AFRL-VA-WP-TP-2003-304

COOPERATIVE CONTROL FOR UAV's SEARCHING RISKY ENVIRONMENTS FOR TARGETS

Matthew Flint
Emmanuel Fernández-Gaucherand
Marios Polycarpou

MARCH 2003

Approved for public release; distribution is unlimited.

AIR VEHICLES DIRECTORATE
AIR FORCE RESEARCH LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7542

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
March 2003	Journal Article Preprint	

4. TITLE AND SUBTITLE
COOPERATIVE CONTROL FOR UAV's SEARCHING RISKY ENVIRONMENTS FOR TARGETS

5a. CONTRACT NUMBER
F33615-01-C-3150

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
69199F

6. AUTHOR(S)
Matthew Flint
Emmanuel Fernández-Gaucherand
Marios Polycarpou

5d. PROJECT NUMBER
ARPF

5e. TASK NUMBER
04

5f. WORK UNIT NUMBER
T3

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Cincinnati
Department of Electrical and Computer Engineering and Computer Science
Cincinnati, OH 45221-0030

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Vehicles Directorate
Air Force Research Laboratory
Air Force Materiel Command
Wright-Patterson Air Force Base, OH 45433-7542

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
AFRL/VACA

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)
AFRL-VA-WP-TP-2003-304

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
To be presented at the Conference on Decision and Control, Maui, HI, 9-12 Dec 03.

14. ABSTRACT ***(Maximum 200 Words)***

A software architecture is presented, which introduces several agents which focus on different aspects of path planning for multiple autonomous unmanned aerial vehicles (UAV's) that are searching an uncertain and threatening environment for targets. One agent models threats in the environment. Another develops a model of the environment that allows targets to be defined by individual probability distribution. Lastly, an agent is presented that utilizes the information from the other agents to generate a near optimal path plan using a Dynamic Programming algorithm.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT	b. ABSTRACT	c. THIS PAGE			Mark Mears
Unclassified	Unclassified	Unclassified	SAR	18	19b. TELEPHONE NUMBER *(Include Area Code)* (937) 255-8685

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18

Cooperative Control for UAV's Searching Risky Environments for Targets *

Matthew Flint † Emmanuel Fernández-Gaucherand ‡ Marios Polycarpou §

Dept. of Electrical and Computer Eng. and Computer Science
University of Cincinnati,
Cincinnati, Ohio 45221-0030, USA

2-14-03

Abstract

A software architecture is presented, which introduces several agents which focus on different aspects of path planning for multiple autonomous unmanned aerial vehicles (UAV's) that are searching an uncertain and threatening environment for targets. One agent models threats in the environment. Another develops a model of the environment that allows targets to be defined by individual probability distribution. Lastly, an agent is presented that utilizes the information from the other agents to generate a near optimal path plan using a Dynamic Programming algorithm.

1 Introduction

Unmanned aerial vehicles (UAV's), such as the Predator [1], have been receiving an increasing amount of attention recently. The versatility of these craft, and their relative inexpensiveness, make them appealing for use in many areas where the use of manned aircraft would be too dangerous to the pilot or too costly. However, the Predator, while unmanned, still requires a substantial amount of manpower to operate, as it is still piloted, albeit remotely from a ground station. It is desired, then, that future UAV's, similar to the Boeing X-45A [2], have higher levels of autonomy.

However, directing autonomous agents such as these to behave in an "intelligent" manner constitutes a very interesting and challenging problem, especially in cases where there exist many constraints on the control of the agents. This is particularly true in the case of multiple autonomous unmanned aerial vehicles searching for targets in an uncertain environment. While the problem is a type of "search" problem, there are many factors present that set this problem apart from many of the classical search problems, such as those discussed in [6]. Even some standard methods of search become less desirable in the presence of a priori information and multiple vehicles, where the concept of cooperation among the vehicles is brought to the forefront.

The great bulk of the work in cooperative path planning and decision is in the area of "point-to-point" paths, where the vehicles have a given destination or set of destinations to travel to, as in [7] and [8]. While this area has produced many good results, the methods utilized are not really suitable to search, where a vehicle's final destination is secondary to the exact route it takes (and the targets discovered thereon.) The first step in the cooperative search path planning decision process is to model the environment in a suitable manner for autonomous route planning. Some of these methods have been proposed recently in [5], [9], and in previous work on this topic that has been evolving in several other papers by the authors of this work: [3] and [4]. However, this paper differs from the previous work by creating an formal software architecture.

*This work was supported by DARPA under the MICA (Mixed Initiative Control of Automa-teams) project.
†flintmd@email.uc.edu
‡emmanuel@ececs.uc.edu
§polycarpou@uc.edu

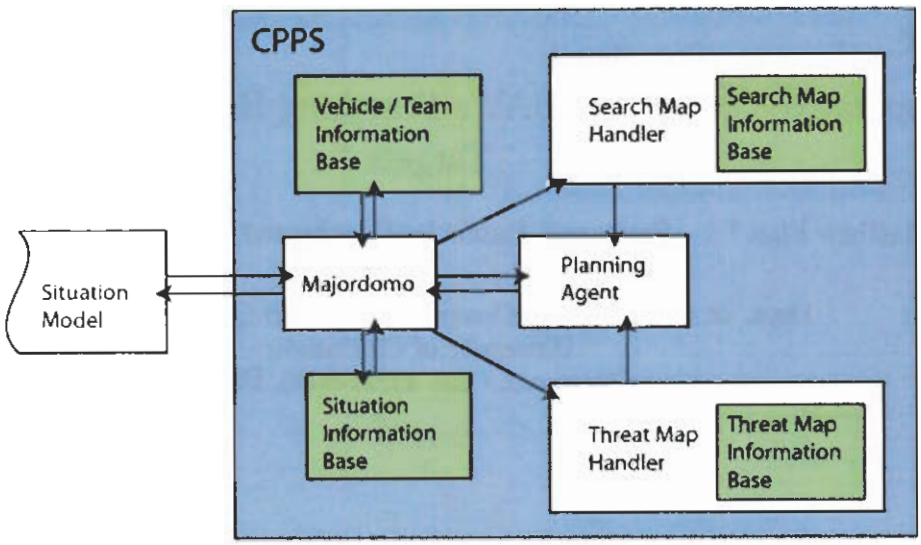

Figure 1: The architecture of the CPPS Agent.

Within this architecture there are now additional elements which have been added to the previous stochastic modeling and decision process formulation for the cooperative UAV search problem. This includes removing the assumption that all of the targets' distributions are completely independent and identically distributed. Also, threats are explicitly represented in the model of the environment and in the planning algorithm. Additionally, a more refined definition of gain is proposed.

Figure 1 shows the internal architecture of the CPPS module. The boxes show the boundaries of structures within the CPPS module, with the arrows showing the interfaces and the flow of information involved. The blue box shows the boundaries of the CPPS module. The boxes with green backgrounds show information bases, which passively store information. The white boxes show the agents, who are the actors in the module, and are described in the following sections. These agents utilize the information from the information bases and coordinate with one another to perform the functions of the module. The Situation Model that exists outside the module acts as a central information base and as the gateway to the human users, the simulator, and the other automated agents and reasoners that handle higher level tasks.

Section 2 is a discussion of the Search Map Handler, an agent which handles information about the probability of discovering targets in various regions of the environment. The Threat Map Handler, the agent that deals with threats, i.e. the danger posed to a vehicle travelling in various parts of the environment, is discussed in section 3. The next section, 4, is concerned with the keystone agent: the Planning Agent, which takes the information from the other agents and uses it to produce the best path available to the vehicle. Section 5 gives some very basic simulation results, and Section 6 gives conclusions and continuing work.

2 The Search Map Handler

For this analysis, any object for which the vehicles are searching is called a target, regardless of what other qualities the target may have (threat value, priority, etc). A target which is suspected to exist in the environment but has not been detected is called "suspected." A target that a vehicle detects, but does not locate with certainty is called "detected.". It is assumed that with each detected target there exists an uncertainty region of variable size that the target is known to be within. An uncertainty region which covers a lot of area means that the target's position is not known with much certainty. It is also assumed that there is some threshold, such that if this area is sufficiently small, the target will be considered as fully "found," and the vehicles will no longer be trying to gain more certainty about its location, and it ceases to be a candidate target for the search. Of course, if the target is destroyed, then it would also cease to be a candidate for the search.

2.1 Search Gain Defined

The Search Map Handler should produce a value that represents how valuable certain areas of the environment are to search. This value is termed "search gain." The search gain function should maximize the number of real targets (as opposed to false targets) detected or found. However, as the location and even the existence of the targets are based on random values, the expected number of targets detected will have to be maximized instead. In order to do this, the vehicle will have to choose which cells of the map to search. To find the maximum number of targets, a vehicle will want to search cells that have the highest number of expected targets in them. This expectation is based on the number of possible targets in the cell and three other factors. For a target i, let

- F_x^i be the event that target i is detected in cell x.
- T_x^i be the event that target i actually is in cell x.
- E^i be the event that target i exists (is a real target).

In order for a target to be within a cell (event T_x^i,) the target must first exist (event E^i), thus $T_x^i \subset E^i$. Also, since if a target exists, it must be somewhere,

$$E^i = \bigcup_x T_x^i. \tag{1}$$

The expected probability of discovering target i in cell x is determined by the probability of events F_x^i, T_x^i, and E^i all occurring (i.e. $P(F_x^i \cap T_x^i \cap E^i)$). But, since $T_x^i \subset E^i$, $P(F_x^i \cap T_x^i \cap E^i) = P(F_x^i \cap T_x^i)$, which can be written as

$$P(F_x^i \cap T_x^i) = P(F_x^i | T_x^i) P(T_x^i) \tag{2}$$

by conditional probability. Since event E^i completely includes event T_x^i,

$$P(T_x^i) = P(T_x^i \cap E^i) = P(T_x^i | E^i) P(E^i). \tag{3}$$

Putting these two equations together,

$$P(F_x^i \cap T_x^i) = P(F_x^i | T_x^i) P(T_x^i | E^i) P(E^i). \tag{4}$$

$P(F_x^i | T_x^i)$ is the probability that a target is detected by a sensor search of cell x given that the target is in cell x. This is defined as the sensor efficiency, ρ, which for this work is assumed to be the same, and constant, for all cells and targets and to be available a priori (e.g. as a function of the sensor type.) Also, $P(E^i)$, the probability that target i is a real target, is labeled as ζ^i. Lastly, $P(T_x^i | E^i)$ is the probability that, given that a target i is a real target, it is in a particular cell x. This can be found from the given distribution (uniform over the uncertainty area by default) of the target or from a priori information. This is not assumed to be the same for every target.

Let $\bar{w}_x^i$ be a random variable such that

$$\bar{w}_x^i = \begin{cases} 1 & : \text{ if } F_x^i, T_x^i, \text{ and } E^i \text{ all occur.} \\ 0 & : \text{ otherwise.} \end{cases} \tag{5}$$

This means that $\bar{w}_x^i = 1$ if target i is detected in cell x and is 0 otherwise. Define the set of all possible targets in a cell x as G_x. Let σ be a function that takes as an argument a cell (or a collection of cells), and returns the search gain for searching that cell (or the sum of the search gains for searching each individual cell in that collection of cells.) The search gain is then calculated by taking the expectation for finding targets in a cell x over $\{\bar{w}_x^i\}_{i \in G_x}$, as shown here:

$$\begin{aligned} \sigma(x) &= \mathrm{E}\{\#\text{ Targets detected in } x\} \\ &= \sum_{i \in G_x} \mathrm{E}\{\bar{w}_x^i\} \\ &= \sum_{i \in G_x} \{P(F_x^i \cap T_x^i)\} \\ &= \sum_{i \in G_x} \{\rho \zeta^i P(T_x^i | E^i)\}. \end{aligned} \tag{6}$$

2.2 The Search Map

It is possible to efficiently store the values necessary to calculate the search gain from Eq. 6. First, the map is defined as a bounded rectangular area divided into discrete, equal-area cells. For each cell of the map there exists an array (or vector) that stores several values. The first value in the array is the product of the vehicle's search: a value that represents the reduction in uncertainty about what is in the cell because of the action of the vehicles. Let ν_x, the number of times a cell has been searched, represent this value for cell x.

The second and subsequent values in the array are calculated based on information about particular targets that have some probability of being located in the cell. For each target, this can be found by taking the area in which that target may be found (e.g. the uncertainty region), and determining which of the map cells are within this area. Then, a value known as the relative probability value (C_x^i) for target i in cell x is calculated, using the distribution that is given for that target. If no information is available a priori to create this distribution, it is assumed to be uniformly distributed over the whole search area. Each cell of the map holds an array (of the same size as the number of elements of G_x) of these C_x^i values for each target. Thus the total size of the array for each cell x is the number of targets that may be in the cell plus one (for ν_x).

Note that when the probability value of one cell for a specific target changes, the value of every other cell where that target is possible would also change. Updating several cells any time a single one changes is not very efficient, especially when there are a large number of cells. Thus the relative probability values are defined such that only the value of the cell that is searched is changed; the others do not change.

How these relative probability values can then be used to calculate the search gain is shown. Let P_x^i be the probability of target i being in cell x (given that the target exists,) and let N_c^i be the number of cells in the uncertainty area of the target. So,

$$P_x^i = P(T_x^i|E^i) = \frac{A_x C_x^i}{\sum_{x=1}^{N_c^i} A_x C_x^i}. \tag{7}$$

Now, A_x is a constant value for every cell, so

$$P_x^i = \frac{C_x^i}{\sum_{x=1}^{N_c^i} C_x^i}. \tag{8}$$

Then letting

$$V^i = \frac{1}{\sum_{x=1}^{N_c^i} C_x^i}, \tag{9}$$

equation 7 simplifies to

$$P_x^i = V^i C_x^i. \tag{10}$$

Since V^i and ζ^i are the same for every cell for a particular i, these values do not need to be stored in the arrays for the cells of the map. Instead, these values can be stored as a list which records, for each target i, the ζ^i value and V^i value for that target. And so, (using Eq. 6 and Eq. 10) the search gain can now be written as

$$\sigma(x) = \sum_{i \in G_x} \{\rho \zeta^i V^i C_x^i\}), \tag{11}$$

where each of these values is available as a priori information (ρ) or from the search map (ζ^i, V^i, C_x^i).

2.3 Updating The Search Map

For each target $i \in G_x$ and for all x that are searched by a vehicle at a certain time step (i.e. the cells that fall within the vehicle's sensor's footprint,) there are three (mutually exclusive) things that can occur before the next time step. They are

- Target not detected.
- Target detected but not found.
- Target detected and fully found, or target destroyed.

2.3.1 Target not Detected

Let $\bar{F}^i_{x_1}$ be the event that target i is not detected on a search of cell x_1. Now, on a search of the cell, either a target is detected or it is not, so $P(\bar{F}^i_{x_1}) + P(F^i_{x_1}) = 1$. This occurs regardless of whether the target actually exists in the cell or not, so $P(\bar{F}^i_{x_1}|T^i_{x_1}) + P(F^i_{x_1}|T^i_{x_1}) = 1$. So,

$$P(\bar{F}^i_{x_1}|T^i_{x_1}) = 1 - P(F^i_{x_1}|T^i_{x_1}). \tag{12}$$

Since $P(F^i_{x_1}|T^i_{x_1}) = \rho$,

$$P(\bar{F}^i_{x_1}|T^i_{x_1}) = (1-\rho). \tag{13}$$

Now, when a cell is searched and target i not detected, the map would like to record the updated probability that target i is in that cell (i.e. that it is really still there.) Assuming that the probability of existence of the target does not change due to this sensor sweep, the updated probability that the target is located in the cell after the search (given that the target exists) is given by conditional probability as

$$P(T^i_{x_1}|(\bar{F}^i_{x_1} \cap E^i)) = \frac{P((\bar{F}^i_{x_1} \cap E^i)|T^i_{x_1})P(T^i_{x_1})}{P(\bar{F}^i_{x_1} \cap E^i)}. \tag{14}$$

The fact that $T^i_x \subset E^i$ means that if $T^i_{x_1}$ is given then event E^i is given as well. This and Eq. 3 let Eq. 14 be written as

$$P(T^i_{x_1}|\bar{F}^i_{x_1}, E^i) = \frac{P(\bar{F}^i_{x_1}|T^i_{x_1})P(T^i_{x_1}|E^i)P(E^i)}{P(\bar{F}^i_{x_1}|E^i)P(E^i)}. \tag{15}$$

The event whose probability is given by $P(\bar{F}^i_{x_1}|E^i)$ occurs under two conditions. The first is if the target is not located in the cell, an event with probability $1 - P(T^i_{x_1}|E^i)$. The second is if the target is in the cell, but that it was not detected, an event with probability $P(\bar{F}^i_{x_1}|T^i_{x_1})P(T^i_{x_1}|E^i)$. Since these two events are mutually exclusive, the probabilities are additive. Using Eq. 13, this sum can be written as

$$\begin{aligned} P(\bar{F}^i_{x_1}|E^i) &= 1 - P(T^i_{x_1}|E^i) + (1-\rho)P(T^i_{x_1}|E^i) \\ &= 1 - \rho P(T^i_{x_1}|E^i). \end{aligned} \tag{16}$$

Thus, Eq. 15 becomes

$$P(T^i_{x_1}|\bar{F}^i_{x_1}, E^i) = \frac{(1-\rho)P(T^i_{x_1}|E^i)}{1 - \rho P(T^i_{x_1}|E^i)}. \tag{17}$$

Note that this update equation is valid only for the cell x_1 referred to by event $\bar{F}^i_{x_1}$, even though the probabilities of the target being located in all of the other cells on the map are also changed. Let x_2 be an arbitrary cell that is not referred to by event $\bar{F}^i_{x_1}$. The post priori (after event $\bar{F}^i_{x_1}$) probability of this cell (again, given that target i exists) is given by

$$P(T^i_{x_2}|\bar{F}^i_{x_1}, E^i) = \frac{P(\bar{F}^i_{x_1}|T^i_{x_2})P(T^i_{x_2}|E^i)}{P(\bar{F}^i_{x_1}|E^i)}. \tag{18}$$

It is assumed for the purposes of this paper that the sensor will not misplace the target if it is detected, so if the target is in cell x_2 it will not be detected in cell x_1 and the probability $P(\bar{F}^i_{x_1}|T^i_{x_2}) = 1$. Using this and Eq. 16, Eq. 18 becomes

$$P(T^i_{x_2}|\bar{F}^i_{x_1}, E^i) = \frac{P(T^i_{x_2}|E^i)}{1 - \rho P(T^i_{x_1}|E^i)}. \tag{19}$$

The map handler now wishes to record the new (post priori) probability values on the map. However, the map does not store the probability ($P^i_{x_1}$) values directly. Instead, the relative probability values ($C^i_{x_1}$) are stored. Let x_1 be the cell that was searched and is being updated, and x_2 be any other arbitrary cell with $C^i_{x_2} \neq 0$. To simplify the notation somewhat, a prime ($'$) is used to represent a post priori value, (i.e. $P^{i\,\prime}_{x_n} = P(T^i_{x_n}|\bar{F}^i_{x_n}, E^i)$ for $n = \{1, 2\}$.) Now, the relative probability values are defined so that

$$\frac{C^i_{x_1}}{C^i_{x_2}} = \frac{P^i_{x_1}}{P^i_{x_2}} \tag{20}$$

and

$$\frac{C_{x_1}^{i\,\prime}}{C_{x_2}^{i\,\prime}} = \frac{P_{x_1}^{i\,\prime}}{P_{x_2}^{i\,\prime}}. \tag{21}$$

However, $C_{x_2}^{i\,\prime} = C_{x_2}^{i}$, since only $C_{x_1}^{i}$ was searched. Using Eq. 17 and Eq. 19,

$$\frac{C_{x_1}^{i\,\prime}}{C_{x_2}^{i}} = \frac{P_{x_1}^{i\,\prime}}{P_{x_2}^{i\,\prime}} = \frac{(1-\rho)P_{x_1}^{i}}{P_{x_2}^{i}}. \tag{22}$$

Now, using equation 20,

$$\frac{C_{x_1}^{i\,\prime}}{C_{x_2}^{i}} = \frac{(1-\rho)C_{x1}^{i}}{C_{x_2}^{i}}. \tag{23}$$

Therefore,

$$C_{x_1}^{i\,\prime} = (1-\rho)C_{x_1}^{i}. \tag{24}$$

So for the case where a cell is searched and a target is not detected, use Eq. 24 to update the value of the array for cell x for that target. No other cell value is changed, but the value of V^i must also be updated. Since the changes to the map are made incrementally and one cell at a time, this allows the value of V^i to be changed in the same manner, so that a running value can be easily maintained. Now,

$$V^i = \frac{1}{C_1 + \ldots + C_{x_1} + \ldots + C_{N_c}^{i}} \tag{25}$$

It is desired that the C_{x_1} term in the denominator be updated with the new value $C_{x_1}^{i\,\prime}$. Thus, V^i can be updated (to $V^{i\prime}$) by using

$$V^{i\prime} = \frac{1}{\frac{1}{V^i} + (C_{x_1}^{i\,\prime} - C_{x_1}^{i})}. \tag{26}$$

2.3.2 Target Detected

If a target i is detected, then remove all previous entries for that target (i.e. all the C_x^i values) from the arrays of all of the cells. Now, the C_x^i values must be generated using the new target information based on the new distribution that is available because the target was detected. Then store this information in the proper arrays of the cells of the map, making sure to record the ζ^i value for that target also. Next, the C_x^i values must be updated to account for any searching in the cells that has already occurred previously. For each cell in the area of the new distribution of that target, apply equations 24 to the map values a number of times equal to the value of ν_x for that cell. The value of V^i is also calculated and stored.

2.3.3 Target Found / Destroyed

This requires the removal of the target from the map entirely, since the target is no longer a candidate for search, either because it does not exist anymore, or because its location is known with enough certainty. In this case, remove it from the array of possible targets for every cell. Its value will no longer be used when calculating the number of (uncertain) targets expected to be found in a search of any particular cell.

3 The Threat Map Handler

This section describes how threats are modeled in a form that can be incorporated efficiently by the Threat Map Handler and stored by the Threat Map Information Base.

The level of danger to a flying UAV may depend on several factors, which include any one or several threats from opposing forces, including SAM sites or random ground fire, or the terrain itself in the form of hills, mountains, etc. Since several of these factors have a direct relationship to altitude, this must also be explicitly accounted for.

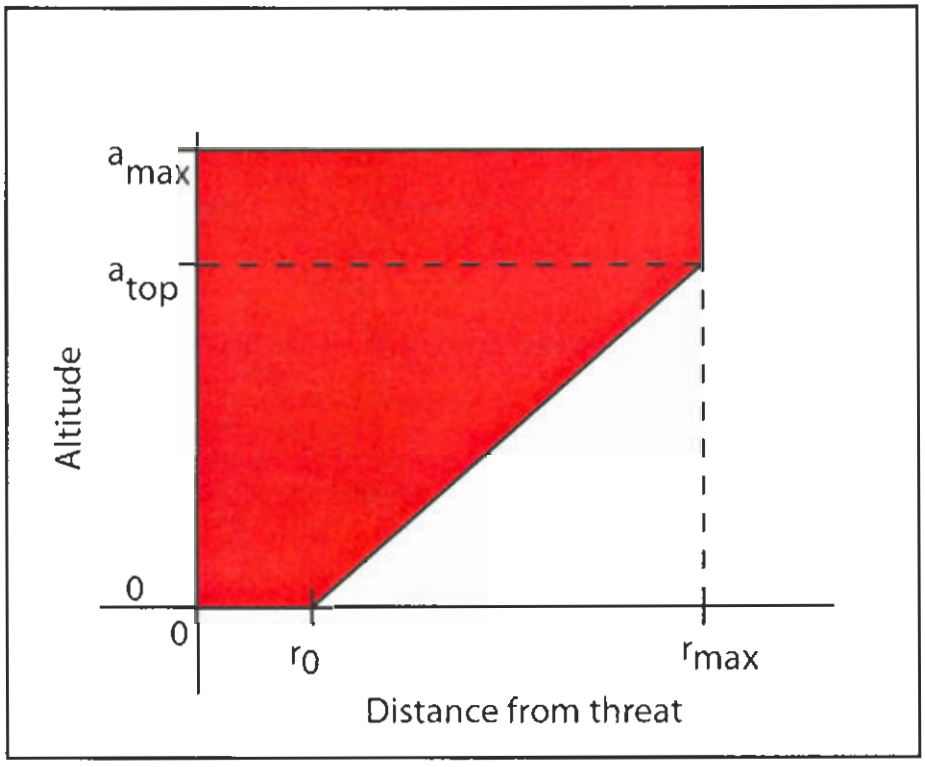

Figure 2: A simplified threat profile.

Let a_u be the altitude of the UAV. Let a_g be the altitude of the ground or some other terrain feature into which the UAV could crash if they are at the same altitude. Thus, the probability of being destroyed by the terrain (D_g) is

$$D_g = \begin{cases} 0 & a_u > a_g \\ 1 & a_u \leq a_g \end{cases} \tag{27}$$

This is true anywhere in the environment, but since a_g can vary in different regions of the environment, a map must record the value for a_g in different regions of the environment. A two dimensional map, like the search map, will work. Also, like the search map, this terrain map is divided into a grid of equal sized cells, the size of which can be adjusted from run to run and does not necessarily have to be correlated with the size of the cells for the search map.

There is assumed to be an altitude (a_f) above which the vehicle is safe from ground fire, but below which it is threatened. Let D_f be the probability of being destroyed by ground fire. This value is given as:

$$D_f = \begin{cases} 0 & a_u > a_f \\ F & \text{otherwise} \end{cases} \tag{28}$$

The value of $F \in [0, 1]$ is assumed for this paper to be a constant value over the entire environment.

Let r_u^i be the distance from the point directly below the UAV (at 0 altitude) to threat i. This is a function of the x and y locations of the vehicle (x_u, y_u), and is calculated from the threat location (x^i, y^i). Let $r_s^i(a_u)$ be the minimum safe distance from the threat for a vehicle at altitude a_u, which must be calculated using the threat range profile for threat i. The threat range profile is assumed to be known a priori and to be look something like that shown in Fig. 2.

Fig. 2 shows several key values of the profile. The distance r_0 is the largest distance at which the SAM can hit a vehicle at 0 altitude. The distance r_{max} is the farthest away the SAM site can fire. The altitude a_{max} is the highest altitude that a vehicle can be fired upon by the SAM site. The altitude a_{top} is the

value above which altitude does not change the threat range (until a_{max} is reached). It is assumed that the minimum altitude is 0, and that the minimum safe distance boundary can be given as a line. Thus,

$$r_s^i(a_u) = \begin{cases} 0 & a_u > a_{max} \\ r_{max} & a_{top} \leq a_u \leq a_{max} \\ \frac{a_u}{a_{top}}(r_{max} - r_0) + r_0 & 0 < a_u < a_{top} \end{cases} \tag{29}$$

Let D_t^i be the probability of being destroyed by threat i. This is given by

$$D_t^i = \begin{cases} 0 & r_u^i > r_s^i(a_u) \\ T^i & \text{otherwise} \end{cases} \tag{30}$$

The value of $T^i \in [0, 1]$ for threat i is assumed for this paper to be a constant for over the whole area covered by that.

The threat map handler records the location and type of all SAM's. Then, for a given UAV's location, it can calculate the threat level from a SAM to the UAV using the information about the profile for that type of SAM. Then, to determine the total threat level to a UAV, the map handler calculates the values for D_g, D_f, and D_t^i for for all threats $i = 1, \ldots, n$ that threaten a vehicle at a given (x_u, y_u, a_u) location using Eq. 27, Eq. 28, and Eq. 30. The total threat (d) facing a vehicle at that location in the environment can then be found by:

$$d = 1 - [(1 - D_g)(1 - D_f)(1 - D_t^1) \ldots (1 - D_t^n)] \tag{31}$$

4 The Planning Agent

The Planning Agent takes the information from the other agents and uses it to decide the path that produces the most expected targets detected over (ideally) the entire lifetime of the vehicle, which is N time steps, or (non-ideally) a smaller, more feasible, planning horizon, which is defined as q ($q < N$) time steps.

In order for the vehicles to plan their trajectories, they need to know information about the state of the environment. The true state of the environment, after discretization, is represented by: (1) an information base like those maintained by the agents, except that it is updated by every vehicle at every time step (producing an ideal information base); and (2) by the locations and headings of the vehicles. This true state is denoted by x_k^*. The state, as perceived by an individual vehicle, is denoted as x_k. Under ideal conditions, every vehicle will perceive the state as being the true state ($x_k^* = x_k$,) but in the non-ideal case, each vehicle may have a different perception based on the information available to it.

The choice of which path to travel for a vehicle at time k is the decision, or control u_k, where $u_k \in U$, where U is a set that contains the choices that the vehicle can take (for example: turn 15° left, go straight, or turn 15° right.) Let J_k be defined as the "cost-to-go" function from time step k to N. Then, the best path at time step k can be found from utilizing the DP recursion over the planning horizon (ideal) k to N or (non-ideal) k to $k + q$:

$$J_k(x_k) = \max_{u_k \in U}(\{g(x_k, u_k) + J_{k+1}(f(x_k, u_k)\}) \tag{32}$$

where $g(x_k, u_k)$ is the one-step gain function. Note that if the expectation of the stochastic elements is encapsulated and contained within the gain function then this is a deterministic equation, and can be solved using a shortest-path algorithm.

To ensure cooperation, the other vehicles are modeled as stochastic elements, where a random quantity w_k is used to represent the loss in search gain actually received by the vehicle compared to what was expected when the decision was made because of interference by another vehicle. The vehicles can then use the expected result of this w_k to plan where to go to avoid undue interference.

The one-step gain at each step is then the expected search gain one would get if no interference were present minus the expected amount of search gain one would lose from another vehicle interfering,

$$g(x_k, u_k) = E\{\sigma - w_k\} \tag{33}$$

This one-step gain is then used in the DP recursion (Equation (32)) to determine the expected optimal path.

The amount of interference a vehicle expects is a function of ρ, σ, and the probability of another vehicle interfering, which is denoted as Ψ. So

$$E\{w_k\} = \rho\,\sigma\Psi \tag{34}$$

Adding in the interference calculations, we have

$$g = E\{\sigma - w_k\} = \sigma - \rho\sigma\Psi \tag{35}$$

Where σ is the search gain (with no interference,) ρ is the sensor efficiency, and Ψ is the probability of interference. This value $\Psi \in [0,1]$ provides penalties to the vehicles for searching areas that have some probability of being searched by another vehicle first. In effect, it forces the vehicles to spread apart. This is covered in much more detail in [3] and [4].

4.1 Adding Threats into the Planning algorithm

Threats can be incorporated into the decision model by recognizing what happens to a vehicle if it enters a threatened area: 1.) nothing happens (i.e. the vehicle travels as normal) or 2.) the vehicle is shot down, or crashes, and is destroyed. There is assumed, for the moment, to be no intermediate states (i.e. no damaged state.) Having a vehicle destroyed means that the vehicle is no longer available to search for the remainder of this mission (or any future mission, either.) So, there is now a new state that a vehicle can enter at any time during the mission: dead. If a vehicle is destroyed at time k, it will move to this state for time step $k+1$ and its one-step gain (g_{k+1}) for this state and for all future states will be 0 (since the vehicle cannot leave the dead state.) However, any gain it had previously received it will not lose.

Additionally, if the vehicle is destroyed, there is also a cost for not being available to use in future missions. So, J_N, the final reward at the end of the mission, now has two different values. If a vehicle is destroyed during the mission, J_N should be 0, as this represents that the vehicle cannot provide any more gain in any future missions. If a vehicle is not shot down, J_N should be X, where X is a positive reward for not being destroyed, and represents the value of having the vehicle available for future missions.

4.2 Utilizing the Information from the Search and Threat Map Handlers

Continuing with this analysis, we can now write the expected cost-to-go (or reward) for taking an arbitrary path by expanding the recursion Eq. 32:

$$\begin{aligned}\mathrm{E}\{J_k\} = (1-d_k)[\sigma_k - \rho\sigma_k\Psi_k + \\ (1-d_{k+1})[\sigma_{k+1} - \rho\sigma_{k+1}\Psi_{k+1} + \ldots + \\ (1-d_{k+q})[\sigma_{k+q} - \rho\sigma_{k+q}\Psi_{k+q} + X]]].\end{aligned} \tag{36}$$

Renaming $(1-d_i)$ as s_i for arbitrary i, factoring out σ, and letting $I_i = (1-\rho\Psi_i)$, Eq. 36 can be written as

$$\mathrm{E}\{J_k\} = s_1[\sigma_k I_k + s_{k+1}[\sigma_{k+1}I_{k+1} + \ldots + s_{k+q}[\sigma_{k+q}I_{k+q} + X]]] \tag{37}$$

which then can be decomposed into succinct time steps by letting

$$\delta_i = \prod_{j=1}^{i} s_j \tag{38}$$

and then noting that now

$$\mathrm{E}\{J_k\} = [\sigma_k\delta_k I_k] + [\sigma_{k+1}\delta_{k+1}I_{k+1}] + \ldots + [\sigma_{k+q}\delta_{k+q}I_{k+q}] + \delta_{k+q}X \tag{39}$$

So all the terms for a certain time step are collected together. Since the one-step gain at a particular time step depends only on the current and previous time steps, these can be converted to a cost (distance) and can be solved as a shortest-path problem like those found in [3] and [4].

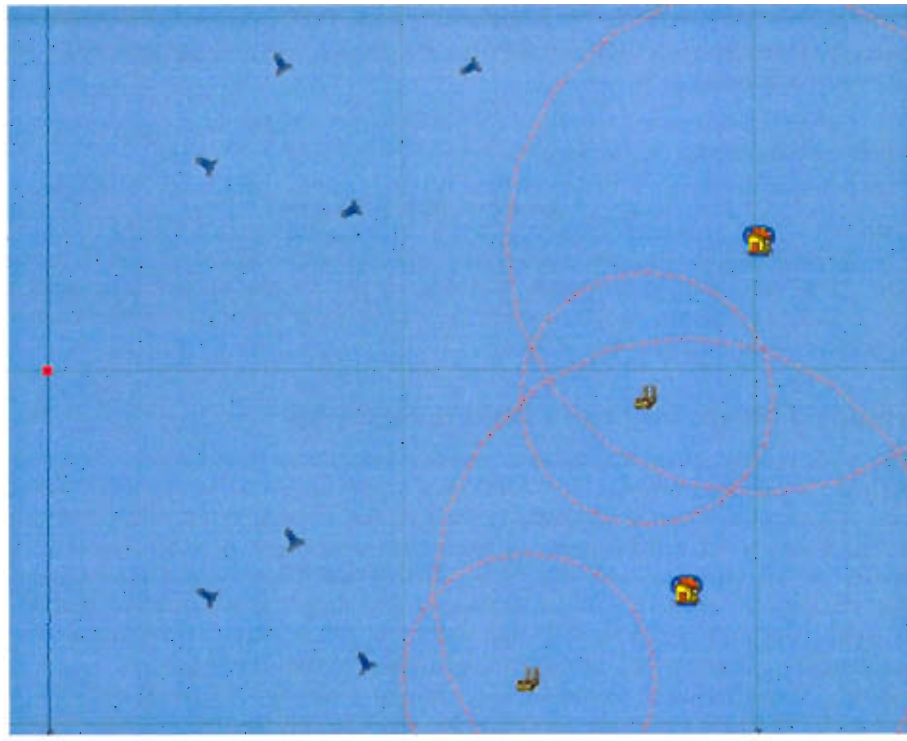

Figure 3: Vehicles Searching a Region with Threats.

5 Simulation

Results for this work are obtained by using the Boeing Open Experimental Platform (OEP).

Figure 3 shows a picture of a trial being conducted in the OEP. Two teams of vehicles are shown, one above the -1 degree latitude line, and one below. The threats, two long range SAMs and two medium range SAMs are shown on the right, with their accompanying threat range rings. These rings show the minimum safe distance at 5000ft altitude.

Additional simulation studies are underway in order to explore the vehicle's behavior and the efficacy of making tradeoffs in the values used in the algorithm.

6 Conclusions

The architecture of the Cooperative Search Path Planner allows focus on particular aspects of the cooperative search path planning problem to be refined in an efficient manner. Advances in the sensor model and the way that information about targets and uncertainty are given and stored are covered by the Search Map Handler. The level of risk posed to a vehicle by various threats in the environment is maintained by the Threat Map Handler. Then, the Planning Agent utilizes this information provided by the other agents in a Dynamic Programming algorithm to find near optimal solutions.

However, there are still many avenues to pursue to refine this work further. The ability to include a variable value of ρ in the formulation allows for multiple types of sensors to be incorporated into the model. Additionally, finding ways of including the effect of removing the assumption that $P(\bar{F}^i_{x_1}|T^i_{x_2}) = 1$ allows for a more robust sensor model. Creating more complex threat environments is also possible by allowing F and T^i to vary. Also, of particular interest is the value Ψ, which is currently implemented by reducing the gain seen by a vehicle, but without taking the threats into account. But, with threats, there can be an additional component that takes into account the fact that other vehicles may be shot down before they can

interfere. This would lessen the amount of penalty to the search gain which forces the vehicles away from one another when searching critical, but dangerous, areas. Additionally, the presence of targets that may be detected and require another vehicle to take a second look could eliminate the effect of the interference in certain cases.

References

[1] Anonymous. Predator uav. Internet Fact Sheet, 2002. `http://www2.acc.af.mil/library/factsheets/predator.html`.

[2] Anonymous. Unmanned combat air vehicle x-45. Internet Fact Sheet, 2002. `http://www.boeing.com/phantom/ucav.html`.

[3] M. Flint, E. Fernández-Gaucherand, and M. Polycarpou. Stochastic models for cooperative control of multiple autonomous uav's searching for targets in an uncertain environment. *Submitted to MOR Journal*, 2002.

[4] M. Flint, M. Polycarpou, and E. Fernández-Gaucherand. Cooperative control for multiple autonomous uavs searching for targets. In *2002 Conference on Decision and Control*, pages 2823–28, 2002.

[5] J. P. Hespanha and H. H. Kizilocak. Efficient computation of dynamic probabilistic maps. In *Proc. of the 10th Mediterranean Conf. on Control and Automation*, 2002.

[6] B.O. Koopman. *Search and Screening: General principles with Historical Application*. Pergarnon, New York, 1980.

[7] A. Richards, J. Bellingham, M. Tillerson, and J. How. Co-ordination and control of multiple uavs. In *Proc. of the 2002 AIAA GNC Conference*, 2002.

[8] A. Richards and J. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proc. of the 2002 Amer. Contr. Conf.*, 2002.

[9] Y. Yang, A. A. Minai, and M. M. Polycarpou. Decentralized cooperative search in uav's using opportunistic learning. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2002.