2010 American Control Conference
Marriott Waterfront, Baltimore, MD, USA
June 30-July 02, 2010

Partitioned Searching and Deconfliction: Analysis and Flight Tests

Christopher Lum, Juris Vagners, Jung Soon Jang, and John Vian

Abstract— **Searching a complex environment for a hidden target is a common problem encountered by many autonomous systems. Many modern autonomous systems use a hierarchical structure for mission management where different algorithms perform different tasks to give agents desired behavior. This work investigates a search policy that guarantees both an exhaustive search of the map and conflict free paths of all agents. Agents formulate control decisions for a fixed number of time steps using a modular algorithm that allows parameterizations of agent capabilities. High fidelity simulation and flight test data using multiple autonomous vehicles are used to verify and validate the algorithms in real time.**

I. Introduction

This paper investigates an algorithm which can efficiently coordinate a team of possibly heterogeneous agents and ensure that the team performs an exhaustive search of the map and simultaneously guarantees agent flight paths are collision free.

Searching a space for a target has been a well studied field. 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. [1, 2] 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. [3] 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 [4] developed exhaustive searching strategies that also provide guarantees about map coverage using 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 Cortes [5] and Frazzoli [6, 7]. Searching and map building has been studied extensively by the robotics community as well. Groups such as Fox et al. [8] have looked at generating searching algorithms with the ideas of exploration and map building in mind. Previous work at the University of Washington [9, 10] explored the use of various techniques to address the search problem with a group of heterogeneous agents.

C. Lum and J. Vagners are with the Department of Aeronautics and Astronautics, University of Washington, Seattle, WA, USA [lum, vagners]@u.washington.edu

J. Jang and J. Vian are with the Boeing Company, Seattle, WA, USA [jungsoon.jang, john.vian]@boeing.com

These methods, while effective for their respective applications, contain shortcomings when addressing the search problem considered in this application. To address this, a search policy is presented in this paper. Each agent determines a desirable coordinate to visit in the future by solving 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. Once this desirable coordinate is determined, a straight path that transitions the agent from its current location to the desirable coordinate can be applied. An additional feature of this algorithm is the guarantee of collision free flight paths while engaged in the search.

These algorithms are then tested in a high fidelity simulation environment developed by Boeing Research and Technology. Once proved in simulation, the algorithms are seamlessly transitioned to the Vehicle Swarm Technology Laboratory (VSTL), an indoor test facility equipped to test multiple agent scenarios.

Section II describes the notion of occupancy based maps and their features. These maps provide the framework for the search strategy which is described in Section III. Section IV presents flight test results and performance metrics. Finally, Section V presents conclusions and some 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 formalized by Elfs [11] but functionality such as Bayesian score updates and time varying models are added to the maps to accommodate the algorithm.

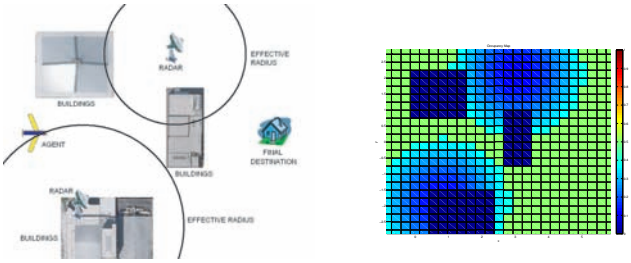
A. Occupancy Based Maps Definition

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 limits the score of any given cell in the map to the range of $[0, 1]$ and is similar to a two dimensional, uncoupled, discretized probability density function [12]. The spatial domain of the occupancy based map, B, consists of a box where x is between x_{min} and x_{max} and similarly for the y dimension.

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 $\overline{z} \in B \subset \Re^2$ at a certain time step $k \in \Re$. In other words, $x_w : B \times \Re \rightarrow \Re$. 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. An example of embedded obstacles and reward areas in an occupancy based map is shown in Figure 1.



(a) Physical environment with soft and hard obstacles

(b) Occupancy based map representation of environment

Fig. 1. 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.

The world state can be propagated forward in time to create an estimate of the world d steps in the future [10]. This estimate is referred to as $\hat{x}_w(k+d, \overline{z})$ and for most applications in this paper, the future estimate is assumed to be identical to the current world state.

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

In a similar fashion, the set of cell centers which have the highest score in the map is denoted as $\tilde{B}_{max}$.

The set of all locations that agent i can reach in d steps is referred to as the agent's reachable set, $B_{R_i} \subseteq B$. This application defines B_{R_i} as

$$B_{R_i} = \{\overline{z} \in B \mid ||\overline{z} - \overline{z}_{agt_i}|| \leq R_{i_{max}}\} \tag{1}$$

Eq. 1 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_i} may not be a perfect circle as described in Eq. 1. In this case, it simply becomes more difficult to define and compute B_{R_i} but the following analysis is not affected by the geometry of B_{R_i}.

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

$$\tilde{B}_{R_i} = B_{R_i} \bigcap \tilde{B} \tag{2}$$

B. Updating Map Cell 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 $X_k = x_B$ denotes 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. Although this scenario is not presented in this work, reliability factors can change during the mission as long as they remain in the range $[0, 1]$. For example, lowering the reliability factor during the mission can be used to model failing or destroyed sensors on an agent. The probabilistic sensor model can be formed as $p(Z_t = z_A | X_t = x_B) = 1 - \frac{1}{2}(h+1)$ and $p(Z_t = z_B | X_t = x_B) = \frac{1}{2}(h+1)$.

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 which yields the following Bayesian update rule.

$$s_k = \begin{cases} \frac{s_{k-1}(1-h)}{1+(1-2s_{k-1})h} & \text{if } Z_k = z_A \\ \frac{s_{k-1}(1+h)}{1+(2s_{k-1}-1)h} & \text{if } Z_k = z_B \end{cases} \tag{3}$$

In Eq. 3, 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} [9].

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

III. Search Strategy

On a simplistic level, the search problem involves choosing a desirable location, $\overline{z}_i^\star$, for the agent i to search within the next d steps. Previous work investigated formulating a search policy where agents had no explicit knowledge of each other [9]. Without explicit cooperation, there are scenarios where two or more agents might choose the same location to search. This is undesirable for several reasons. In terms of safety, the agents must now have some type of reactive collision avoidance [13] since collision free trajectories are not guaranteed at the planning level. Furthermore, the performance of the system may suffer if there are multiple agents searching the same cells. One method to alleviate these problems is to partition the search space. In essence, this is a divide and conquer approach. The search space can be partitioned using a Voronoi diagram, which is a tessellation of a Euclidean space and is described in a purely mathematical sense by Okabe [14]. This paper investigates applying them to previously developed search algorithms. In this sense, the location of the agents are considered to be generators for the Voronoi polygons $P = \{p_1, p_2, ..., p_n\} = \{\overline{z}_{agt_1}, \overline{z}_{agt_2}, ..., \overline{z}_{agt_n}, \}$.

The Voronoi diagram embeds information about an agent's position relative to the other agents. Each agent now has an influence on the diagram and each agent will have an influence on the algorithm. The degradation of algorithm scalability is immediately visible when employing this type of cooperation as the Voronoi diagram requires at least $O(n\ log\ n)$ at best [14] where n is the number of generators (or agents). Previously, the single agent search strategy scaled linearly with n.

Formulating the search policy requires the definition of several intermediate values.

A. Intermediate Variables

The point in $\tilde{B}_{max}$ that is closest to agent i is given as

$$\overline{z}_{H_i} \in \underset{\overline{z} \in \tilde{B}_{max}}{\text{arg minimize}} ||\overline{z} - \overline{z}_{agt_i}|| \tag{4}$$

With $\overline{z}_{H_i}$ defined, the distance between agent i and $\overline{z}_{H_i}$ is given by

$$d_i = ||\overline{z}_{H_i} - \overline{z}_{agt_i}|| \tag{5}$$

An example of these distances and locations is shown in Figure 2. In this figure, the black dots represent $\tilde{B}$ (the cell centers of the occupancy map). The black dots circled in purple represent $\overline{z} \in \tilde{B}_{max}$. These are the cell centers that correspond to cells which have the highest score in the map. As defined previously, locations inside the dashed red circles represent B_{R_i} (agent i's reachable set). The black dots within the dashed red circles represent $\tilde{B}_{R_i}$ (the discrete approximation of agent i's reachable set).

The distance d_i is computed by checking the distance between agent i and each $\overline{z} \in \tilde{B}$ with the minimum value assigned as d_i. The corresponding location $\overline{z} \in \tilde{B}$ which yields d_i for agent i is denoted as $\overline{z}_{H_i}$.

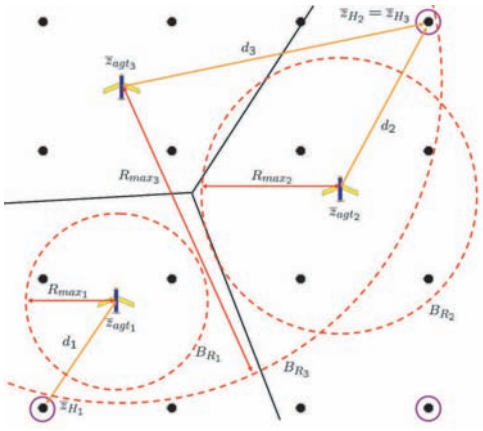


Fig. 2. Example with 3 agents showing d_i and $\overline{z}_{H_i}$.

Note that $\overline{z}_{H_i}$ may not be in the agent's reachable set (either the compact set B_{R_i} or its discrete approximation $\tilde{B}_{R_i}$). In other words, it is possible that $\overline{z}_{H_i} \notin B_{R_i}$ and $\overline{z}_{H_i} \notin \tilde{B}_{R_i}$. This is the case with both agent 1 and agent 2 but not agent 3 in Figure 2.

The Voronoi edges are shown as the solid black lines. The Voronoi polygon associated with each agent is enclosed by the solid black lines.

$$V(\overline{z}_{agt_i}) = \left\{\overline{z} \mid ||\overline{z} - \overline{z}_{agt_i}|| \leq ||\overline{z} - \overline{z}_{agt_j}||\right\} \text{ for } i \neq j \tag{6}$$

Note that it is possible that $\overline{z}_{H_i} \notin V(\overline{z}_{agt_i})$. In Figure 2, this is the case for agent 3 where the point $\overline{z}_{H_3}$ is not in agent 3's Voronoi polygon. It is also possible that $\overline{z}_{H_i} = \overline{z}_{H_j}$ for some $i \neq j$ as shown with agent 2 and agent 3 in Figure 2.

Most of the following analysis requires identifying the agent which is closest to the set $\tilde{B}_{max}$. The index of the agent closest to the set $\tilde{B}_{max}$ is denoted as I. The index I is given by (recall that $I_n = \{1, 2, ..., n\}$)

$$I \in \underset{i \in I_n}{\text{arg minimize}}\ d_i \tag{7}$$

Therefore, $\overline{z}_{H_I}$ denotes the location of the point in $\tilde{B}_{max}$ that is closest to any agent.

The point $\overline{z}_{H_i}$ is used to define the point $\overline{z}'_{h_i}$ as

$$\overline{z}'_{h_i} \in \begin{cases} \underset{\overline{z} \in \tilde{B}_{R_i} \bigcap V(\overline{z}_{agt_i})}{\text{arg minimize}} ||\overline{z} - \overline{z}_{H_i}|| & \text{if } A \neq \emptyset \\ \overline{z}_{agt_i} & \text{otherwise} \end{cases} \tag{8}$$

Where $A = \tilde{B}_{R_i} \bigcap V(\overline{z}_{agt_i})$.

With $\overline{z}'_{h_i}$ defined, it is convenient to define a flag ζ_i as

$$\zeta_i = \begin{cases} 1 & \text{if } ||\overline{z}'_{h_i} - \overline{z}_{H_i}|| \geq ||\overline{z}_{agt_i} - \overline{z}_{H_i}|| \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Physically, $\zeta_i = 1$ if the point $\overline{z}'_{h_i}$ is further away from $\overline{z}_{H_i}$ than the agent's current position of $\overline{z}_{agt_i}$. An example showing when $\zeta_i = 1$ is shown in Figure 3. In this situation, the set $\tilde{B}_{R_i} \bigcap V(\overline{z}_{agt_i})$ is a single point and therefore, Eq. 8

chooses it as $\overline{z}'_{h_i}$ (denoted by the black dot enclosed by the orange circle). In this case, $\overline{z}'_{h_i}$ is obviously farther away from $\overline{z}_{H_i}$ than $\overline{z}_{agt_i}$, so $\zeta_i = 1$.

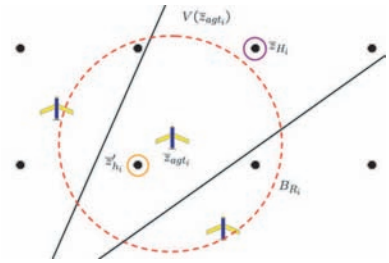


Fig. 3. Situation showing $\zeta_i = 1$.

With the flag ζ_i defined, the point $\overline{z}_{h_i}$ is given by

$$\overline{z}_{h_i} = \begin{cases} \overline{z}'_{h_i} & \text{if } \zeta_i = 0 \\ \overline{z}_{agt_i} & \text{if } \zeta_i = 1 \text{ and } i \neq I \\ \overline{z}_{S_I} & \text{if } \zeta_i = 1 \text{ and } i = I \end{cases} \tag{10}$$

where

$$\overline{z}_{S_I} = \overline{z}_{agt_I} + R_{max_I} \frac{\overline{z}_{H_I} - \overline{z}_{agt_I}}{||\overline{z}_{H_I} - \overline{z}_{agt_I}||} \tag{11}$$

Previous analysis in [10] showed that the point $\overline{z}_{h_i}$ has some interesting properties:

1) $\overline{z}_{h_i} \in B_{R_i} \bigcap V(\overline{z}_{agt_i})$
2) $||\overline{z}_{h_i} - \overline{z}_{H_i}|| \leq ||\overline{z}_{agt_i} - \overline{z}_{H_i}||$ for $i \in I_n \setminus \{I\}$
3) $||\overline{z}_{h_I} - \overline{z}_{H_I}|| < ||\overline{z}_{agt_I} - \overline{z}_{H_I}||$ for $i = I$

These properties can be used to develop the control law with explicit cooperation between agents.

B. Reward Function

The desirability of a given location, $\overline{z}$, to agent i is measured with a reward function of the following form

$$\hat{J}_{0_i}(\overline{z}) = \alpha \hat{x}_w(k+d, \overline{z}) + \eta \cdot (\beta f_\chi(\overline{z}) + \gamma f_d(\overline{z})) \tag{12}$$

In Eq. 12 it is understood that all of the parameters α, β, and γ and all the functions η, $f_\chi()$, and $f_d()$ are specific to agent i. The variable η is defined as

$$\eta = \max_{\overline{z} \in \tilde{B}_R \bigcap V(\overline{z}_{agt_i})} \hat{x}_w(k+d, \overline{z}) \tag{13}$$

In Eq. 12, the function $f_\chi()$ is given by

$$f_\chi(\overline{z}) = \begin{cases} 0 & \text{if } \overline{z} \text{ in same cell as current agent} \\ \Delta & \text{otherwise} \end{cases} \tag{14}$$

where $\Delta = 1 - \frac{q(\chi_{agt}, \pi/2 - \text{atan2}(\overline{z}_2 - y_{agt}, \overline{z}_1 - x_{agt}))}{\pi}$

In Eq. 14, the function $q(a, b)$ computes the absolute angular difference between the two angles, a and b. Note that the range of $f_\chi()$ is $[0, 1]$.

The function $f_d()$ of Eq. 12 is given by

$$f_d(\overline{z}) = \begin{cases} \frac{||\overline{z} - \overline{z}_{agt}||}{R_{max}} & \text{if } \overline{z} \in B_R \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

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. This function is used to encourage the agent to move farther and spend more time at a higher velocity, thereby covering more ground on the map.

C. Control Law with Explicit Cooperation

With the intermediate variables and reward function defined, the control law to incorporate explicit cooperation is now of the form

$$\overline{z}_i^{\star'} \in \begin{cases} \arg \underset{\overline{z} \in \tilde{B}_{R_i} \bigcap V(\overline{z}_{agt_i})}{\text{maximize}} \hat{J}_{0_i}(\overline{z}) & \text{if } \tilde{B}_{R_i} \bigcap V(\overline{z}_{agt_i}) \neq \emptyset \\ \overline{z}_{agt_i} & \text{otherwise} \end{cases} \tag{16}$$

$$\overline{z}_i^\star \in \begin{cases} \overline{z}_{h_i} & \text{if } \alpha \hat{x}_w(k+d, \overline{z}_i^{\star'}) < \delta \\ \overline{z}_i^{\star'} & \text{otherwise} \end{cases} \tag{17}$$

Agent i chooses $\overline{z}_i^\star$ as the next location to search and the process repeats once the agent reaches $\overline{z}_i^\star$.

D. Exhaustive Searching and Strategic Collision Avoidance

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. This paper shows 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

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

All of these assumptions are reasonable and can be implemented easily.

Theorem 3.1: The point $\overline{z}_i^\star$ is always reachable by agent i and remains in the agent's Voronoi polygon, $\overline{z}_i^\star \in B_{R_i} \bigcap V(\overline{z}_{agt_i})$.

Proof: In the case where $\overline{z}_i^\star = \overline{z}_i^{\star'}$, it can be seen from Eq. 16 that $\overline{z}_i^{\star'} \in B_{R_i} \bigcap V(\overline{z}_{agt_i})$ since the feasible set of the maximization problem is precisely $B_{R_i} \bigcap V(\overline{z}_{agt_i})$ and it was previously shown that $\overline{z}_{agt_i} \in B_{R_i} \bigcap V(\overline{z}_{agt_i})$.

In the case where $\overline{z}_i^\star = \overline{z}_{h_i}$, the properties of $\overline{z}_{h_i}$ guarantee that $\overline{z}_{h_i} \in B_{R_i} \bigcap V(\overline{z}_{agt_i})$ (end of Section III-A). ■

If the same assumptions are satisfied, several guarantees about algorithm performance can be made. However, unlike previous results [9] where guarantees are shown for all agents in the team, most of the guarantees for the case of explicit cooperation can be shown only for the case of $i = I$.

Theorem 3.2: Under assumptions A.3, A.4, A.5, and the previously described search strategy with explicit cooperation, if agent I is not currently at the point $\overline{z}_{H_I}$ the agent must move to a point other than $\overline{z}_{agt_I}$ once the quantity $\alpha x_w(k, \overline{z}_{agt_I})$ decreases below the value δ.

Proof: Assuming that $\overline{z}_i^{\star'} = \overline{z}_{agt_I}$, if the quantity $\alpha x_w(k, \overline{z}_{agt_I})$ decreases below the value δ, then Eq. 17 will assign $\overline{z}_i^\star = \overline{z}_{h_i}$. The properties of $\overline{z}_{h_i}$ can be applied to show that $\overline{z}_i^\star$ is strictly closer to $\overline{z}_{H_i}$ than $\overline{z}_{agt_I}$ (end of Section III-A). Since the agent is not at the point $\overline{z}_{H_I}$, the agent must move to a point other than $\overline{z}_{agt_I}$. ■

Theorem 3.3: Under the previously described search strategy, agent I must choose either a cell center which has non-zero score, $\overline{z}_{h_I}$, or both as the point $\overline{z}_I^\star$.

Proof: The point $\overline{z}_i^\star$ is assigned according to Eq. 17. If the point $\overline{z}_i^{\star'}$ has a score of zero, then $\alpha\hat{x}_w(k+d, \overline{z}_i^{\star'}) = 0$ which is less than the value of δ for any $\delta \in (0, 1]$ so Eq. 17 will assign $\overline{z}_i^\star = \overline{z}_{h_i}$.

Alternatively, if $\alpha\hat{x}_w(k+d, \overline{z}_i^{\star'}) \geq \delta$, then Eq. 17 will assign $\overline{z}_i^\star = \overline{z}_i^{\star'}$. Since $\delta, \alpha \in (0, 1]$, $\alpha\hat{x}_w(k+d, \overline{z}_i^{\star'}) \geq \delta \iff \hat{x}_w(k+d, \overline{z}_i^{\star'}) > 0$ which shows that the point associated with $\overline{z}_I^\star$ has a non-zero score. ■

Theorem 3.4: Under assumption A.4, A.5 and the previously described search strategy, if agent I chooses $\overline{z}_{h_I}$ as $\overline{z}_I^\star$ and $\overline{z}_{h_I} \neq \overline{z}_{H_I}$, then at the next time step the point $\overline{z}_{H_I}$ will remain unchanged and some agent $i \in I_n$ will move closer to the point $\overline{z}_{H_I}$.

Proof: The properties of $\overline{z}_{h_i}$ (end of Section III-A) showed that $||\overline{z}_{h_I} - \overline{z}_{H_I}|| < ||\overline{z}_{agt_I} - \overline{z}_{H_I}||$ so if the agent chooses $\overline{z}_I^\star = \overline{z}_{h_I}$, at the next time step, it will be closer to the point $\overline{z}_{H_I}$ than it was before. If $\overline{z}_{h_I} \neq \overline{z}_{H_I}$, then the agent cannot search that cell so the score will not decrease and at the next step, $\overline{z}_{H_I}$ will still be in the set $\tilde{B}_{max}$. Another agent cannot search this same location due to fact that $\overline{z}_i^\star \in V(\overline{z}_{agt_i})$ and therefore, $\overline{z}_{H_I} \in V(\overline{z}_{agt_I})$.

Furthermore, since the distance between the agent and the same point $\overline{z}_{H_I}$ is decreased, this same point will be chosen by Eq. 4 as the point $\overline{z}_{H_i}$. Note that the index is i, not I in the last sentence. This is because it is possible that the index of the agent that is closest to the set $\tilde{B}_{max}$ may change at the next time step. However, it will change only if another agent moves closer to the set $\tilde{B}_{max}$ than the current agent located at $\overline{z}_{agt_I}$. This does not affect the proof in the sense that the point $\overline{z}_{H_I}$ does not change, simply the index of whichever agent happens to be closest to $\overline{z}_{H_I}$ at the next step. ■

Using these results, we can show that the control law with explicit cooperation yields an exhaustive map search.

Theorem 3.5: Under the previously described assumptions and search strategy with explicit cooperation, $x_w(k, \overline{z}) \rightarrow 0 \ \forall \overline{z} \in B$ (the scores of all cells in the map will approach 0).

Proof: Theorem 3.2 guarantees that agent I cannot remain in a single cell indefinitely and Theorem 3.3 ensures that it must choose either a cell of non-zero score, $\overline{z}_{h_I}$, or both as the point $\overline{z}_I^\star$ at any given time step. If agent I chooses a cell of non-zero score, previous work [9] showed that under the negative update rule of Eq. 3, 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 3.3 is that the agent chooses $\overline{z}_{h_I}$ and $x_w(k, \overline{z}_{h_I}) = 0$. By definition of $\overline{z}_{H_I}$, if $x_w(k, \overline{z}_{H_I}) = 0$, then the map has been completely covered since all scores are at most 0. Therefore, assuming that the map has not yet been entirely searched, $x_w(k, \overline{z}_{h_I}) = 0 \Rightarrow \overline{z}_{h_I} \neq \overline{z}_{H_I}$.

Theorem 3.4 ensures that if if $\overline{z}_{h_I} \neq \overline{z}_{H_I}$, choosing $\overline{z}_{h_I}$ as the point $\overline{z}_I^\star$ does not change the value of $\overline{z}_{H_I}$ at the next time step.

At this next time step, the agent once again must choose a cell with non-zero score, $\overline{z}_{h_I}$, or both. If the agent continues to choose $\overline{z}_{h_I}$, eventually $\overline{z}_{H_I} \in \tilde{B}_{R_I}$ and at this point, choosing a cell with non-zero score, $\overline{z}_{h_I}$, 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, $x_w(k, \overline{z}) \rightarrow 0 \ \forall \overline{z} \in B$. ■

IV. Results

Since the algorithm is proven to exhaustively search the map using collision free flight paths, it can be tested in both simulation and flight tests.

A. Boeing Hardware Environment

The algorithms were tested in the Vehicle Swarm Technology Laboratory (VSTL) developed by the Boeing Research and Technology group [15, 16]. This facility provides a large, indoor flight test arena where heterogenous teams may conduct various types of missions. The autonomous algorithms for each vehicle are executed on dedicated computers and the position information of all vehicles are captured with a system of cameras and coordinated pulses of light. The overall laboratory is shown in Figure 4. Data acquisition at 100Hz with sub 40ms latency is possible with this system for a large number of vehicles. The number of controlled vehicles is limited to 14 due to software and hardware architectures.

The flight test vehicles are heavily-modified, commercially available quad-rotor helicopters as shown in Figure 5.

Algorithms are developed in C and C++ using the Microsoft Visual Studio environment. The algorithms can then be tested using a high fidelity numerical simulation environment developed in Matlab and Simulink. Once the algorithms are verified in simulation, they can be seamlessly integrated onto the actual hardware for real time operations. This facility has been used previously to validate other strategic, autonomous algorithms using this work flow [17].

B. Performance Metrics

In order to gauge performance, several metrics are used. Perhaps the most intuitive is to sum the scores of all the

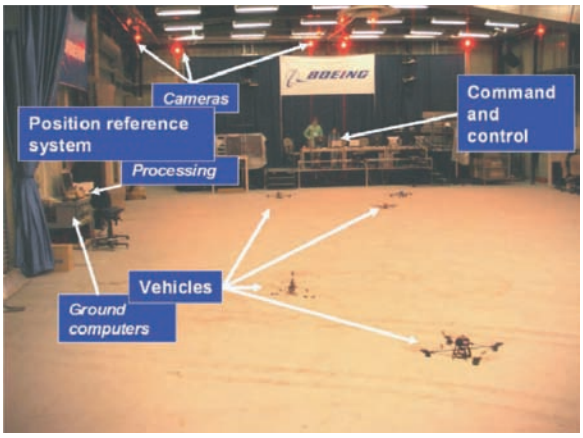


Fig. 4. Vehicle Swarm Technology Laboratory (VSTL) developed by the Boeing Research and Technology group.

Fig. 5. Quadrotor vehicle equipped with reference markers.

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

$$S(i,k) = \sum_{\overline{z} \in \tilde{B}} x_w(k, \overline{z}) \tag{18}$$

The average values across n runs are simply

$$S_{ave}(k) = \frac{1}{n} \sum_{i \in I_n} S(i,k) \tag{19}$$

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

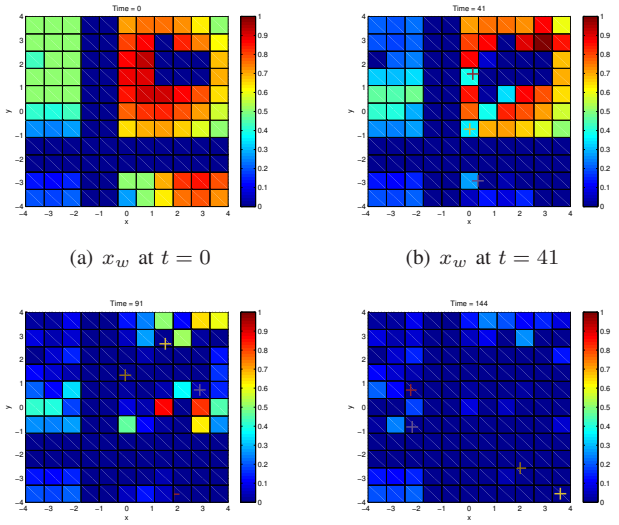
$$S_{max}(k) = \max_{i \in I_n} S(i,k) \tag{20}$$

$$S_{min}(k) = \min_{i \in I_n} S(i,k) \tag{21}$$

Additional performance metrics and measures are described in [10].

C. Simulation Results

The proposed search strategy is tested in the previously described simulation environment provided by Boeing. The progression of the occupancy map and the trajectories of the agents are shown in Figure 6.



(a) x_w at $t = 0$ (b) x_w at $t = 41$

(c) x_w at $t = 91$ (d) x_w at $t = 144$

Fig. 6. Full algorithm with Voronoi partitioning trajectories for 4 agents in Boeing simulation environment.

In this figure, the agents are denoted by colored crosses. Initially only a single agent is involved in the search (Figure 6(a)). Later, more agents join the search using the same policy (Figure 6(b)-6(d)). As expected, the agent's trajectories are collision free since they constantly remain in their own Voronoi polygon. Furthermore, it can be seen that the map is exhaustively searched.

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. 19. The best and worst case scenarios for the series of runs (in terms of map coverage) are also computed using Eq. 21 and Eq. 20, respectively. The algorithms are run on 3 different search scenarios which are denoted S1, S2, and S3. The results for scenario 3 are presented in Figure 7 (the traces for the other scenarios display similar trends).

Although a Monte Carlo simulation with only 20 runs may seem statistically insignificant, these runs are used to simply verify the previous analysis and are not meant to be exhaustive or used as a statistical proof of coverage. Map coverage has been mathematically guaranteed..

In Figure 7, the solid line represents $S_{ave}(k)$ and the dashed line represents $S_{max}(k)$ for the corresponding search strategy.

The lawn mower algorithm is a fairly simplistic, heuristic search which emulates how one might mow a lawn. The gradient climb algorithm has the agent's moving to the surrounding cell with highest score. The strategy labeled as "Algorithm" refers to previous related work without the Voronoi partitioning [9]. The randomized Voronoi strategy is similar to work by Frazzoli [6]. The strategy described in this paper is referred to as the "Algorithm w/ Voronoi". And finally, the raster scan policy is a standard policy where the agents move up and down rows while moving across the

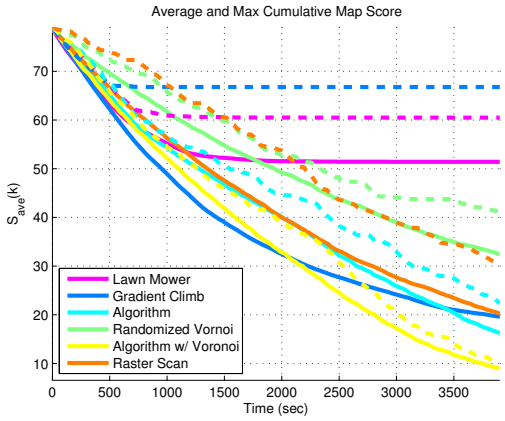


Fig. 7. $S_{ave}(k)$ and $S_{max}(k)$ for scenario 3.

map.

The performance of the various search strategies using $S_{ave}(k)$ and $S_{max}(k)$ as metrics is summarized in Table I. In this table, 1 corresponds to the best performance and 6 corresponds to the worst performance.

TABLE I

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	6	6	6	6	6	5
Randomized Voronoi	5	5	5	4	4	4
Raster Scan	4	4	4	5	5	3
Gradient Climb	3	1	3	3	2	6
Full Algorithm	2	3	2	2	3	2
Full Algorithm w/ Voronoi	1	2	1	1	1	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, full algorithm, then full algorithm with Voronoi partitioning. This is the expected result and shows that the best performance and guarantee of map coverage is achieved with the full algorithm. Furthermore, it shows that the performance is further increased (and the coverage guarantee is preserved) when augmenting the full algorithm with explicit cooperation between agents through the Voronoi partitioning. 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. This occurs because the areas to be searched in scenario 2 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 and full algorithm with Voronoi partitioning 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 algorithm described in this paper (Algorithm w/ Voronoi) 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).

D. Experimental Results

The trajectories for actual hardware flight tests are similar to those shown in Figure 6. The cumulative map score and the map score variance vs. time for a flight test with three agents is shown in Figure 8.

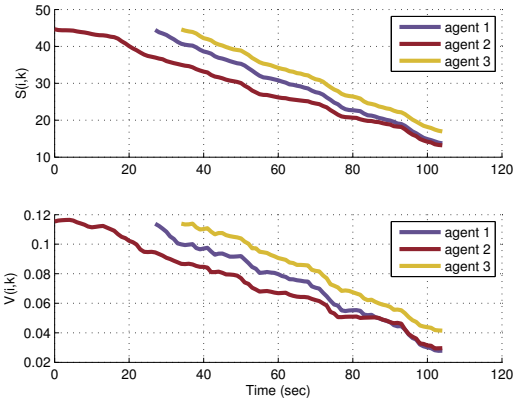


Fig. 8. $S(i,k)$ and $V(i,k)$ vs. time during flight test of two agents with user interaction.

The scenario which generated this figure involved three agents who maintained separate beliefs of the world state (thereby creating three unique $S(i,k)$ and $V(i,k)$ traces). The agents were sequentially added to the search mission. In this case, agent 2 started searching at $t = 0$ seconds. After approximately 30 seconds had elapsed, agent 1 was added to the team. Finally, at $t \approx 35$ seconds, agent 3 was added to the team. This shows the modular nature of the algorithm and how agents can be dynamically added and removed from the team. In all cases, the map scores and variances are driven to zero given sufficient time as the agents exhaustively search the map.

V. CONCLUSIONS

This paper presents a searching algorithm that can be used to coordinate a large number 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. Each agent 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. The Voronoi partitioning of the agents allows the agents to remain collision free since they only choose flight paths which remain within their own Voronoi polygon. This feature also allows the agents to remain spread out and cover the search area more effectively.

In earlier versions of the algorithms, there was no explicit cooperation between agents in the team. Instead, 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 resulting emergent behavior is that the team performs a coordinated search. Explicit cooperation between agents is incorporated using the Voronoi partitioning which simultaneously increases performance and computational costs.

These algorithms are verified using an advanced numerical simulator from the VSTL. They are then transitioned to the hardware test bed and validated using multiple vehicles in real time flight tests. Future research directions involve using the versatile Boeing test bed to investigate human/automaton interactions using the VSTL interface. Currently user interaction with the autonomous systems are somewhat onesided where users dictate the behavior. Current research at the University of Washington are directed towards two way communication and information flow between human operators and autonomous systems with the goal of improving performance and mission success rates.

VI. 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). The authors would also like to thank Dr. Kristi Morgansen from the University of Washington and David Halaas, Paul Pigg, Paul Robinette, Nina Hester, and Dr. Stefan Bieniawski from the Boeing Company and for their assistance.

References

[1] 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.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] Lum, C. W. and Vagners, J., "A Modular Algorithm for Exhaustive Map Searching Using Occupancy Based Maps," *Proceedings of the 2009 Infotech@Aerospace Conference*, Seattle, WA, April 2009.

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

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

[12] 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.

[13] Nair, S. and Marsden, J., "Collision Avoidance and Surveillance Measures for Multivehicle Systems," Tech. rep., California Institute of Technology, Pasadena, CA, 2008.

[14] Okabe, A., Boots, B., and Sugihara, K., *Spatial Tessellations Concepts and Applications of Voronoi Diagrams*, John Wiley and Sons, 1996.

[15] Bieniawski, S., Pigg, P., Vian, J., Bethke, B., and How, J., "Exploring Health-Enabled Mission Concepts in the Vehicle Swarm Technology Lab," *Proceedings of 2009 Infotech@Aerospace Conference*, Seattle, WA, 2009.

[16] Halaas, D., Bieniawski, S., Pigg, P., and Vian, J., "Control and Management of an Indoor, Health Enabled, Heterogeneous Fleet," *Proceedings of the 2009 Infotech@Aerospace Conference*, Seattle, WA, 2009.

[17] Nigam, N., Bieniawski, S., Kroo, I., and Vian, J., "Control of Multiple UAVs for Persistent Surveillance: Algorithm Description and Hardware Demonstration," *Proceedings of the AIAA Infotech@Aerospace Conference*, Seattle, WA, 2009.