Proceedings of the 42nd IEEE
Conference on Decision and Control
Maui, Hawaii USA, December 2003

Pursuit-Evasion Strategies for Teams of Multiple Agents with Incomplete Information

Adonis Antoniades H. Jin Kim Shankar Sastry
Department of Electrical Engineering & Computer Sciences
University of California at Berkeley
Berkeley, CA 94720, USA
{adonis,jin,sastry}@eecs.berkeley.edu

Abstract— **In this paper, we investigate search strategies for multi-player pursuit-evasion games, in which a team of pursuers try to detect and capture multiple intelligent evaders. These are games with incomplete information for which there is no complete theory of existence of optimal solutions. We present a number of heuristic pursuit strategies and compare their performance using capture time as the performance metric. We demonstrate that reducing sensing overlap between pursuers and avoiding over-assignment of pursuers to target locations improve the performance of pursuit policies, and then derive some conclusions for their further development.**

I. INTRODUCTION

This paper presents algorithms for pursuit-evasion games (PEG) with multiple pursuers and evaders. In the pursuit-evasion game which we consider, a pursuer (or a group of pursuers) searches for an evader (or a group of evaders). A suitable physical analogy would be a military surveillance mission or a public safety operation with Unmanned Aerial Vehicles (UAV) and Unmanned Ground Vehicles (UGV) serving as the pursuers and evaders respectively. In this scenario, the UAVs would be sent to search a region for intruding UGVs or people in danger. These are complex dynamic games with incomplete information for which there is no existence theory for optimal solutions. Consequently we design and compare the performance of a number of heuristically good strategies. This comparison would be useful as a basis for the future "learning" controllers or the studies of finite horizon games.

In our setting, all the players act in a bounded rectangular area which we will refer to as a *stage*. The pursuers can sense only a finite region around them, which presents the difficulty of *partial observability*. Based on the history of such limited sensing information, the pursuers try to outsmart the evaders and eventually capture them. The evaders can either move around randomly in this region, or actively try to avoid the pursuers. The game ends when all evaders are captured. Some work has been done in developing pursuit policies assuming accurately mapped environments and considering worst-case motion of evaders [1]. Such deterministic policies guarantee detection of an evader but can be too conservative for many practical applications. The lack of complete information about the world suggests that the problem should be approached in a probabilistic framework. In [2], we used the *probability map* for searching in partially observable environments, and presented a computationally feasible pursuit policy which sends the pursuers to the local maximum point of the probability map. In [3], we presented the global-max policies in addition to the local-max policies presented in [2]. In parallel to this theoretical work, we have been developing a platform of UAVs and UGVs as a test-bed of multi-agent coordination and control. In [4], a real-time control system for regulation and navigation of a UAV was developed. In [5], we presented a distributed hierarchical system architecture for pursuit-evasion games, described the implementations of the navigation, communication and sensing layers of teams of UAVs and UGVs, and tested the global-max policies and the local-max policies on a team of UAVs and UGVs.

In this paper, we extend these ideas to the case of multiple evaders. To do this, we incorporate overlap-reducing algorithms into our previous work. These new algorithms will reduce the amount of sensing overlap between pursuers, thus distributing them more efficiently. Furthermore, the evaders follow an evasion policy that actively tries to steer them away from pursuers. Because of these new considerations, our problem becomes inherently difficult to analyze theoretically. Here we develop heuristic algorithms and test the policies of pursuers with varying sensing capabilities against faster evaders with enhanced intelligence. The focus of our work is on pursuit policies. For a detailed discussion of a probabilistic approach to map-building and localization, refer to [6].

The remaining parts of this paper are organized as follows: Section II describes our game framework that will be used in the development of pursuit-policies. Section III describes the pursuit policies. In Section IV, we present the simulation results, comparing the performance of global/local policies with/without the consideration of maximizing the collective visibility region of pursuers while varying the game parameters such as the sensing size and number of agents. Section V concludes the paper with the future directions.

II. PROBLEM FORMULATION

In this section, we consider a system of N robots in a two-dimensional environment, searching for a multiple, unknown number of evading robots. We abstract the physical world as a grid world, and play a general game using the following rules and parameters:

This work was done while the first author was a graduate student at UC Berkeley. He is now with National ICT Australia (NICTA).

0-7803-7924-1/03/$17.00 ©2003 IEEE

- We have n pursuers and m evaders.
- The pursuers have a square sensing field. Each side of the square is d cells long.
- All the pursuers move every other step, and evaders move every step. Within each time step, all the moving players change locations simultaneously.
- At time steps when pursuers do not move they can still detect an evader that lies in their sensing region.
- An evader is captured if it has been sensed by pursuers for k times. These steps do not have to be consecutive.
- The game is not symmetric in that pursuers never get destroyed, as evaders never attack.
- The game ends when all evaders have been captured.

A. Discrete Game Model

We overlay a grid on top of our stage, which essentially divides it into cells of some pre-specified size. Each cell is mapped to a node x on the graph and let X be the set of all nodes on the graph. This is demonstrated in Fig. 1(a), and from now on, we will use the terms *cell* and *node* interchangeably.

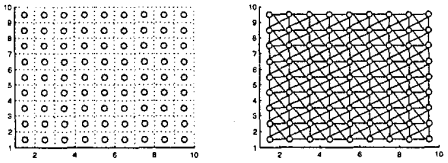

Fig. 1. Pursuit-evasion game model (a) Mapping cells to nodes, (b)Introducing edges to model the motion dynamics

Players are allowed to move to one-step reachable nodes from their current node. In our graph framework, this is represented by edges that connect each node to the cells that can be reached in one time-step. This definition of one-step reachable neighbors also includes the cells in the diagonal, as shown in Fig. 1(b). In addition, in order to challenge our pursuers further, evaders are assumed to move every time-step whereas pursuers move every other time-step.

B. The probability map

We define the measurement history Y_t to be the set of all the sensing data of each pursuer, along with the locations of each pursuer for all time steps up to and including time step t. The probability map assigns to each cell the probability of an evader being present in the cell and define:

$$p_e(x, t|Y_t)$$

to be the probability of an evader being in cell x at time t given the measurement history Y_t. In essence, the probability map is a two-dimensional table with positive numbers that add up to one. There exist some rules for updating the probability map at each time step, and these were presented in previous work ([2], [5]).

C. Multiple evaders-Multiple probability maps

In the previous work that considered one evader, only one probability map was used. A natural extension to the case of multiple evaders would be to use multiple probability maps. We have implemented this idea in the following way: We have one probability map that represents the probability of evader presence, assuming that no evader has been detected. So, in essence, the question that this map is trying to answer is the following: "Provided the pursuers have not detected an evader yet, what is the probability that an evader exists in the cells they are not currently observing?" We will call this map the *general map* and its probability density function $p_g(x, t|Y_t)$. Now upon detecting an evader we create another probability map, which is specific to that evader. This map works in the same way as the general probability map with one minor difference: each time the evader is being detected in a cell, that cell is assigned a probability equal to 1 and the other cells are reset to zero. We define

$$p_{e_i}(x, t|Y_t)$$

to be the probability of evader i being in cell x at time t given the measurement Y_t. Every new evader detected is assigned its own probability map. Thus at every time step we have the general map always present, and one or more evader maps active.

D. Assigning Identities to evaders

Once pursuers detect an evader they may lose detection for a few time steps and then may start detecting it again. How can the pursuers tell that the evader they are detecting now is the same one that they were detecting before? We assume the following: If the evader remains undetected for a time period T then the pursuers have technically lost the evader so deactivate the probability map we built for it. If it remains undetected for less than T time steps then we assume that if a pursuer detects the same evader again, it can tell if it is the same one that they detected before or if it is a new one. In a real physical scenario, this will involve some test concerning physical properties of the detected evader such as dynamics of motion, color, shape, location, etc.

E. Combining probability maps

Assume that at step t the pursuers are detecting k evaders. This means, that the pursuers have generated k evader maps and one general map. We furthermore assume that the actions of evaders are independent, so the presence of one evader in one cell does not affect the probability of another evader being present in the same cell. Since we never know the complete number of evaders, we use the general map as the probability map of the $k+1$ evader which we suspect to be out there but we have not detected yet. Under these assumptions, we have $k+1$ probability maps which we need to combine into one map. This one compact map that we aim to produce will function like a reward map. We assume

that each pursuer has sufficient power and time to attempt to capture all the evaders that lie in one cell. So, provided we are being rewarded with r_{ixt} units for attempting to capture evader i in cell x at time t, then the total expected reward associated with cell x at time t given the measurement history Y_t is given by:

$$\rho(x,t|Y_t) = p_g(x,t|Y_t) \cdot r_{(k+1)xt} + \sum_{i=1\ldots k} [p_{e_i}(x,t|Y_t) \cdot r_{ixt}]$$

We assume that the same reward is given for attempting to capture any given evader in any given cell at time t and we denote this by ϱ_t. Let I be the set of indices of all evaders that the pursuers are currently detecting. Then:

$$\rho(x,t|Y_t) = \varrho_t \cdot \left[p_g(x,t|Y_t) + \sum_{i \in I} [p_{e_i}(x,t|Y_t)] \right]$$

The choice of value for ϱ_t will depend on the real life scenario that we are trying to simulate. For the sake of simplicity, we will from now on use ϱ_t as a normalizing constant set to $\frac{1}{k+1}$. To avoid confusion with the terminology introduced in the next section we will refer to $\rho(x,t|Y_t)$ as the *single cell reward.*

The creation of the single cell reward map is demonstrated in Fig. 2. Note that the dark areas are areas of higher evader presence probability (higher rewards). Fig. 3 shows an instance of the game. The solid-filled squares with no outer squares represent the evaders, and those with with the outer square represent the pursuers, with the outline marking their sensing region. Finally, the unfilled outer squares denote the boundary of our stage. No pursuer or evader can move out of this boundary.

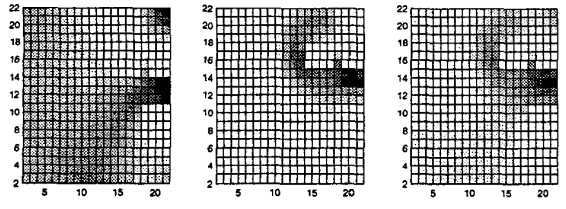

Fig. 2. Combining the general map and one evader map to obtain the reward map: (a) the general map, (b) an active evader map, and (c) the single cell reward map

III. SEARCH ALGORITHMS

In this section we formally describe our search policies, which correspond to the intelligence of the pursuing agents. We define the state-space for both pursuers and evaders to be X. That means that the state of a player is the cell in which it exists, i.e. their location. Define $x_{p_k}(t)$ to be the state of pursuer k at time-step t. Similarly define X_{e_i} to be

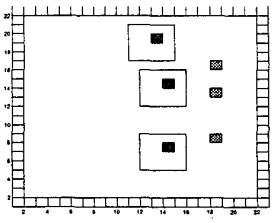

Fig. 3. An instance of the game between three pursuers and three evaders

the random variable denoting the state of evader i and let $x_{e_i}(t)$ be its realization at time step t.

A. Local-max policy

This is the simplest of policies, whose property was proved in [2]. Let $U(x_{p_k}(t))$ denote the one-step reachable set for pursuer k when the pursuer is in cell x at time t. A reward for moving to cell y is defined to be the sum of the single cell rewards associated with the sensing area of the pursuer, provided the pursuer was centered in cell y. Let $S_k(y)$ denote the set of all cells that lie within the sensing area of pursuer k located at the cell y. Then the **total** reward associated with pursuer k moving to cell y is given by:

$$R(y,t) = \sum_{z \in S_k(y)} [\rho(z,t+1|Y_t)]$$

where $\rho(z,t+1|Y_t)$ is the single cell reward, which was defined to be the expected reward associated with cell z at time $t+1$ given the measurement history Y_t. Under the local-max policy pursuer k moves to the cell $x_{p_k}(t+1)$ that gives the highest total reward, that is:

$$x_{p_k}(t+1) = \arg \max_{y \in U(x_{p_k}(t))} [R(y,t)]$$

B. Global-max policy

This policy attempts to find the evaders using a global perspective. Define $d(x,y)$ to be the Manhattan distance from cell x to cell y, that is, if cells x and y lie in a two-dimensional space with coordinates (x_1,x_2) and (y_1,y_2) respectively then:

$$d(x,y) = \max(|x_1 - y_1|, |x_2 - y_2|)$$

The total reward associated with cell y and pursuer k is given by:

$$R(y,t) = \frac{\rho(y,t+1|Y_t)}{d(y,x_{p_k}(t))}$$

Under the global-max policy, pursuer k finds the cell $\hbar_{p_k}(t+1)$ that gives the highest total reward, that is:

$$\hbar_{p_k}(t+1) = \arg \max_{y \in X} [R(y,t)]$$

where X is the set of all cells in our grid. Having found its target cell, the pursuer moves straight toward it. We call

the line joining a pursuer's current location with its target location the *principal direction.* So this policy seeks to move the pursuers toward the cells with the maximum distance-discounted probability of evader presence. In order to model this, we let $\vec{p}$ be the vector pointing to the principal direction. Thus, in a two-dimensional space, we will have $\vec{p} = x_1\vec{i} + x_2\vec{j}$ for some x_1 and x_2 and $\vec{i}, \vec{j}$ pointing in the increasing column and row directions, respectively. Then, $x_{p_k}(t+1)$ will be the cell that corresponds to a physical translation of cell $x_{p_k}(t)$ by $sign(x_1)$ columns and $sign(x_2)$ rows. For example, if $\vec{p} = 3\vec{i} - 2\vec{j}$ then $x_{p_k}(t+1)$ is the cell that lies one column to the right and one row below $x_{p_k}(t)$.

C. Local-max with no overlap

This is an enhanced version of the local-max policy. In this policy pursuers decide on their next move sequentially. After a pursuer k has made a choice of action then the cells $z \in S(x_{p_k}(t+1))$ are assigned a single cell reward of -1. Hence, for pursuer $k+1$, actions that would result in moving to a region that has a sensing overlap with pursuer k will be heavily penalized. The effect of this is to minimize observation overlap.

D. Global-max with no overlap

This is an enhanced version of the global-max policy that also tries to reduce sensing overlap between pursuers. Under this policy the pursuers choose their target locations exactly like in the global-max policy. The difference, however, is that they do not necessarily move straight toward that location. Instead, they are given three options: move along the principal direction, or move along directions that lie 45° to the left and right of the principal direction. The pursuer makes the choice between the three directions using the local-max with no overlap policy. The 45° angle choice is made to ensure that the two extra directions retain forward momentum toward the target location. To summarize, under this policy, we employ the global-max policy deciding on a target location, and the local-max with no overlap policy deciding on how the pursuer gets to that target location.

E. Global-subregional

This policy is very similar to global-max with no overlap in the way pursuers move toward their target locations. It is much different, though, in the way pursuers select those target locations. We divide the gridworld into subregions that are as big as the sensing area of each pursuer. This is demonstrated in Fig. 4 where we used a 16×16 grid and a 3×3 pursuer sensing region.

Let L denote the set of all these subregions and for each subregion $l \in L$, let $F(l)$ denote the set of all cells x that lie in subregion l. Then the total reward associated with subregion l at time t is given by:

$$R(l,t) = \max_{x \in F(l)} [\rho(x, t+1|Y_t)] \ .$$

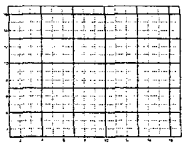

Fig. 4. Dividing the map into subregions

Once the total reward for each subregion is calculated, we assign pursuers to locations in the following manner: Let ℓ be defined to be the subregion with the maximum reward value:

$$\ell = \arg\max_{l \in L} [R(l,t)]$$

and let $C(\ell)$ be the cell in ℓ that has the maximum single cell reward:

$$C(\ell) = \arg\max_{x \in F(\ell)} [\rho(x, t+1|Y_t)] \ .$$

Then ℓ gets assigned to the nearest pursuer with the Manhattan distance of pursuer k from $C(\ell)$ used as the distance metric. Let the identity of the pursuer assigned to subregion ℓ be denoted by I_ℓ. Then:

$$I_\ell = \arg\min_{i \in (1....n)} [d(x_{p_i}, C(\ell))]$$

We then eliminate ℓ from L and reiterate this assignment procedure with the remaining subregions and the other pursuers except I_ℓ until all pursuers have been assigned to locations. Once all assignments are completed, the pursuers move to their target locations in the same manner as in the global-max with no overlap policy.

F. Evasion Policy

The evaders actively try to avoid the pursuers and they use a potential function to try to stay away from each other, from pursuers, and from the inaccessible locations. To enhance their ability to do so, we gave the evaders perfect information concerning the location of pursuers. At time step t evader k is sitting in cell $x_{e_k}(t)$ and can move to any cell $x \in U(x_{e_k}(t))$. Each potential next location is given a cost and the evader uses a greedy policy to select the cell with the lowest cost as its location for the next step. Define the cost associated with cell y and evader k at time t to be:

$$V(y) = \sum_{z \in X} \left[v(z) \, (1/9)^{d(z, x_{e_k}(t))} \right]$$

where the sum is taken over all the cells in the grid and $v(z)$ is given by:

$$v(z) = \begin{cases} 2.0 & \text{if } z \text{ lies within the sensing area of a pursuer} \\ 0.2 & \text{if } z \text{ contains an evader} \\ 1.0 & \text{if } z \text{ contains an obstacle(e.g. boundary wall)} \\ 0 & \text{otherwise} \end{cases}$$

Game Parameters				Median Capture Time				
Grid size	Sensing area size	Num. of pursuers	Num. of evaders	Global-subregional	Local-max no overlap	Global-max no overlap	Local-max with overlap	Global-max with overlap
20 x 20	4 x 4	4	3	191	174	266	253	500+
20 x 20	4 x 4	4	5	214	218	305	383	500+
20 x 20	5 x 5	3	3	162	166	212	345	500+
20 x 20	5 x 5	3	5	214	192	264	456	500+
20 x 20	2 x 2	8	3	531	435	512	1000+	1000+
20 x 20	4 x 4	2	3	1500+	863	1500+	1500+	1500+

TABLE I

SIMULATION RESULTS FOR THE FIVE PURSUIT POLICIES

Evader k moves to the cell $x_{e_k}(t+1)$ that minimizes the received cost:

$$x_{e_k}(t+1) = \arg \min_{y \in U(x_{e_k}(t))} [V(y)]$$

Fig. 5 shows time shots of a game with three evaders avoiding three pursuers that move horizontally towards them. The pursuers in this example are not using any of the pursuit policies described above (i.e. sweeping the state-space from left to right), and the evaders smartly avoid detection.

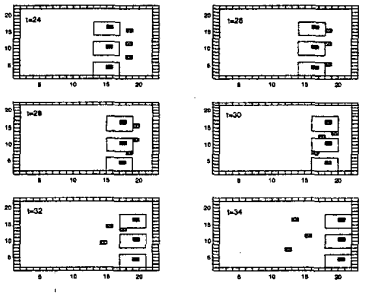

Fig. 5. Evaders escaping when pursuit strategies are not intelligent.

IV. SIMULATIONS

In this section, we evaluate the effectiveness of the pursuit policies proposed in Section III in simulations. We ran simulations in Matlab to compare the performance of the five different pursuit policies, with capture time of all evaders being the performance metric. We did this for six different game settings and for each setting we ran 200 simulations. For each simulation we assigned the player's initial positions randomly ensuring, however, that no evader starts in the sensing area of a pursuer. Finally, the five policies were tested using the same initial conditions for each simulation run.

To save time, the simulation was stopped at 500 simulation steps in the first four game settings, and at 1000 and 1500 steps in the more challenging fifth and sixth settings. To account for this truncation, the median capture time was used as the performance metric of the policies. The results are summarized in Table I.

A. Best performing policies

The results suggest that the global-subregional and the local-max with no overlap policy perform the best among the five suggested pursuit policies, with the local-max with no overlap policy winning in the last two more challenging scenarios. The overall better performance of these two policies can be attributed to two factors:

1) They reduce sensing overlap between pursuers.
2) They do not over-assign pursuers to target locations.

The first point follows from comparing the local-max and global-max policies with overlap to their counterparts with no overlap. It is readily seen that the provision for reducing sensing overlap significantly improved their performance. The second point is demonstrated in Fig. 6(a) through Fig. 6(c). As shown in Fig. 6(a), the global-max with no overlap policy assigns all the pursuers to one specific target area, thus underutilizing resources.

B. Local versus global policies

In the last two game settings we can see that the local-max with no overlap policy outperforms the global-subregional. This is a result of an unfavorable property of global policies: they assign pursuers to target locations of high reward, but they cannot predict if that reward will remain high until the time the pursuer gets there. The local-max policy, however, ensures that the predicted reward will be received on the next time step. In the global-subregional policy, two adjacent subregions l_1 and l_2 of high reward value will be assigned to two different pursuers, p_1 and p_2 respectively. If p_1 happens to be much closer to l_1 than p_2 is to l_2 then p_1 will complete searching l_1, and on the next time step it will be assigned to l_2 as it is the closest pursuer to it. Meanwhile, p_2 was traveling towards l_2 and now will have to be assigned to

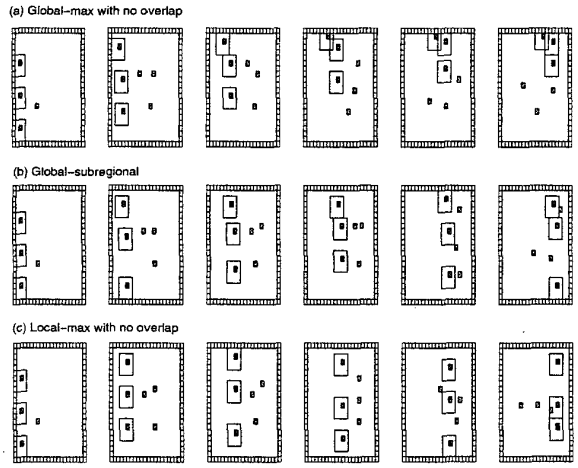

Fig. 6. Three pursuers versus three evaders under different pursuit policies, (a) Global-max with no overlap, (b) Global-subregional, and (c) Local-max with no overlap

another location. This constant reassignment practice results in a resource allocation inefficiency, which accounts for the worse performance of the global-subregional policy.

C. Time costs and other observations

Comparing the algorithms behind the policies, we see that the local-max policies are computationally less expensive. In the case of the global-max and global-subregional policies we need to search over the whole state-space X in order to find the desired target areas. The local-max policy, however, only searches in $U(x_{p_k}(t))$ which is a small subset of X. Furthermore, $U(x_{p_k}(t))$ is of fixed size so the computational complexity does not increase with the dimensions of the grid-world.

Another interesting conclusion can be inferred from the last two game settings in which we have the same total sensing power of 32 cells distributed among eight and two pursuers, respectively. The simulation results indicate that distributing sensing power among a greater number of pursuers significantly decreases capture time.

V. CONCLUSIONS AND FUTURE WORK

A. Conclusions

In this paper, we presented a number of pursuit algorithms for solving complex multi-player pursuit-evasion games. In the games we consider, a team of pursuers try to detect and capture multiple evaders that actively try to avoid capture. It was demonstrated that reducing sensing overlap between pursuers and avoiding over-assignment of pursuers to target locations improve the performance of pursuit policies in terms of capture time. The strategies were heuristic because we do not as yet have theory to support the existence of solutions to multi-player games with partial information. Our analysis, however, will be useful as a basis for developing policy classes for the future "learning" controllers or the study of finite horizon games.

B. Future Work

We expect that incorporating concepts of lookahead horizons and discounts in determining target locations will further improve the performance of our pursuit policies. In developing the heuristic algorithms, we assumed that a pursuer has access to perfect information concerning the sensing data of all other pursuers. As a result, we were able to develop a centralized probability map that was used by pursuers to better organize information and assign target areas. This assumption will be greatly challenged in a real-life scenario with UAVs and UGVs, as in most cases perfect communication between agents is not possible. Sensor network approaches can remedy this limitation and [7] describes a promising framework for adding sensor networks to distributed pursuit-evasion games.

VI. ACKNOWLEDGMENTS

This research was supported by the U.S. Army Research Office under MURI grant DAAD19-02-1-0383 "ACCLIMATE" and DARPA program MICA under contract number F33615-01-C-3150.

VII. REFERENCES

[1] S. LaValle, D. Lin, L. Guibas, J.C. Latombe, and R. Motwani. Finding an unpredictable target in a workspace with obstacles. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 732–742, 1997.

[2] J. Hespanha, H.J. Kim, and S. Sastry. Multiple-agent probabilistic pursuit-evasion games. In *Proc. of 38th IEEE CDC*, pages 2432–2437, Dec. 1999.

[3] H. J. Kim, R. Vidal, O. Shakernia, D. H. Shim, and S. Sastry. A hierarchical approach to probabilistic pursuit-evasion games with unmanned ground and aerial vehicles. In *Proc. of 40th IEEE Conference on Decision and Control*, December 2001.

[4] D. H. Shim. *Hierarchical Control System Synthesis for Rotorcraft-based Unmanned Aerial Vehicles*. PhD thesis, University of California at Berkeley, 2000.

[5] R. Vidal, O. Shakernia, H.J. Kim, D. H. Shim, and S. Sastry. Probabilistic pursuit-evasion games: Theory, implementation and experimental evaluation. *IEEE Transaction on Robotics and Automation*, 18(5):662–670, Oct. 2002.

[6] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002. to appear.

[7] Bruno Sinopoli, Courtney Sharp, Luca Scenato, Shawn Schaffert, and Shankar Sastry. Distributed control application within sensor networks. In *Proc. of the IEEE special issue on distributed sensor networks*, Nov. 2003.