Probabilistic Pursuit-Evasion Games: Theory, Implementation and Experimental Evaluation

René Vidal[†], Omid Shakernia, H. Jin Kim, Hyunchul Shim, Shankar Sastry

Abstract— **We consider the problem of having a team of Unmanned Aerial Vehicles (UAV) and Unmanned Ground Vehicles (UGV) pursue a second team of evaders while concurrently building a map in an unknown environment. We cast the problem in a probabilistic game theoretic framework and consider two computationally feasible greedy pursuit policies: *local-max* and *global-max*. To implement this scenario on real UAVs and UGVs, we propose a distributed hierarchical hybrid system architecture which emphasizes the autonomy of each agent yet allows for coordinated team efforts. We describe the implementation of the architecture on a fleet of UAVs and UGVs, detailing components such as high-level pursuit policy computation, map building and inter-agent communication, and low-level navigation, sensing, and control. We present both simulation and experimental results of real pursuit-evasion games involving our fleet of UAVs and UGVs and evaluate the pursuit policies relating expected capture times to the speed and intelligence of the evaders and the sensing capabilities of the pursuers.**

Keywords— **Multi-robot systems, pursuit-evasion games, multi-agent coordination and control, autonomous vehicles.**

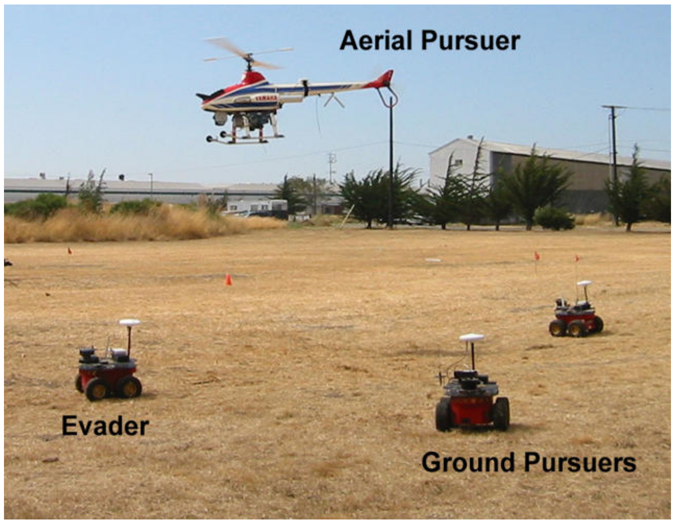

Fig. 1. The Berkeley AeRobot test bed for pursuit-evasion games.

I. Introduction

The BErkeley AeRobot (BEAR) project [1] is a research effort at UC Berkeley that encompasses the disciplines of hybrid systems theory, navigation, control, computer vision, communication, and multi-agent coordination. The goal of our research is to integrate multiple autonomous agents with heterogenous capabilities into a coordinated and intelligent system that is modular, scalable, fault-tolerant, adaptive to changes in task and environment, and able to efficiently perform complex missions. This paper highlights the theory, implementation and evaluation of probabilistic pursuit-evasion games on the BEAR test bed of Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs) shown in Figure 1.

We consider a pursuit-evasion game scenario in which a team of UAVs and UGVs acting as *pursuers* attempts to capture *evaders* within a bounded but unknown environment. We cast the problem in a probabilistic game theoretic framework that combines pursuit-evasion games and map building in a *single* problem, which avoids the conservativeness inherent to classical worst-case approaches. We consider two computationally feasible pursuit policies: *local-max* and *global-max*. We prove that for the global-max policy there exists an upper bound on the expected capture time which depends on the size of the arena, and the number, speed and sensing capabilities of the pursuers.

[†]Corresponding author. The authors are with the Department of EECS, University of California at Berkeley, 301 Cory Hall, Berkeley CA 94720-1774, USA. Phone: (510) 643-2382, Fax: (510) 642-1341, E-mail: {rvidal,omids,jin,hcshim,sastry}@robotics.eecs.berkeley.edu.

In order to implement this pursuit-evasion game scenario on a fleet of UAVs and UGVs, we propose a distributed hierarchical hybrid system architecture that segments the control task into different layers of abstraction: high-level pursuit policy computation, map building and inter-agent communication; and low-level tactical planning, navigation, regulation and sensing. Our architecture is modular and scalable, allowing one to "divide and conquer" a complex large scale system by developing and integrating simpler components. Unlike the traditional *sense-model-plan-act* decomposition, our architecture takes into consideration the dynamics of each agents so that our system can achieve real-time performance.

We evaluate the proposed probabilistic framework and hierarchical architecture through simulation and experimental results involving our fleet of UAVs and UGVs. Using the expected capture time as the performance criterion, we compare the local-max and global-max pursuit policies on numerous situations, varying the speed and intelligence of the evaders and the sensing capabilities of the pursuers. Our experimental results support our theoretical work by showing that the global-max policy outperforms the local-max policy in a realistic situation in which the dynamics of each agent are included and computer vision is used to detect the evaders. Furthermore, our experiments show that the global-max policy is robust to changes in the conditions of the game: even though it is designed for a randomly moving evader, the policy is also successful in catching an intelligent evader.

A. Previous Research in Pursuit-Evasion Games

The classical approach to pursuit-evasion games is to first build a map of the terrain and then play the game in a known environment. For the map building stage, several techniques have been proposed, see e.g. [2] and references therein. Most of them are based on Bayesian estimation and are implemented using the expectation-maximization algorithm [3]. The main problem with these map building techniques is that they are time consuming and computationally expensive, even in the case of simple two dimensional rectilinear environments [4]. On the other hand, most of the literature in pursuit-evasion games, see e.g. [5], [6], [7], [8], [9], [10], assumes worst case motion for the evaders and an accurate map of the environment. In practice, this results in overly conservative pursuit policies if applied to inaccurate maps built from noisy measurements.

In [11] the pursuit-evasion game and map building problems are combined in a single probabilistic framework. The basic scenario considers multiple pursuers trying to capture a single randomly moving evader. In [12] we extended the scenario to consider multiple evaders and proposed a simple vision-based algorithm for evader detection. We also included supervisory agents, such as a helicopter, that can detect evaders but not capture them. In parallel with our theoretical work on pursuit-evasion games, we have been developing a test bed for multi-agent coordination and control. In [13] we presented a real-time control system for regulation and navigation of a UAV. In [12] we presented an architecture for pursuit-evasion games and described the implementations of the navigation, communication and sensing layers. In [14] we presented the implementation of the high-level mission coordination, including the components for pursuit policy computation and map building.

Recent work on pursuit evasion games considers evaders that actively avoid the pursuers, as described in [15] where a dynamic programming solution to a Stackelberg equilibrium of a partial information Markov process is proposed. There has also been work on vision-based pursuit-evasion games, where the pursuers use optical flow to determine the number of moving evaders as well as their position and orientation [16]. Implementation and evaluation of these two techniques is forthcoming.

B. Previous Research in Multi-Robot Systems

The multi-agent pursuit-evasion game scenario considered in this paper fits within the general framework of multi-robot systems. There exists a large body of literature in multi-robot systems addressing problems such as machine learning techniques for multi-agent systems [17], architectures for multi-robot cooperation [18], hybrid algorithms for multi-agent control [19], multi-robot localization [20], distributed sensor fusion [21] and formation control [22].

Also, there have been many applications of multi-robot systems to robot soccer. We refer the reader to [23] and [24] for centralized coordination and control of multiple robots, and to [25], [26] and [27] for completely distributed systems.

II. Pursuit-Evasion Scenario

This section describes the theoretic foundations for probabilistic pursuit-evasion games, including map building, pursuit policies, and evasion policies. We also describe a vision-based algorithm for obstacle and evader detection.

Notation. We denote by $(\Omega, \mathcal{F}, \mathrm{P})$ the relevant *probability space* with Ω the set of all possible events related to the pursuit-evasion game, $\mathcal{F}$ a family of subsets of Ω forming a σ-algebra, and $\mathrm{P} : \mathcal{F} \to [0,1]$ a probability measure on $\mathcal{F}$. Given two sets of events $A, B \in \mathcal{F}$ with $\mathrm{P}(B) \neq 0$, we write $\mathrm{P}(A|B)$ for the *conditional probability of A given B*. Bold face symbols are used to denote random variables.

A. Probabilistic Framework

Consider a finite two-dimensional environment $\mathcal{X}$ with n_c square cells containing an unknown number of fixed obstacles and let $\mathbf{x}_p \subset \mathcal{X}$ ($\mathbf{x}_e \subset \mathcal{X}$) be the set of cells occupied by the n_p pursuers (n_e evaders). Pursuers and evaders are restricted to move to cells in which there is no other pursuer, evader or obstacle.

Each pursuer collects information about $\mathcal{X}$ at discrete time instants $t \in \mathcal{T} \triangleq \{1, 2, \ldots\}$. Each measurement $\mathbf{y}(t)$ is a triple $\{\mathbf{v}(t), \mathbf{e}(t), \mathbf{o}(t)\}$ taking values in a measurement space $\mathcal{Y}$, where $\mathbf{v}(t)$ denotes the measured positions of the pursuers and $\mathbf{e}(t)$ ($\mathbf{o}(t)$) is a set of cells where evaders (obstacles) are detected. We let $\mathcal{Y}^*$ be the set of all finite sequences of elements in $\mathcal{Y}$, and $\mathbf{Y}_t \in \mathcal{Y}^*$ be the sequence of measurements $\{\mathbf{y}(1), \ldots, \mathbf{y}(t)\}$ taken up to time t. In practice, measurements are taken within a certain subset of $\mathcal{X}$: *the visibility region.* We denote the visibility region of pursuer k (evader i) at time t as $V_{p_k}(t)$ ($V_{e_i}(t)$).

Sensor information is assumed to be imperfect. We use a simple sensor model based on the probability of false positives $p \in [0,1]$ and false negatives $q \in [0,1]$ of a pursuer detecting an evader or an obstacle. However, we assume that pursuers have perfect knowledge of their own locations, that is $\mathbf{v}(t) = \mathbf{x}_p(t)$[1].

Capture of an evader is defined as follows: Let $x_{p_k}(t) \in \mathbf{v}(t)$ and $x_{e_i}(t) \in \mathbf{e}(t)$ be the estimated positions of ground pursuer k and evader i at time t, respectively. We say that evader i is captured by ground pursuer k at time t if $x_{e_i}(t) \in V_{p_k}(t)$ and $d(x_{p_k}(t), x_{e_i}(t)) \leq d_m$ where $d(\cdot,\cdot)$ is a metric in $\mathcal{X}$ and d_m is a pre-specified *capture distance.* Captured evaders are removed from the game. The *capture time* of all evaders is defined as $\mathbf{T}^* = \max_{i=1\cdots n_e} \mathbf{T}_i^*$, where $\mathbf{T}_i^*$ is the time instant at which evader i is captured. Notice that aerial pursuers can detect and share information about the positions of evaders, but not capture them.

B. Map Building

We assume that pursuers are able to identify each evader separately and that each evader moves independently of the other evaders. Therefore, without loss of generality, we will assume $n_e = 1$ and omit the subscript identifying

[1]This assumption is unrealistic in general, although valid when GPS is used for pursuer localization and vision is used for evader detection.

the evaders. Let $p_e(x, \tau \mid Y_t)$ be the posterior probability of the evader being in cell x at time τ, given the measurement history $\mathbf{Y}_t = Y_t$. Similarly, let $p_o(x \mid Y_t)$ be the conditional probability of having an obstacle in cell x given Y_t. At each t, pursuers have estimates of the evader and obstacle *maps* $p_e(x, t \mid Y_{t-1})$ and $p_o(x \mid Y_{t-1})$, obtain a new measurement $\mathbf{y}(t)$ and recursively estimate $p_o(x \mid Y_t)$ and $p_e(x, t+1 \mid Y_t)$ in three steps: First, pursuers compute $p_e(x, t \mid Y_t)$ as:

$$\begin{cases} 0 \text{ if } x \in \mathbf{o}(t) \cup \mathbf{v}(t) \setminus \mathbf{e}(t) \text{ or the evader is captured} \\ \alpha p_e(x, t \mid Y_{t-1}) P(e \mid x, v, Y_{t-1}) \qquad \text{otherwise,} \end{cases} \tag{1}$$

where α is a normalizing constant independent of x, and $P(e|x, v, Y_{t-1}) = P(\mathbf{e}(t) = e|\mathbf{x}_e(t) = x, \mathbf{v}(t) = v, \mathbf{Y}_{t-1} = Y_{t-1})$

$$= \begin{cases} 0 & x \in \mathbf{v}(t) \\ p^{k_1}(1-p)^{k_2} q^{k_3}(1-q)^{k_4} & \text{otherwise.} \end{cases} \tag{2}$$

Here, for each x, k_1 is the number of false positives, k_2 is the number of true negatives, k_3 is the number of false negatives, and k_4 is the number of true positives. Recall that p and q are the probability of the sensor reporting false positives and false negatives, respectively.

Second, pursuers compute the obstacle map $p_o(x|Y_t)$ as:

$$\begin{cases} \frac{(1-q)p_o(x|Y_{t-1})}{(1-q)p_o(x|Y_{t-1}) + p(1-p_o(x|Y_{t-1}))} & x \in V_p(t) \cap \mathbf{o}(t) \\ \frac{q p_o(x|Y_{t-1})}{q p_o(x|Y_{t-1}) + (1-p)(1-p_o(x|Y_{t-1}))} & x \in V_p(t) \setminus \mathbf{o}(t) \\ 1 & x \in \mathbf{v}(t) \cap \mathbf{o}(t) \\ 0 & x \in \mathbf{v}(t) \setminus \mathbf{o}(t) \\ p_o(x \mid Y_{t-1}) & \text{otherwise,} \end{cases} \tag{3}$$

where $V_p(t) = \cup_{k=1}^{n_p} V_{p_k}(t)$.

Finally, in order to compute $p_e(x, t+1|Y_t)$, pursuers assume a Markov model for the motion of the evader which is determined by the probability $\rho \in [0, 1/8]$ that the evader moves to an unoccupied cell in $\mathcal{A}(x)$, where $\mathcal{A}(x)$ is the set of (up to eight) cells adjacent to x. The evader map $p_e(x, t+1|Y_t)$ is updated as:

$$(1-|\mathcal{A}(x)|\rho) p_e(x, t|Y_t) + \rho(1-p_o(x|Y_t)) \sum_{\bar{x} \in \mathcal{A}(x)} p_e(\bar{x}, t|Y_t). \tag{4}$$

C. Pursuit Policies

Given the measurement history $\mathbf{Y}_t$, the pursuers need to decide where to move at the next time instant. Let $\mathbf{u}(t) \triangleq [\mathbf{u}_1(t), \ldots, \mathbf{u}_{n_p}(t)]$ be the desired position of the pursuers at time t. Since two pursuers must not occupy the same cell, $\mathbf{u}(t)$ is an element of the control action space $\mathcal{U} \triangleq \left\{\{v_1, \ldots, v_{n_p}\} : v_i \in \mathcal{X}, v_i \neq v_j \text{ for } i \neq j\right\}$. We define a *pursuit policy* as the random function $\mathbf{g} : \mathcal{Y}^* \to \mathcal{U}$:

$$\mathbf{g}(\mathbf{Y}_t) \triangleq \mathbf{u}(t+1) = [\mathbf{u}_1(t+1), \cdots, \mathbf{u}_{n_p}(t+1)]. \tag{5}$$

We measure the performance of a specific pursuit policy $\bar{g}$ by the expected capture time $E_{\bar{g}}[\mathbf{T}^*] \triangleq E[\mathbf{T}^*|\mathbf{g} = \bar{g}]$. Since the dependence of $E_{\bar{g}}[\mathbf{T}^*]$ on the pursuit policy $\bar{g}$ is in general very complex [11], instead of finding the optimal policy that minimizes $E_{\bar{g}}[\mathbf{T}^*]$, we look for efficiently computable sub-optimal policies with good performance. To this end, we first introduce the notion of a persistent pursuit policy [11] and show that it guarantees a certain degree of success for the pursuers. We then present two computationally efficient greedy policies and show that one of them satisfies the persistence property, given certain assumptions on the distribution of the obstacles and the sensing models.

C.1 Persistent on the average pursuit policies

A specific pursuit policy $\bar{g} : \mathcal{Y}^* \to \mathcal{U}$ is said to be *persistent on the average* if there is an integer T and some $\epsilon > 0$ such that, for each $t \in \mathcal{T}$, the conditional probability of capturing an evader on the set of T consecutive time instants starting at t is greater than or equal to ϵ, i.e.,

$$\mathrm{P}(\mathbf{T}^* \in \{t, t+1, \ldots, t+T-1\} \mid \mathbf{g} = \bar{g}, \mathbf{T}^* \geq t) \geq \epsilon. \tag{6}$$

We call T the *period of persistence*. Persistent on the average pursuit policies satisfy the following [11]:

Lemma 1: If $\bar{g} : \mathcal{Y}^* \to \mathcal{U}$ is a persistent on the average pursuit policy with period T, then $\mathrm{P}(\mathbf{T}^* < \infty \mid \mathbf{g} = \bar{g}) = 1$, and $E_{\bar{g}}[\mathbf{T}^*] \leq T\epsilon^{-1}$, with ϵ as in (6).

Lemma 1 shows that for a pursuit policy which is persistent on the average, the probability of capturing the evaders in finite time is equal to one. Moreover, Lemma 1 gives a simple upper bound on the expected capture time. The following lemma (proved in [28]) gives a sufficient condition for a policy to be persistent on the average.

Lemma 2: Let $\mathcal{Y}_t^{\neg\mathrm{fnd}}$, be the set of all sequences of measurements of length t, associated with an evader not being found up to time t. A sufficient condition for a pursuit policy $\bar{g}$ to be persistent on the average with period T is the existence of some $\delta > 0$ such that, for each $t \in \mathcal{T}$ and each $Y \in \mathcal{Y}_{t+T-2}^{\neg\mathrm{fnd}}$, there is some $\tau \in \{t-1, t, \ldots, t+T-2\}$ for which $\mathrm{P}(\mathbf{T}^* = \tau + 1|\mathbf{Y}_\tau = Y_\tau, \mathbf{g} = \bar{g}) \geq \delta$. In this case (6) holds with:

$$\epsilon \triangleq \begin{cases} \frac{1}{T}(1 - \frac{1}{T})^{T-1} & \delta \geq \frac{1}{T} \\ \delta(1-\delta)^{T-1} & \delta < \frac{1}{T}. \end{cases} \tag{7}$$

C.2 Admissible policies, obstacle density & sensor models

In order to apply Lemma 2 to a specific pursuit policy, we will need some extra assumptions on the dynamics of the pursuers, the distribution of obstacles, and the sensor models used for evader detection.

First, we restrict our attention to pursuit policies that respect the dynamics of each vehicle. Let $\mathcal{U}_{p_k}(x_{p_k}) \subset \mathcal{X}$ be the set of cells that pursuer k can reach from cell x_{p_k} in one time step, provided that those cells are empty, and let $\mathcal{U}(x_p) \triangleq \prod_{k=1}^{n_p} \mathcal{U}_{p_k}(x_{p_k})$[2]. We say that a pursuit policy $\bar{g}$ is *admissible* if for every sequence of measurements $Y \in \mathcal{Y}^*$ we have $\bar{g}(Y) \in \mathcal{U}(x_p)$, where $x_p = (x_{p1}, \ldots, x_{p_{n_p}})$ are the pursuers' positions specified by the last measurement in Y.

Next, we assume that the density of obstacles in the environment is small enough so that any cell in $\mathcal{X}$ can be reached in a finite amount of time. More formally:

[2]The one-step reachable set $\mathcal{U}_{p_k}(x_{p_k})$ can be computed off-line as a parametric function of x_{p_k}, using polynomial time algorithms based on robust semi-definite programming as shown in [29].

Assumption 1: For any $v_o, v_f \in \mathcal{X}$ there exists a finite sequence $\{x_p(0), \ldots, x_p(t) : v_o = x_p(0), v_f = x_p(t), t \in \mathcal{T}\}$ such that $x_p(\tau) \in \mathcal{U}_p(x_p(\tau-1))$ for each $\tau \in \mathcal{T}$.

Finally, we assume that in a single time step, the conditional probability of the evader being at a cell $x \in \mathcal{X}$ does not decay by more than a certain amount, unless one pursuer reaches x—in which case the probability of the evader being at x may decay to zero if the evader is not there—or if it is possible to conclude from the measured data that an obstacle is at x with probability one. Such an assumption holds for most sensor models.

Assumption 2: There is a positive constant $\gamma \leq 1$ such that for any sequence $Y_t \in \mathcal{Y}_t^{\neg\text{fnd}}$ of $t \in \mathcal{T}$ measurements for which the evader was not captured,

$$p_e(x, t+1|Y_t) \geq \gamma p_e(x, t|Y_{t-1}), \tag{8}$$

for any $x \in \mathcal{X}$ for which (i) x is not in the list of pursuers positions specified by the last measurement in Y_t and (ii) the probability of there being an obstacle in any given location given the measurements up to time t is strictly less than 1 for any pursuit policy $\bar{g}$.

C.3 Greedy policies

Given the assumptions in the previous section, we now focus on finding efficiently computable sub-optimal pursuit policies. We consider the following *greedy* policies: *local-max* and *global-max*. Both policies try to maximize the probability of capturing an evader at the next time instant; the difference being that local-max searches only one-step reachable cells, while global-max searches the entire map.

Local-max Policy. Under this policy, pursuer k moves to the cell in the one-step reachable set with the highest probability of containing an evader over all the evader maps, that is:

$$\mathbf{u}_k(t+1) = \underset{x \in \mathcal{U}(x_{p_k}(t))}{\operatorname{argmax}} \max_{i=\{1 \ldots n_e\}} p_{e_i}(x, t+1|Y_t),$$

where $p_{e_i}(x, t+1|Y_t)$ represents the probability of evader i being in cell x at time $t+1$ given the measurements Y_t.

Notice that the local-max policy is advantageous in scalability, since it assigns more importance to local measurements by searching only in $\mathcal{U}(x_{p_k})$ regardless of the size of the environment $\mathcal{X}$. This policy is computationally efficient, and can be computed independently by each pursuer in a decentralized pursuit-evasion game. However, it can be shown that in general the local-max policy is not persistent on the average.

Global-max Policy. The *global-max* policy searches over the entire map in order to compute the control that maximizes the probability of capturing an evader. Therefore, it is more computationally intensive and does not scale as well as the local-max policy with the size of $\mathcal{X}$. However, as we will show below, it has the nice property that it is persistent on the average.

Take an arbitrary sequence of measurements $Y \in \mathcal{Y}^*$, and compute the cell in the map with the maximum probability of having an evader:

$$x^* \triangleq \underset{x \in \mathcal{X}}{\operatorname{argmax}} \max_{i \in \{1, \ldots, n_e\}} p_{e_i}(x, t+1, | Y_t). \tag{9}$$

Next define the desired positions of the pursuers as:

$$x_p^* \triangleq \underset{x_{p_1}, \ldots, x_{p_{n_k}} \in \mathcal{U}}{\operatorname{argmax}} \max_{i \in \{1, \ldots, n_e\}} \sum_{k=1}^{n_p} p_{e_i}(x_{p_k}, t \mid Y_t) \tag{10}$$

Now define the global-max pursuit policy $g(Y)$ as:

$$g(Y) \triangleq \operatorname{nav}(x_p^*, k^*, Y) \tag{11}$$

where $k^* \in \{1, \ldots, n_p\}$ is the integer for which $x^* = x^*_{p_{k^*}}$. Here nav : $\mathcal{U} \times \{1, \ldots, n_p\} \times \mathcal{Y}^* \to \mathcal{U}$ is an underlying "navigation policy" which takes a list of desired positions for the pursuers x_p^*, together with measurements Y_t and produces a position reachable in a single time step that is "one step closer" to x_p^*, or concludes that there is an obstacle at x_p^*.

Theorem 1: If assumptions 1 and 2 hold, then the global-max policy $g : \mathcal{Y}^* \to \mathcal{U}$ is an admissible pursuit policy which is persistent on the average with period $T \triangleq d$, where d is the maximum number of steps needed to travel from one cell to any other. Moreover $\mathrm{P}(\mathbf{T}^* < \infty \mid \mathbf{g} = g) = 1$, and $E_g[\mathbf{T}^*] \leq T\epsilon^{-1}$, with ϵ as in (7) and $\delta = \frac{\gamma^d}{n_c}$ in Lemma 2.

Proof: By definition of the navigation policy, g is admissible. In order to prove that g is persistent on the average, we show that the hypotheses of Lemma 2 hold. From the definition of x^* in (9) we have that for any $t \in \mathcal{T}$:

$$\max_{i \in \{1, \ldots, n_e\}} p_{e_i}(x^*, t \mid Y_t) \geq \frac{1}{n_c},$$

where n_c is the number of cells in $\mathcal{X}$. Now since a pursuer takes at most d steps to reach x^*, by following policy g there must be some $\tau \in \{t, \ldots, t+T-2\}$ with $T = d$ such that there is a pursuer just 1 step away from x^*. Consider such a time τ and set $x_p = g(Y_\tau)$. Therefore, the conditional probability of finding an evader at time $\tau+1$ is given by $h_g(Y_\tau) \triangleq$

$$\mathrm{P}(\mathbf{T}^* = \tau+1 | \mathbf{Y}_\tau = Y_\tau, \mathbf{g} = g) = \max_{i \in \{1, \ldots, n_e\}} \sum_{k=1}^{n_p} p_{e_i}(x_{p_k}, \tau | Y_\tau).$$

By Assumption 2 and the fact that it takes at most d steps to reach x^* we have:

$$h_g(Y_\tau) \geq \max_{i \in \{1, \ldots, n_e\}} p_{e_i}(x^*, \tau \mid Y_\tau) \geq \delta \triangleq \frac{\gamma^d}{n_c} > 0.$$

Applying Lemmas 1 and 2 finishes the proof. ■

D. Intelligent Evasion

Even though the pursuit policies described in the previous section are designed to work for a randomly moving evader only, in Section IV we will apply these policies to the case of an intelligent evader. We allow an intelligent evader to build a map of obstacles and pursuers and to employ either a local-min or global-min policy so as to minimize the probability of being captured. The local-min and global-min evasion policies are defined similarly to the local-max and global-max pursuit policies.

E. Vision-based detection of obstacles and evaders

Assume that the pursuers are equipped with a camera to sense the environment. We show how to estimate the position of obstacles and evaders from their observed positions in the image plane, the pose (rotation and translation) of the camera and that of the pursuer. We define the coordinate frames: (a) Inertial frame, (b) UAV frame, (c) Camera base, (d) Camera head, and (e) UGV frame, and let $g_{ij} \triangleq (R_{ij}, p_{ij}) \in SE(3)$ be the relative pose of frame i with respect to frame j. Also, let $\hat{w} \in so(3)$ be the skew symmetric matrix associated with axis $w \in \mathbb{R}^3$ and (e_1, e_2, e_3) be the usual basis for $\mathbb{R}^3$. If the observer is a UAV, then $g_{ab} = (\exp(\hat{e_3}\psi)\exp(\hat{e_2}\theta)\exp(\hat{e_1}\phi), p_{ab})$, where (ψ, θ, ϕ) are the estimates of the yaw, pitch and roll angles of the helicopter and $p_{ab} \in \mathbb{R}^3$ is the estimate of its position. $g_{bc} = (R_{bc}, p_{bc})$ is a predefined (known) transformation and $g_{cd} = (\exp(\hat{e_2}\alpha)\exp(\hat{e_1}\beta), 0)$, where (α, β) are the estimates of the pan and tilt angles of the camera. Then, the orientation of the camera head with respect to the fixed inertial frame is then given by:

$$(R_{ad}, p_{ad}) = (R_{ab}R_{bc}R_{cd}, R_{ab}p_{bc} + p_{ab}). \tag{12}$$

Let $\mathbf{x}$ be the estimate of the position of an obstacle (evader) in the image plane. Then its 3D position is obtained as:

$$q = \frac{(z_0 - e_3^T p_{ad})}{e_3^T R_{ad} A^{-1}\mathbf{x}} R_{ad} A^{-1}\mathbf{x} + p_{ad} \tag{13}$$

where z_0 is the (fixed) z-coordinate of the evader on the ground, assuming a flat terrain, and $A \in \mathbb{R}^{3\times 3}$ is the camera calibration matrix.

If obstacles (evaders) are being observed by a ground pursuer, equation (13) can still be applied with minor changes. Replace frame (b) by frame (e), the UGV frame. Then $R_{ae} = \exp(\hat{z}\gamma)$, where γ is the estimate of the heading of the UGV, p_{ae} is the estimate of the position of the observer and g_{ec} is also a predefined (fixed) transformation.

The vision system is also used to estimate the visibility region of each vehicle. For a ground pursuer or evader, the visibility region is defined as the trapezoid whose vertices are computed from equation (13) applied to the vertices of a fixed rectangle located below the horizon on the image plane. For an aerial pursuer, since the camera is pointing down, the rectangle on the image plane is chosen as the whole image, which results in an approximately rectangular visibility region.

III. System Architecture

In order to implement the pursuit-evasion game scenario on real UAVs and UGVs, we propose a hierarchical hybrid system architecture that segments the control of each agent into different layers of abstraction as shown in Figure 2. The different layers allow for the same high-level intelligent control strategies to be applicable to both UAVs and UGVs. By abstracting away the details of sensing and control of each agent, we gain the interoperability of a unified framework for high-level intelligent pursuit policies across all platforms.

This section gives an overview of the different layers of abstraction of our system architecture and some details about the implementation on our fleet of UAVs and UGVs. Our architecture design was inspired by the architectures of Automated Highway Systems [30], Air Traffic Management Systems [31], and Flight Management Systems [32].

A. High-Level: Strategy Planner and Map Builder

The *Strategy Planner* is responsible for the high-level intelligent control of the vehicles, i.e., the pursuit policy computation described in Section II-C. It maintains a state-space of the system useful for mission planning and tasks the agents according to mission objectives.

The *Map Builder* gathers sensor information from each vehicle and computes probabilistic maps with the locations of obstacles and evaders as described in Section II-B.

B. Low-Level: Tactical Planner, Regulation and Sensing

The *Tactical Planner* uses the state information maintained by the strategy planner for controling the motion of each vehicle. It converts strategic plans into a sequence of way-points or flight modes which are used by the *Trajectory Planner* to produce a realizable and safe trajectory based on a dynamic model of the vehicle and safety routines such as obstacle avoidance. The final trajectory is sent to the *Regulation Layer*, which performs the real-time control of the vehicle along the specified trajectory.

Each vehicle makes observations about the environment using a vision system and about its state using a variety of sensors for position and orientation. Sensor fusion techniques are used to improve the quality of the measurements.

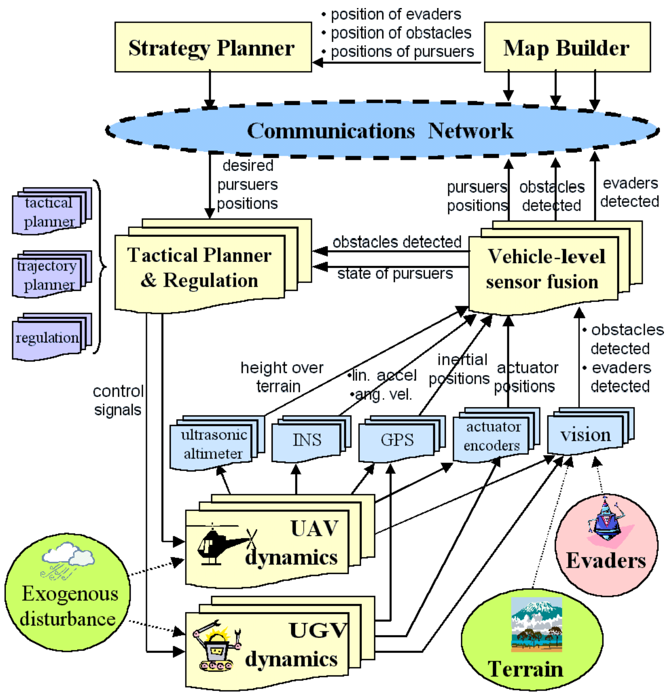

Fig. 2. System Architecture: Strategy planning and map building are implemented in MATLAB and run in a laptop which is also used for visualization. Tactical planning, regulation and sensing are implemented in C++ and run in the UAV or UGV computers.

C. Implementation of High-Level

We implemented the *strategy planner* and *map builder* in a MATLAB/Simulink environment as part of a unified platform on which to conduct both simulations and experiments. Furthermore, we used a TCP interface to connect the MATLAB-based strategy planner and map builder with the UAVs and UGVs through the wireless LAN.

With this unified platform we are able to seamlessly combine experiments and simulations. In simulation mode, the strategy planner sends control commands over TCP to a UAV simulator obtained from system identification [13] and to a UGV simulator. Visibility regions are simulated according to the state variables of each vehicle, and the detection of evaders and obstacles is simulated with probabilistic sensor models. In experiment mode, the *same* strategy planner sends commands over TCP to the actual UAVs and UGVs, while the *same* map builder receives vehicle locations from the GPS/INS, and visibility region and locations of obstacles and evaders from the vision system.

D. Implementation of Low-Level

Our UAV fleet consists of custom-designed UAVs based on Yamaha R-50 and R-MAX industrial helicopters. The trajectory planner and regulation layers are implemented in C on an embedded PC running the QNX real-time OS. The low level controller has a TCP interface with which it asynchronously receives desired setpoints from the high level strategic planner, and reports its current position. The vision system used to detect obstacles and evaders is implemented in C++ on a second PC running Linux. See [13] and [33] for further details.

Our UGV fleet consists of ActivMedia Pioneer 2-AT all-terrain ground robots. The tactical/trajectory planner and regulation layer run on a microcontroller, while the vision system runs on a PC running Linux. See [12] for details.

UAVs and UGVs share the following components for sensing and communication: IEEE 802.11b wireless LAN connectivity, differential GPS, a PC104 Pentium 233MHz-based PC running Linux, and a color-tracking vision system. All these components are described in detail in [12].

IV. Simulation and Experimental Results

In this section, we present both simulation and experimental results of real pursuit-evasion games on our fleet of UAVs and UGVs using the unified experiment/simulation platform described in Section III-C. The experiments are designed to evaluate the proposed pursuit policies on numerous realistic situations, relating the expected capture time to the speed and intelligence of the evaders and the sensing capabilities of the pursuers[3].

A. Simulation Results

Table I presents mean capture time of 10 simulations between 3 pursuers and 1 evader with random initial conditions. Simulations 1–4 evaluate the performance of the two pursuit policies against a randomly moving evader for two types of visibility regions: An omni-directional view S_{p_k}[4] and a trapezoidal view T_{p_k}[5]. Simulations 5–8 evaluate the performance of the global-max policy with trapezoidal view for different speeds and levels of intelligence of the evader.

TABLE I
Simulation Results

Exp	Purs. Policy	Purs. Speed	Evad. Policy	Evad. Speed	Visib. Region	Capt. Time
1	L-max	0.3	Rand	0.3	Omni	279s
2	L-max	0.3	Rand	0.3	Trap	184s
3	G-max	0.3	Rand	0.3	Omni	86s
4	G-max	0.3	Rand	0.3	Trap	67s
5	G-max	0.3	Rand	0.5	Trap	56s
6	G-max	0.3	Rand	0.1	Trap	92s
7	G-max	0.3	G-min	0.1	Trap	151s
8	G-max	0.3	G-min	0.5	Trap	168s

B. Experimental Results

Table II presents results of real experiments between 3 UGV pursuers and 1 UGV evader. Figure 3 shows the evolution of Experiment 1 through photographs and corresponding snapshots created by the map builder. The darker cells in the map represent regions with higher probability of having an evader. Figures 4 and 5 show the map building snapshots for Experiments 2 and 3, respectively.

TABLE II
Experimental Results

Exp	Purs. Policy	Purs. Speed	Evad. Policy	Evad. Speed	Visib. Region	Capt. Time
1	G-max	0.3	Rand	0.1	Omni	105s
2	G-max	0.3	Rand	0.1	Trap	42s
3	G-max	0.3	Rand	0.5	Trap	37s

Figure 6 shows snapshots from Experiment 4: A game with 1 UAV and 2 UGV pursuers and 1 evader. The game parameters were similar to those in Table II: Pursuer speed was 0.3 m/s, evader speed was 0.1 m/s, the evader moved randomly, pursuers had trapezoidal visibility regions and followed the global-max policy. The capture time was 30s.

C. Discussion of Simulation and Experimental Results

Capture Time vs. Visibility Region: Simulations 1–4 in Table I and experiments 1–3 in Table II show that, regardless of pursuit policy, pursuers with trapezoidal vision outperform those with omni-directional vision. Even though at a given time instant both visibility regions cover approximately the same number of cells, a pursuer with a trapezoidal view can change its heading, thus covering many more *new* cells than a pursuer with an omni-directional view. This agrees with natural predator/prey systems.

[3]All the experiments are performed in a $20m \times 20m$ environment with $1m \times 1m$ square cells, $p = q = 0.1$ and $d_m = 1.5m$.

[4]$S_{p_k}(t)$ is a square of side $5m$, centered at $x_{p_k}(t)$.

[5]$T_{p_k}(t) \triangleq \triangle(x_{p_k}(t), 45^\circ, 7m) \backslash \triangle(x_{p_k}(t), 45^\circ, 1m)$, where $\triangle(x, \theta, h)$ denotes an isosceles triangle with vertex x, height h and angle θ.

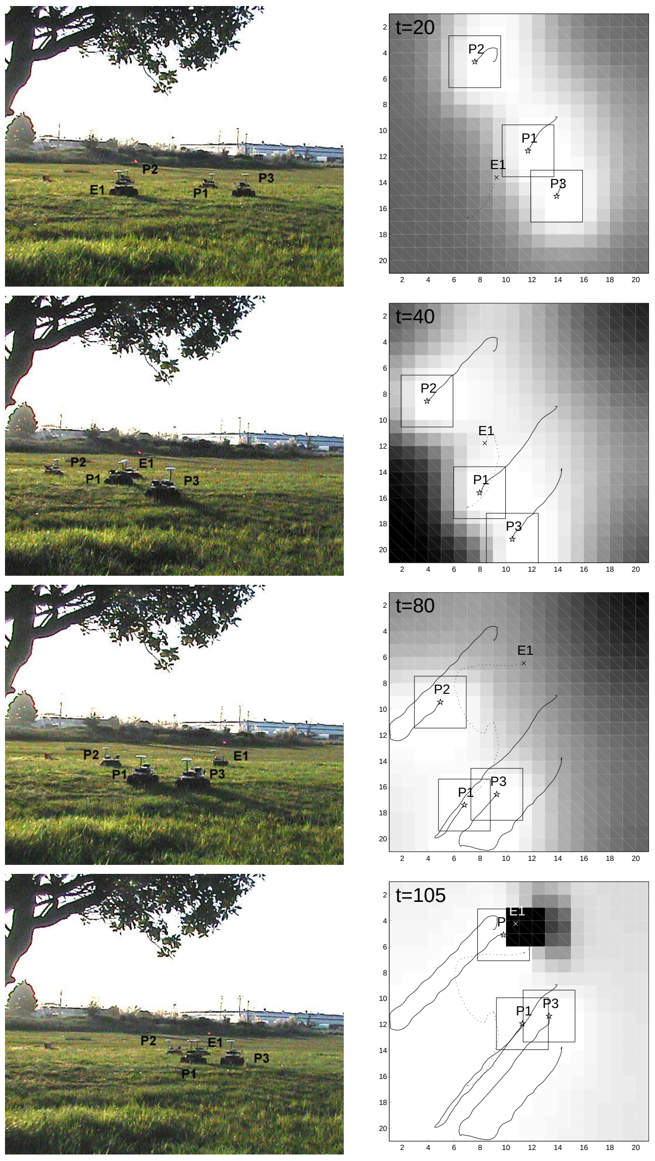

Fig. 3. Experiment 1: An actual game between 3 UGV pursuers and 1 UGV evader. The pursuers P1, P2 and P3 ($\star$) move at 0.3 m/s and use the global-max policy with an omni-directional visibility region. The evader E1 moves randomly at 0.1 m/s.

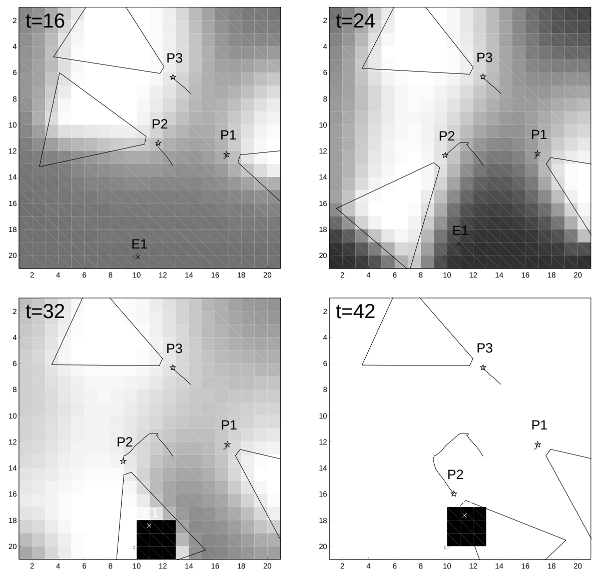

Fig. 4. Experiment 2: Three UGV pursuers vs. a *slow* UGV evader.

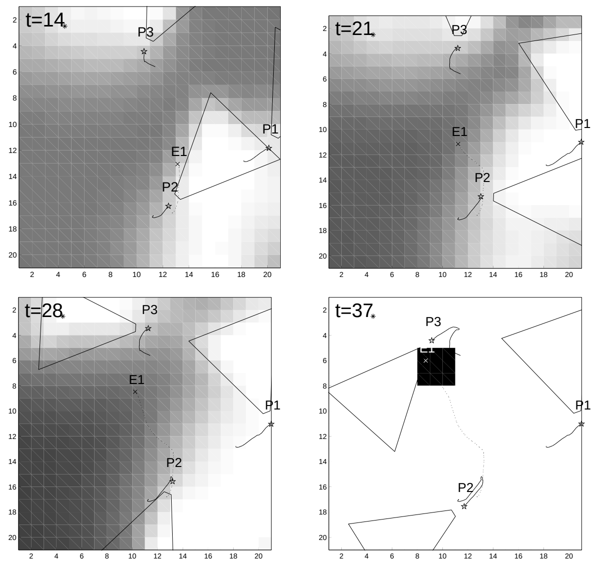

Fig. 5. Experiment 3: Three UGV pursuers vs. a *fast* UGV evader.

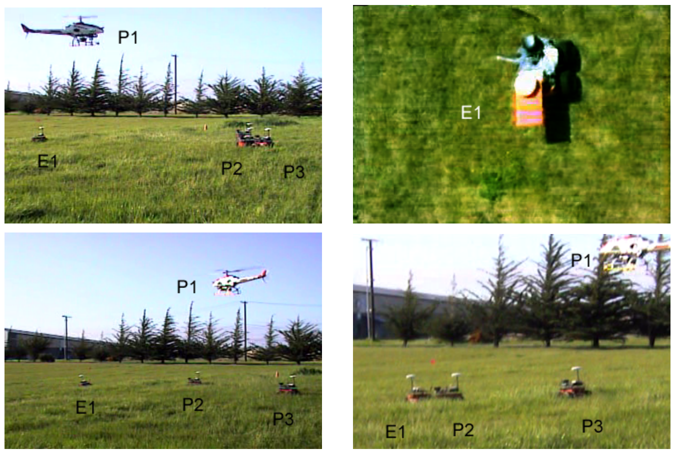

Fig. 6. Experiment 4: 1 UAV and 2 UGV pursuers vs. 1 UGV evader. Clockwise from top left: Initial configuration, evader on the left. UAV pursuer detects evader. UGV pursuers head towards global-max. A UGV pursuer captures evader.

Capture Time vs. Pursuit Policy: Simulations 1–4 in Table I show that the global-max policy generally outperforms the local-max policy. This is expected due to the fact that the global-max policy is persistent on average, while the local-max policy is not.

Capture Time vs. Evasion Policy: Simulations 5–8 in Table I evaluate the global-max pursuit policy against an evader following either a random or a global-min evasion policy. Since the global-max policy is designed for a randomly moving evader, there is no guarantee that the expected capture time will be finite for the case of an intelligent evader. We conclude from the simulations that it takes longer to capture an intelligent evader than a randomly moving one. Also, for a fast evader it takes 300% longer to capture an intelligent one than a randomly moving one, while for a slow evader it takes only 64% longer.

Capture Time vs. Evader Speed: Simulations 5 and 6 in Table I show that it takes longer to capture a slightly faster random evader than a slower random evader. This is because a faster random evader visits more cells in the map, increasing the chances of being detected. This argument can be applied to Figure 5: The higher speed of E1 allows it to move away from the visibility region of P2 for $t \in [0, 14]$, but E1 soon moves into the visibility region of P3 and is quickly captured.

UAV pursuer vs. UGV pursuer: Simulation results in [34] and Experiment 4 show that the local-max policy has a similar performance with either a UAV or UGV pursuer, while the global-max policy performs better with a UAV pursuer.

V. Conclusions

We presented a probabilistic approach to pursuit-evasion games involving UAVs and UGVs. We considered two computationally feasible greedy pursuit policies: *local-max* and *global-max*. We proved that for the global-max policy there exists an upper bound on the expected capture time which depends on the size of the arena and the number, speed and sensing capabilities of the pursuers. Next, we presented an implemention of the scenario on a fleet of UAVs and UGVs based on a hierarchical hybrid system architecture. Finally, we presented several experiments, evaluating the performance of the proposed pursuit policies with respect to speed and intelligence of the evaders and sensing capabilities of the pursuers. Our results show that the global-max pursuit policy outperforms the local-max policy in a realistic situation in which the dynamics of each agent are included and computer vision is used to detect the evaders.

Acknowledgment

The authors thank João Hespanha, Cory Sharp and Shahid Rashid for their contributions to this work. We also thank the support of ONR grants N00014-97-1-0946 and N00014-00-1-0621, and ARO MURI grant DAAH04-96-1-0341.

References

[1] BErkeley Aerial Robot (BEAR) Project homepage, “`http://robotics.eecs.berkeley.edu/bear`,” .

[2] S. Thrun, W. Burgard, and D. Fox, “A probabilistic approach to concurrent mapping and localization for mobile robots,” *Mach. Learning & Autonomous Robots*, vol. 31, no. 5, pp. 1–25, 1998.

[3] A. Dempster, N. Laird, and D. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” *Journal of the Royal Statistical Society*, vol. 39, no. B, pp. 1–38, 1977.

[4] X. Deng, T. Kameda, and C. Papadimitriou, “How to learn an unknown environment I: The rectilinear case,” *Journal of the ACM*, vol. 45, no. 2, pp. 215–245, March 1998.

[5] R. Isaacs, *Differential Games*, John Wiley & Sons, 1965.

[6] T. Başar and G. Olsder, *Dynamic Noncooperative Game Theory*, Number 23. SIAM, 2nd edition, 1999.

[7] 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 ICRA*, 1997, vol. 1, pp. 732–742.

[8] S. LaValle and J. Hinrichsen, “Visibility-based pursuit-evasion: an extension to curved environments,” in *Proc. of IEEE Conf. on Robotics and Automation*, 1999, pp. 1677–1682.

[9] I. Suzuki and M. Yamashita, “Searching for a mobile intruder in a polygonal region,” *SIAM Journal on Computing*, vol. 21, no. 5, pp. 863–888, Oct. 1992.

[10] L. Stephens and M. Merx, “The effect of agent control strategy on the performance of a DAI pursuit problem,” in *Proceedings of the 1990 Distributed AI Workshop*, 1990.

[11] J. Hespanha, H. Kim, and S. Sastry, “Multiple-agent probabilistic pursuit-evasion games,” in *Proc. of 38th IEEE Conf. on Decision and Control*, 1999, vol. 3, pp. 2432–2437.

[12] R. Vidal, S. Rashid, C. Sharp, O. Shakernia, J. Kim, and S. Sastry, “Pursuit-evasion games with unmanned ground and aerial vehicles,” in *Proc. of IEEE ICRA*, 2001, vol. 3, pp. 2948–2955.

[13] H. Shim, H.J. Kim, and S. Sastry, “Hierarchical control system synthesis for rotorcraft-based unmanned aerial vehicles,” in *Proc. AIAA Conf. on Guidance, Navigation & Control*, 2000.

[14] J. Kim, R. Vidal, H. Shim, O. Shakernia, and S. Sastry, “A hierarchical approach to probabilistic pursuit-evasion games with unmanned ground and aerial vehicles,” in *Proc. of 40th IEEE Conf. on Decision and Control*, 2001, pp. 634–639.

[15] J. Hespanha, M. Prandini, and S. Sastry, “Probabilistic pursuit-evasion games: a one-step Nash approach,” in *Proc. of 39th IEEE Conf. on Decision and Control*, 2000, pp. 2272–2277.

[16] R. Vidal and S. Sastry, “Vision based detection of autonomous vehicles for pursuit-evasion games,” in *Proc. of IFAC World Congress on Automatic Control*, 2002, (To appear).

[17] P. Stone and M. Veloso, “Multiagent systems: A survey from a machine learning perspective,” *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, 2000.

[18] L. E. Parker, “Alliance: An architecture for fault-tolerent multi-robot cooperation,” *IEEE Transactions on Robotics and Automation*, vol. 14, no. 2, pp. 220–240, 1998.

[19] A. Timofeev, F. Kolushev, and A. Bogdanov, “Hybrid algorithms of multi-agent control of mobile robots,” *Int. Joint Conference on Neural Networks*, vol. 6, pp. 4115–4118, 1999.

[20] S. Roumeliotis and G. Bekey, “Collective localization: a distributed Kalman filter approach to localization of groups of mobile robots,” in *Proc. of IEEE ICRA*, 2000, pp. 2958–2965.

[21] A Stroupe, M. Martin, and T. Balch, “Distributed sensor fusion for object position estimation by multi-robot systems,” in *Proc. IEEE Conf. on Robotics and Automation*, 2001, pp. 1092–1098.

[22] J. Desai, V. Kumar, and J. Ostrowski, “Control of changes in formation for a team of mobile robots,” in *Proc. of IEEE Conf. on Robotics and Automation*, 1999, pp. 1556–1561.

[23] K. Han and M. Veloso, “Perception, reasoning and learning of multiple agent systems for robot soccer,” in *Proc. of IEEE Conf. on Robotics and Automation*, 1998, vol. 4, pp. 3510–3515.

[24] T-Y Kuc, I-J Lee, and S-M Baek, “Perception, reasoning and learning of multiple agent systems for robot soccer,” in *IEEE Int. Conf. on Systems, Man, & Cybernetics*, 1999, pp. 728–733.

[25] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara, “Robocup: A challenging problem for AI and robotics,” in *RoboCup-97: Robot Soccer World Cup I*. 1998, pp. 1–19, Springer, Lecture Notes in Artificial Intelligence, no. 1395.

[26] H. Kitano, M. Asada, I. Noda, and H. Matsubara, “Robocup: Robot world cup,” *IEEE Robotics and Automation Magazine*, vol. 5, no. 3, pp. 30–36, 1998.

[27] M. Asada, E. Uchibe, and K. Hosoda, “Cooperative behavior acquisition for mobile robots in dynamically changing real worlds via vision-based reinforcement learning and development,” *Artificial Intelligence*, vol. 110, pp. 275–292, 1999.

[28] J. Hespanha, H. Kim, and S. Sastry, “Multiple-agent probabilistic pursuit-evasion games,” Tech. Rep., Dept. of EECS, University of California at Berkeley, March 1999.

[29] R. Vidal, S. Schaffert, O. Shakernia, J. Lygeros, and S. Sastry, “Decidable and semi-decidable controller synthesis for classes of discrete time hybrid systems,” in *Proc. of 40th IEEE Conf. on Decision and Control*, 2001, pp. 1243–1248.

[30] J. Lygeros, D.N. Godbole, and S. Sastry, “Verified hybrid controllers for automated vehicles,” *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 522–539, Apr. 1998.

[31] G. Pappas, C. Tomlin, J. Lygeros, D. Godbole, and S. Sastry, “A next generation architecture for air traffic management systems,” in *Proc. of 36th IEEE CDC*, Dec. 1997, pp. 2405–2410.

[32] T. Koo, F. Hoffmann, H. Shim, B. Sinopoli, and S. Sastry, “Hybrid control of an autonomous helicopter,” in *Proceedings of IFAC Workshop on Motion Control*, 1998, pp. 285–290.

[33] C. Sharp, O. Shakernia, and S. Sastry, “A vision system for landing an unmanned aerial vehicle,” in *Proc. of IEEE Int. Conf. on Robotics and Automation*, 2001, pp. 1720–1727.

[34] S. Rashid, “Design and Implementation of Multi-Agent Control: Pursuit & Map Building,” M.S. thesis, UC Berkeley, 2000.