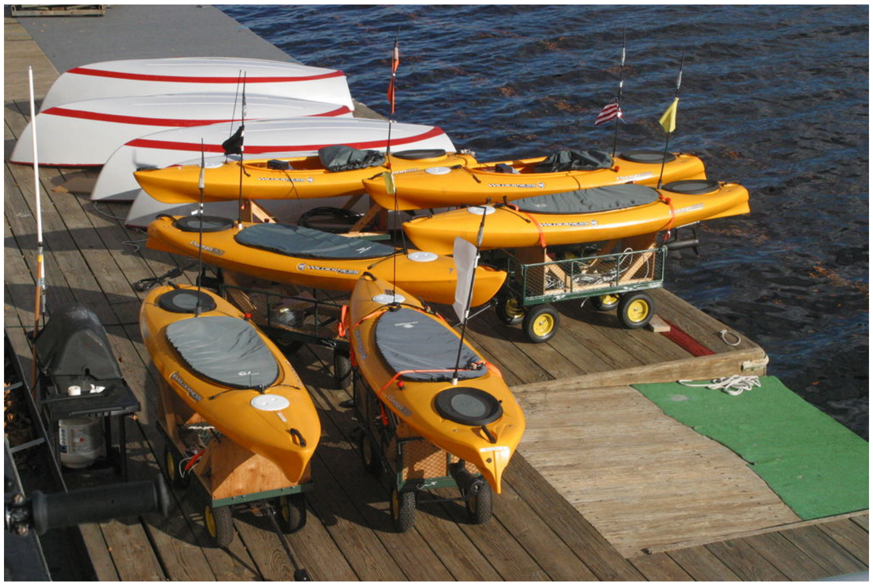


Figure 1-4: The SCOUT platform of autonomous kayaks are a relatively inexpensive robotic platform. The low cost and ease-of-use make it possible for research groups to own and operate several of these vehicles. Image courtesy Mike Benjamin.

1.3.3 Autonomy Software

The autonomy software for this project is a suite of software referred to as "MOOS," the Mission Oriented Operating Suite (by Paul Newman [31]), coupled with an interval-programming-based multi-objective optimization helm called pIvPHelm by Mike Benjamin [9]. The collective suite is referred to as MOOS-IvP.

MOOS is a platform-independent software suite that facilitates communication between processes that are typically written by different users. Each process gets information by talking *only* with the central MOOS database, MOOSDB (see Figure 1-5). The independence of each process from every other process prevents problems with one process from taking down the whole system. To share information, processes publish data to the MOOSDB in a (Name, Value) pair. Other processes that are interested in this data subscribe to the (Name) variable and are informed of updates to this variable by the MOOSDB. The modularity of this system enforces firm, clear interfaces between processes that desire to communicate with one another. For instance, the GPS software might publish variables called `GPS_LAT` and `GPS_LON` containing the latitude and longitude of the vehicle respectively and an in-

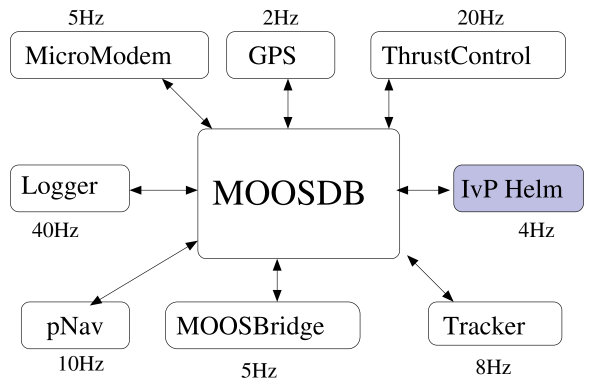


Figure 1-5: The Mission Oriented Operating Suite (MOOS) by Paul Newman is a middle-ware software suite that facilitates communication between different processes in a publish-subscribe architecture. Image courtesy Mike Benjamin.

ternal compass might publish `COMPASS_HEADING`. The navigation software (e.g., pNav) would subscribe to these variables and publish `NAV_X` and `NAV_Y`, its best estimate as to the location of the vehicle. Other software could subscribe to this navigation data and publish a desired heading and speed, while rudder and thruster control software could subscribe to the desired heading and speed data and actually interface with those physical controls.

MOOS, as robot middleware, does not provide much in the way of vehicle autonomy. The MOOS software distribution does provide a simple helm process called pHelm. Multi-objective optimization between behaviors is implemented in another MOOS process called pHelmIvP a.k.a. the IvP Helm. [8] In the MOOS publish-subscribe architecture, pHelmIvP subscribes to variables such as `NAV_X`, `NAV_Y`, `NAV_HEADING`, `NAV_SPEED`, and others (such as the position, heading, and speed of other vehicles) and typically publishes `DESIRED_HEADING` and `DESIRED_SPEED`.

pHelmIvP produces the desired heading and speed for a vehicle by efficiently weighing and optimizing the combination of objective functions produced by individual behaviors (see Figure 1-6). As a simple example, suppose a waypoint behavior produces an objective function that desires a course that would take the vehicle to a desired waypoint at a specified speed (see Figure 1-7). Also suppose there is a behavior that desires to limit the rate of turn of a vehicle to prevent kinking a tow cable. In some architectures, the helm would choose one behavior over another to control the vehicle. In our example this would either cause the vehicle never to reach the waypoint, or the vehicle to turn so sharp it kinks the cable. Other architectures would simply combine the single-decision output of each behavior. In our example this could violate the turn limit. pHelmIvP solves this multi-objective optimization problem by efficiently searching over the entire decision space (all discrete headings and speeds) and choosing the heading and speed that maximizes the combined objective funtions.

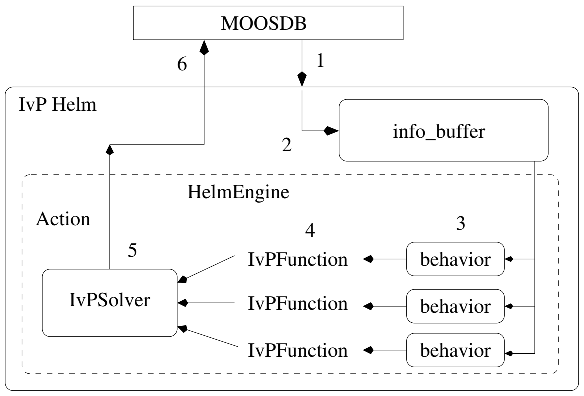


Figure 1-6: pHelmIvP uses the interval programming method to solve the competing objectives of different behaviors. Image courtesy Mike Benjamin.

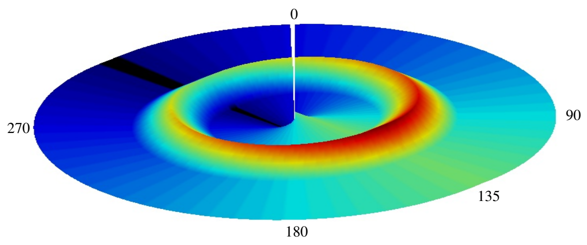


Figure 1-7: In this visualization of an objective for a waypoint behavior, the behavior would like to procede to a point at heading 135 degrees, but it is willing to take courses that take it near the point too. Image courtesy Mike Benjamin.

1.3.4 A Note on Coverage

In many applications of multiple vehicle problems, the goal is to "cover" an area with some kind of sensor. This includes such applications as mine deployment and mine hunting, robotic automobile painting [15], automated cleaning robots [30], and others. It is important to distinguish the different types of coverage that can be achieved. [20]

There are three basic types of coverage:

1. **Blanket or Field Coverage** is a static way of arranging your search elements such that you minimize the probability of a target appearing undetected inside the search area.

2. **Barrier Coverage** is a static arrangement of search elements that attempts to minimize the probability that a target passes undetected through the barrier.

3. **Sweep Coverage** is a dynamic arrangement of search elements that moves through the search area, attempting to both maximize the number of detections per time and to minimize the number of missed detections.

It is also important to recognize the capabilities of the targets when designing a coverage program. For static targets, any patterns that cover the entire area are essentially equivalent detection-wise. Moving targets, however, could conceivably do something like follow behind the searcher, preventing their detection even if the entire area is “covered” by the moving search elements. For adversarial moving targets like these, some form of moving barrier coverage is desired.

1.3.5 Relevance of Cooperative, Autonomous, Adaptive Search

The applications of this research are valuable for both military and civilian purposes. The Navy is currently funding research initiatives for cooperative search platforms that can search large (100-km by 100-km) areas. [32] There are also combined Navy and civilian efforts, such as PLUSNet, that are attempting to expand the communication capabilities of underwater vehicles. [24, 40]

1.4 A Novel Approach to Adaptive Autonomous Search

Current algorithms for mine hunting assume two features of the search vessel:

1. Mine hunting is the only task the vehicle is trying to complete, allowing the search algorithm to completely determine the actions of the vehicle.

2. The sensor's detection ability is constant for all regions of the search area and does not depend on environmental conditions or vehicle state (these parameters are referred to as the sensor's A and B values, see Section 2.1).

These two assumptions preclude using an adaptive algorithm for mine hunting. By violating these assumptions I propose a novel solution of decomposing the search area into smaller cells that allows for adaptive autonomous mine hunting. In particular, the algorithm is adaptive to changes in:

1. The mission, by allowing multiple objectives to be completed simultaneously, and

2. The environment, by adjusting the sensor's A and B parameters.

1.5 Criteria for Success

Defining success for search tasks is not easy and various papers use different definitions of success. [39] The elements that comprise an "efficient" search algorithm greatly depend on the objectives of the searcher. For example, efficiency might be defined based on the time of search, the search platform's cost or effectiveness, the number of missed and/or detected targets, or some other objective, such as finding a cleared path between two points.

That being said, success for this project will be defined as:

1. Creating a set of tools that enable a user to plan, simulate, and conduct a search operation over a specified area.

2. Conducting a series of simulations and experiments to show that these tools allow users to evaluation the effectiveness of their own search efforts.

1.6 Chapter Summary

In this chapter we have described the problem that current search algorithms face as being non-adaptive to real-world parameters such as the environment or mission objectives. We have also described mine countermeasures (MCM) in terms of detection, classification, identification, and neutralization phases. Several real autonomous vehicles and their capabilities were examined and some of the autonomy software that runs on them was described. The thesis of this work is that because current search algorithms assume that mine hunting is the only vehicle task and that the sensor parameters are constant, those algorithms are unable to adapt to changing environmental and mission parameters. To remove those assumptions, we use a cellular-decomposition method to create a framework that allows environmental and mission adaptation.

The next section describes mine countermeasure theory and how it applies to the cellular decomposition.

Chapter 2

Theory

2.1 Mine Countermeasure Theory

Planning for mine countermeasures (MCM) operations takes on a probability-based framework that trades the risk of missing targets for the time spent to cover the search area. [21, 36] The current objective of this planning operation is to determine the number of search tracks, N, across the search area of channel width C to attain the desired clearance level P (average probability of neutralizing mines) (see Figure 2-1).

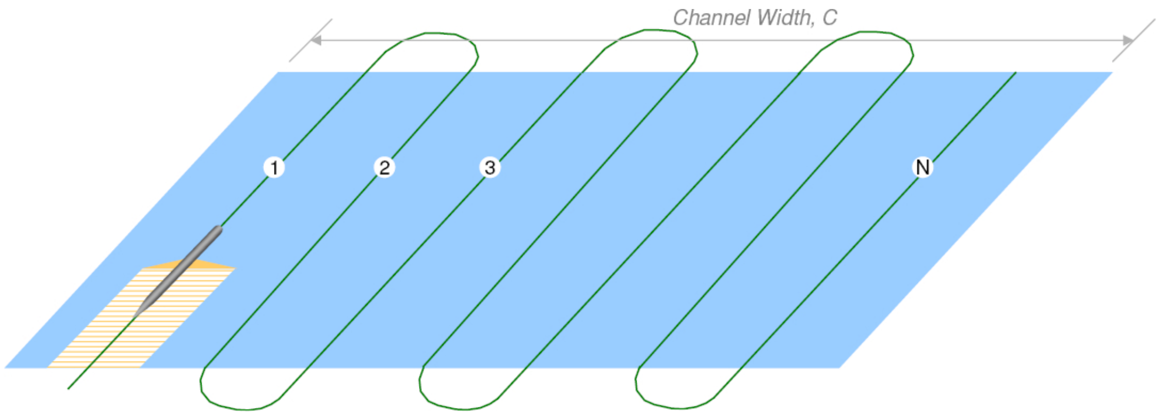


Figure 2-1: When planning for a mine countermeasures operation, the mission commander specifies a desired clearance level P, for the search area. The result of the planning is N, the number of search tracks to cover the entire channel width, C. Image courtesy Rafael Rodriguez.

The probability that at any instant a given searcher detects a given target depends on the relative locations of the searcher and target, the physical capabilities of the sensor, the physical characteristics of the target, and the environment. Because the probability of detection is very low for long distances between the searcher and target, it is possible to calculate the probability of detection for a target at a given lateral range (distance between a target the straight line trajectory of a searcher) by integrating the instantaneous probability of detection from $-\infty$ to ∞. See Figure 2-2.

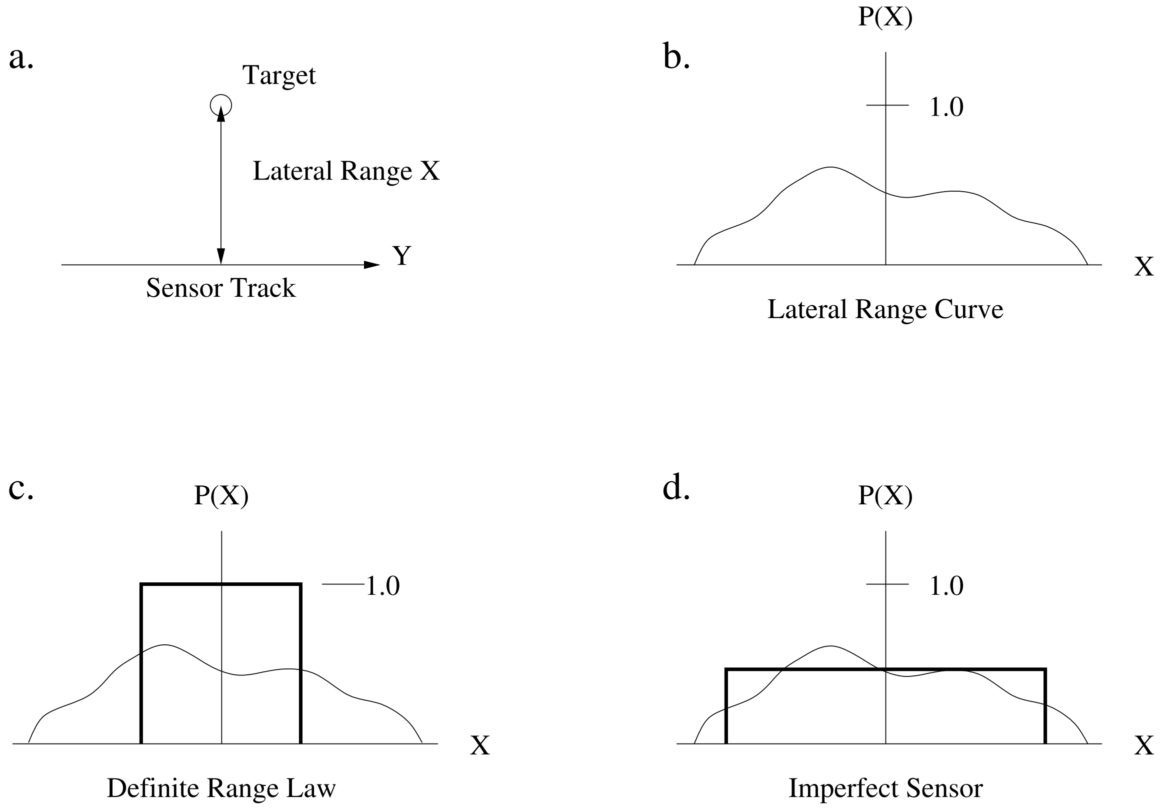


Figure 2-2: In a., the sensor passes along a target at lateral range X. Integrating the instantaneous detection probability (a function of X and sensor and target characteristics), we obtain a fictional b., the lateral range curve. This curve goes to 0 for large X and may not be symmetric, depending on the sensor configuration. Approximations for b. include c., which assumes that all targets within a fixed distance are detected (cookie cutter), and d., which more accurately models the sensor's performance (imperfect sensor). Image originally appears in [21].

For our purposes then, a sensor can be completely described by its nominal detection probability, B, and its nominal range (width), A. See Figure 2-3.

The mission commander for an MCM operation specifies a desired clearance level, P, which gives the probability that any particular mine has not been neutralized. For

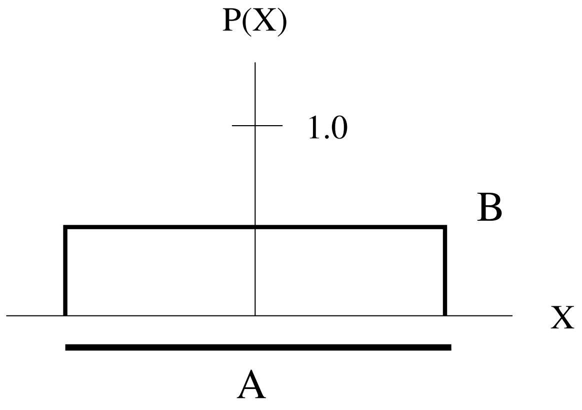


Figure 2-3: This model completely describes the sensor using its detection probability, B, and its range, A.

example, if the mission commander desires 96% clearance, that allows a 4% chance each mine has not been neutralized. In a field of 100 mines, it would be acceptable for four of them not to be neutralized.

With a perfect sensor ($B = 1$), the clearance level is just equal to the portion of the search area covered by the sensor (see Equation 2.1). For an imperfect sensor, the probability of missing any particular mine is inversely proportional to the number of times the entire area has been scanned.

$$P = \begin{cases} \max(N * A/C, 1) & B = 1 \\ 1 - (1 - B)^{N*A/C} & B < 1 \end{cases} \tag{2.1}$$

Further complicating clearance operations is that the probability of identification $P_{id} < 1$, the probability of neutralization $P_n < 1$, and there are some mines that cannot be detected, μ. (See Figure 2-4) If the commander desires an overall clearance

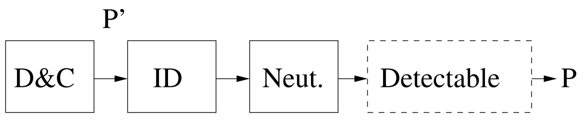


Figure 2-4: Modifications need to be made to P to account for limits in P_{id}, P_n, and μ. That is to say, if the commander desires 90% clearance and the product of P_{id}, P_n, and $1 - \mu$ is 97%, we can set the desired clearance level of our algorithm, P' to $90\%/97\% = 92.7\%$ to satisfy the commander's goal.

level of P, we can correct for these factors by increasing the coverage to

$$P' = P/(P_{id} * P_n * (1 - \mu)) \tag{2.2}$$

Because we can correct for these factors by increasing P, we will ignore any complications caused by these factors. In the case where $P > (P_{id} * P_n * (1 - \mu))$, no number of paths can satisfy the desired clearance level and a later minesweeping effort will have to be conducted.

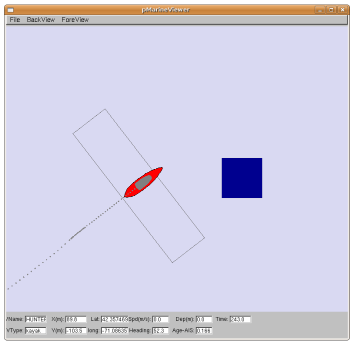
2.2 Applications to Small Cells

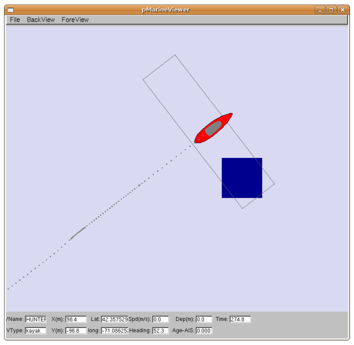
The previous discussion applies to a large search area where it makes sense to talk about the number of paths that cover the search area. It is possible to apply this reasoning to small cells, which is at the heart of what we are trying to do.

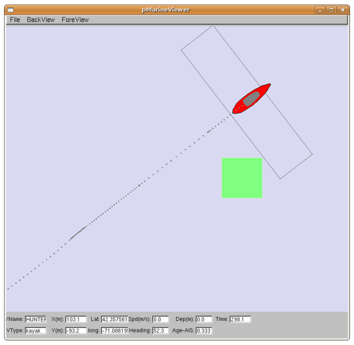
Recall that the clearance level, P, is an average of $1-$(the probability that an undetected target is present). Also recall that B is the probability that the sensor will detect a given target as it passes by the target. Importantly, it is assumed that detection events are independent for each target and for each pass of the sensor over the target. However, this assumption might not hold for some targets or environments. For example, some targets have a strong directionality which makes them easily detectable from one direction but not from another. Applying this to a small cell, every time a cell comes into view and then leaves, we must update the clearance level of that cell. Assuming detection independence, the equation for updating a cell that has been sensed is in Equation 2.3.

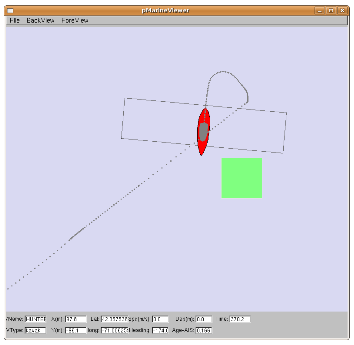
$$\begin{aligned} \text{new prob. missed} &= \text{(prob. prev. missed)} * \text{(prob. missed this time)} \\ 1 - P_{new} &= (1 - P_{old})(1 - B) \\ P_{new} &= 1 - (1 - P_{old})(1 - B) \end{aligned} \tag{2.3}$$

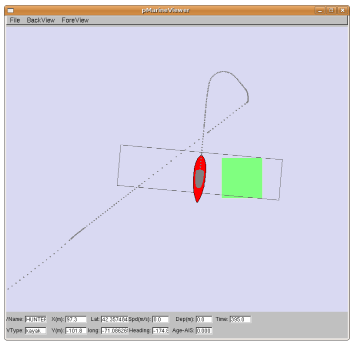
Figure 2-5 shows a graphical procession of what happens when a sensor is passed over a cell multiple times.


(a)


(b)


(c)


(d)


(e)

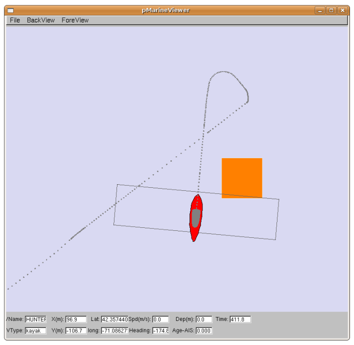

(f)

Figure 2-5: In this figure, a sensor is passed over a single cell. As the sensor enters the cell (b), nothing is changed. When the sensor has completely passed the center of the cell (c), its clearance is updated according to Equation 2.3. Here, the cell has changed from 0 clearance (blue), to .5 clearance (teal). On the second pass, (d)-(f), the cell is updated from .5 to .75 (orange); which is two passes of a .5 sensor. These updates are communicated to other searchers to adjust their own internal grids.

The average clearance for a group of cells is the average of each cell's clearance weighted by its relative area. For equal-sized cells, this is equivalent to summing the clearances and dividing by the number of cells.

Average Clearance $P = \frac{\sum_{i=1}^{N} P_i}{N}$

2.3 Signal Detection Scoring Metrics

The scoring metric used to evaluate a potential search algorithm greatly influences the design of that algorithm. For instance, with a metric that greatly punishes missing targets and only lightly penalizes incorrectly declaring a target, the algorithm may decide to declare many of the objects it sees as targets. In addition to the standard signal detection theory scoring (see Table 2.3), the score might be affected by factors such as the time of search (penalizing long search times), the clearance level, the number of search vessels used, amount of energy expended, or the completion of certain objectives, such as finding a lane [33] from one side of the area to the other.

While not explored in this research, it would be beneficial to try to create a scoring metric that uses classic signal detection theory but that also incorporates each cell's clearance level, P, when assigning points. For example, if a cell has a high clearance level, meaning that not many targets should be missed there, the penalty for misses would be increased.

Table 2.1: Basic signal detection theory terminology. Here, there are two states for each cell, either there are targets present or there are no targets present. There are also two possible outputs of the search effort, declaring a target present or not present.

	Target Present	Target Not-Present
Declare Target	Hit	False Positive
Declare No Target	Miss	Correct Rejection

To simplify the calculation of scores, we will use classic signal detection scoring. We will hold the other possible scoring variables (such as time or clearance level) constant to compare search algorithms. The actual scoring tool/metric is discussed in Appendix B.3.

2.4 Chapter Summary

In this chapter we have given an overview of the current state of mine countermeasure theory, including the current goal of operations planning of calculating N, the number of paths to use when covering a channel of width C. The clearance level, P, is a measure of how likely it is that a target has not been neutralized. This clearance level can be calculated for an entire search area (which is currently done), or it can be applied to a cellular decomposition of the search area (this research). Finally, the scoring metric used to evaluate a search algorithm greatly influences the design of that algorithm. For this research, we use classic signal detection theory while holding other variables (such as time or clearance level) constant.

In the next chapter we describe the software modules that implement this cellular decomposition framework and the search algorithms.

Chapter 3

Software Framework and Modules

The principle development work for this project is the creation of three software modules: pSensorSim, pArtifactMapper, and bhv_SearchArtifact. These software modules form a chain that starts with simulated targets and ends with desired heading and speed commands for the vehicle. pSensorSim loads the artifacts to simulate and publishes the artifacts that it detects. pArtifactMapper subscribes to these artifact updates and forms a clearance map of the area of interest. The search behavior then subscribes to this clearance map and makes decisions about where the vehicle should go next. See Figure 3-1 for a diagram of the relationship. Any of the modules can be replaced with components that follow the same interface. For example, the sensor model could be replaced with the actual sensor hardware, and the search behavior could be replaced with any other search algorithm. Configuration information for all of the software is in Appendix B.

3.1 pSensorSim

pSensorSim is a MOOS process that simulates the output of a fictional sensor given a "threat laydown," which is a list of targets to simulate.

The algorithm for sensing is shown in Algorithm 1. The basic process is to maintain a list of artifacts that can currently be "seen" by the sensor and store it. On the next iteration, for each *new* artifact, pick a random number [0, 1). If the random

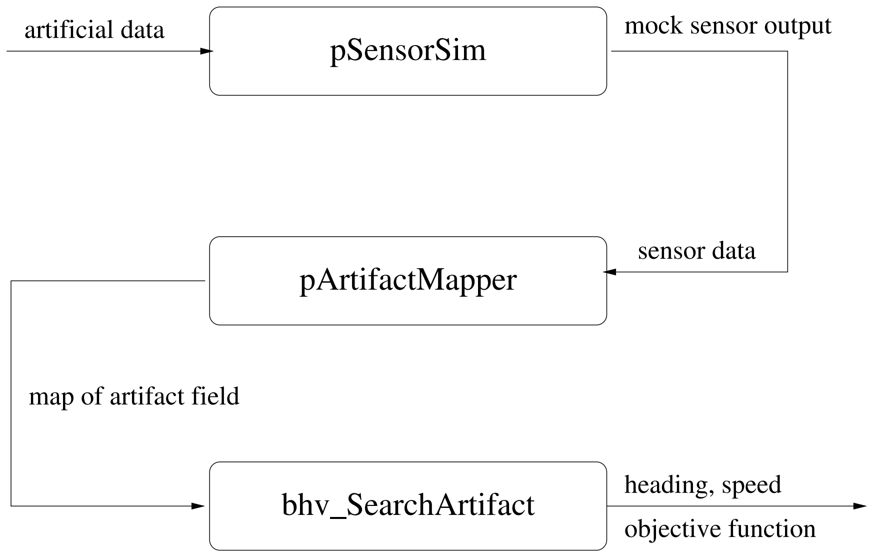


Figure 3-1: This diagram displays the flow of data between the three parts of the search toolchain. pSensorSim models a simple sensor and outputs the artifacts that it detects. pArtifactMapper remembers this output and also keeps track of which cells the vehicle has covered. This artifact map is used by the search behavior (bhv_SearchArtifact) to produce an objective function for the helm to solve.

number is less than B (the sensor detection capability), emit the artifact as a detected artifact for processing by other modules. Now store the list of all seen artifacts for the next iteration.

Another essential function of pSensorSim is to simulate variations in the sensor's A and B values. Recall that A is the effective width of the sensor while B is the probability that the sensor will detect any given target. pSensorSim uses a simple two-value sensor B parameter that changes the sensor's normal value to a reduced value for a short period of time. For example, the sensor might have a B value of .8 for 60 seconds and then drop to .25 for 10 seconds before repeating the cycle. There is plenty of room for further work in modeling a sensor's B value given the state of the vehicle and the bottom environment. See Chapter 5.2.

3.2 pArtifactMapper

pArtifactMapper's role is to maintain two data structures: 1) a list of all the detected artifacts, and 2) a clearance level associated with each cell of the search grid.

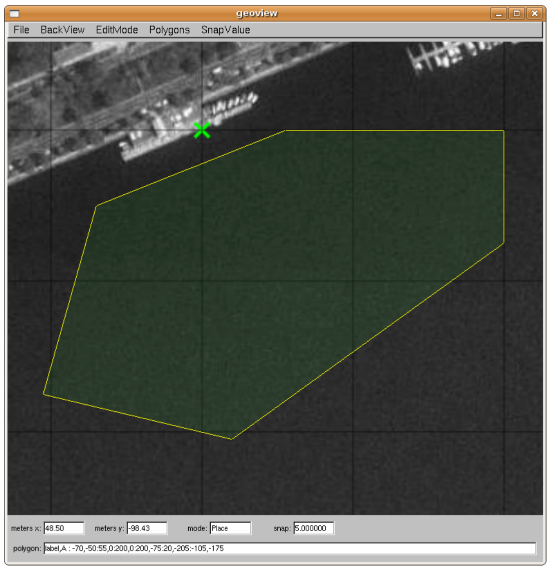
```
input  : threat laydown
output: detected artifacts
load(threat laydown);
locArtifacts ← Detect($currx, curry, currhdg$);
while $true$ do
    newArtifacts ← Detect($currx, curry, currhdg$);
    foreach artifact $in$ newArtifacts $and not in$ locArtifacts do
        x ← Rand($[0, 1)$);
        if x < $B$ then
            Publish(artifact);
        end
    end
    locArtifacts ← newArtifacts;
    /* Publish the sensor values so pArtifactMapper can adjust its
       internal values                                            */
    Publish(Sensor $A$);
    Publish(Sensor $B$);
end
```
Algorithm 1: Algorithm for pSensorSim. See Section 3.1 for a full description.

First, some terminology. The *search area* is the physical region of interest, defined by a convex polygon. The convex polygon requirement is not a function of anything that pArtifactMapper can perform, but rather because the search behavior, which shares the map, currently only functions with convex polygons. This requirement could be relaxed with a reworked behavior.

3.2.1 Discretization Method

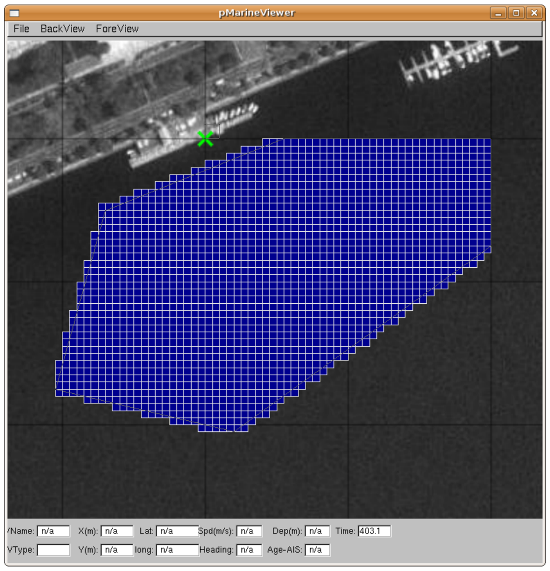
The search area polygon is completely tiled with equal-sized, square elements, forming a *search grid.* The total area of the search grid fully includes the area of the search area. See Figure 3-2. The size of the cells is specified by the user. For most of this work we have been using 5-m squares. This number seems to be small enough that it is within the error bounds of a sensor (approximately 8-m [41]), but large enough that the number of cells is manageble.

Other constraints might limit the maximum number of cells that can be described. For instance, if an AUV allocates 10-bits to describe the cell index number, the number of cells is limited to 1024 (2^{10}), regardless of an optimal size for the cell.



meters x: 48.50 meters y: -98.43 mode: Place snap: 5.000000
polygon: label,A : -70,-50:55,0:200,0:200,-75:20,-205:-105,-175

(a) A search area defined by a convex polygon.



VName: n/a X(m): n/a Lat: n/a Spd(m/s): n/a Dep(m): n/a Time: 403.1
VType: Y(m): n/a long: n/a Heading: n/a Age-AIS: n/a

(b) The search grid that completely covers the search area.

Figure 3-2: The search area, defined by a convex polygon (a), can be broken into discrete, equal-sized, square elements that completely cover the area (b)

Also, square cells might not be the best shape for the elements. Some discretization methods use hexagonal cells because of their close-packing ability and because it removes ambiguity as to which cells are considered neighbors. [42]

3.2.2 Grid Updates

When pArtifactMapper receives a detected artifact notice from pSensorSim, it stores the artifact in the cell location that corresponds to the artifact's X and Y values.

Using the theory developed in Section 2.2, for all cells that have entered the sensor's effective area and then left (i.e., those cells have been scanned), those cells have their clearances updated.

$$\begin{aligned} P_{new} &= 1 - (1 - P_{old})(1 - B) \\ P_{new} &= P_{old} + B(1 - P_{old}) \end{aligned}$$

After updating a cell, this information needs to be communicated to other searchers as a grid update. Grid updates take the form of index, prev_clearance, new_clearance. The previous clearance is included in the update in case platforms want to try to reconstruct a sequence of grid updates. To communicate these updates to other vehicles, other MOOS processes deliver the message in whatever form is appropriate. For communication over TCP/IP channels, pMOOSBridge forms a connection to both MOOSDBs and pushes updates in one direction. Underwater, acoustic modems are used to communicate data at slow rates. Figure 3-3 shows how one community might be passing these updates.

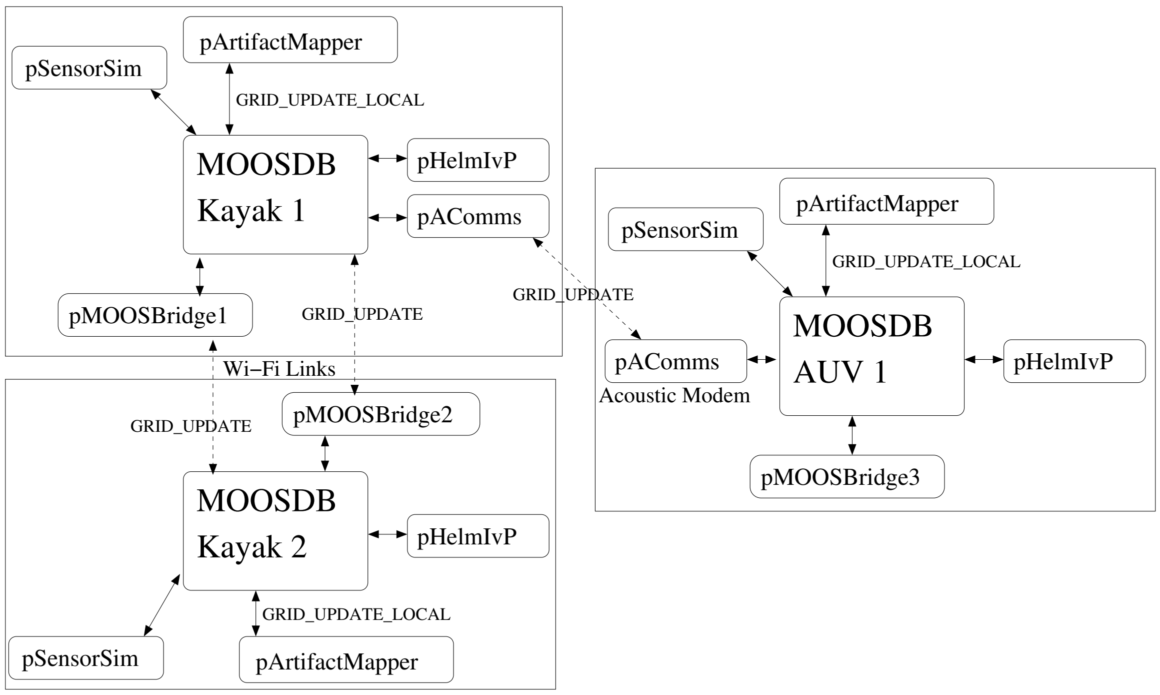


Figure 3-3: This diagram displays one possible way for grid updates (and other MOOS variables) to be communicated between vehicles. For the two surface vehicles with Wi-Fi connections, pMOOSBridge maintains a TCP/IP connection to both databases and is able to publish updates from one MOOSDB to another. In this example, one of the kayaks is outfitted with an acoustic modem to communicate with the AUV underwater. A different set of MOOS processes would be responsible for formatting the updates and ensuring delivery (summarized as pAComms). Our simulations and experiments were conducted with kayaks equipped with Wi-Fi so pMOOSBridge was used to communicate between vehicles.

The algorithm for pArtifactMapper is summarized in Algorithm 2.

```
input  : detected artifacts, Sensor $A$, Sensor $B$, grid configuration
output: clearance map
$P \leftarrow 0$;
locCells $\leftarrow$ SeeCells(currx, curry, currhdg, Sensor $A$);
while true do
    if detectedartifact then
        cell.add(artifact);
    end
    newCells $\leftarrow$ SeeCells(currx, curry, currhdg, Sensor $A$);
    foreach cell in locCells and not in newCells do
        $P_{old} \leftarrow$ cell.Clearance();
        $P_{new} \leftarrow 1 - (1 - P_{old})(1 - \text{Sensor } B)$;
        cell.Clearance() $\leftarrow P_{new}$;
        Publish (index, $P_{old}$, $P_{new}$);
    end
    locCells $\leftarrow$ newCells;
end
```

Algorithm 2: Algorithm for pArtifactMapper. See Section 3.2 for a full description.

3.3 Search Behaviors

The search algorithm is arguably the most important component of an adaptive search platform. For example, certain sensors, such as side-scan sonar, require the vehicle to travel in a fairly straight line in order to get good data. Other sensors, such as magnetic field gradiometers, have little dependency on turn rate. A search algorithm that tries to turn frequently would result in poor data from the side-scan sonar but would not affect the magnetic field gradiometer. Conveniently, in this framework the search algorithm is also the most replaceable component.

Because of the prevalance of side-scan sonar for mine hunting tasks, we will assume that turns are to be avoided to minimize lost data collection opportunities.

Our example will utilize two vehicles, one that mimics an AUV and performs a simple lawnmower behavior (bhv_Lawnmower), and a second vehicle that mimics a surface craft (ASC). This scenario is plausible because when an ASC and AUV opperate in cooperation, both vehicles benefit. The AUV gets accurate location updates from the ASC without wasting time and energy to surface, while the surface craft is able to act as a relay for reliable, fast, efficient communications with the AUV.

Our vision for how these vehicles would cooperate is to have the AUV perform a standard lawnmower pattern over the search area while the ASC tries to perform two objectives. First, it needs to stay close to the AUV for reliable communications and location updates. Second, because it knows something about the patches that the AUV is "missing," or cells that still have low clearance levels, the ASC can try to go out of its way to cover those cells.

pHelmIvP's interval programming architecture allows us to easily find a solution for these competing objectives. See Figures 3-4 and 3-5 for an example.

3.3.1 bhv_SearchArtifact

bhv_SearchArtifact is a simple greedy algorithm that chooses the desired heading and speed based on which choice gives the most delta in coverage.

We start with what the delta for an individual cell is:

$$\begin{aligned}
\Delta P &= 1 - ((1-P)(1-B)) - P \\
&= 1 - (1 - P - B - PB) - P \\
&= B - BP \\
\Delta P &= B(1-P) \qquad (3.1)
\end{aligned}$$

Intuitively, this makes sense, because ΔP is just the effect of passing a B-grade sensor over an area that has a $(1-P)$ probability of having something undetected in that cell.

Now for any given speed and heading, we can calculate which cells will be covered within a time horizon. We then add the ΔP values for each cell in that set to get the utility of travelling in that heading at that speed.

This algorithm is summarized in Algorithm 3.

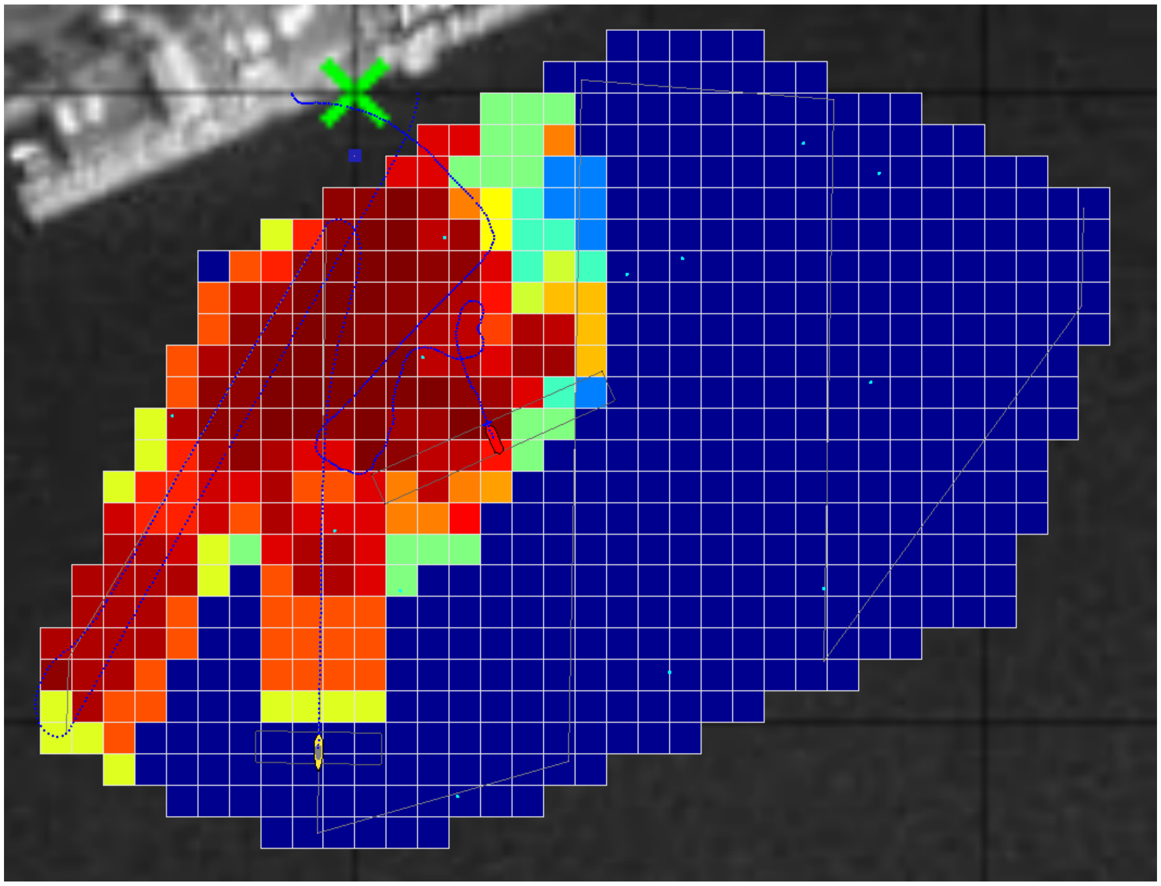
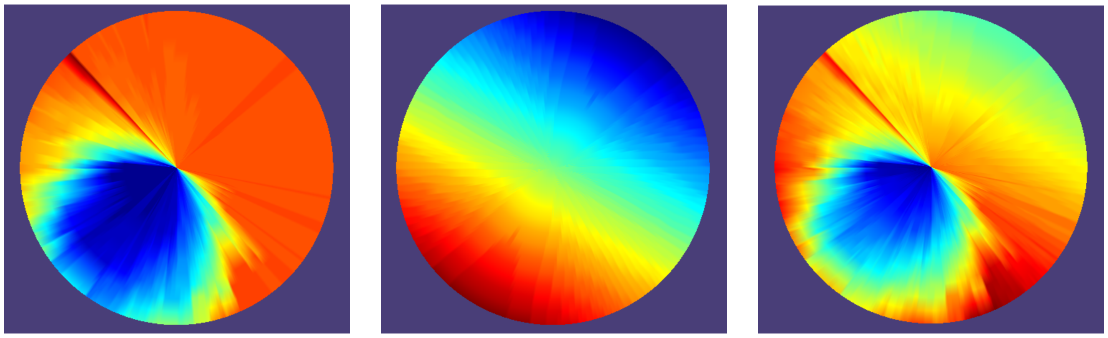


Figure 3-4: This is an example setup that demonstrates how multiple objectives can be satisfied simultaneously by the interval programming method of pHelmIvP. The satellite underlay is from the MIT Sailing Pavilion (the structure in the upper left). The faint, light-blue dots are artifacts published by pSensorSim in the threat laydown. Lines in the image are the boundaries between cells of the search grid. The color of each cell represents its clearance level, dark blue is 0 clearance, dark red is nearly 100% clearance. The dark blue trails behind the vehicles represents the path that vehicle has travelled. The yellow vehicle (near the bottom) is running a lawnmower behavior. The red vehicle (near the middle) is running both a search behavior and a cut range behavior to the other vehicle. See Figure 3-5 for the objective functions in play.



(a) bhv_SearchArtifact (b) bhv_CutRange (c) Collective Function

Figure 3-5: In this figure, two behaviors are active in the setup of Figure 3-4. The plots represent objective functions produced by each behavior. The plots are defined over heading and speed with higher speeds further out radially from the center. Colors represent the utility with red representing high utility and blue lower. (a) is bhv_SearchArtifact. It is expressing desired headings and speeds that cover the unexplored cells. (b) is a Cut Range behavior that is expressing desired headings and speeds that take the vehicle closer to the target vehicle. The influence of this behavior (weight) increases with increasing distance, making it more important when the vehicle is far from the target. (c) is the weighted sum of the other two functions. The helm efficiently finds the peak of this collective function. In this case, the desired behavior is to travel at top speed (1.0 m/s) at heading 156.0 (0 is due North, 90 is due East, etc.).

```
input  : grid configuration, grid updates
output: heading, speed objective function for pIvPHelm decision

/* Reduce decision space from $n$ cells to smaller range          */
locCells ← Cells within TopSpeed $*$ TimeHorizon;

/* Compute one-time costs for each cell                           */
foreach cell in locCells do
    Compute($\Delta P$, range, cell heading)
end

/* Pre-compute which cells are intersected at each heading        */
foreach Heading do
    foreach cell in locCells do
        if cell is within coverage at TopSpeed then
            HdgVector.add(cell)
        end
    end
end

/* Compute utility for each Heading and Speed                     */
foreach Heading and Speed do
    Util ← 0;
    foreach cell in HdgVector do
        if cell is within coverage at Speed then
            Util += $\Delta P$
        end
    end
    return Util
end

/* Scale range so the maximum is 100 and the minimum is 0         */
ScaleUtilities(0, 100);
return Utilities;
```

Algorithm 3: Algorithm for bhv_SearchArtifact. See Section 3.3.1 for a full description.

3.3.2 The bhv_Lawnmower Behavior

The bhv_Lawnmower behavior is a component of the IvP helm that implements the lawnmower pattern generation algorithm described in Appendix B.1. It is basically a waypoint following behavior that takes as inputs the parameters of the pattern (search area, path radius, path heading, etc.) instead of a list of waypoints.

3.4 Auxillary Tools

pLawnmower is a helper utility that can generate lawnmower patterns for other MOOS processes that would like to use them. It takes as input a polygon to generate the lawnmower pattern in and some parameters such as the starting point, initial heading and turn direction, and path radius. The algorithm is described in Appendix B.1.

The artifact field generator (artfieldgen) takes a polygon and number of targets and generates a uniform random distribution of targets within the polygon. We use this to create our threat laydowns prior to mission execution. The algorithm is described in Appendix B.2.

scoreartfield takes in a file dump from pArtifactMapper (containing clearances and detected artifacts), a threat laydown or actual target location list, and a scoring metric (points for each hit, miss, false alarm, and correct rejection) and outputs the score. The scoring utility using classic signal detection theory and is described in Appendix B.3.

3.5 Chapter Summary

In this chapter we have described the three major tools produced in this work: pSensorSim, pArtifactMapper, and bhv_SearchArtifact. pSensorSim simulates a fictional sensor by publishing the artifacts that it detects. pArtifactMapper subscribes to this data and creates a clearance map of the search grid. bhv_SearchArtifact subscribes to this map and produces a desired heading and speed for the vehicle that greedily tries to explore low-clearance cells in the search grid.

In the next chapter we will test the functionality of these tools by simulating some multi-vehicle scenarios and experimentally validating them.

Chapter 4

Simulated and Experimental Mission Validation

For our experimental vehicles we used three kayaks of the SCOUT platform [18]. The kayaks are equipped with Garmin 5 Hz GPS sensors for location information. Because these particular vehicles lacked compasses, the heading data was taken from the GPS unit which uses an unknown but probably difference between the last two measurements algorithm for computing the heading. This turned out to be a big disadvantage because the GPS heading data only becomes reliable at speeds greater than approximately 1 m/s. The effects of noisy heading data are described in section 4.1.

For all of the missions below, the scoring metrics in Table 4.1 were used. Because our sensor does not produce actual false alarms,[1] the overall score for an algorithm is directly proportional to the average clearance level. We also used a constant threat-laydown of 30 uniform-randomly placed targets in a polygon of 15,100 m^2.

In all, seven mission simulations and nine experimental missions were analyzed. We divided the missions into five scenarios. The results are summarized in Table 4.2. Detailed analysis for each mission type is presented in the sections.

[1]The false alarms listed in the mission results are from situations where a target exactly on the boundary between two cells is considered to be "in" both cells by pSensorSim but only in one of the cells by the scoring utility.

Table 4.1: The scoring metrics used for evaluating the performance of the algorithms are listed here. Misses are heavily penalized (reflecting a high cost of not knowing about targets) while false alarms are mildly penalized (reflecting the relative ease of re-evaluating targets).

HitPoint:	10
MissPoint:	-1000
FalseAlarmPoint:	-10
CorrRejPoint:	10

Table 4.2: Summary of Simulations and Experiments: This table summarizes the results gathered from both simulated and experimental testing of different parameters for cooperative, autonomous searching. In the time column, faster is better. Simulations were typically faster because the vehicles have a more difficult time turning on the water. In the score column, higher is better. Because our scoring metric gave -1000 points for each missed target, missing a few targets quickly put some scores into negative values. The first four scenarios all used the same vehicle and sensor parameters. In the two missions of the fifth scenario, one sensor was made to be wider and the other sensor was made more accurate.

Scenario, Page No.; Summary	Time [Min:Sec]	Score
4.1, pg. 51; A fast-moving search kayak follows a slow-moving AUV. Both vehicles have a 20-m, .5 B sensor with drops to .25 B for 10 seconds out of 30. We discover that slew-rate limiting does not function properly during experiments because of the noisy heading data.	Sim: 16:42 Exp: 21:20	Sim: -1090 Exp: -1820
4.2, pg. 53; We repeat scenario 4.1 but we turn off the slew-rate limit. This allows us to collect data, but experiments overstate the ability of the sensor by over-scanning cells because of noisy heading data.	Sim: 16:11 Exp: 22:00	Sim: 1940 Exp: 3960
4.3, pg. 55; We remove adaptivity from the vehicles by having both perform lawnmower patterns over the search grid. This holds time constant to allow us to compare clearance levels.	Sim: 17:11 Exp: 24:00	Sim: -2060 Exp: 3980
4.4, pg. 59; We again remove adaptivity from the vehicles but this time each vehicle runs a pattern over half of the search area.	Sim: 11:00 Exp1: 14:00 Exp2: 12:00	Sim: -2100 Exp1: -2100 Exp2: -6120
4.5.1, pg. 63; The lead AUV performs a lawnmower pattern with a 40-m wide, .5 B sensor while the kayak follows behind with a 20-m wide .8 B sensor.	Sim: 10:48 Exp1: 12:52 Exp2: 12:45	Sim: 1980 Exp1: 3010 Exp2: 930
4.5.2, pg. 65; We reverse the previous experiment and have the kayak lead with a wide sensor while the slower AUV follows with the more accurate sensor. Experiment 1 used a 10-m swath for the lawnmower pattern while Experiment 2 used a 20-m swath.	Sim1: 14:09 Exp1: 16:52 Sim2: 7:37 Exp2: 10:07	Sim1: 970 Exp1: 5980 Sim2: -2100 Exp2: -40

4.1 Scenario 1: An AUV with a Kayak Tender

In this scenario we model an AUV being followed by an ASC tender (in this case, an autonomous kayak). As described in Section 3.3, this scenario is plausible because it allows the kayak to provide reliable, fast communications between mission control and the AUV.

Both vehicles have 20-m wide, .5 B sensors with drops to .25 B for 10 seconds every 30 seconds. Importantly, the slew-rate limit feature of pSensorSim and pArtifactMapper is set at 10 degrees/sec. The sensor and mapper return no results when the vehicle is turning beyond this limit. Slew-rate is defined as the difference in heading data points divided by the time difference between those two points. For noisy data collected at a fairly fast rate (5 Hz GPS sensor), the noise quickly overwhelms the slew-rate.

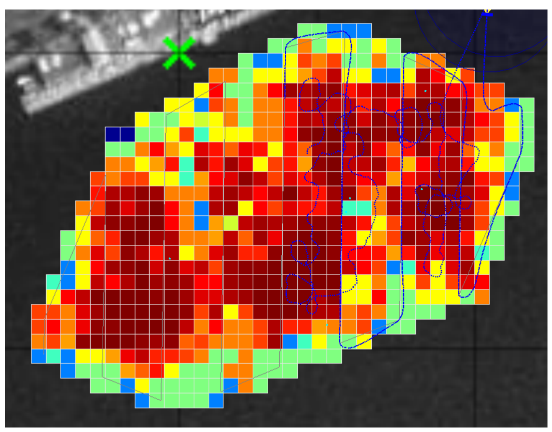
Vehicle 1, the AUV, travels at 1.0-m/s and is setup to perform a lawnmower pattern with a 10-m radius (half the sensor width).

Vehicle 2, the kayak, travels at 1.5-m/s and runs both bhv_SearchArtifact and bhv_CutRange to the AUV.

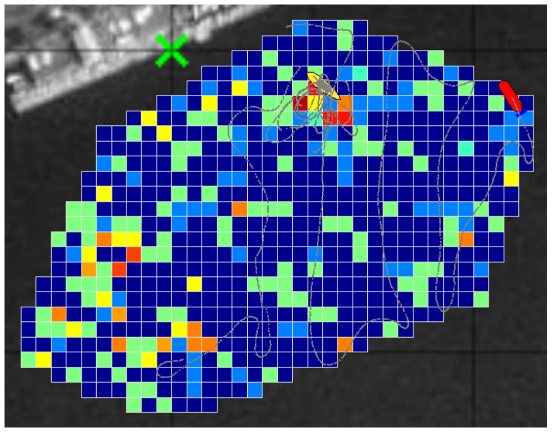
Figure 4-1 shows the results of both a simulated run (a) and the experimental run (b).

As can be seen in the figure, noisy heading data cause the vehicle to believe that it was turning much more than it actually was. The sensor and mapper both interpreted this as turning too fast and did not provide regular data.

The results of these missions are presented in Table 4.3. Notice especially how in simulation the clearance was a reasonable 78%, but that in the actual water mission the performance was a miserable 18.7%. This result shows the importance of trying to experimentally validate results that have been shown in simulation.



(a)



(b)

Figure 4-1: In this figure, the sensor and artifact mapper are configured with a slew-rate limiter that prevents them from outputting data when the vehicle is turning faster than 10 degrees/sec. In simulation (a), this works fine. In the experiment on the water (b), however, the noise in the heading data causes no data to be returned. For the rest of the missions we were forced to turn off the slew-rate limit. This also means that the rest of the missions over-state the clearance of cells that pop in and out of the sensor range because of heading noise. See mission 4.2.

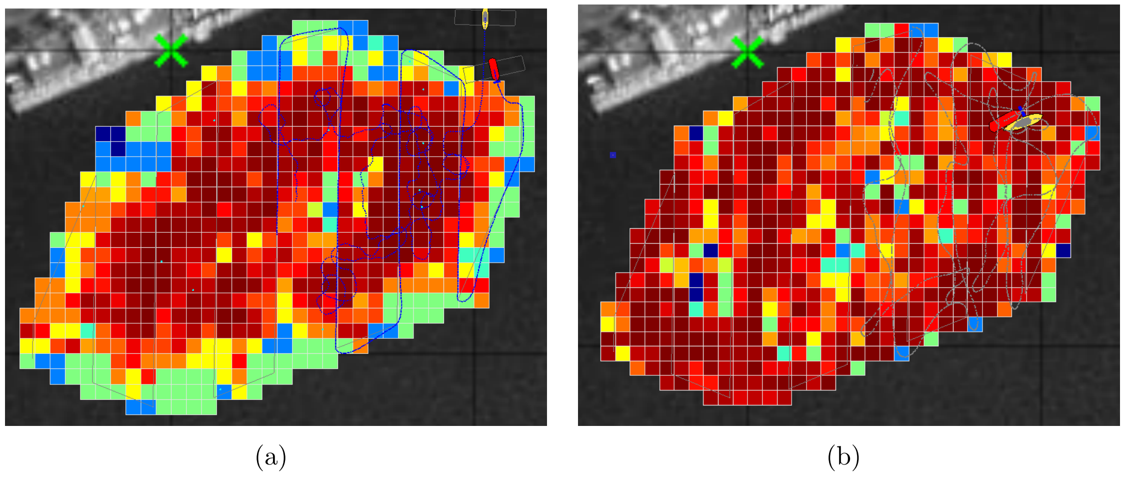
Table 4.3: Simulation and experimental results for mission 4.1

	Simulation	Experiment
Score:	-1090	-1820
Time:	16:42	21:20
Ave. Clearance:	77.9%	18.7%
Actual Clearance:	70.0%	13.0%
Hits:	21	4
Misses:	7	24
False Alarms:	4	1
Correct Rejs:	574	577

4.2 Scenario 2: An AUV with a Kayak Tender, No Slew-Rate Limit

In this scenario we repeat the parameters from scenario 4.1 but with the slew-rate limit turned off. The vehicle now scans cells more easily, especially while turning or when the noisy heading reading causes cells to "pop" in and out of view of the sensor. This cell "popping" occurs because when the slew-rate is multiplied by the half-width of the sensor, the distance that the tip of the sensor moves can be significant. This moving edge can quickly cover and uncover a cell, causing pArtifactMapper to update the clearance of that cell multiple times. This effect could be corrected by cleaning the heading data or using some other metric to determine when a cell has been cleared.

Figure 4-2 shows the effect of turning slew-rate limiting off for both the simulated mission and the experiment. Table 4.4 summarizes the results.



(a) (b)

Figure 4-2: In this figure, the sensor and artifact mapper are configured without the slew-rate limiter. (a) shows the simulation results while (b) shows the experimental results.

Comparing the results for mission 4.1 and 4.2 in Tables 4.3 and 4.4, we see that the effect of turning off the slew-rate limit is minimal for simulations (77.9% vs. 76.7%) but significant for experiments (18.7% vs. 87.1%).

Table 4.4: Simulation and experimental results for mission 4.2

	Simulation	Experiment
Score:	1940	3960
Time:	16:11	22:00
Ave. Clearance:	76.7%	87.1%
Actual Clearance:	80.0%	86.6%
Hits:	24	26
Misses:	4	2
False Alarms:	4	4
Correct Rejs:	574	574

4.3 Scenario 3: Two Vehicles with Full Lawnmower Patterns

In this mission the two vehicles do not do any adaptation to the search grid. Each vehicle runs a lawnmower pattern, one with paths at 90 degrees, one with paths at 0 degrees. This allows us to hold time constant to see how the clearance level differs from adaptive missions.

The results are summarized in Table 4.5.

Table 4.5: Simulation and experimental results for mission 4.3

	Simulation	Experiment
Score:	-2060	3980
Time:	17:11	24:00
Ave. Clearance:	70.7%	84.6%
Actual Clearance:	66.7%	86.6%
Hits:	20	26
Misses:	8	2
False Alarms:	2	3
Correct Rejs:	576	575

Figure 4-3 is a plot of the actual vehicle paths overlaid on the lawnmower patterns that the vehicles tried to run. During the experiment there was a moderate wind blowing from the east to the west. This is most noticible on HUNTER1's north-south trackline, where the vehicle is blown to the west. HUNTER2's attempts to travel east into the wind show some buffeting but its downwind runs are stable. This unexpected result shows the importance of accurate navigation data during search tasks and the importance of being able to adapt to changing environmental conditions. That is, after determining that the wind (or current) was coming from a particular direction, the vehicle could change its behavior to minimize negative consequences.

We can compare the results for mission 4.2 and 4.3 in Tables 4.4 and 4.5 to see what gain there is in using an adaptive behavior over just regular lawnmower. In simulation, the adaptive mission produced a clearance of 76.7% in approximately

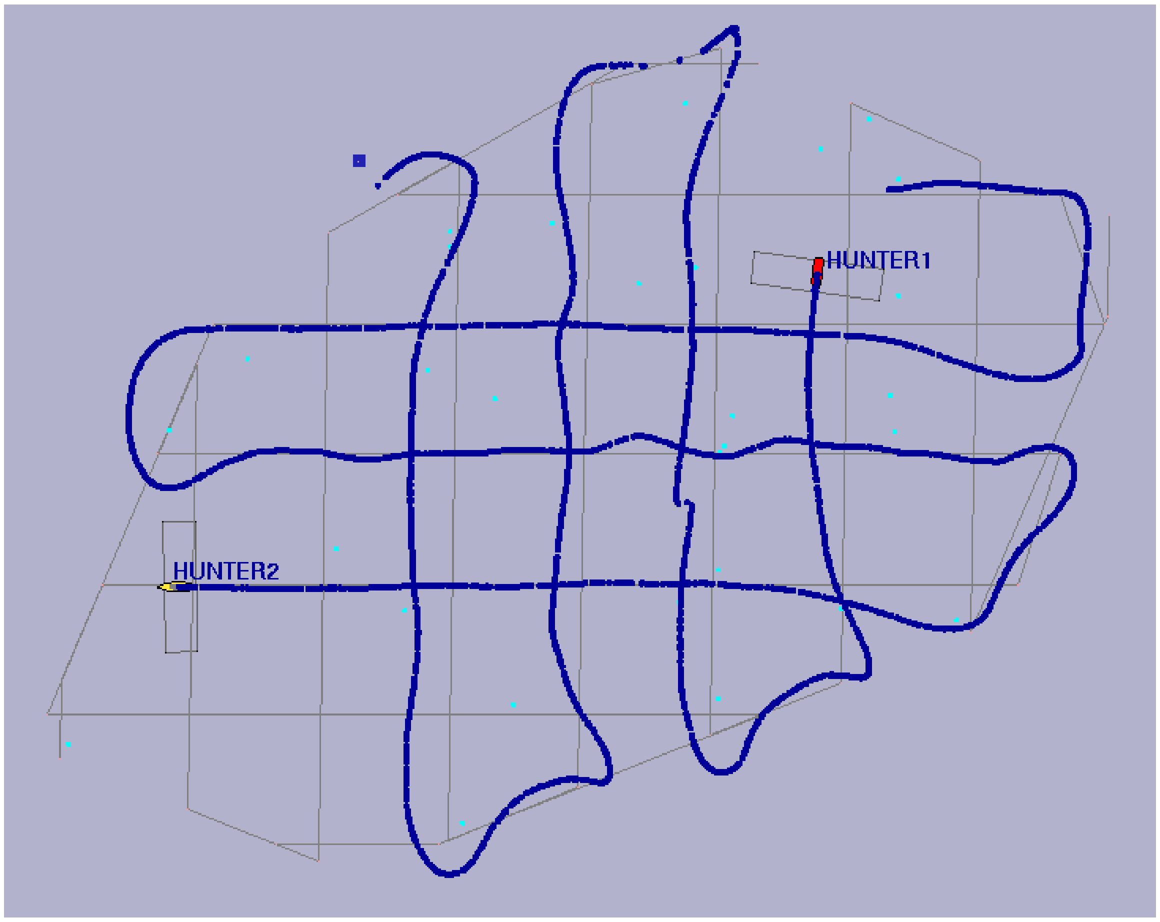


Figure 4-3: This plot shows the actual vehicle paths for mission 4.3 overlaid on the lawnmower patterns that the vehicles tried to run. During the experiment there was a moderate wind blowing from the east to the west. This is most noticible on HUNTER1's north-south trackline, where the vehicle is blown to the west. HUNTER2's eastern tracklines show that the wind did affect it head-on, but its western tracklines are fairly straight.

16:11. The pure lawnmower mission produced a clearance of 70.7% in approximately 17:11. Similarly, on the water the adaptive behavior cleared 87.1% in approximately 22:00 while the pure lawnmower cleared 84.6% in approximately 24:00. In both cases we experience an incremental gain in both time and the average clearance.

4.4 Scenario 4: Two Vehicles Partitioning the Search Area

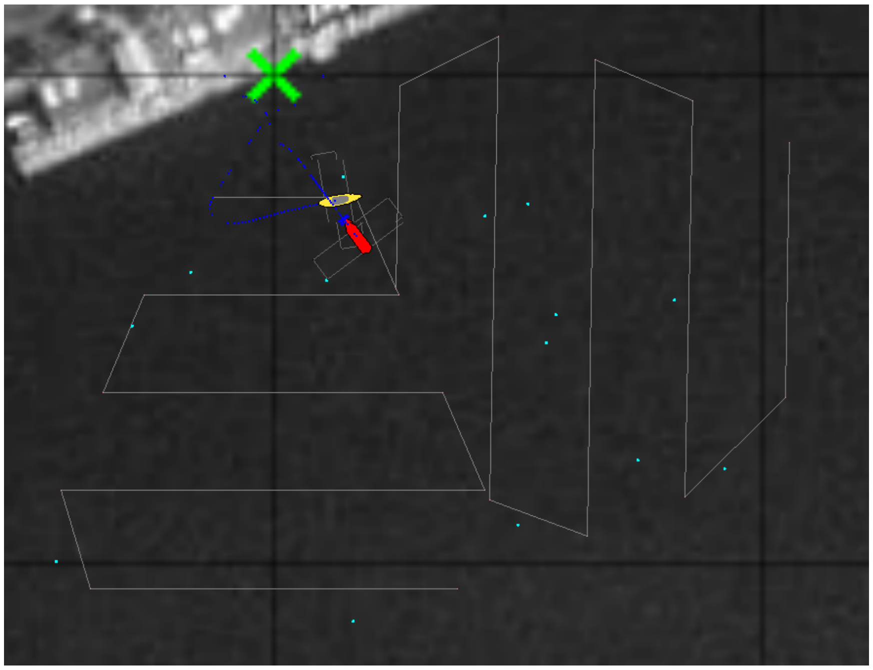
In the previous mission we held time constant by having both vehicles cover the entire search grid. In this mission we reduced the time of search by splitting the search grid in approximately half and having one vehicle lawnmower over each portion. When each vehicle finished its half, it returned to the dock. Each vehicle had a 20-m wide sensor with .5 B, dropping to .25 for 10 seconds every 30. Collision avoidance was also turned on.

The results are summarized in Table 4.6. Ideally, each cell in the grid would be covered once by a pass of a sensor. This would give us an expected clearance of $.5 * 20/30 + .25 * 10/30 = .41667$. The 50% coverage in simulation is probably caused by overlap from the two vehicles near the center of the search grid and from when the vehicles turn at the end points. The even higher coverage in the experiments is caused by the noisy heading data over-scanning cells near the periphery of the sensor.

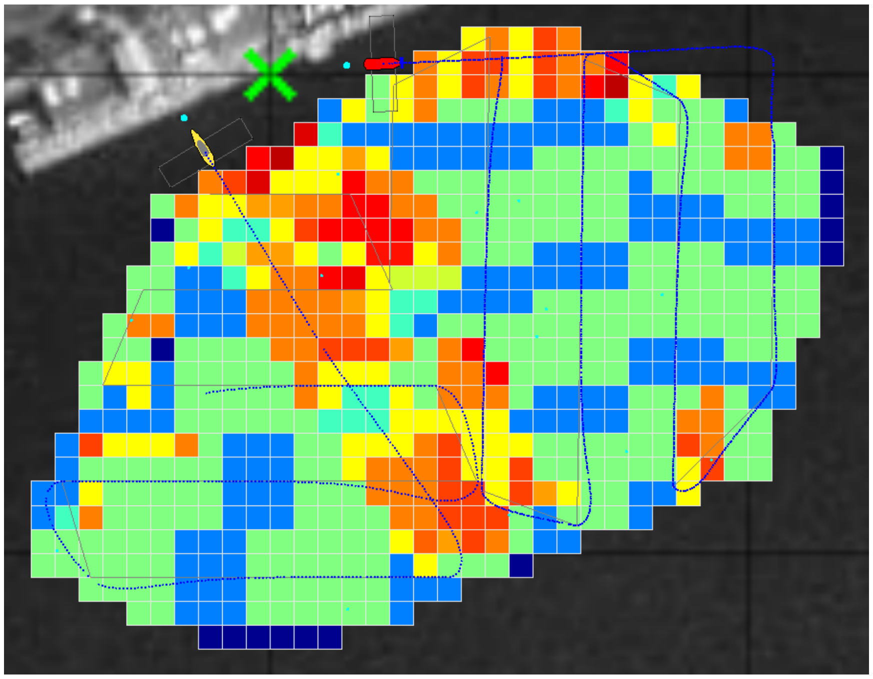
Table 4.6: Simulation and experimental results for mission 4.4. Experiment 1 and 2 are two different runs of the exact same mission profile. The cause of the much lower score for run 2 is unknown. The average clearance is reasonably close to that of run 1, but the actual clearance is much lower. Because of the small target set (30 mines) and the heavy penalty for missing targets (-1000 points), small deviations from the mean are heavily represented in the score. These two runs could represent relative extremes from a mean.

	Simulation	Experiment 1	Experiment 2
Score:	-2100	-2100	-6120
Time:	11:00	14:00	12:00
Ave. Clearance:	50.0%	61.5%	57.6%
Actual Clearance:	66.7%	66.7%	53.3%
Hits:	20	20	16
Misses:	8	8	12
False Alarms:	4	4	3
Correct Rejs:	574	574	575

Figure 4-4 shows the subpatterns that were run by the two vehicles and the completed grid in simulation.



(a)



(b)

Figure 4-4: One common approach to using multiple vehicles in search applications is to divide the search area into nearly equal sized partitions and to assign one vehicle to each partition. To cover the search area, the two vehicles will run lawnmower patterns as in (a). The simulation results of the search effort are shown in (b). As described in Figure 3-4, dark blue represents 0 clearance, light blue .25, teal .5, and dark red 1.00.

4.5 Scenario 5: Simulating Different Sensors

Part of the adaptability aspect of this thesis is to provide for cooperation when the vehicles have different sensors. We now give one of the vehicles a 40-m wide, .5 B sensor (again with 10/30 second drops to .25), but we give the second vehicle a narrower, more accurate sensor–20-m wide, .8 B with 10/30 seconds drops to .6.

In the first trials, we give the wide sensor to the slower AUV and have it perform the lawnmower pattern. In the second example we give the faster kayak the wide sensor and have it perform the lawnmower patterns.

4.5.1 The AUV Leading with a Wide Sensor

In this mission the AUV has a 40-m wide, .5 B sensor (10/30 second drops to .25 B). It performs a lawnmower pattern with radius 20-m (half the sensor width, so single coverage on the search area) over the search area at 1.0 m/s. The kayak performs the cut range and artifact search behaviors with a 20-m wide, .8 B sensor (10/30 drops to .6 B).

The results are summarized in Table 4.7. We first notice that the difference between experiment 1 and experiment 2 are minor for the average clearance and time (the slight difference in actual clearance is probably a statistical anomaly). This means that adding collision avoidance to the vehicles did not impact mission performance but in real applications could prevent the loss of vehicles from collisions.

Figure 4-5 is the clearance map output for the simulation run.

Table 4.7: Simulation and experimental results for mission 4.5.1. Experiment 1 was conducted with collision avoidance on for both vehicles. Experiment 2 turned off the lead vehicle's collision avoidance so that it would not be disturbed by the chase vehicle.

	Simulation	Experiment 1	Experiment 2
Score:	1980	3010	930
Time:	10:48	12:52	12:45
Ave. Clearance:	81.0%	82.7%	85.8%
Actual Clearance:	80.0%	83.3%	76.7%
Hits:	24	25	23
Misses:	4	3	5
False Alarms:	2	1	4
Correct Rejs:	576	577	574

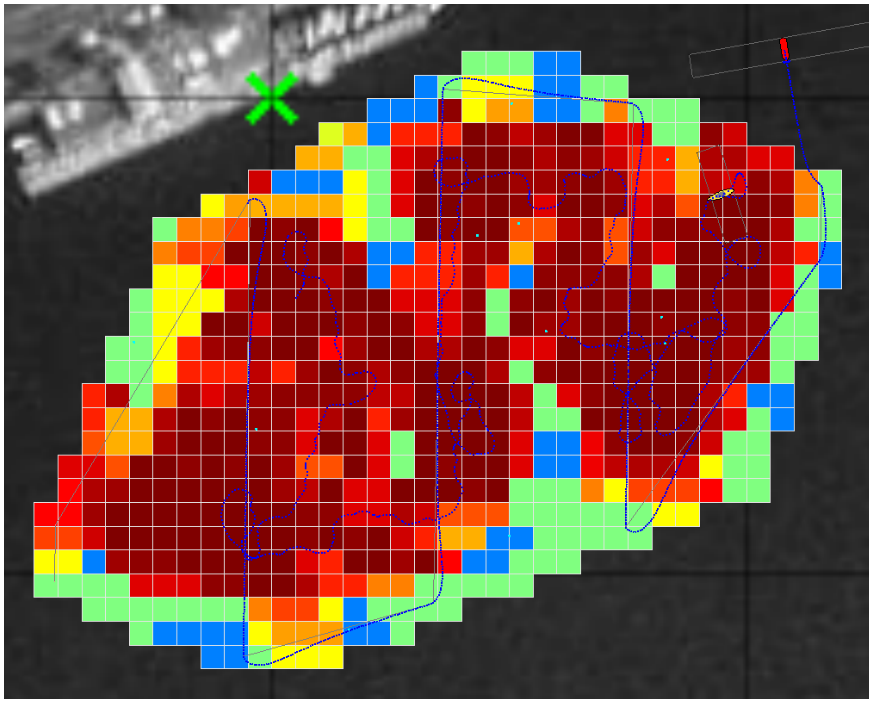


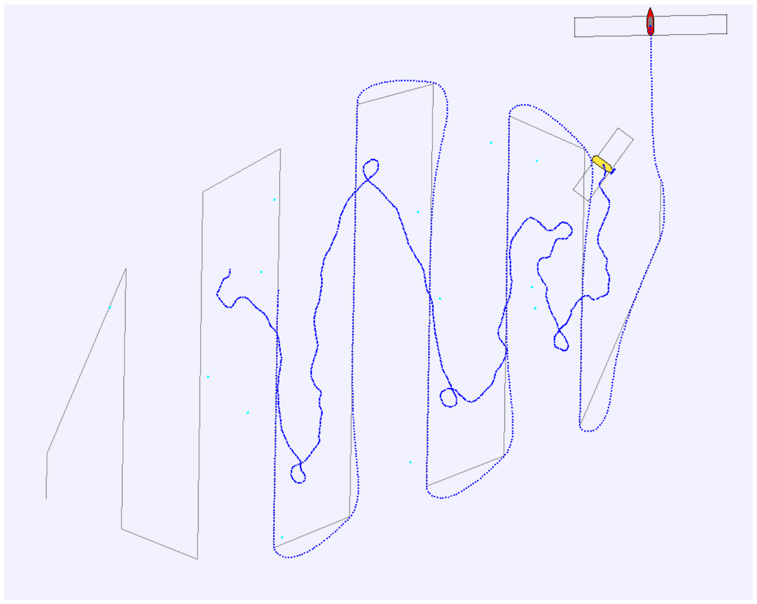
Figure 4-5: The clearance map produced during the simulation run of mission 4.5.1. Dark blue represents 0 clearance, light blue .25 clearance, teal .5 clearance, and dark red nearly 1.0 clearance. The satellite underlay is of the MIT Sailing Pavilion (white buildings at upper left). The tiny blue dots represents targets in the threat laydown. The two vehicles are at the completion of their run in the upper right corner. The dark blue path behind each vehicle is that vehicle's path. If a mission commander was given this clearance map, he or she would know which areas have been heavily cleared (dark red) and which have not (dark blue). The commander would also have a map of the detected targets to avoid or identify and neutralize.

4.5.2 The Kayak Leading with a Wide Sensor

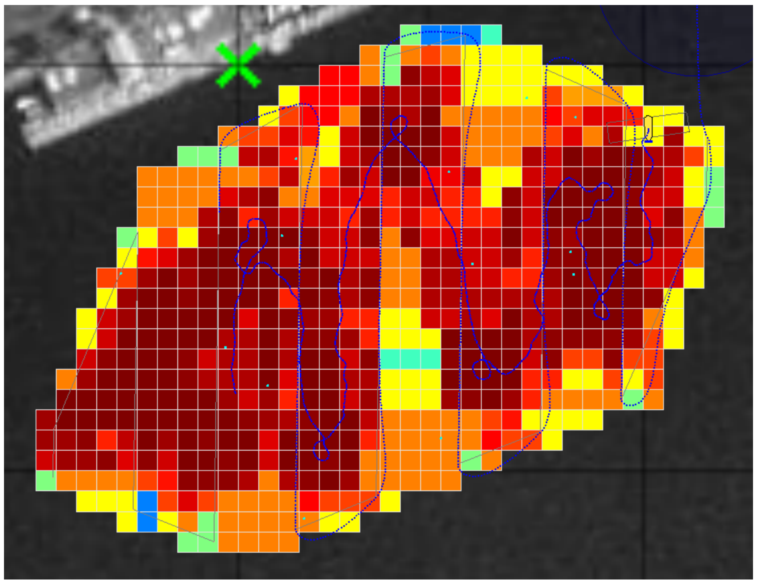
In this mission we give the kayak the wide sensor and make it the lead vehicle that performs a lawnmower pattern. The AUV has a narrower, more accurate sensor. This conceivable configuration could result from a single mine hunting surface craft that might be responsible for one or several AUVs. The AUVs would have to both stay near the surface ship for tending while simultaneously exploring interesting/less-covered areas.

The results for this mission are in Table 4.8. In Experiment 1, we used a lawnmower radius of only 10-m, for a path-to-path distance of 20-m, half of the lead sensor width. This means that the lead sensor covered every part of the search area twice, which is why the clearance level is so high and the time is 70% longer than Experiment 2. Experiment 2 used a radius of 20-m as was used in mission 4.5.1. Comparing the two missions, we see that with the faster lead vehicle we get the anticipated decrease in search time but we also see a decrease in coverage. In simulation we increase the coverage by 11% but increase the time by 41%. Experimentally we see a 4% increase in coverage (compared to the previous experiment 1) for a 27% increase in time.

Figure 4-6 shows that the AUV spent much of its time trying to catch up to the lead vehicle. That is, the cut range behavior had significant influence on the vehicle. Further work needs to be done to see how to balance the cut range behavior and bhv_SearchArtifact in this scenario.



(a)



(b)

Figure 4-6: When the lead vehicle can travel faster than the following vehicle (mission 4.5.2), the follower spends much of its time playing catch-up, (a). In this image the actual vehicle paths are in dark blue. The waypoints for the lead vehicle are represented by the grey line. The final clearance map for simulation 1 is pictured in (b). Notice that the gap of least coverage in the bottom-center of the image occurred where the chasing vehicle was trying to catch up to the lawnmowing vehicle. In the clearance map, dark red is heavily cleared areas while dark blue, light blue, teal, yellow, and orange represent increasing levels of clearance.

Table 4.8: Simulation and experimental results for mission 4.5.2. In this mission the faster kayak is leading with the wide sensor. Experiment 1 used a 10-m swath radius (one-quarter of the sensor width, double coverage). Experiment 2 used a 20-m swath radius for single coverage.

	Simulation 1	Experiment 1	Simulation 2	Experiment 2
Score:	970	5980	-2100	-40
Time:	14:09	16:52	7:37	10:07
Ave. Clearance:	85.9%	95.8%	69.3%	78.6%
Actual Clearance:	76.7%	100.0%	66.7%	76.7%
Hits:	23	28	20	22
Misses:	5	0	8	6
False Alarms:	2	4	4	2
Correct Rejs:	576	574	574	576

Chapter 5

Conclusions and Recommendations for Further Work

5.1 Conclusions

This project set out to investigate the feasibility of using autonomous marine vehicles to cooperatively search a given area. To that effect, it has been shown that cooperative search is possible, even when it can be difficult to evaluate how effective the cooperation has been. Evaluation of cooperative search algorithms has always been difficult because of the number of objectives users might wish to optimize. Time of search, energy used, number of vehicles, number of targets detected, lane-finding, and others are all reasonable ways of scoring a search algorithm.

The simple greedy search behavior described here seems to function well when it cooperates with some other behavior that guides it towards exploring the whole search area, such as following a lawnmower-based lead vehicle. However, when a vehicle is in the middle of an unexplored grid and attempts to just run the search behavior, the vehicle tends to just drive around in circles because every direction seems equally good. This algorithm assumes that the targets have some kind of uniform distribution and that the utility in clearing any one cell is the same as any other cell. This means that the average clearance (based on the cells) is usually close to the actual clearance (based on the total targets found). This assumption is often used in MCM planning because it simplifies some calculations used for analytic solutions. A more intelligent behavior could be written that attempts to find the patterns that naturally exist in deployed mine fields, such as lines or clusters of mines. Being able to exploit these features should vastly decrease search time, allowing the vehicle to finish early or spend more time scanning difficult areas.

Our sensor model also assumes that the B value of the sensor is uniform along its fixed width, A. We assumed constant B along the sensor width because we have no model that suggests another distribution. We believe that by modelling random sensor dropouts we can begin to get an understanding of how a non-uniform B might affect the search, but more research into sensors is needed.

Our experimental research was quite hampered by the noisy heading data that forced us to turn off the slew-rate limiting in pSensorSim and pArtifactMapper. This feature was designed to penalize rapid turns and by shutting it off our simulations and experiments overstate the clearance level at the end of the search. A new differential-GPS unit installed on one of the kayaks should return more accurate heading data and perhaps allow us to turn the slew-rate limiter back on.

To make the problem more realistic we would like to have data about the performance of mine sensors under various conditions of vehicle/sea-state, sea-bottom state, and mine type. Unsurprisingly, unclassified research data of this type is difficult to come by. With this data we could modify pSensorSim to look at the vehicle state (rate of roll, rate of pitch, etc.), the sea-bottom state (sandy, rocky, extremely rocky, etc.) and the mine type (moored, surface, buried, etc.) to determine a more accurate probability of detection.

Another major issue with CAD/CAC is the prevalence of false positives. Our sensor does not emit false positives but it could be set to do so by adding fake targets to the threat laydown. We did not evaluate the effect of false positives on our performance.

Our experimental operations were limited to ranges of about 200-m at the MIT Sailing Pavilion because of Wi-Fi connectivity. Actual search areas can be dozens to hundreds of kilometers on a side. The performance of the greedy behavior probably becomes more inefficient as the search area grows without some structure imposed on the searcher. Further research is needed to test this.

The next experiment in this line of research should focus on testing how different parameters affect the search performance. Some suggested parameters include the relative weights of bhv_SearchArtifact and bhv_CutRange, the distances at which the cut range behavior becomes active, and the sizes and shapes of the search areas. Thinking beyond this experiment, getting access to a vehicle with an actual sensor to test on would be a valuable experience to determine if this framework and algorithm function well on actual hardware.

In conclusion, we hope that this research has somehow pushed the boundary of what is currently being done with autonomous MCM. We came into this field with little experience in mine countermeasures, both a handicap because our knowledge of current practices was weak and a benefit because we have no entrenched dogmas limiting our creativity. There remains much to be done in the field and we look forward to seeing how our colleagues assimilate our research into their own.

5.2 Recommendations for Further Work

Our work has only skimmed the surface of what is possible with autonomous, cooperative, adaptive search. Here are a few ideas for where future work can proceed.

1. **Software:**
   (a) Implement Artifact and ArtVector classes for better processing abilities.
   (b) Rethink polygon strings to be more consistent with one another.
   (c) Create utilities to take a string into a map (or multi-map) and vice-versa.
   (d) Create a behavior file/mission file creation tool, especially for multiple vehicles.
   (e) Modify splug to be recursive, allow #includes with $(data).
   (f) Let the helm dynamically decide how many pieces to search, perhaps using some form of iterative deepening.
   (g) Design a more compact, efficient communications packet between searchers.

2. **Algorithms:**
    (a) Add slew-rate control to bhv_SearchArtifact so that it does not drop out.
    (b) Add steady-heading behavior to the mix to get the vehicle to drive straight.
    (c) Decrease the ΔUtil of cells that are farther away and in directions not near the current heading.
    (d) Experiment with different threat laydown options instead of uniform random.
    (e) Create a search algorithm to find a path between two sides of the search area (lane finding).
3. **Variables:**
    (a) More simulation of behavior performance with different shaped/sized search areas.
    (b) Use larger/smaller cell sizes, and/or try to use hexagon cells.
    (c) Change the distance that the follow vehicle tries to stay within.
    (d) Change the look-ahead time the search behavior uses.
4. **Research:**
    (a) There is always room for improvement in D&C algorithms, especially with regards to eliminating false positives.
    (b) More work needs to be done to determine how to accurately model a sensor's A and B values given the current state of the vehicle and the state of the environment.

Appendix A

Configuration Blocks

The configuration blocks in this section are designed to assist other users of the MOOS-IvP suite in configuring the software modules. Each one of the blocks is suitable for pasting into a .moos file used for running MOOS programs.

A.1 pSensorSim

```
//------------------------------------------
// pSensorSim config block
ProcessConfig = pSensorSim
{
   AppTick   = 4
   CommsTick = 4

   // Add single artifacts
   // Artifact = X=10,Y=5

   // Read in an artifact file (looks like above lines)
   ArtifactFile = mines.art

   // Configure the basic sensor parameters
   Sensor_A      = 20 // meters
   Sensor_B      = .5 // between [0, 1]

   // Configure the optional sensor parameters
   // All must be set to enable dropout simulation
   Drop_Period   = 30 // How often should it begin
```

```
                      //     drop periods? seconds
   Drop_Duration = 10 // How long should the dropout
                      //     last? seconds
   Drop_B        = .25 // B value during dropouts

   // Optional, beyond this slew rate (degrees/sec),
   // the sensor does not detect anything
   Max_Slew      = 10
}

//Messages posted:
//SENSOR_A, Sensor A value
//SENSOR_B, Sensor B value
//DETECTED_ARTIFACT_LOCAL, detected artifact string
//VIEW_POLYGON, displays the sensor
//VIEW_POINT, simulated target points at load
```

A.2 pArtifactMapper

```
//-------------------------------------------
// pArtifactMapper config block
ProcessConfig = pArtifactMapper
{
   AppTick   = 4
   CommsTick = 4

   // The polygon to look at
   GridPoly =  label,ArtField1:-20,-30:50,10:120,
                      -20:100,-80:0,-120:-50,-100

   // The size of each cell, in meters
   GridSize = 5.0

   // Optional, beyond this slew rate (degrees/sec),
   // the mapper does not update cells
   Max_Slew      = 10
}

//Messages posted:
//ARTIFACTMAP_REFRESH, t/f posts the entire grid to the MOOSDB
//ARTIFACTMAP_EXPORT, t/f saves the current data to a
                      file as vehicle_currenttime.map
//GRID_CONFIG, posted for pMV
//GRID_AVG_CLEARANCE, average clearance of the grid
```

```
//ARTIFACTGRID_DELTA_LOCAL, delta to pass to other processes
//GRID_DELTA_LOCAL, delta to pass to pMV
```

A.3 bhv_SearchArtifact

```
initialize   DEPLOY = false
initialize   RETURN = false

//------------------------------------------------
Behavior = BHV_SearchArtifact
{
  name      = bhv_SearchArtifact
  pwt       = 100
  condition = DEPLOY = true
  condition = RETURN = false

  // Polygon search grid
  // USE THE SAME AS pARTIFACTMAPPER OR IT WILL HAVE ODD BEHAVIOR
  searchgrid = label,ArtField1:-20,-30:50,10:120,
                     -20:100,-80:0,-120:-50,-100@5.0,5.0

  // Time to look ahead, in seconds
  time_horizon = 20
}

//Messages posted:
//TIME_TO_EXIT_GRID_+grid_label, time to leave the grid
//osSPD, ownship speed
//dtg, distance to edge of grid
```

A.4 bhv_Lawnmower

```
initialize   DEPLOY = false
initialize   RETURN = false

//------------------------------------------------
Behavior = BHV_Lawnmower
{
  name      = bhv_lawnmower
  pwt       = 100
  condition = DEPLOY = true
  condition = RETURN = false
  // Polygon to patrol, required (can also be called poly)
  polygon =     -20,-30:50,10:120,-20:100,-80:0,-120:-50,-100

  // Angle for pattern, 0 = N, 90 = E
  ang     =   1  // optional, default = 0

  // Radius, half the distance between lines
  radius  =   10  // required, in meters

  // By default, pLawnmower will make a full pattern
  // You can also specify a start point to make
  // a partial pattern (useful for mid-mission changes
  full    =  true   // optional, default = true
  //x       =   10  // required for full, in meters
  //y       =   -71  // required for full, in meters
  //clockwise =   // optional, default = true,

  // Value to snap segment points to
  snap    =  .25   // optional, in meters

  // Options for the waypoint navigating
  lead      = 15.0 // optional, trackline lead distance, in meters
  speed     = 3.0  // speed to waypoints, in m/s

  waypoint_radius = 2.0  // size of point in meters
  nm_radius       = 10.0 // radius for capture in meters

  updates   = LAWNMOWER_UPDATES // update MOOS variable, case sens.
}

//Messages posted:
//VIEW_SEGLIST, seglist that is being traversed
```

A.5 pLawnmower

```
//------------------------------------------
// pLawnmower config block
ProcessConfig = pLawnmower
{
   AppTick   = 4
   CommsTick = 4

   // No configuration options for pLawnmower
}
```

```
//Messages posted:
//VIEW_POLYGON, polygon asked about
//VIEW_SEGLIST, seglist that corresponds for pMV
//LAWNMOWER_SEGLIST, seglist that for consuming processes
```

Appendix B

Utility Descriptions

B.1 pLawnmower

pLawnmower is a helper utility that can generate lawnmower patterns for other MOOS processes that would like to use them. This algorithm is the same one used in bhv_Lawnmower.

Given the convex polygon, an initial point, and an initial heading, generate a segment from the start point to the boundary of the polygon.

Push this segment onto the lawnmower pattern.

Copy the previous segment and shift it by distance $2*radius$. If a clockwise initial turn is specified, shift it at angle initial heading + 90, otherwise, shift it at initial heading - 90.

Because of the polygon convexity this new segment must intersect the polygon (that is, at least part of the segment must be contained in the polygon), if it doesn't, we are done and can return the current lawnmower pattern.

There are four possible cases with these two possible endpoints:

1) p0, p1 inside polygon

2) p0 inside, p1 outside polygon

3) p0 outside, p1 inside polygon

4) p0, p1 outside polygon

For each of these four cases we can determine which direction we need to project the point (p0, p1) to move it so it itersects with the boundary.

Once these two points are generated, the one closest to the last endpoint on the lawnmower pattern is pushed on first, followed by the second point.

To generate a FULL lawnmower pattern, the center of the polygon can be chosen to be the initial point and a partial pattern can be generated for each direction (initial heading and initial heading + 180). By reversing one of these patterns and removing the center point from both, they can be stitched together to get a full lawnmower pattern.

B.2 artfieldgenerator

The artifact field generator takes a polygon and generates a uniform random distribution of targets within the polygon.

Given a convex polygon, generate the smallest rectangle that holds the whole polygon. Choose a random x and random y uniform over the bounds of the rectangle. Check to see if the chosen point is contained within the polygon. If it is not, choose x and y again. If it does, check to see if this target has been picked before. The same target can be created multiple times because the targets are snapped to a discrete grid.

B.3 scoreartfield

As noted in Section 2.3, our utility uses classic signal detection theory when scoring the output of pArtifactMapper.

Assume that with small enough cells, if there are m targets and n declares, that there are $\min(m, n)$ hits.

If $m > n$ (more targets than declarations), there are $m - n$ misses. If $n > m$ (more declarations than targets), there are $n - m$ false alarms.

If there are no targets and no declares in the cell, that is a correct rejection.

Each result (hit, miss, false alarm, correct rejection) is multiplied by its scoring value as specified by the user.

Appendix C

Photographs

The following photos were taken by Mike Benjamin at the MIT Sailing Pavilion on the Charles River in Cambridge, MA, during water operations May 14, 2008.

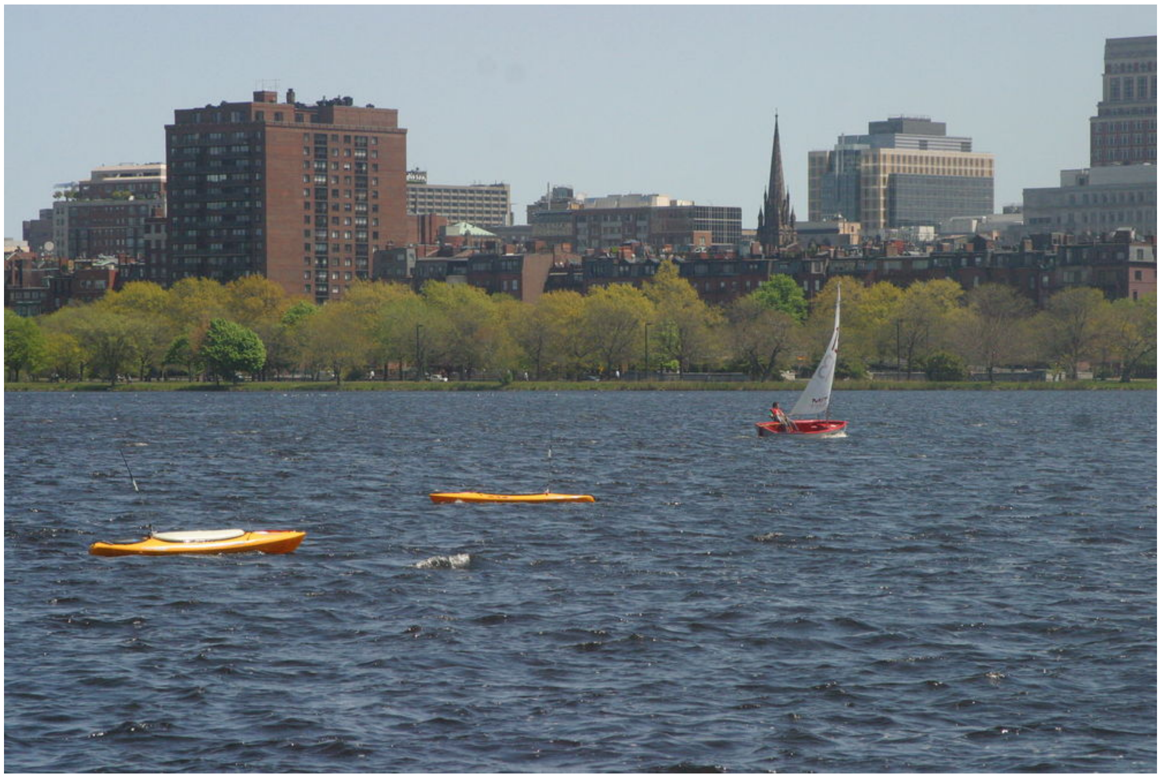


Figure C-1: Two kayaks operating on the river. One of the kayaks is acting like an AUV while the other is simulating a surface craft.

Figure C-2: The author checks to see if the kayaks are disturbing any other vessels on the water.

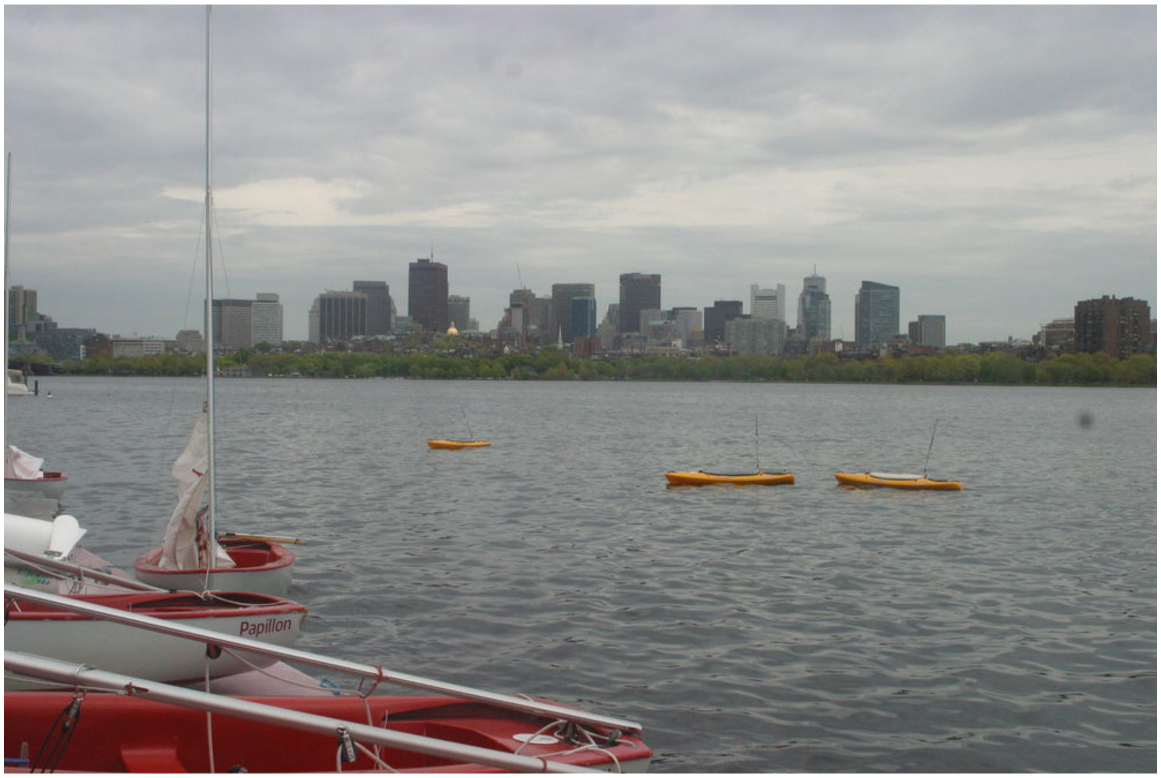


Figure C-3: Three SCOUT kayaks waiting for deployment commands from mission control.

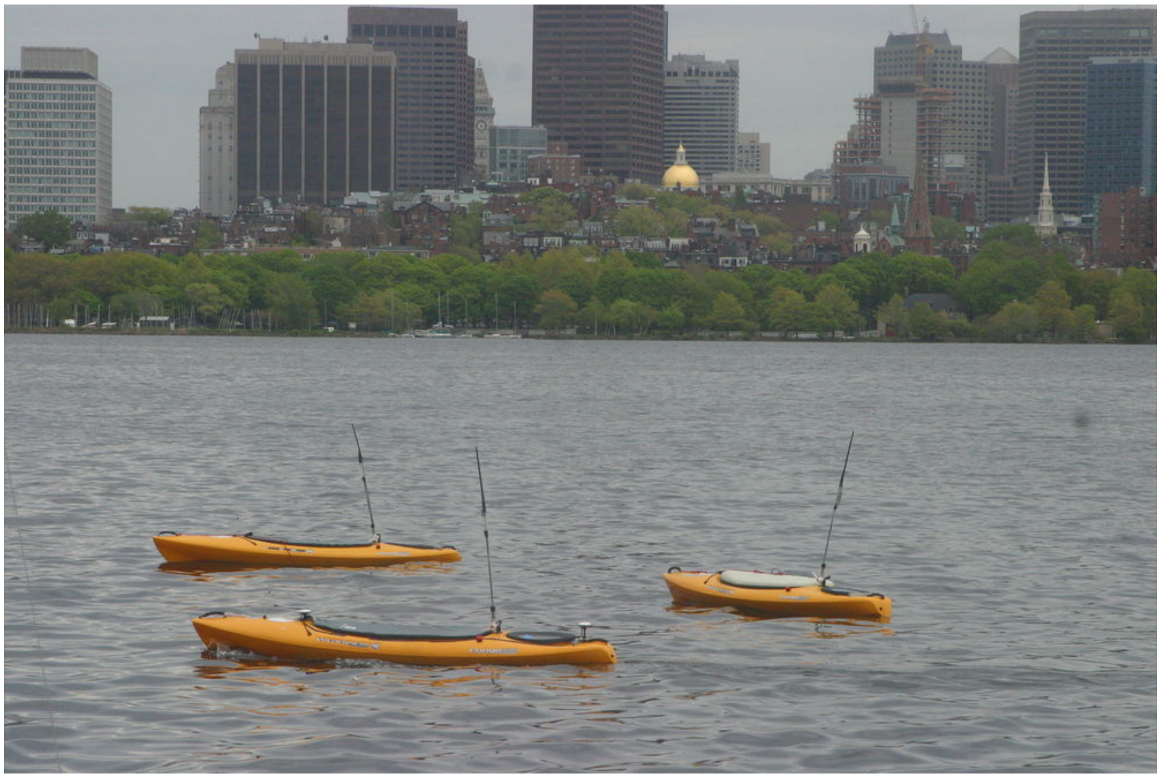


Figure C-4: Three SCOUT kayaks operating on the Charles River.

Bibliography

[1] T. Abdelzaher, B. Blum, Q. Cao, D. Evans, J. George, S. George, T. He, L. Luo, SH Son, R. Stoleru, et al. Envirotrack: An environmental programming model for tracking applications in distributed sensor networks. *ICDCS 04*, 2004.

[2] E.U. Acar and H. Choset. Sensor-based coverage of unknown environments: Incremental construction of morse decompositions. *The International Journal of Robotics Research*, 21(4):345, 2002.

[3] E.U. Acar, H. Choset, A.A. Rizzi, P.N. Atkar, and D. Hull. Morse decompositions for coverage tasks. *The International Journal of Robotics Research*, 21(4):331, 2002.

[4] E.U. Acar, H. Choset, Y. Zhang, and M. Schervish. Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods. *The International Journal of Robotics Research*, 22(7-8):441–466, 2003.

[5] P.N. Atkar, A. Greenfield, D.C. Conner, H. Choset, and A.A. Rizzi. Uniform coverage of automotive surface patches. *The International Journal of Robotics Research*, 24(11):883, 2005.

[6] T. Balch and R.C. Arkin. Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1(1):27–52, 1994.

[7] M.R. Benjamin. *Interval Programming: A Multi-Objective Optimization Model for Autonmous Vehicle Control*. PhD thesis, Brown University, Providence, RI, 2002.

[8] M.R. Benjamin. The interval programming model for multi-objective decision making. Tech. Rep. AIM-2004-021, Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA, Sep 2004.

[9] M.R. Benjamin. A guide to the IvP helm for autonomous marine vehicle control. Technical report, Massachusetts Institute of Technology, 2006.

[10] M.R. Benjamin, J. Curcio, J. Leonard, and P. Newman. Navigation of unmanned marine vehicles in accordance with the rules of the road. *International Conference on Robotics and Automation (ICRA)*, 2006.

[11] M.R. Benjamin, M. Grund, and P. Newman. Multi-objective optimization of sensor quality with efficient marine vehicle task execution. *International Conference on Robotics and Automation (ICRA)*, May 2006.

[12] M.R. Benjamin, J.J. Leonard, J.A. Curcio, and P.M. Newman. A method for protocol-based collision avoidance between autonomous marine surface craft. *Journal of Field Robotics*, 23(5):333–346, 2006.

[13] R.A Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, April 1986.

[14] H. Choset. Coverage of Known Spaces: The Boustrophedon Cellular Decomposition. *Autonomous Robots*, 9(3):247–253, 2000.

[15] H. Choset. Coverage for robtics–a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31(1):113–126, 2001.

[16] TR Clem. Sensor technologies for hunting buried sea mines. In *Oceans 2002*, volume 1, pages 450–462. MTS/IEEE, 2002.

[17] DC Conner, A. Greenfield, PN Atkar, AA Rizzi, and H. Choset. Paint deposition modeling for trajectory planning on automotive surfaces. *Automation Science and Engineering, IEEE Transactions on [see also Robotics and Automation, IEEE Transactions on]*, 2(4):381–392, 2005.

[18] J. Curcio, J. Leonard, and A. Patrikalakis. SCOUT–a low cost autonomous surface craft for research in cooperative autonomy. *IEEE Oceans*, 2005.

[19] L. Freitag, M. Grund, S. Singh, J. Partan, P. Koski, and K. Ball. The WHOI micro-modem: An acoustic communications and navigation system for multiple platforms. *IEEE Oceans Conference*, 2005.

[20] D.W. Gage. Sensor abstractions to support many-robot systems. *Proceedings of SPIE Mobile Robots VII*, pages 18–20, 1992.

[21] D.W. Gage. Randomized search strategies with imperfect sensors. *Proceedings of SPIE Mobile Robots VIII*, 2058:270–279, 1993.

[22] P. Gao and LM Collins. A theoretical performance analysis and simulation of time-domain EMI sensor data for land mine detection. *Geoscience and Remote Sensing, IEEE Transactions on*, 38(4):2042–2055, 2000.

[23] E. Gat and G. Dorais. Robot navigation by conditional sequencing. *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 1293–1299, 1994.

[24] M. Grund, L. Freitag, J. Preisig, and K. Ball. The PLUSNet underwater communications system: Acoustic telemetry for undersea surveillance. *OCEANS 2006*, pages 1–5, 2006.

[25] G.K. Hartmann et al. *Weapons that Wait: Mine Warfare in the US Navy.* Naval Institute Press, 1979.

[26] S. Hert, S. Tiwari, and V. Lumelsky. A terrain-covering algorithm for an AUV. *Autonomous Robots*, 3(2):91–119, 1996.

[27] L.R. Howell, W.C. Littlejohn, C. Guastella, and N. Rodriguez-Casanova. Defining surf zone crawler search strategies for minefield reconnaissance. *OCEANS, 2001. MTS/IEEE Conference and Exhibition*, 1, 2001.

[28] J. Leonard, A. Bennett, C. Smith, and H. Feder. Autonomous underwater vehicle navigation. Technical memorandum, Massachusetts Institute of Technology, Marine Robotics Laboratory, 1998.

[29] VJ Lumelsky, S. Mukhopadhyay, and K. Sun. Dynamic path planning in sensor-based terrain acquisition. *Robotics and Automation, IEEE Transactions on*, 6(4):462–472, 1990.

[30] D. MacKenzie and T. Balch. Making a clean sweep: Behavior based vacuuming. *AAAI Fall Symposium, Instationating Real-World Agents*, 1996.

[31] P.M. Newman. MOOS - a mission oriented operating suite. Technical Report OE2003-07, Massachusetts Institute of Technology, Department of Ocean Engineering, 2003.

[32] Office of Naval Research. BAA-07-028: Undersea cooperative cueing and intervention. Broad Agency Announcement, April 2007.

[33] S.A. Redfield. Lane finding using homogeneous groups of cooperating autonomous vehicles. *Autonomous Underwater Vehicles, 2004 IEEE/OES*, pages 26–31, 2004.

[34] J.A. Rice. Undersea networked acoustic communication and navigation for autonomous mine-countermeasure systems. *Proceedings of the 5th International Symposium on Technology and the Mine Problem*, 2002.

[35] M. Richardson, P. Valent, K. Briggs, J. Bradley, and S. Griffin. NRL mine burial experiments. *Proceedings of the Second Australian-American Joint Conference on Technologies of Mine Countermeasures*, pages 27–29, 2001.

[36] R.R. Rodriguez. Unmanned systems autonomy and mcm performance. Technical Talk, March 2008.

[37] W.R. Scott Jr and J.S. Martin. Experimental investigation of the acousto-electromagnetic sensor for locating land mines. *Proc. SPIE*, 3710:209–214, 1999.

[38] W.R. Scott Jr, C. Schroeder, and J.S. Martin. An acousto-electromagnetic sensor for locating land mines. *SPIE, AeroSense, Detection and Remediation Technologies for Mines and Minelike Targets III, Orlando, FL*, pages 176–186, 1998.

[39] J.R. Stack and C.M. Smith. Combining random and data-driven coverage planning for underwater mine detection. *OCEANS 2003. Proceedings*, 5, 2003.

[40] M.S. Stewart and J. Pavlos. A means to networked persistent undersea surveillance. *Technology Symposium*, 2006.

[41] R. Stokey, T. Austin, B. Allen, N. Forrester, E. Gifford, R. Goldsborough, G. Packard, M. Purcell, and C. von Alt. Very shallow water mine countermeasures using the REMUS AUV: A practical approach yielding accurate results. *OCEANS, 2001. MTS/IEEE Conference and Exhibition*, 1, 2001.

[42] E. Stoneking and J. Hosler. Path planning algorithms for the adaptive sensor fleet. *Advances in the Astronautical Sciences*, 121:207–217, 2005.

[43] I.A. Wagner, M. Lindenbaum, and A.M. Bruckstein. Smell as a computational resource-a lesson we can learn from the ant. *Proc. ISTCS*, 96:219–230, 1996.

[44] IA Wagner, M. Lindenbaum, and AM Bruckstein. Distributed covering by ant-robots using evaporating traces. *Robotics and Automation, IEEE Transactions on*, 15(5):918–933, 1999.

[45] R. Wiegert. Magnetic anomaly guidance system for mine countermeasures using autonomous underwater vehicles. In *Oceans 2003*, volume 4. MTS/IEEE, 2003.

[46] R. Wiegert, B. Price, and J. Hyder. Magnetic anomaly sensing system for mine countermeasures using high mobility autonomous sensing platforms. In *Oceans 2002*, volume 2, pages 937–944. MTS/IEEE, 2002.

[47] R. Williams. Design and experimental evaluation of an autonomous surface craft to support AUV operations. Master's thesis, MIT, February 2007.

[48] Y. Zhang, M. Schervish, EU Acar, and H. Choset. Probabilistic methods for robotic landmine search. *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, 3, 2001.