A Modular Algorithm for Exhaustive Map Searching Using Occupancy Based Maps

Christopher W. Lum[*], and Juris Vagners[†]

Autonomous Flight Systems Laboratory

University of Washington, Seattle, WA, 98195, USA

Searching for a target in a complex environment is a common problem encountered by many autonomous systems. This work considers the problem of searching for targets using a team of heterogeneous agents. The system maintains a grid-based world model which contains information about the probability that a target is located in any given cell of the map. Agents formulate control decisions for a fixed number of time steps using a modular algorithm that allows for individual capabilities and characteristics of individual agents to be encoded in several parameters. This paper investigates one aspect of the search strategy and presents a solution that guarantees total map coverage. The resulting search patterns executed by agents guarantee an exhaustive search of the map in the sense that all cells will be searched sufficiently to ensure that the probability of a target being located in any given cell is driven to zero.

Nomenclature

B	Set of $\overline{z}$ values defining spatial domain of occupancy based map
B_R	Locations reachable by agent in d steps
$\tilde{B}$	Set of $\overline{z}$ values defining center of occupancy based map cells
$\tilde{B}_R$	Cell centers reachable by agent in d steps
d	Prediction horizon
$f_\chi()$	Reward function for course deviation
$f_d()$	Reward function for distance
$f_h()$	Reward function for high score cell
h	Sensor reliability factor in range $[0,1]$
$J_0()$	Total reward function for $(\wp_2)$
L_x, L_y	Width and height of map cell in x-direction and y-direction, respectively
N_x, N_y	Number of columns and rows, respectively, of occupancy based map
$(\wp_1)$	Subproblem of creating future world state estimates
$(\wp_2)$	Subproblem of finding desirable cells for agent to search
$(\wp_3)$	Subproblem of finding trajectories for agent flight path
$p(A\|B)$	Conditional probability of A given B
P_E, P_N	East and north position of agent, respectively
$q(a,b)$	Calculates absolute angular difference between angles a and b
R_{max}	Maximum distance agent can travel in a d steps
s_k	Score of given occupancy map cell at step k ($p(X_k = x_B)$ at step k)
V_{max}	Maximum velocity of agent
$\overline{w}^\star$	Controls/waypoints which take agent from current location to $\overline{z}^\star$
x_A, x_B	Event of target not in cell and target in cell states, respectively
x_{min}, x_{max}	Minimum and maximum x value of occupancy based map
$x_w(k,\overline{z})$	Actual state of the world at time step k and location $\overline{z}$

[*]Research Assistant, Dept. of Aeronautics and Astronautics, lum@u.washington.edu, AIAA student member
[†]Professor Emeritus, Dept. of Aeronautics and Astronautics, vagners@aa.washington.edu, AIAA member

$\hat{x}_w(k, \overline{z})$	Estimated world state at time step k and location $\overline{z}$
$\overline{x}_{agt}(k)$	Agent state $(P_E\ P_N\ \chi)^T$ at step k
$\overline{x}_{tgt}(k)$	State of the target at time step k
$\hat{x}_{tgt}(k)$	Estimated target state at time step k
y_{min}, y_{max}	Minimum and maximum y value of the occupancy based map
$\overline{z}$	(x, y) coordinate $(P_E\ P_N)^T$
$\overline{z}^{\star}$	Most desirable cell in $(\wp_2)$
$\overline{z}_{agt}$	Agent's current position
Z	Set of all cell centers that the agent can reach in d steps
z_A, z_B	Observation of target not in cell and in cell, resectively
$\overline{z}_H$	Location of cell center with highest score in map and closest to agent
$\overline{z}_h$	Location of cell center in agent's reachable set closest to $\overline{z}_H$
$\overline{z}_s$	Location of cell center of same cell that agent is currently in
α	Scalar tradeoff parameter for $\hat{x}_w(k + d, \overline{z})$
β	Scalar tradeoff parameter for $f_\chi(\overline{z})$
ΔT	Time between steps
η	Maximum score of cell within agent's reachable set
γ	Scalar tradeoff parameter for $f_d(\overline{z})$
$\Gamma(a, b)$	Predicted future state of the world given estimated target state a and world state b

I. Introduction

The overwhelming majority of current missions tasked to autonomous systems revolve around intelligence, reconnaissance, and surveillance (ISR). This work seeks to increase the autonomy and capabilities of a heterogeneous team of agents involved in a search and surveillance type mission. The system maintains a world model which includes an estimate of possible target states and the state of the environment. The issue of compelling agents to converge on possibly moving targets and continuing to search new regions is formulated as a model predictive control problem. The world model is propagated forward in time and strategic decisions are made based on the predicted future state of the world. Agents formulate control decisions for a fixed number of time steps by optimizing an objective function that allows for control and timing constraints.

The overall objective of this work is to develop robust and scalable control laws to apply to a heterogeneous group of agents so that they operate in a cooperative fashion to achieve a common goal. The goal in this application considers the specific mission of coordinating a group of agents to search a two dimensional space for a target based. Although this goal may seem narrow in scope, many of the technologies developed for this application are easily adaptable to more general tasks encountered in autonomous systems. This paper investigates a modular algorithm which can efficiently coordinate a team of possibly heterogeneous agents and ensure that the team performs an exhaustive search of the map.

This type of search mission has been previously studied by several other groups. Classically, it has been addressed as classical exploration missions by Monekosso[1] and Dollarhide[2] et al. It also fits the model of search and rescue applications as shown by Kurabayashi,[3] Kantor,[4] and Jennings[5] et al. A survey of classical searching techniques by Benkoski[6] provides additional background and references.

One modern approach that has become popular is to consider possible target locations as a continuous or discrete probability density function. Groups such as Durrant-Whyte et al.[7,8,9] studied the problem of searching for a target using a Bayesian probabilistic approach and have investigated some of the communication issues involved in such a search. Polycarpou et al.[10,11,12] applied optimization techniques to generate search patterns over a finite amount of steps. Many of these methods are successful and effective but have difficulty providing guarantees on target detection and map coverage. To address this, Erignac[13] developed exhaustive searching strategies that also provide guarantees about map coverage with ideas based on pheromone maps. Coverage of maps and domains have also been studied in the context of minimum service time to spontaneously occurring targets. Many of these techniques use Voronoi partitioning of the domain to maintain agent separation and coverage. This approach has been studied by Du,[14] Cortes[15],[16] and Frazzoli[17,18] . Searching and map building has been studied extensively by the robotics community as well. Groups such as Fox et al.[19] have looked at generating searching algorithms with the ideas of exploration and map building in mind. Others such as Baillieul et al.[20] and Hoffman[21] have posed the searching mission in an information theoretic framework which provides useful metrics for search effectiveness. Previous work at the University of Washington by Rubio et al.[22] investigated searching algorithms in the context of adaptive algorithms. Previous work explored the use of novel optimization techniques to address the search problem.[23]

These methods, while effective for their respective applications, contain shortcomings when addressing the search problem considered in this application. To address the problem at hand, a modular search algorithm is presented in this paper. The algorithm is modular in the sense that it is comprised of three main steps. Each step is somewhat independent of the others and different algorithms can be used to accomplish the same goal within any of the three main steps.

The first step involves propagating the world state (the occupancy map) forward in time. By providing a predictive aspect to the problem, each agent can then make control decisions based on the predicted future state of the world rather than only using the current information. Once the system generates a predicted future world state, each agent determines a desirable coordinate to visit in the future. The definition of "desirable" is formed as a numerical optimization problem. This formulation allows for each agent in the team to have a different set of parameters; therefore, each agent in the team can have its own notion of desirability. Finally, once this desirable coordinate is determined, a path that transitions the agent from its current location to the desirable coordinate can be applied.

The focus of this paper is an algorithm for solving the second step of the process which will guarantee that the team exhaustively searches the map for the target. The first[23] and third[24] steps have already been addressed in previous works.

The search strategy is formulated with a heterogeneous team in mind. Each agent may have different capabilities and bandwidths. For example, fast agents (Figure 1(a)) are able to maneuver more effectively but require faster processing speeds. In contrast, slower agents (Figure 1(b)) are more sluggish but the bandwidth requirements decrease as well. The search strategy must be scalable and able to accommodate these differences.

(a) The GeoRanger autonomous air vehicle

(b) The SeaFox autonomous surface vehicle

Figure 1. Possible agents of a heterogeneous team involved in searching mission.

Section II describes the notion of occupancy based maps and their features. These maps provide the framework for the search strategy which is initially derived as a single agent search strategy and is described in Section III. Section IV then analyzes the proposed search strategy and shows that it yields an exhaustive map search. Simulation results and performance metrics are discussed in Section V. Finally, Section VI presents conclusions and future directions of research.

II. Occupancy Based Maps

In order to effectively search a two dimensional domain for a target, the system must keep track of the state of the world in terms of possible target locations. To do this, an occupancy based map is employed. These constructs were originally developed by Elfs[25,26] but functionality such as Bayesian score updates and time varying models are added to the maps to accommodate the algorithm.

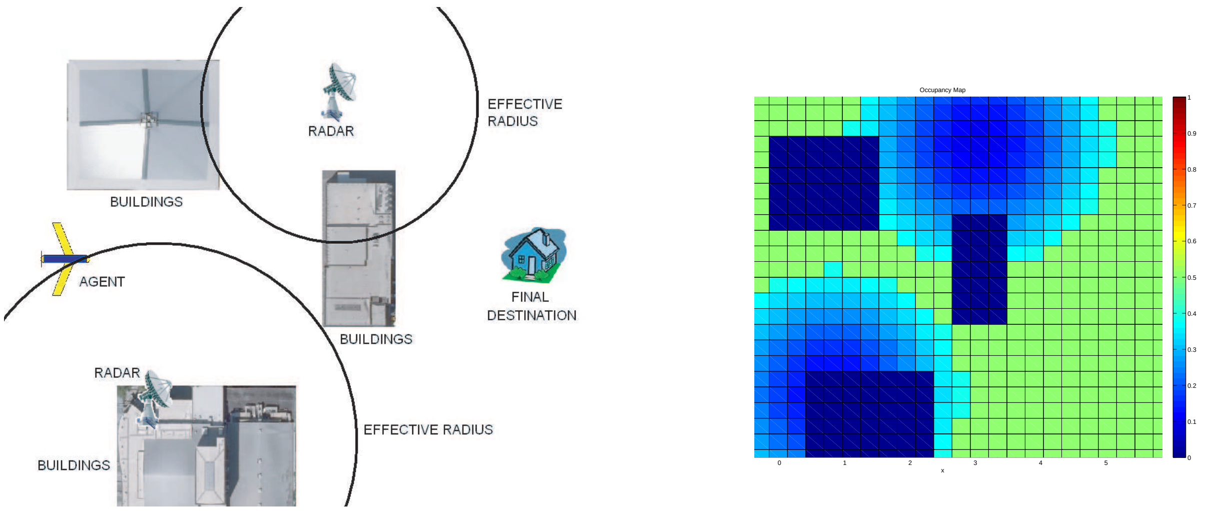
A. Defining Occupancy Based Maps

The search domain is discretized into rectangular cells. Each cell is assigned a score which is the probability that the target is located in that cell. This is similar to a two dimensional, discretized probability density function.[7] The spatial domain of the occupancy based map consists of a box where x is between x_{min} and x_{max}. Similarly for the y dimension.

$$B = \left\{ \bar{z} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \mid z_1 \in [x_{min}, x_{max}], z_2 \in [y_{min}, y_{max}] \right\} \tag{1}$$

The occupancy based map, $x_w()$, is a function defined over the set $B \times \Re$ which assigns a score in the range $[0,1]$ to each element $\bar{z} \in B \subset \Re^2$ at a certain time step $k \in \Re$. In other words, $x_w : B \times \Re \to \Re$. The score of a given cell represents the probability that the target is located in that cell.

The occupancy based map is shared and updated by all agents involved in the search. At each time step, guidance decisions for each agent are computed based on this map. The state of the map at any time k is also referred to as the world state. This reflects the fact that the map represents the possible locations of targets and other objects in the environment. In essence, the system's belief of the state of the world is embedded in the state of the occupancy based map. For example, the map can represent the locations of obstacles and reward areas in the environment. The idea of embedding obstacles in the map is similar to defining obstacles in a configuration space.[27] An example of embedded obstacles and reward areas in an occupancy based map is shown in Figure 2.



(a) Physical environment with soft and hard obstacles (b) Occupancy based map representation of environment

Figure 2. Abstraction of urban environment using occupancy based maps.

In this example, the dark blue sections represent cells with zero scores (hard obstacles) and the green sections represent scores of 0.5 (neutral values). Furthermore, there are regions which correspond to soft obstacles that should be avoided if possible but entering these regions does not violate a constraint. These sections are represented by the lighter blue shades with scores ranging from 0 to 0.5. If there were regions that were beneficial to the agent, these could be assigned scores greater than 0.5.

It is useful to define the cell center set, $\tilde{B}$, as all values $\bar{z}$ which correspond to the center of a cell in the occupancy based map.

$$\tilde{B} = \left\{ \bar{z} \middle| \bar{z} = \begin{pmatrix} x_{min} + L_x(i - 1/2) \\ y_{min} + L_y(j - 1/2) \end{pmatrix}, i = 1, 2, \ldots, N_x, j = 1, 2, \ldots, N_y \right\} \tag{2}$$

B. Updating Map Scores

The occupancy based map is dynamic and can be updated either by agents involved in the mission, external sources, or other means. The agents are able to modify the map to reflect their findings during the search mission. Each agent in the team is able to search the cell at its current location using its sensor. The discrete state space of the cell is simply $X_k = \{x_A, x_B\}$ where $X_k = x_A$ corresponds to the target not in the cell and $X_k = x_B$ meaning that the target is in the cell. In a similar fashion, the agent may make one of two sensor measurements, $Z_t = z_A$ (observe target not in cell) and $Z_t = z_B$ (observe target in cell). As mentioned previously, the score of a given cell in the occupancy based map reflects the scalar probability that the target is located in that cell at the current time step k. For convenience, the score of the cell at time step k is denoted $s_k = p(X_k = x_B)$.

To model a heterogeneous team of agents with stochastic sensors, each agent's sensor is assigned a reliability factor $h \in [0, 1]$. A value of $h = 0$ implies that the sensor is completely unreliable and no information can be gained from this sensor. Conversely, $h = 1$ corresponds to a completely reliable sensor that can ascertain if the target is or is not located in the agent's current cell in a single measurement. The probabilistic sensor model can be formed as

$$\begin{aligned} p(Z_t = z_A | X_t = x_B) &= 1 - \tfrac{1}{2}(h + 1) \\ p(Z_t = z_B | X_t = x_B) &= \tfrac{1}{2}(h + 1) \end{aligned} \tag{3}$$

Assuming that the state of any given occupancy map cell score is not affected by the action of taking a measurement, the probabilistic score of a given occupancy based map cell can be updated using the sensor model of Eq. 3 which yields the following Bayesian update rule.

$$s_k = \begin{cases} \frac{s_{k-1}(1-h)}{1+(1-2s_{k-1})h} & \text{if } Z_k = z_A \\ \frac{s_{k-1}(1+h)}{1+(2s_{k-1}-1)h} & \text{if } Z_k = z_B \end{cases} \tag{4}$$

In Eq. 4, s_k represents the score of the occupancy map cell ($s_k = x_w(k, \overline{z})$ for $\overline{z}$ in some cell region).

This update rule has several interesting properties. It can be shown that the scores of each cell either monotonically increase or decrease with each sensor measurement for the majority of values of h and s_{k-1}. It is trivial to see that if $s_{k-1} = 1$ then $s_k = 1$ and if $s_{k-1} = 0$, then $s_k = 0$ for either case of $Z_k = z_A$ or $Z_k = z_B$ and for all values of h. The physical significance of this is that if the target is absolutely certain to either be in the cell or not, then no further updates will change this fact. Similarly, for $Z_k = z_A$, if $h = 0$ then $s_k = s_{k-1}$ and if $h = 1$ then $s_k = 0$. This implies that if the sensor is completely unreliable ($h = 0$) and making a measurement with this sensor will not change the state of the cell. Conversely, if the sensor is completely reliable with $h = 1$, then only a single measurement of $Z_k = z_A$ is all that is required to change the score of the cell to 0. A similar situation arises for $Z_k = z_B$ where if $h = 0$ then $s_k = s_{k-1}$ but if $h = 1$ then $s_k = 1$. This is because z_B with $h = 1$ is the event of detecting the target with a 100% reliable sensor. The more interesting cases are when s_{k-1} and h are in the interval $(0, 1)$.

Theorem II.1. *Under the update rule of Eq. 4, $s_k < s_{k-1}$ if $Z_k = z_A$ and $s_k > s_{k-1}$ if $Z_k = z_B$ $\forall s_{k-1}, h \in (0, 1)$.*

Proof. For the case of $Z_k = z_A$ the burden is to now show that $s_k < s_{k-1} \forall k$ for the cases where $s_{k-1}, h \in (0, 1)$.

One can examine the denominator of Eq. 4 and note that $\forall s_{k-1} \in (0, 1)$ the term $1 - 2s_{k-1} > -1$. Therefore, the denominator term must be greater than $1 - h$. Noting that for all $h \in (0, 1)$ the term $1 - h > 0$. Therefore, the denominator term must always be positive.

Furthermore, it is easy to see that $1 - s_{k-1} > 0$ and $2h > 0$ for these conditions. So one can write

$$0 < 2h(1 - s_{k-1}) \tag{5}$$

$$1 - h < 1 + h - 2hs_{k-1} \tag{6}$$

$$\frac{1 - h}{1 + h - 2hs_{k-1}} < 1 \tag{7}$$

$$\tag{8}$$

Multiplying both sides by s_{k-1} yields the final result

$$\frac{s_{k-1}(1 - h)}{1 + (1 - 2s_{k-1})h} = s_k < s_{k-1} \;\; \forall h, s_{k-1} \in (0, 1) \tag{9}$$

It is trivial to show that $s_k > 0\ \forall k$. This in conjunction with Eq. 9 shows that with enough sensor measurements of $Z_k = z_A$, the score of a given cell will proceed monotonically towards 0.

A similar proof can be applied for the case of $Z_k = z_B$ with the result showing that with enough sensor measurements of $Z_k = z_B$, the score of a given cell will proceed monotonically towards 1. □

A standard Markov assumption is used to remove the dependence of the next score on past scores, and therefore only one version of the occupancy based map must be maintained at any given time step. The map can be made time varying using a number of techniques.[28]

One final aspect of system modeling that concerns updating the map is how the agents update the map when they encounter a target. If the agent searches the cell where the target is located and successfully finds the target, it can update the scores of its current cell and those around it. The sensor reliability factor is also used to model a sensor which may miss a positive target identification. If h is low, there is a high probability that the agent will not find the target even if it searches the correct cell. This behavior affects several performance metrics such as the average time to target detection, which is discussed later in Section V.

The occupancy based map and its associated features provides a versatile framework from which to build a searching algorithm.

III. Single Agent Search Strategy

This section concentrates on how a single agent is to find an optimal location to search using the occupancy based map. The goal for the agent is to converge on regions of high score (a high probability that the target is located there). Once an agent locates an anomaly, it should loiter there until a positive identification can be made. The overall flow of the search strategy for a single agent is shown below in Figure 3.

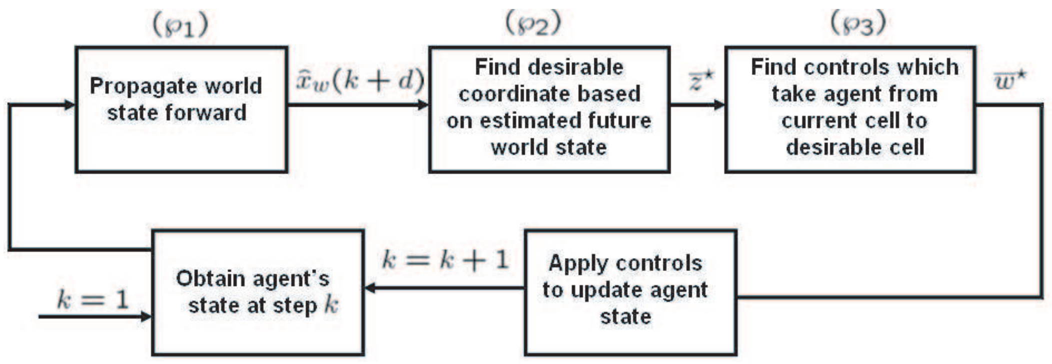

($\wp_1$)
Propagate world state forward
$\hat{x}_w(k+d)$
($\wp_2$)
Find desirable coordinate based on estimated future world state
$\overline{z}^\star$
($\wp_3$)
Find controls which take agent from current cell to desirable cell
$\overline{w}^\star$
$k=1$
Obtain agent's state at step k
$k=k+1$
Apply controls to update agent state

Figure 3. Flow diagram for single agent search strategy.

The process is comprised of three subproblems which are referred to as ($\wp_1$), ($\wp_2$), and ($\wp_3$). The process starts by finding the agent's state at the current time, $\overline{x}_{agt}(k)$. Next, ($\wp_1$) is solved to obtain the estimated world state at time $k + d$. Next, ($\wp_2$) is solved to find a desirable coordinate, $\overline{z}^\star$, which the agent will visit within the next d steps. Finally, ($\wp_3$) consists of finding an optimal set of waypoints/controls, $\overline{w}^\star$, which will take the agent from its current location to the location of the desirable coordinate found in ($\wp_2$).

A. ($\wp_1$) Predictive World Model

The first problem, ($\wp_1$), involves projecting the current state of the world forward in time to create an estimate of the world state at step $k + d$. Previous work investigated using various linear predictive models to generate future estimates of the world state[23],[29] In practice, any methods may be used to accomplish this goal. This aspect of the control algorithm is not the main focus of this paper and will not be covered further.

Once the the state of the world is estimated at step $k + d$, the system attempts to find a coordinate that has desirable properties and is located within the agent's reachable set. This is addressed in ($\wp_2$).

B. $(\wp_2)$ Desirable Location Selection

The agent needs to choose a location that it must travel to based on the estimated future state of the world. This section describes a method that can be used to choose a desirable cell to search.

1. *Defining the Reward Function*

In $(\wp_1)$, the state of the world is propagated forward in time by d steps. The subproblem $(\wp_2)$ concerns finding a desirable location (a desirable $\overline{z}$ value) for the agent to travel to in d steps. The desirability of a location is determined by a reward function of the following form

$$J_0(\overline{z}) = \alpha \hat{x}_w(k+d,\overline{z}) + \eta\left(\beta f_\chi(\overline{z}) + \gamma f_d(\overline{z})\right) + \delta f_h(\overline{z}) \tag{10}$$

The reward function is a combination of terms which model both the environment and agent states. For example, the term $\hat{x}_w(k+d,\overline{z})$ measures the estimated state of the world d steps into the future. The function $f_\chi()$ is a function which penalizes heading changes. In a similar fashion, $f_d()$ rewards locations which are farther away from the agent. Finally, $f_h()$ is an indicator function which serves to drive the agent towards the highest score in the map. This describes the general behavior of the cost function, but a more detailed look at its construction is necessary.

With the cost function defined, the most desirable location is then found via an optimization scheme using Eq. 10 as an objective function. Typically, the set over which one optimizes Eq. 10 is the set of all locations that the agent can reach in d steps. This set is determined by parameters such as its maximum velocity and time step. The set of all locations that the agent can reach in d steps is referred to as the agent's reachable set, $B_R \subseteq B$. This application defines B_R as

$$B_R = \{\overline{z} \in B \mid ||\overline{z} - \overline{z}_{agt}|| \leq R_{max}\} \tag{11}$$

Eq. 11 assumes that the agent has no turn rate limits and the agent has simple planar kinematics. In a practical application where there may be saturation concerns, it is possible that B_R may not be a perfect circle as described in Eq. 11. In this case, it simply becomes more difficult to define and compute B_R but the following analysis is not affected by the geometry of B_R.

It is useful to also define the set of cell centers that the agent can reach in d steps. This is simply

$$\tilde{B}_R = B_R \bigcap \tilde{B} \tag{12}$$

In Eq. 10, the function $f_\chi()$ is given by

$$f_\chi(\overline{z}) = \begin{cases} 0 & \text{if } \overline{z} \text{ in same cell as current agent} \\ 1 - \frac{q\left(\chi_{agt}, \pi/2 - \text{atan2}(\overline{z}_2 - y_{agt}, \overline{z}_1 - x_{agt})\right)}{\pi} & \text{otherwise} \end{cases} \tag{13}$$

In Eq. 13, the function $q(a,b)$ computes the absolute angular difference between the two angles, a and b. A simple absolute value of the difference of a and b is not sufficient and some simple heuristics are included in the function $\eta()$ of Eq. 10 to take care of situations such as where $a = 1\cdot\pi/180$ radians and $b = 359\cdot\pi/180$ radians. A simple absolute value of the difference would return an angle of $358\cdot\pi/180$ radians, which is incorrect. However, the function $\eta()$ returns the correct angular difference of $2\cdot\pi/180$ radians. Note that the range of $f_\chi()$ is $[0,1]$. An example of this function is shown in Figure 6(c).

The function $f_d()$ of Eq. 10 is given by

$$f_d(\overline{z}) = \begin{cases} \frac{||\overline{z} - \overline{z}_{agt}||}{R_{max}} & \text{if } \overline{z} \in B_R \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

The function $f_d()$ effectively rewards locations that are farther away from the current agent position until the distance R_{max} is reached; past this point, the function returns 0. An example of this function is shown in Figure 6(d).

Finally, the function $f_h()$ of Eq. 10 is an indicator function. To define this function, it becomes necessary to first define some intermediate variables. The first of these variables is given by

$$\tilde{B}_{max} = \left\{\overline{z} \in \tilde{B} \mid \hat{x}_w(k+d,\overline{z}) \text{ is maximum } \forall \overline{z} \in \tilde{B}\right\} \tag{15}$$

The set $\tilde{B}_{max}$ is simply the set of cell centers that have the highest score in the map. Note that this set may have more than one cell center. For further clarification, the relationship between the sets $\tilde{B}_{max}$, B, B_R, and $\tilde{B}_R$ are shown in Figure 4.

A single cell center from $\tilde{B}_{max}$ can be chosen using one of two methods. The first method involves choosing the point in $\tilde{B}_{max}$ which is closest to the agent. This location is designated as $\overline{z}_H$

$$\overline{z}_H \in \underset{\overline{z} \in \tilde{B}_{max}}{\text{arg minimize}} \ ||\overline{z} - \overline{z}_{agt}|| \tag{16}$$

An alternative method for choosing $\overline{z}_H$ is to use

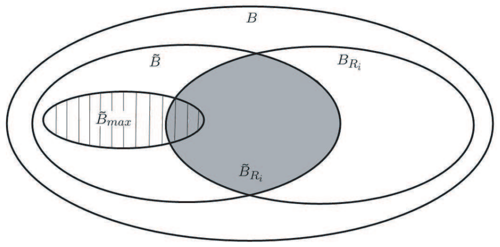
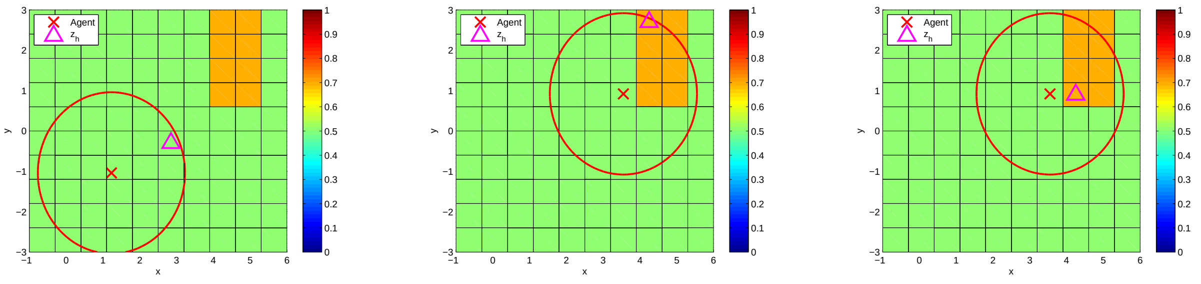


Figure 4. Relationships between sets B, $\tilde{B}$, $\tilde{B}_{max}$, B_{R_i}, and $\tilde{B}_{R_i}$ (subscript i denotes set for agent i).

$$\overline{z}_H \in \begin{cases} \underset{\overline{z} \in \tilde{B}_{max}}{\text{arg minimize}} \ ||\overline{z} - \overline{z}_{agt}|| & \text{if } \tilde{B}_{max} \bigcap \tilde{B}_R = \emptyset \\ \underset{\overline{z} \in \tilde{B}_{max} \bigcap \tilde{B}_R}{\text{arg maximize}} \ ||\overline{z} - \overline{z}_{agt}|| & \text{otherwise} \end{cases} \tag{17}$$

The method for choosing $\overline{z}_H$ in Eq. 17 entails first checking if the set $\tilde{B}_{max} \bigcap \tilde{B}_R = \emptyset$. If this is true, this implies that there are no cells that have the highest score of the map in the agent's reachable set. If this is the case, Eq. 16 and Eq. 17 will choose the same point for $\overline{z}_H$. This can be seen in Figure 5(a). The two methods deviate if there is a cell with the highest score of the map within the agent's reachable set. In this case, Eq. 16 would still choose the cell in $\tilde{B}_{max}$ which is closest to the agent as shown in Figure 5(c). On the other hand, Eq. 17 would instead choose the cell with the maximum score that is farthest from the agent, but is still in the agent's reachable set as shown in Figure 5(b). Either method can be used and each has advantages and disadvantages which can be used to tailor to behavior of the agents for different scenarios. For example, using Eq. 17 tends to generate behaviors where the agents must move a large distance within d steps. This leaves little flexibility when it comes to planning a path from the agent's current location to $\overline{z}^\star$ (defined in Eq. 21). Alternatively, Eq. 16 tends to choose $\overline{z}^\star$ locations which are close to the agent's current location and therefore, the path planning algorithm[24] in $(\wp_3)$ will be allow to choose paths other than straight paths to $\overline{z}^\star$.



(a) $\overline{z}_h$ using either Eq. 16 or 17 (b) $\overline{z}_h$ using Eq. 17 (c) $\overline{z}_h$ using Eq. 16

Figure 5. Differences between using Eq. 16 and 17 for choosing $\overline{z}_H$.

With the point $\overline{z}_H$ defined, the point in the agent's reachable cell centers that is closest to $\overline{z}_H$ is given by

$$\overline{z}_h \in \underset{\overline{z} \in \tilde{B}_R}{\text{arg minimize}} \ ||\overline{z} - \overline{z}_H|| \tag{18}$$

With $\overline{z}_H$ and $\overline{z}_h$ defined, the function $f_h()$ is simply given as

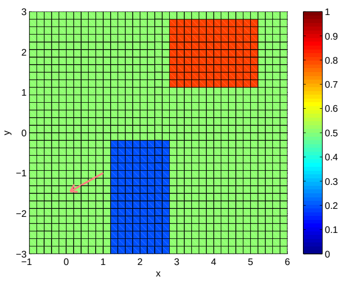
$$f_h(\overline{z}) = \begin{cases} 1 & \text{if } \overline{z} \text{ is in cell containing } \overline{z}_h \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

An example of the $f_h()$ function is shown in Figure 6(e).

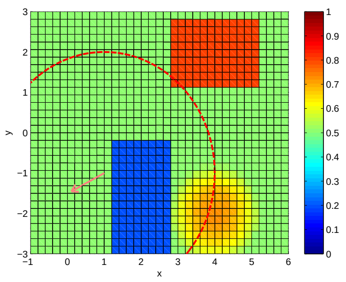
The final parameter in the reward function is the variable η which is the maximum score within the agent's reachable set.

$$\eta = \max_{\bar{z} \in \tilde{B}_R} \hat{x}_w(k+d, \bar{z}) \tag{20}$$

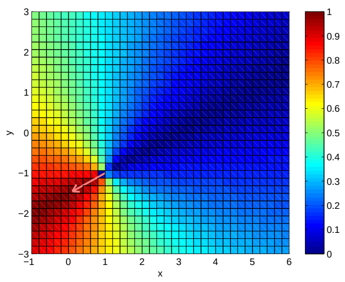
An example of the various functions that make up the reward function are shown below in Figure 6.



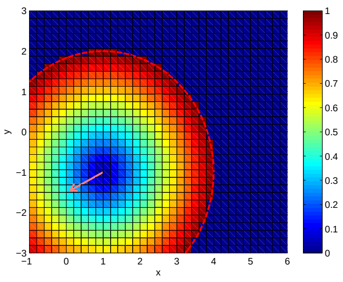
(a) $x_w(k, \bar{z})$



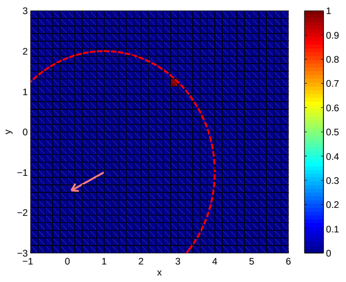
(b) $\hat{x}_w(k+d, \bar{z})$. $\alpha = 0.5$



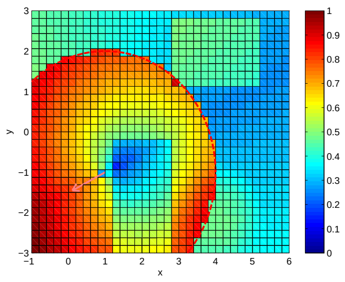
(c) $f_\chi(\bar{z}).\chi_{agt} = 243°$. $\beta = 0.5$



(d) $f_d(\bar{z})$. $\gamma = 0.5$



(e) $f_h(\bar{z})$. $\delta = 0.05$



(f) $J_0(\bar{z})$

Figure 6. Various functions which compose the reward function, $J_0()$.

Figure 6(a) shows the state of the world at the current time. This is the system's belief of the world without making any estimates of the target location.

Figure 6(b) shows the estimated state of the world in d steps. In this case, an external source has provided information that the target is likely located in the lower right corner. This estimate can be made by a team member or another source in the system. The dashed red line shows the maximum distance the agent can reach in d steps. Maximizing this function over $\tilde{B}_R$ requires choosing the cell with the highest score within the dashed red line.

Figure 6(c) shows the $f_\chi()$ function for the particular case of the agent having a course angle of 243°. Maximizing this function requires picking a location $\bar{z}$ which minimizes the course change required to visit this location ($\bar{z}$ locations which are in line with the pink arrow).

Figure 6(d) shows the $f_d()$ function. Maximizing this function corresponds to choosing a $\bar{z}$ location which is farthest away from the current location.

Figure 6(e) shows the indicator function $f_h()$. In this scenario, the set $\tilde{B}_{max}$ is the set of cell centers which maximize $\hat{x}_w(k+d, \tilde{B})$. These correspond to the cell centers of the dark red rectangle in Figure 6(b). From the set $\tilde{B}_{max}$, $\bar{z}_H$ is chosen as the cell that is closest to the agent (the lower left corner point). Finally, the point $\bar{z}_h$ is determined as the point within the agent's reachable cell centers which is closest to $\bar{z}_H$. As can be seen, maximizing $f_h()$ requires simply choosing $\bar{z}$ in the same cell as $\bar{z}_h$.

Figure 6(f) shows the normalized total reward function. The reward function can typically exceed 1 but it is normalized so that the max value is 1 for plotting purposes.

The parameters α, β, γ, and δ can be chosen differently to reflect different agent capabilities. For example, small α values yield agents which will not be adverse to large course changes. This may be appropriate for more agile agents such as small UAVs. Conversely, large α values correspond to agents which may deem a coordinate more desirable if it has a somewhat low score but is in line with its current course. This may be appropriate for cumbersome agents like large boats. One important aspect of Eq. 10 is that α, β, γ, δ, $\hat{x}_w()$, $f_\chi()$, $f_d()$, and $f_h()$ are all in the range $[0, 1]$. This is explained further in Section ??.IV.B.

Previous work[23] investigated using adaptive sampling methods to provide quasi-optimal solutions of Eq. 10 over the feasible set B_R. Although this method worked well for most cases, it was nondeterministic. The stochastic nature when selecting $\overline{z}^\star$ makes analysis difficult. Therefore, instead of optimizing $J_0()$ over the compact set B_R, the problem is reduced to optimizing over the feasible set of $\tilde{B}_R$. With this approximation, the most desirable cell, and solution to $(\wp_2)$ is given by

$$(\wp_2) \quad \overline{z}^\star \in \arg \underset{\overline{z}\in\tilde{B}_R}{\text{maximize}}\ J_0(\overline{z}) \tag{21}$$

The problem formulated in Eq. 21 differs in that the feasible set is manageably finite. Therefore, the optimization is reduced to an exhaustive search over the set $\tilde{B}_R$. An additional benefit of this method is that it can be easily implemented by evaluating the reward function at each cell center in the set $\tilde{B}_R$.

C. $(\wp_3)$ Transition From Current Location to Desired Location

The final subproblem, $(\wp_3)$, concerns finding feasible waypoints which take the agent from the current location, $\overline{z}_0$, to the desirable cell location $\overline{z}^\star$ found in $(\wp_2)$. Any path planning algorithm may be used as long as they meet the requirement that the agent arrives at the point $\overline{z}^\star$ in d steps or less. Previous work investigated addressing this problem as a convex optimization scheme[23] and using a graph-based approach.[24] Many other authors have investigated the path planning in complex environments using evolutionary programming,[30] rapidly exploring random trees,[31] and weighted regions.[32]

The path planning algorithm used to solve $(\wp_3)$ is not the focus of this paper, so the reader is referred to the previously mentioned papers for more information.

IV. Exhaustive Searching

The primary goal of a search mission is to find one or more targets which are located somewhere in the domain. One would be concerned if there are conditions where the target might be able to hide in the environment and avoid detection by the agents. We will show that under a reasonable set of assumptions, the agents are guaranteed to visit all cells in the map with non-zero score sufficiently often to drive the cell scores to zero. In other words, they will exhaustively search and cover the entire area of interest. The assumptions are

$$\text{Assumptions:} \quad \begin{array}{ll} h \in (0,1) & (A.1) \\ \delta \in (0,1] & (A.2) \\ R_{max} \geq \max(L_x, L_y) & (A.3) \\ Z_k = z_A \ \ \forall k & (A.4) \\ \hat{x}_w(k+d,\overline{z}) = x_w(k,\overline{z}) \ \ \forall k & (A.5) \\ x_w(0,\overline{z}) \in [0,1) \ \ \forall \overline{z} & (A.6) \\ \delta > \beta + \gamma & (A.7) \end{array}$$

The physical meaning of assumption A.1 is to ensure that each agent has a sensor that can be relied upon to some degree. Eliminating the possibility that $h = 0$ ensures that with each sensor measurement, the score of the searched cell decreases. Note that the case of $h = 1$ implies an infinitely reliable sensor which is actually the best scenario. However, this requires rewording several following theorems so it is simply excluded using this set of assumptions. All of the following analysis can apply for the case of $h = 1$ with slight changes.

Recall from Section 1, the scalars α, β, γ, and δ are in the range of $[0, 1]$. Assumption A.2 ensures that $\delta \neq 0$. Its significance will become clear in Section IV.B.

Assumption A.3 effectively specifies a minimum size of the set $\tilde{B}_R$. This states that there must be more than one cell center in $\tilde{B}_R$. Furthermore, this guarantees that $\tilde{B}_R$ includes the cells centers to the North, East, South, and West of the current agent location and ensures that the agent can move in any direction and eventually reach any other cell under an appropriate control law.

Assumptions A.4, A.5, and A.6 deal with the nature of the complete coverage of the map. Complete coverage implies the agents will search each cell in the map sufficiently to drive its score to zero. In a scenario involving complete coverage of the map, the target is located in the last cell that the agents would search or is simply not located in the map at all. These assumptions ensure that the scores of the map are never increasing and can in fact decrease under Eq. 4.

Assumption A.7 is crucial for the proof of complete coverage and is explained in Section IV.B.

All of these assumptions are reasonable and can be implemented easily. Under these assumptions we will show that the agents operating under this strategy will exhaustively search the map and drive all cell scores towards zero given sufficient time.

A. Decreasing Scores

As the agents search a cell, it is desired that the score of the cells decrease. We show that the scores of each cell monotonically decrease with each sensor measurement.

Theorem IV.1. *Under assumptions A.1, A.4, A.5, and A.6 and the previously described and search strategy, the score of any given occupancy map cell with a score not equal to* 0 *or* 1 *will monotonically decrease towards 0.*

Proof. Recall that the score of the cell is updated via Eq. 4 with $Z_k = z_A \; \forall k$. It is trivial to see that if $s_{k-1} = 1$ then $s_k = 1$ and if $s_{k-1} = 0$, then $s_k = 0$. The burden is to now show that $s_k < s_{k-1} \; \forall k$ for the cases where $s_{k-1} \in (0, 1)$. This was already proved in Theorem II.1

It is trivial to show that $s_k > 0 \; \forall k$. This, in conjunction with Eq. 9, shows that with enough sensor measurements of $Z_k = z_A$, the score of a given cell will proceed monotonically towards 0. □

B. Guaranteed Coverage

Theorem IV.1 shows that if a cell is searched a sufficient number of times, its score will decrease towards zero. In order to show that the agents exhaustively search the map, the burden is to show that under the given control law, the agent will visit each cell enough times to decrease the scores to zero.

Theorem IV.2. *Under assumptions A.3, A.4, A.5, and the previously described search strategy, if an agent is not currently in the cell containing* $\overline{z}_H$*, the agent must move to a new cell once the quantity* $\alpha x_w(k, \overline{z}_{agt})$ *decreases below the value* δ*.*

Proof. If the agent is not in the cell containing $\overline{z}_H$, then it cannot be at the location $\overline{z}_h$ either since A.3 $\Leftrightarrow$ $|\tilde{B}_R| > 1$ and by definition of $\overline{z}_h$, $\overline{z}_h$ must be closer to $\overline{z}_H$ than the current agent position. This implies that $f_h(\overline{z}_{agt}) = 0$. Similarly, by definition in Eq. 13, $f_\chi(\overline{z}_{agt}) = 0$. The maximum possible reward gained by choosing the same cell as the current agent (denoted $\overline{z}_s$) is

$$J_0(\overline{z}_s) = \alpha x_w(k, \overline{z}_s) + \eta\gamma f_d(\overline{z}_s) \tag{22}$$

By definition, $\overline{z}_h \in \tilde{B}_R$. It is possible that $x_w(k, \overline{z}_h) = f_\chi(\overline{z}_h) = 0$ (if the score is already zero and the cell is located directly behind the agent) so the minimum possible value of the reward function at this point is

$$J_0(\overline{z}_h) = \eta\gamma f_d(\overline{z}_h) + \delta \tag{23}$$

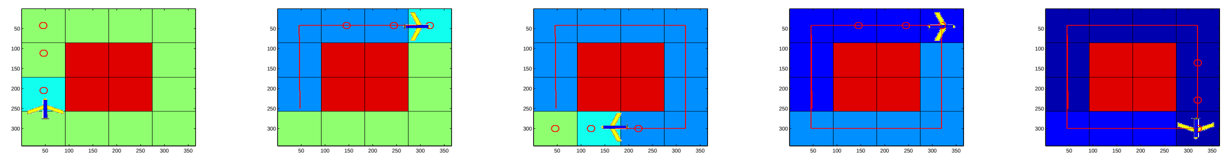
If the control solution from ($\wp_3$) performs correctly, the agent should be located at the point $\overline{z}^\star$ from ($\wp_2$) and $f_d(\overline{z}_s) = 0$. However, the proof can be applied for the case where the agent is not necessarily located on a cell center at all times. From a geometric perspective, the maximum value of $f_d(\overline{z}_s) = (L_x^2 + L_y^2)^{1/2}/R_{max}$ which corresponds to the agent located in the corner of the cell. The distance to any other cell in $\tilde{B}_R$ must be at least that if not greater, so $f_d(\overline{z}_h) \geq f_d(\overline{z}_s)$. This result, combined with Eq. 22 and Eq. 23, yields the result that

$$\{\alpha x_w(k, \overline{z}_s) < \delta\} \Rightarrow \{J_0(\overline{z}_h) > J_0(\overline{z}_s)\} \tag{24}$$

□

This shows the existence of a point with a higher score than the agent's current cell. This gives a lower bound of the score when the agent must choose to leave the current cell. It is possible that the agent will chose a different cell before $\alpha x_w(k, \overline{z}_s) < \delta$.

Theorems IV.1 and IV.2 ensure that an agent searching a non-zero score cell will decrease the score consistently and an agent cannot remain at a given cell indefinitely. Combined, these two ensure movement of the agent to different cells but does not guarantee that the map is covered. In fact, there are counter examples where an agent can constantly choose cells of nearly zero score and never search all of the map, thereby ignoring certain cells which have non-zero scores. So far, the previous theorems only required assumptions A.1 - A.6. The example in Figure 7 shows that although assumptions A.1- A.6 may be satisfied, this is not sufficient to guarantee exhaustive map coverage.



(a) $t = 0$ (b) $t = 195$ (c) $t = 325$ (d) $t = 520$ (e) $t = 845$

Figure 7. Example with $\alpha = 0.15$, $\beta = 0.95$, $\gamma = 0.85$, $\delta = 0.10$ showing a map not being covered due to Assumption A.7 violation.

In this example, the agent continues to fly around the perimeter of the map and never explores the interior. This is due to the fact that the cost of changing heading to turn into the interior of the map is too high relative to the other terms in the cost function. In order to guarantee complete coverage of the map, the restriction of A.7 must be enforced.

Theorem IV.3. *Under assumption A.7 and the previously described search strategy, the agent must choose either a cell which has non-zero score, contains $\overline{z}_h$, or both as the solution to $(\wp_2)$.*

Proof. Assume there exists a point $\overline{z}_0$ which has a score of zero and is not $\overline{z}_h$. In this case $x_w(k, \overline{z_0}) = f_h(z_0) = 0$. The maximum value of the reward function at this point is

$$J_0(\overline{z}_0) = \eta(\beta + \gamma) \tag{25}$$

The minimum value of the reward function obtained by choosing $\overline{z}_h$ is

$$J_0(\overline{z}_h) = \delta \tag{26}$$

Since $\eta \in [0, 1]$, these two results along with the assumption $\delta > \gamma + \beta$ ensures that $J_0(\overline{z}_h) > J_0(\overline{z}_0)$. This guarantees that the agent will not choose a cell if it has zero score and is not $\overline{z}_h$. Therefore, the only other options are to choose a cell which has a non-zero score, contains $\overline{z}_h$, or both. $\square$

Theorem IV.4. *Under assumptions A.4, A.5 and the previously described search strategy, if Eq. 16 is used to choose $\overline{z}_H$ and the agent chooses $\overline{z}_h$ as the solution to $(\wp_2)$ and $\overline{z}_h \neq \overline{z}_H$, then at the next time step the point $\overline{z}_H$ will remain unchanged unless it is searched by another agent.*

Proof. Consider the distance between $\overline{z}_H$ and $\overline{z}_{agt}$ at a given time step k $(||\overline{z}_H - \overline{z}_{agt}||)$. By definition, $\overline{z}_h \neq \overline{z}_H \Leftrightarrow \overline{z}_H \notin \tilde{B}_R$. So if the agent chooses $\overline{z}_h$ as the solution to $(\wp_2)$ and $\overline{z}_h \neq \overline{z}_H$, the agent must move closer to the point $\overline{z}_H$ in the sense that $||\overline{z}_H - \overline{z}_{agt}||$ cannot increase. Since the point $\overline{z}_h$ is not equal to $\overline{z}_H$ then the agent cannot search the cell containing $\overline{z}_H$ and score at $\overline{z}_H$ will not change. Therefore, at the next time step, the same point $\overline{z}_H \in \tilde{B}_{max}$. Furthermore, since the distance between the agent and the same point $\overline{z}_H$ has decreased, it will be chosen again as $\overline{z}_H$ using Eq. 16.

Of course, if the point $\overline{z}_H$ is searched first by another agent, its score may decrease and it is possible that at the next step the same point is no longer in $\tilde{B}_{max}$ due to the fact that the score decreased. $\square$

A similar theorem can be stated for the case where $\overline{z}_H$ is chosen using Eq. 17.

Theorem IV.5. *Under assumptions A.4, A.5 and the previously described search strategy, if Eq. 17 is used to choose $\bar{z}_H$ and the agent chooses $\bar{z}_h$ as the solution to $(\wp_2)$ and $\bar{z}_h \neq \bar{z}_H$, then at the next time step the point $\bar{z}_H$ will either remain unchanged or be located in the agent's reachable set, $\tilde{B}_R$ unless it is searched by another agent.*

Proof. Recall that if $\tilde{B}_{max} \bigcap \tilde{B}_R = \emptyset$ (there are no cells with maximum score within the agent's reachable set), then both Eq. 16 and Eq. 17 choose the same cell center for $\bar{z}_H$, so the proof of Theorem IV.4 applies and shows that as long as $\tilde{B}_{max} \bigcap \tilde{B}_R = \emptyset$, the point $\bar{z}_H$ will remain unchanged under these assumptions. The difference between Eq. 16 and Eq. 17 occurs when $\tilde{B}_{max} \bigcap \tilde{B}_R \neq \emptyset$.

Assume at step k, the $\tilde{B}_{max} \bigcap \tilde{B}_R = \emptyset \Rightarrow \bar{z}_H \notin \tilde{B}_R$. Assuming that the agent chooses $\bar{z}_h$ as the solution to $(\wp_2)$ and $\bar{z}_h \neq \bar{z}_H$, the previous analysis shows that $\bar{z}_H$ does not change. However, at the next step $k+1$, if the same point $\bar{z}_H \in \tilde{B}_R$, then it is possible that Eq. 17 will not choose the same point $\bar{z}_H$ again because there might be another member of $\tilde{B}_{max} \bigcap \tilde{B}_R$ which is farther away from $\bar{z}_{agt}$. However, it is guaranteed that $\tilde{B}_{max} \bigcap \tilde{B}_R \neq \emptyset$ since the same point $\bar{z}_H$ from step k is in $\tilde{B}_{max}$ and $\tilde{B}_R$ under these assumptions, thus ensuring that $\bar{z}_H \in \tilde{B}_R$.

Of course, if the point $\bar{z}_H$ is searched first by another agent, its score may decrease and it is possible that at the next step the same point is no longer in $\tilde{B}_{max}$ due to the fact that the score decreased. □

Figure 5 can be used as a visual aid for understanding the proofs of both Theorems IV.4 and IV.5. These theorems can be used to guarantee that the team performs an exhaustive search of the map given the proper conditions.

Theorem IV.6. *Under the previously described assumptions and search strategy, $x_w(k, \bar{z}) \to 0 \ \forall \bar{z} \in B$ (the scores of all cells in the map will approach 0).*

Proof. Theorem IV.2 guarantees that an agent cannot remain in a single cell indefinitely and Theorem IV.3 ensures that it must choose either a cell of non-zero score, $\bar{z}_h$, or both as the solution to $(\wp_2)$ at any given time step. If the agent chooses a cell of non-zero score, Theorem IV.1 ensures that the score of that cell is monotonically decreased towards 0.

If the agent does not choose a cell of non-zero score, the only alternative scenario allowed by Theorem IV.3 is that the agent chooses $\bar{z}_h$ and $x_w(k, \bar{z}_h) = 0$. By definition of $\bar{z}_H$, if $x_w(k, \bar{z}_H) = 0$, then the map has been completely covered since all scores are at most 0. Therefore, assuming that the map has not been entirely searched yet, $x_w(k, \bar{z}_h) = 0 \Rightarrow \bar{z}_h \neq \bar{z}_H$.

Theorem IV.4 and Theorem IV.5 ensure that if $\bar{z}_h \neq \bar{z}_H$, choosing $\bar{z}_h$ as the solution to $(\wp_2)$ does not change the value of $\bar{z}_H$ at the next time step or if $\bar{z}_H$ does change, $\bar{z}_H \in \tilde{B}_R$ at the next time step.

At this next time step, the agent once again must choose a cell with non-zero score, $\bar{z}_h$, or both. If the agent continues to choose $\bar{z}_h$, eventually $\bar{z}_H \in \tilde{B}_R$ and at this point, choosing a cell with non-zero score, $\bar{z}_h$, or both guarantees that a cell of non-zero score is chosen. Therefore, the score of some cell will be ensured to decrease and given sufficient time, $x_w(k, \bar{z}) \to 0 \ \forall \bar{z} \in B$.

□

Note that the results of Theorem IV.6 yield a stronger result than simply ensuring that each cell is visited during the mission. In the specified framework, visiting a cell once may not be enough to guarantee that the target is not located in that cell. In the case where the agent has an unreliable sensor, the amount of information obtained from a single visit to a cell may not decrease the score of that cell sufficiently. Theorem IV.6 and its associated proof show that the map will be exhaustively searched for the target using any number of agents in the sense that each cell is visited a sufficient number of times to drive the scores of all cells in the map to zero.

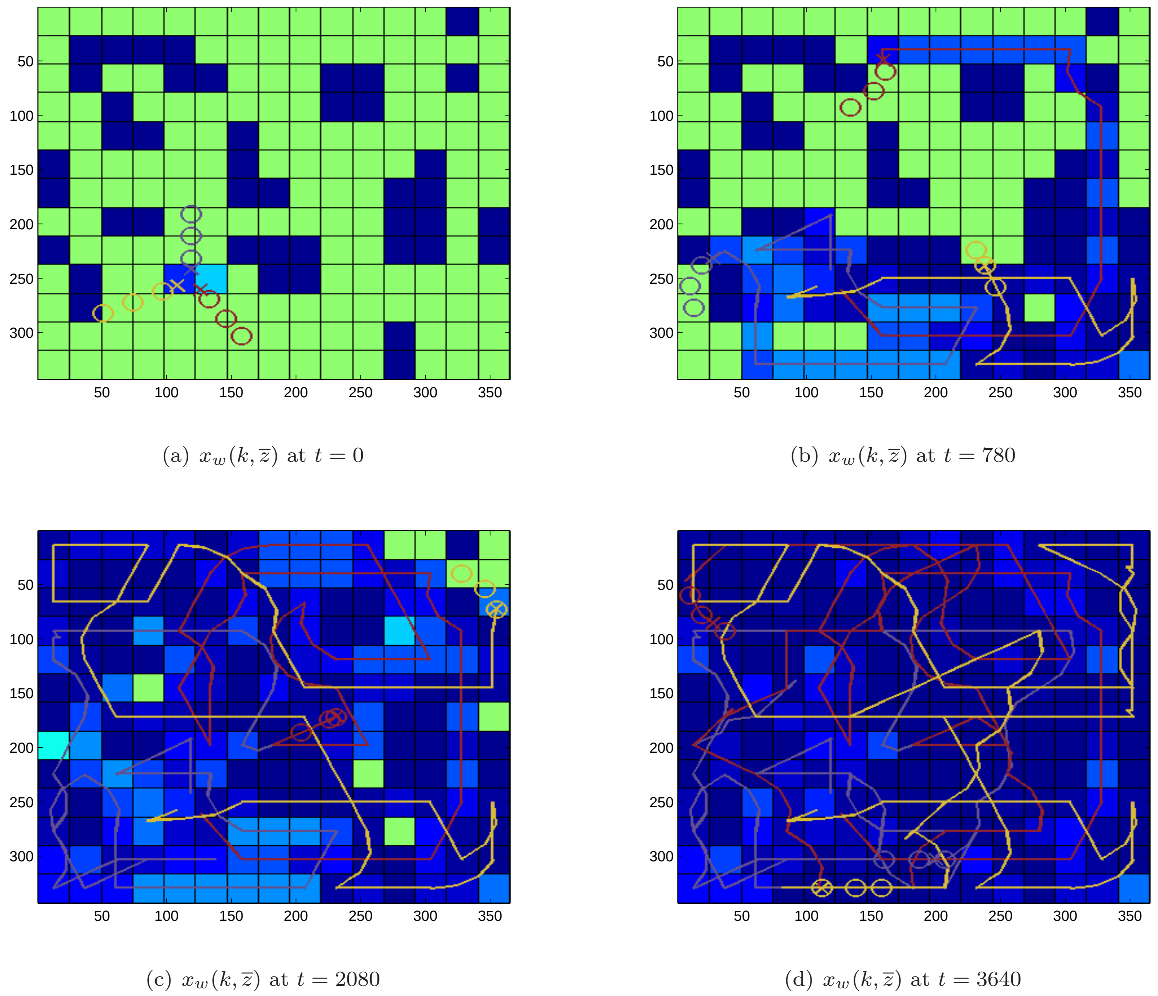
Note that only one agent must be constricted by Assumption A.7 in order to guarantee an exhaustive search by the team. Performance may be increased by relaxing Assumption A.7 for some agents but it cannot be guaranteed that the map will be covered by the agents who do not satisfy Assumption A.7

V. Simulation Results

After analyzing the previously described search algorithm, it can be verified in simulation. A heterogeneous team of agents can be simulated in different environments using the framework described. A heterogeneous team can be modeled by varying parameters in Eq. 10. The parameters used for the different agents using the full algorithm in are listed in Table 1. Simulation results of this test scenario are shown in Figure 8.

Table 1. Parameters of agents in team during search missions ($d = 3$ for all agents) using full algorithm (no explicit cooperation).

Agent	α	β	γ	δ	h	A.7	$\overline{z}_H$ method
1 (purple)	0.75	0.40	0.95	0.15	0.25	No	Eq. 17
2 (red)	0.90	0.50	0.90	0.05	0.35	No	Eq. 17
3 (gold)	0.10	0.05	0.25	0.35	0.55	Yes	Eq. 17



(a) $x_w(k, \overline{z})$ at $t = 0$

(b) $x_w(k, \overline{z})$ at $t = 780$

(c) $x_w(k, \overline{z})$ at $t = 2080$

(d) $x_w(k, \overline{z})$ at $t = 3640$

Figure 8. Full algorithm applied to 3 agents with typical sensors using Eq. 17 to choose $\overline{z}_H$.

The 'x' shows the location of the agent. The 'o' marks illustrate the agent's path, $\overline{w}^\star$, which are the waypoints determined to be the solution to ($\wp_3$). In this situation, the prediction horizon is $d = 3$, so there are 3 'o' marks for each agent. Recall that the only constraint is that the last location of $\overline{w}^\star$ must be equal to $\overline{z}^\star$. Note that only agent 3 (gold) meets the requirement that $\delta > \beta + \gamma$. Having one such agent is sufficient to guarantee complete coverage and eventually the team of agents exhaustively searches the entire map.

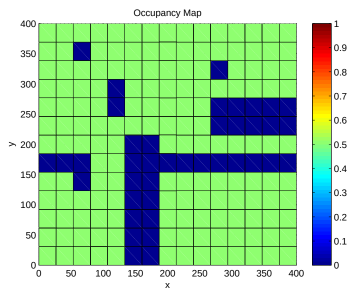
In this scenario, the agent's sensors are considerably degraded. Due to their decreased reliability, a single visit to a cell only decreases the score to slightly less than the original score. This can be seen since the cell colors change from green to a light blue instead of a dark blue. Multiple visits to the same cell are required to change the color to dark blue (a score of near 0). Recall the results from Theorem IV.6 ensure that the agents visit each cell sufficiently often to drive the scores to zero. Therefore, the algorithm routes the agents to initially search the cells of high probability and then revisits cells as required to drive the scores to zero, guaranteeing an exhaustive search of the map. Additional test cases and discussion can be found in.[28]

A. Scenarios and Performance

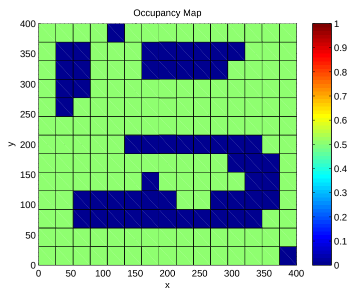
To evaluate the performance of the various algorithms, several test scenarios are created. These scenarios are used to emulate situations and environments where a general search algorithm might be used.

1. *Testing Scenarios*

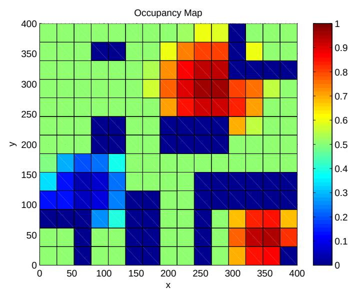
The algorithm can be tested using several different scenarios. Three representative scenarios are shown in Figure 9. Scenario 1 (S1) is used to represent a possible urban scenario where there are hard obstacles such as buildings to avoid. Scenario 2 (S2) represents a maritime environment where the agents might be searching for a lost ship on the water. Finally, scenario 3 (S3) is another urban environment where there is some a priori knowledge regarding the target's possible location.



(a) Scenario 1 (S1)



(b) Scenario 2 (S2)



(c) Scenario 3 (S3)

Figure 9. Several test scenarios used to verify algorithm performance.

2. *Performance Metrics*

In order to gauge performance, several metrics are used. Perhaps the most intuitive is to sum the scores of all the cells. This is directly proportional to the mean of the cell scores. The cumulative map score for a run i at time step k is denoted

$$S(i,k) = \sum_{\overline{z} \in \tilde{B}} x_w(k, \overline{z}) \tag{27}$$

Along a similar vein, the variance of the map scores is defined as

$$V(i,k) = \text{Var}(x_w(k,\overline{z})) \text{ for } \overline{z} \in \tilde{B} \tag{28}$$

The average values across n runs are simply

$$S_{ave}(k) = \frac{1}{n} \sum_{i \in I_n} S(i,k) \tag{29}$$

$$V_{ave}(k) = \frac{1}{n} \sum_{i \in I_n} V(i,k) \tag{30}$$

In terms of map coverage, the best and worse case scenarios can be given by

$$S_{max}(k) = \max_{i \in I_n} S(i,k) \tag{31}$$

$$S_{min}(k) = \min_{i \in I_n} S(i,k) \tag{32}$$

In addition to coverage metrics, performance metrics which measure expected time to target detection can be defined. The number of agents which find the target in run i is given by $\tilde{N}(i)$. The average number of agents which find the target in n runs can be defined as

$$\tilde{N}_{ave} = \frac{1}{n} \sum_{i \in I_n} \tilde{N}(i) \tag{33}$$

The time when the m^{th} agent finds the target is given by $T_m(i)$. Note that by definition, $T_1(i) \leq T_2(i) \leq ... \leq T_M(i)$. The average time to target detection for the encounters is slightly more complicated. This is because in some runs, $T_m(i)$ is undefined (in the case that the m^{th} agent does not find the target). Therefore, the values of $T_m(i)$ which contribute to the average are only those when it is defined

$$T_{m,ave} = \frac{1}{\hat{n}(m)} \sum_{i \in I_{\hat{n}(m)}} T_m(i) \tag{34}$$

Where $\hat{n}(m)$ is the number of runs where at least m agents find the target and $I_{\hat{n}(m)}$ corresponds to the indices where the times occur. The ramifications of this definition are explained in the context of the simulation in Section V.C.2.

B. Alternative Search Strategies for Comparison

The searching algorithm can be compared with other common search strategies to evaluate its performance. One of the simplest search strategies is the simple raster scan. This involves the agents moving in a north/south or east/west lines until the boundary of the search is reached. The agent then moves over by one row and then turns around.

Another heuristic search strategy that is related to the raster scan is the "lawn mower" algorithm. It is referred to as a lawn mower algorithm because the agents follow a set of heuristics which might be similar to a person mowing a lawn and involves the agent moving in a straight line until it encounters an obstacle and turning in a direction clear of obstacles.

A third simple search strategy is the gradient climb algorithm. In this case, the algorithm evaluates the scores of the cells surrounding it (to the north, east, south and west) and then chooses the cell which has the highest score. If two or more cells have the same maximum score, the algorithm chooses the cell which requires the smallest course change to visit. This is similar to a steepest ascent algorithm[33,34] where the feasible directions are only the four cardinal directions.

Finally, the Voronoi partitioning method has been used by several groups to generate coverage algorithms which can be applied in this situation[17,16] . In this case, the Voronoi diagram $\overline{V}$ is generated using the agent's positions as generators. Then the point for the agent to search within the next d steps is chosen to be within the agent's Voronoi polygon and its reachable set. This algorithm is outlined in Figure 10.

For purposes of comparison, the full algorithm refers to the algorithm described in Section III.

```
1  Generate Voronoi diagram of system using agent locations as generators.
2  if $V(\overline{z}_{agt_i}) \bigcap B_{R_i} \neq \emptyset$
3      Choose $\overline{z}_i^\star \in V(\overline{z}_{agt_i}) \bigcap B_{R_i}$
4  else
5      Choose $\overline{z}_i^\star = \overline{z}_{agt_i}$
6  end
7  Generate linear path from $\overline{z}_{agt_i}$ to $\overline{z}_i^\star$
```

Figure 10. Pseudo code for randomized Voronoi partitioning algorithm.

C. Comparison Results

The various comparison algorithms are used in the three different scenarios. The performance metrics defined previously in Section V.A.2 can then be used to judge each algorithm's efficiency.

1. Map Coverage

When evaluating the map coverage by the different strategies, the relevant quantities to analyze are the cumulative map scores and other related metrics. These scenarios involve multiple agents in the team searching the map with no targets.

In order to judge the general behavior of the algorithms, a series of Monte Carlo simulations are used. In these simulations, the performance of each algorithm is gauged over a series of 20 runs and then averaged using Eq. 29. The best and worst case scenarios for the series of runs (in terms of map coverage) are also computed using Eq. 32 and Eq. 31, respectively. The results for scenario 3 are presented in Figure 11 (the traces for the other scenarios display similar trends).

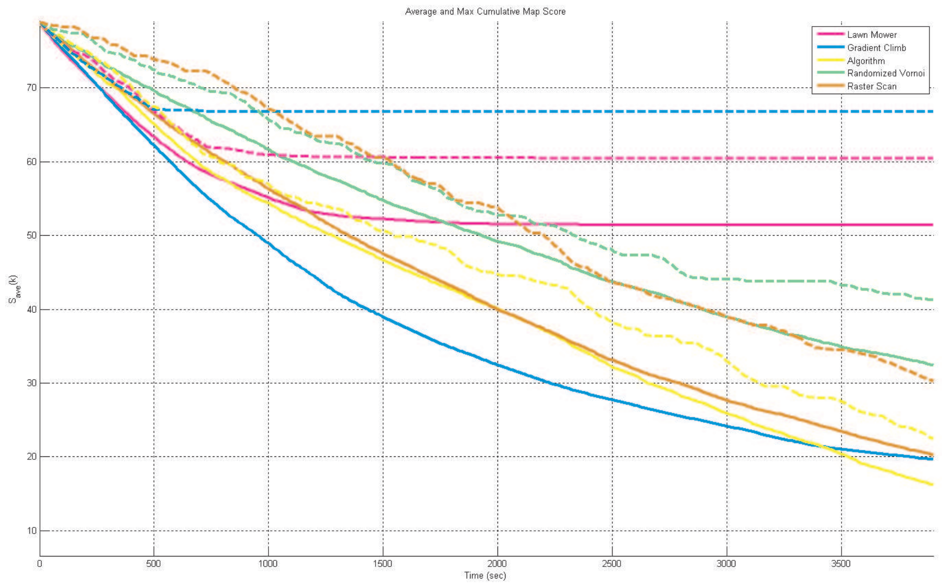


Figure 11. $S_{ave}(k)$ **and** $S_{max}(k)$ **for scenario 3.**

In Figure 11, the solid line represents $S_{ave}(k)$ and the dashed line represents $S_{max}(k)$ for the corresponding search strategy. The performance of the various search strategies using $S_{ave}(k)$ and $S_{max}(k)$ as metrics is summarized in Table 2. In this table, 1 corresponds to the best performance and 6 corresponds to the worst performance.

Table 2. Rankings of search strategies using $S_{ave}(k)$ **and** $S_{max}(k)$ **as metrics (1 = best).**

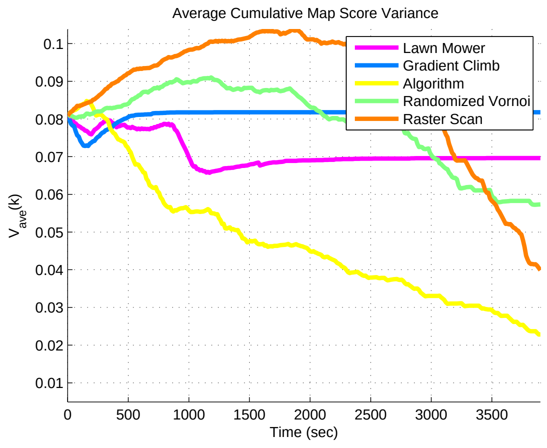
	$S_{ave}(k)$			$S_{max}(k)$		
Strategy	**S1**	**S2**	**S3**	**S1**	**S2**	**S3**
Lawn Mower	5	5	5	5	5	4
Randomized Voronoi	4	4	4	3	3	3
Raster Scan	3	3	3	4	4	2
Gradient Climb	2	1	2	2	1	5
Full Algorithm	1	2	1	1	2	1

Looking at average performance measured by $S_{ave}(k)$, it can be seen that in scenarios 1 and 3, the performance of the algorithms from worst to best appears to be, lawn mower, randomized Voronoi, raster scan, gradient climb, and finally the full algorithm. This is the expected result and shows that the best performance and guarantee of map coverage is achieved with the full algorithm. It should be noted that if the simulation were run for a longer amount of time, it is expected that the raster scan algorithm will eventually outperform the gradient climb when using $S_{ave}(k)$ as the metric for map coverage. Map coverage is guaranteed with the raster scan algorithm, but it is obvious that the performance is suboptimal. Note that in scenario 2, it appears that the gradient climb algorithm performs the best. As mentioned previously, this occurs because the areas to be searched are connected and the environment is fairly simple. If the environment was comprised of long, narrow corridors, the gradient climb algorithm would perform poorly due to the fact that it would not cross over areas of low score whereas the full algorithm would.

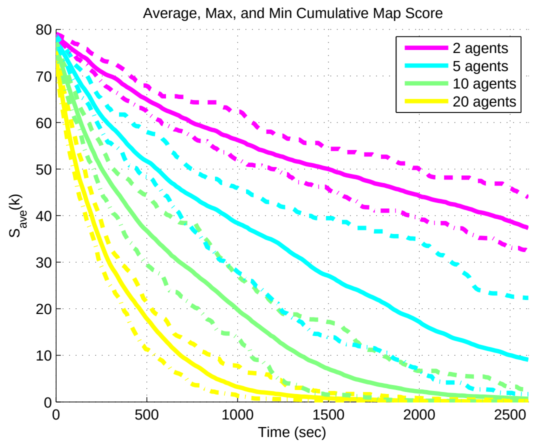
The guarantees of map coverage are more evident when looking at the worst case scenario for map coverage. Recall that $S_{max}(k)$ is a measure of the worst case scenario possible over all test cases. In this case, it is obvious that the full algorithm with explicit cooperation is the best policy to use. Although the gradient climb strategy may work well for some situations, there are situations where it performs the worst out of all the possible strategies ($S_{max}(k)$ for scenario 3).

The variance of the scores, $V_{ave}(k)$ for scenario 3 is shown in Figure 12(a). The results of this figure reinforce the notion that the full algorithm performs the best in terms of driving the variance to zero as well.

Improving performance in a searching mission typically involves increasing the number of agents involved in the search. The main challenge with current unmanned systems is that raising the number of agents greatly increases operator workload required to manage the team. If the team is comprised of heterogeneous agents with different capabilities, the mission management task becomes even more complicated. The benefits of the full algorithm within this framework can be seen when looking at the coverage vs. time for varying number of agents. For example, the effects of varying the number of homogeneous agents operating under the full algorithm strategy are shown shown in Figure 12(b).



(a) $V_{ave}(k)$ for scenario 3



(b) $S_{ave}(k)$ for varying number of agents

Figure 12. Coverage metrics for varying number of agent using scenario 3.

Figure 12(b) displays several interesting phenomena. First, the guarantee of exhaustive map searching is reinforced. Also, the effect of increasing the number of agents involved in the search is evident since in both cases, the time to drive the map scores to zero decreases as the number of agents increase. The effect of increasing the number of agents in the team can be investigated by calculating the settling time for the coverage metric. In this context, the settling time is defined as the time it takes for $S_{ave}(k)$ to drop below to 15% of the initial cumulative map score. As expected, the settling time decreases as the number of agents increases. Notice that the settling time does not decrease in a linear fashion. Instead, the relative increase in performance decreases with each successive addition of a team member. In other words, adding more agents to the team does not greatly increase performance after a certain point.

2. Time to Target Detection

There are several parameters that measure the efficiency of the algorithm in terms of target detection time. One metric is the average number of agents that find the target for a given scenario. Related to this metric is the number of scenarios where at least 1, 2, or 3 agents find the target. All runs for the time to target detection Monte Carlo simulation use 200 time steps with 3 agents.

The values of $\tilde{N}_{ave}$, $\hat{n}(1)$, $\hat{n}(3)$, and $\hat{n}(3)$ for 30 runs are summarized in Table 3.

The easiest metric to interpret is the average number of agents that find the target. This is shown in the first three columns of Table 3. As can be seen, the full algorithm performs the best in all scenarios in terms of getting the most agents to find the target.

It is also worth noting that the raster scan method does not appear to be affected by the different scenarios. This makes sense considering that information about the environment is not taken into account when using this method. It should be noted that in scenario 3, many of the algorithms which use information about the environment (gradient climb and full algorithm) show a decrease in performance. This is because the initial world map (shown previously in Figure 9(c)) has several regions of high score. These would correspond to a priori knowledge of possible target locations. Despite this being the initial state of the world, the locations of the targets are placed according to a uniform distribution across the map. In other words, the initial target location is not distributed according to the distribution shown in Figure 9(c). This simulates a situation where the agents are given inaccurate a priori knowledge before starting the mission, thus leading to a decrease in performance. To simulate a situation with accurate information, the initial target location would need to be distributed according to the distribution shown in Figure 9(c).[35]

The columns for $\hat{n}(m)$ show the number of runs where m agents find the target. Once again, it is apparent that methods which use information about the environment perform the best.

Table 3. Average number of agents which find target and number of scenarios where at least 1, 2, or 3 agents finds the target (30 scenarios each).

	Average # Agents Which Find Target $\tilde{N}_{ave}$			# Runs w/ 1 Encounter $\hat{n}(1)$			# Runs w/ 2 Encounters $\hat{n}(2)$			# Runs w/ 3 Encounters $\hat{n}(3)$		
Strategy	**S1**	**S2**	**S3**	**S1**	**S2**	**S3**	**S1**	**S2**	**S3**	**S1**	**S2**	**S3**
Lawn Mower	0.267	0.367	0.233	5	8	5	3	2	2	0	1	0
Randomized Voronoi	0.467	0.433	0.633	14	13	19	0	0	0	0	0	0
Raster Scan	1.000	1.033	1.200	20	23	23	9	8	11	1	0	2
Gradient Climb	1.400	2.067	1.500	23	27	19	13	24	16	6	11	10
Full Algorithm	2.567	2.400	1.500	28	24	16	26	24	15	23	24	14

Information regarding the average amount of time required to find the target are displayed in Table 4.

Table 4. Average time to target detection by first, second, and third agent (30 scenarios each).

	Average Time for 1st Encounter $T_{1,ave}$			Average Time for 2nd Encounter $T_{2,ave}$			Average Time for 3rd Encounter $T_{3,ave}$		
Strategy	**S1**	**S2**	**S3**	**S1**	**S2**	**S3**	**S1**	**S2**	**S3**
Lawn Mower	143	322	143	500	774	501	N/A	754	N/A
Randomized Voronoi	1114	845	1115	N/A	N/A	N/A	N/A	N/A	N/A
Raster Scan	1045	807	1045	1797	1412	1798	1989	N/A	1989
Gradient Climb	1115	814	1116	1222	1316	1223	1481	1715	1481
Full Algorithm	844	915	844	904	1143	904	1238	1264	1238

The results in Table 4 are not as good of a representation of performance as those shown in Table 3. The reason can be seen from the definition of $T_{m,ave}$ in Eq. 34. These averages are computed only for the situations where the agents find the target. For example, with the lawn mower strategy with scenario 1, from Table 3, it can be seen only 5, 3, and 0 runs out of 30 show at least 1, 2, or 3 agents finding the target, respectively. Therefore, the averages $T_{1,ave}$, $T_{2,ave}$, and $T_{3,ave}$ are computed using only 5, 3, and 0 samples (explaining why $T_{3,ave}$ = N/A). Using the lawn mower strategy, the agents tend to become stuck in limit cycles and therefore, if the agent is going to find the target, it is will happen very quickly or not at all. This shows why the values of $T_{1,ave}$ and $T_{2,ave}$ are low. This can be compared with the full algorithm where it is virtually guaranteed that the agents will find the target but it will take a longer amount of time to do so.

Furthermore, notice that it is not required that $T_{1,ave} \leq T_{2,ave} \leq ... \leq T_{M,ave}$. For example, for the full algorithm with explicit cooperation using scenario 2, $T_{3,ave} < T_{2,ave}$. Once again, this is because the average is computed over a different number of samples.

VI. Conclusion and Further Research

Improving performance in a searching mission typically involves increasing the number of agents involved in the search. The main challenge with current manned systems is that raising the number of agents greatly increases operator workload required to manage the team. If the team is comprised of heterogeneous agents with different capabilities, the mission management task becomes even more complicated. This paper presents a modular and scalable searching algorithm that can be used to coordinate a large numbers of heterogeneous agents involved in a searching mission. The centralized occupancy based map represents the system's belief of the state of the world at a given time. The map can be propagated forward in time to provide the estimate of the future state of the world at step $k + d$. Each agent then decides which coordinate is the most desirable to search in the next d steps. The team can be comprised of different types of agents with different capabilities. The formulation allows each agent to determine what is desirable for its individual capabilities. Each agent then computes control decisions based on the predicted future state of the world. Although these actions may not be optimal in a single step, they will benefit the agent in the future.

Although there is no explicit cooperation between agents in the team, the agents are implicitly coupled through the centralized occupancy map. The algorithm remains scalable because each agent does not need to explicitly know about the existence of other agents. Each agent executes the searching algorithm and the emergent behavior is that the team performs a coordinated search.

The modularity of the algorithm allows the user to tailoring specific parts of the algorithm to address individual agent capabilities. This paper focused mostly on the solution for $(\wp_2)$ and showed that under the proposed solution, the agents are shown to perform an exhaustive search of the map regardless of the methods used to provide solutions to $(\wp_1)$ and $(\wp_3)$.

Future work in this area is directed towards increasing searching performance for all agents. As mentioned previously, it has been observed that using parameters which violate Assumption A.7 tends to yield paths which cover more ground quicker and in a seemingly more efficient manner. Although an exhaustive search of the map is guaranteed under the current conditions, Assumption A.7 hinders increased performance of the system. Work is directed towards tailoring the reward function and finding additional constraints which may be more relaxed than Assumption A.7 and still guarantees coverage. The issue of system performance is tied tightly to the way that $(\wp_3)$ is solved. A closer look at the relationships between $(\wp_2)$ and $(\wp_3)$ is the focus of current research activity. In addition, the target may be moving or evading the searching agents. Therefore the assumption of a static environment is not valid. Investigations into incorporating a dynamic world model and changing scores is also being conducted.

VII. Acknowledgements

This work is sponsored in part by the Washington Technology Center (grants F04-MC2 and F04-MC3), the Osberg Family Trust Fellowship, the Washington NASA Space Grant Consortium, and the Air Force Office of Scientific Research (grant FA-9550-07-1-0528). Dr. Rolf Rysdyk at the Insitu Group also contributed to this work.

References

[1]Monekosso, N. and Remagnino, P., "Robot Exploration Using the Expectation-Maximization Algorithm," *Proceedings of 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Kobe, Japan, 2003.

[2]Dollarhide, R. L., Agah, A., and Minden, G. J., "Evolving Controllers for Autonomous Robot Search Teams," *Artificial Life and Robotics Journal*, Vol. 5, 2002, pp. 178–188.

[3]Kurabayashi, D., Tsuchiya, H., Fujiwara, I., Asama, H., and Kawabata, K., "Motion Algorithm for Autonomous Rescue Agents Based on Information Assistance System," *Proceedings of the 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Kobe, Japan, 2003.

[4]Kantor, G., Singh, S., Peterson, R., Rus, D., Das, A., Kumar, V., Pereiera, G., and Spletzer, J., "Distributed Search and Rescue with Robot Sensor Teams," *Proceedings of the 4th International Conference on Field and Service Robotics*, 2003.

[5]Jennings, J. S., Whelan, G., and Evans, W. F., "Cooperative Search and Rescue with a Team of Mobile Robots," *Proceedings of the International Conference on Advanced Robotics*, 1997.

[6]Benkoski, S. J., Monticino, M. G., and Weisinger, J. R., "A Survey of the Search Theory Literature," *Naval Research*

Logistics, Vol. 38, 1991, pp. 469–494.

[7]Bourgault, F., Furukawa, T., and Durrant-Whyte, H., "Coordinated Decentralized Search for a Lost Target in a Bayesian World," *Proceedings of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, Australian Centre for Field Robotics, Las Vegas, NV, October 2003.

[8]Bourgault, F. and Durrant-Whyte, H. F., "Communication in General Decentralized Filters and the Coordinated Search Strategy," *Proceedings of the 7th International Conference on Information Fusion*, Australian Centre for Field Robotics, Stockholm, Sweden, 2004.

[9]Wong, E.-M. and Bourgault, Frederic Furukawa, T., "Multi-vehicle Bayesian Search for Multiple Lost Targets," *Proceedings of the 2005 IEEE Internationalo Conference on Robotics and Automation*, Barcelona, Spain, April 2005.

[10]Polycarpou, M. M., Yang, Y., and Passino, K. M., "A Cooperative Search Framework for Distributed Agents," *Proceedings of the 2001 IEEE International Symposium on Intelligent Control*, Mexico City, Mexico, 2001.

[11]Flint, M., Polycarpou, M., and Fernandez-Gaucherand, E., "Cooperative Control for Multiple Autonomous UAV's Searching for Targets," *Proceedings of the 41st IEEE Conference on Decision and Control*, University of Cincinnati, Las Vegas, NV, 2004.

[12]Jin, Y., Liao, Y., Minai, A. A., and Polycarpou, M. M., "Balancing Search and Target Response in Cooperative Unmanned Aerial Vehicle (UAV) Teams," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 36, 2006, pp. 571–587.

[13]Erignac, C. A., "An Exhaustive Swarming Search Strategy Based on Distributed Pheromone Maps," Tech. rep., Boeing, Seattle, WA, 2004.

[14]Du, Q., Faber, V., and Gunzburger, M., "Centroidal Voronoi Tessellations: Applications and Algorithms," *Society for Industrial and Applied Mathematics Review*, Vol. 41, 1999, pp. 637–676.

[15]Cortes, J., Martinez, S., Karatas, T., and Bullo, F., "Coverage Control for Mobile Sensing Networks," *IEEE Transactions on Robotics and Automation*, Vol. 20, 2004, pp. 243–255.

[16]Laventall, K. and Cortes, J., "Coverage Control By Robotic Networks with Limited-Range Anisotropic Sensory," *Proceedings of the 2008 American Control Conference*, Seattle, Washington, 2008.

[17]Frazzoli, E. and Bullo, F., "Decentralized Algorithms for Vehicle Routing in a Stochastic Time-Varying Environment," *Proceedings of the IEEE Conference on Decision and Control*, December 2004.

[18]Arsie, A. and Frazzoli, E., "Efficient Routing of Multiple Vehicles with No Communication," *Proceedings of the 2007 American Control Conference*, New York City, New York, 2007.

[19]Fox, D., Ko, J., Konolige, K., Limketkai, B., Schulz, D., and Steward, B., "Distributed Multi-Robot Exploration and Mapping," *Proceedings of the 2nd Canadian Conference on Computer and Robot Vision*, 2005.

[20]Baronov, D. and Baillieul, J., "Search Decisions for Teams of Automata," *Proceedings of the 47th Conference on Decision and Control*, Cancun, Mexico, 2008.

[21]Hoffman, G. M., Waslander, S. L., and Tomlin, C. J., "Distributed Cooperative Search Using Information-Theoretic Costs for Particle Filters, with Quadrotor Applications," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, Keystone, CO, 2006.

[22]Rubio, J. C., Vagners, J., and Rysdyk, R. T., "Adaptive Path Planning for Autonomous UAV Oceanic Search Missions," *Proceedings of the 1st AIAA Intelligent Systems Technical Conference*, 2004.

[23]Lum, C. W., Rysdyk, R. T., and Pongpunwattana, A., "Occupancy Based Map Searching Using Heterogeneous Teams of Autonomous Vehicles," *Proceedings of the 2006 Guidance, Navigation, and Control Conference*, Keystone, CO, August 2006.

[24]Lum, C. W. and Rysdyk, R. T., "Time Constrained Randomized Path Planning Using Spatial Networks," *Proceedings of the 2008 American Control Conference*, Seattle, WA, June 2008.

[25]Elfes, A., "Using Occupancy Grids for Mobile Robot Perception and Navigation," *IEEE Computer*, 1989, pp. 46–57.

[26]Elfes, A., *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*, Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, May 1989.

[27]Latombe, J.-C., *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

[28]Lum, C. W., *Coordinated Searching and Target Identification Using Teams of Autonomous Agents*, Ph.D. thesis, University of Washington, Seattle, WA, March 2009.

[29]Lum, C. W., Rysdyk, R. T., and Pongpunwattana, A., "Autonomous Airborne Geomagnetic Surveying and Target Identification," *Proceedings of the 2005 Infotech@Aerospace Conference*, AIAA, Arlington, VA, September 2005.

[30]Capozzi, B. J. and Vagners, J., "Navigating Annoying Environments Through Evolution," *Proceedings of the 40th IEEE Conference on Decision and Control*, University of Washington, Orlando, FL, 2001.

[31]LaValle, S. M. and Kuffner Jr, J. J., "Randomized Kinodynamic Planning," *International Journal of Robotics Research*, May 2001, pp. 378–400.

[32]Sun, Z. and Reif, J. H., "On Finding Approximate Optimal Paths in Weighted Regions," *Journal of Algorithms*, January 2006, pp. 1–32.

[33]Boyd, S. and Vandenberghe, L., *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.

[34]Rockafellar, R., "Fundamentals of Optimization," Tech. rep., University of Washington, Seattle, WA, 2006.

[35]Thrun, S., Burgard, W., and Fox, D., *Probabilistic Robotics*, MIT Press, 2005.