A Hierarchical Approach to Probabilistic Pursuit-Evasion Games with Unmanned Ground and Aerial Vehicles[1]

H. Jin Kim René Vidal David H. Shim Omid Shakernia Shankar Sastry

Department of Electrical Engineering & Computer Sciences
University of California at Berkeley, Berkeley CA 94720
{jin,rvidal,hcshim,omids,sastry}@robotics.eecs.berkeley.edu

Abstract

We consider the problem of having a team of Unmanned Ground Vehicles (UGV) and Unmanned Aerial Vehicles (UAV) pursue a team of evaders while concurrently building a map in an unknown environment. We cast this problem in a probabilistic game-theoretic framework and consider two computationally feasible pursuit policies: *greedy* and *global-max*. We implement this scenario on a fleet of UGVs and UAVs by using a distributed hierarchical system architecture. Finally, we present both simulation and experimental results that evaluate the pursuit policies relating expected capture times to the speed and intelligence of the evaders and the sensing capabilities of the pursuers.

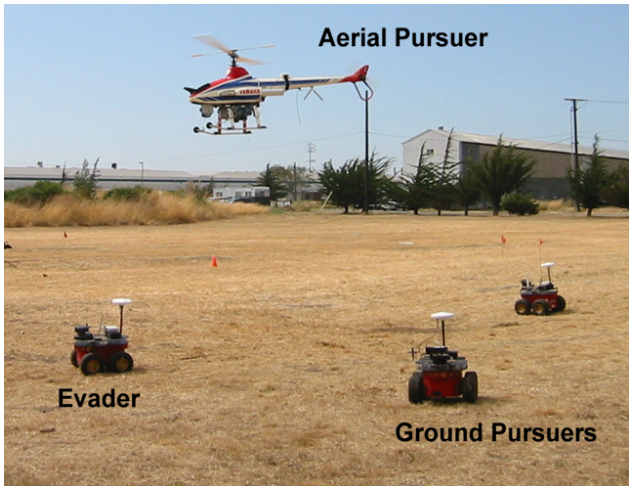

Figure 1: Berkeley test-bed for pursuit-evasion games.

1 Introduction

We consider pursuit-evasion games (PEG) in which a team of Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs) acting as *pursuers* tries to capture *evaders* within a bounded but unknown environment (see Figure 1). The classical approach to this PEG is to first build a terrain map and then play the game in the known environment. However, systematic map building is time consuming and computationally expensive, even in the case of simple two-dimensional rectilinear environments [1]. On the other hand, most of the PEG literature, see e.g. [5], considers worst-case motion for the evaders and assumes 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 [2] we combined pursuit-evasion games and map building in a single probabilistic framework which avoids the conservativeness inherent to the classical worst-case approaches and takes into account inaccurate sensing. We also proved the existence of a *persistent* pursuit policy which guarantees to capture a randomly moving evader in finite time with probability one. In [8] we extended this scenario to consider supervisory pursuers and multiple evaders. The case where an evader actively avoids the pursuers by having access to their information has been investigated in [3], which showed the existence of a one-step Nash solution. The difficulty of implementing this computationally expensive sub-optimal solution in real-time inspires the search for more efficient sub-optimal solutions with good performance. In [8] we considered two computationally feasible pursuit policies: *greedy* and *global-max*, using expected capture time as a performance metric.

In parallel with our theoretical work on pursuit-evasion games, we have been developing a platform of UAVs and UGVs as a test-bed for multi-agent coordination and control. In [10] we developed a real-time control system for regulation and navigation of a UAV. In [11] we presented a distributed hierarchical system architecture for pursuit-evasion games and described the implementations of the navigation, communication and sensing layers for teams of UGVs and UAVs.

In this work, we complete the implementation of our architecture for probabilistic PEG on real UAVs and

[1]This work was supported by grants ONR N00014-97-1-0946, ONR N00014-00-1-0621, and ARO MURI DAAH04-96-1-0341.

UGVs. We implement the high-level mission coordination layer including the components for pursuit policy computation and map building. We also compare pursuit policies relating expected capture times to the speed and intelligence of the evaders and the sensing capabilities of the pursuers. Our results show that in real PEGs between our fleet of UAVs and UGVs, the global-max policy outperforms the greedy policy.

Paper outline: Section 2 describes the PEG scenario. Section 3 describes the design and implementation of an architecture for multi-agent coordination and control. Section 4 presents the simulation and experimental results, and Section 5 concludes the paper.

2 Pursuit-Evasion Scenario

This section describes the theoretic foundations for the PEG scenario following the frameworks described in [2, 8], and a vision-based detection algorithm [11].

2.1 Probabilistic Framework

Consider a finite two-dimensional environment $\mathcal{X}$ with n_c cells that contain an unknown number of fixed obstacles. $\mathbf{x}_p \subset \mathcal{X}$ ($\mathbf{x}_e \subset \mathcal{X}$) is the set of cells occupied by the n_p pursuers (n_e evaders). For each $t \in \{1, 2, \ldots\}$, each pursuer or evader collects information within a certain subset of $\mathcal{X}$: *the visibility region*. We denote the visibility region of pursuer k (evader i) at time t as $V_{p_k}(t)$ ($V_{e_i}(t)$). Each measurement $\mathbf{y}(t)$ is a triple $\{\mathbf{v}(t), \mathbf{e}(t), \mathbf{o}(t)\}$, where $\mathbf{v}(t)$ denotes the measured positions of all the pursuers and $\mathbf{e}(t)$ ($\mathbf{o}(t)$) is a set of cells where each evader (obstacle) was detected. We denote by $\mathcal{Y}^*$ the set of all finite sequences of elements in $\mathcal{Y}$, and for each t denote by $\mathbf{Y}_t \in \mathcal{Y}^*$ the measurement history $\{\mathbf{y}(1), \ldots, \mathbf{y}(t)\}$.

Whenever an evader is captured, that evader is removed from the game. *Capture* is defined as follows: Let $v_k(t) \in \mathbf{v}(t)$ and $x_{e_i}(t) \in \mathbf{e}(t)$ be the estimated positions of pursuer k and evader i at time t, respectively. We say that evader i is captured by pursuer k at time t if $x_{e_i}(t) \in V_{p_k}(t)$ and $d(v_k(t), x_{e_i}(t)) \leq d_m$ where $d(\cdot, \cdot)$ is a metric in $\mathcal{X}$ and d_m is a pre-specified *capture distance*. The *capture time* 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 by one of the pursuers. We assume that capture and the pursuers' own positions are perfectly detected. For the other cells in the visibility region, we use a sensor model based on the probability of false positive detection $p \in [0, 1]$ and false negative detection $q \in [0, 1]$.

2.2 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 can assume $n_e = 1$ for map building purposes. 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 $p_e(x, t \mid Y_t) \triangleq$

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

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

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

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.

Second, pursuers compute $p_o(x \mid Y_t) \triangleq$

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

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

Finally, pursuers compute $p_e(x, t+1 \mid Y_t)$ from $p_e(x, t \mid Y_t)$ assuming that the evader moves randomly [11].

2.3 Pursuit Policies

Pursuer k decides its next desired position and heading $\mathbf{u}_k(t+1)$ within a control action set $\mathcal{U}_{p_k}$ based on the measurement history. We define $\mathcal{U}_{p_k} := \mathcal{X}_{p_k} \times \Psi_{p_k}$, where $\mathcal{X}_{p_k} \subset \mathcal{X}$ denotes the set of one-step reachable cells for pursuer k and $\Psi_{p_k} \subset (-\pi, \pi)$ denotes the set of one-step achievable headings. We call the collection $\mathbf{g} : \mathcal{Y}^* \to \Pi_{k=1}^{n_p} \mathcal{U}_{p_k}$

$$\mathbf{g}(\mathbf{Y}_t) = \left[\ \mathbf{u}_1(t+1), \cdots, \mathbf{u}_{n_p}(t+1) \ \right] \quad (4)$$

a *pursuit policy*.

We measure the performance of a given pursuit policy g, by the expected capture time $E[\mathbf{T}^* \mid \mathbf{g} = g]$. The dependence of the conditional probability of finding an evader at a time t on the pursuit policy is in general very complex. This makes the optimization problem of computing the policy which minimizes expected capture time not suitable for real-time applications.

Following [2, 8], we concentrate on finding efficiently computable policies with good performance. Instead of

considering general policies of the form in equation (4), we implement *memoryless* policies which depend on the probabilistic maps, or *belief states*, $p_o(x \mid Y_t)$ and $p_e(x, t+1 \mid Y_t)$.

Greedy Policy: Under the *greedy* policy, pursuer k moves to the cell in $\mathcal{X}_{p_k}$ with the highest probability of containing an evader over all the evader maps at the next time instant. That is, the desired position and heading, $\mathbf{u}_k(t+1) = [x_d(t+1), \psi_d(t+1)]$, are:

$$\begin{cases} x_d(t+1) = \underset{x \in \mathcal{X}_{p_k}}{\operatorname{argmax}} \ \max\limits_{i=\{1 \ldots n_e\}} p_{e_i}(x, t+1 \mid Y_t) \\ \psi_d(t+1) = \underset{\psi \in \Psi_{p_k}}{\operatorname{argmin}} \mid \angle(x_d(t+1) - v_k(t)) \ - \ \psi \mid^2 . \end{cases}$$

Notice that this policy is advantageous in scalability, since it assigns more importance to local measurements by searching only in $\mathcal{X}_{p_k}$ regardless of the size of $\mathcal{X}$.

Global-Max Policy: Under the *global-max* policy, pursuer k moves to the cell in $\mathcal{X}_{p_k}$ which is closest to the cell in $\mathcal{X}$ with the highest discounted probability of having an evader. The desired position and heading, $\mathbf{u}_k(t+1) = [x_d(t+1), \psi_d(t+1)]$, are:

$$\begin{cases} x_d(t+1) = \underset{x' \in \mathcal{X}_{p_k}}{\operatorname{argmin}} \ d\left(x', \underset{x \in \mathcal{X}}{\operatorname{argmax}} \ \max\limits_{i=\{1 \ldots n_e\}} \frac{p_{e_i}(x, t+1 \mid Y_t)}{d(x, v_k(t))}\right) \\ \psi_d(t+1) = \underset{\psi \in \Psi_{p_k}}{\operatorname{argmin}} \mid \angle(x_d(t+1) - v_k(t)) \ - \ \psi \mid^2 . \end{cases}$$

Here the discount factor encourages a pursuer to explore locations nearby before heading to far-away cells. Thus, if there are two cells in the map with the same high probability of having an evader, the global-max policy guides the pursuer towards the closer one.

2.4 Evasion Policy

A evader can either move randomly, or be smart. In the latter case, it builds a map of obstacles and pursuers and employs a greedy policy (to move to the one-step reachable cell with the minimum probability of being captured at the next time instant) or global-min policy (to move to the one-step reachable cell which is closest to the global location with the minimum discounted probability of being captured at the next time instant).

2.5 Vision-based Detection

The pursuers need to detect the locations of obstacles and evaders to build a probabilistic map. In [11] we described a technique and successful experiments on 3D position estimation based on a sensor fusion of color tracking and GPS information.

The position and heading of the pursuer, and the camera calibration parameters of the vision system determine the region in the environment that is visible to the pursuer. The vertices of the visibility region are computed by applying image projection equations to vertices of a fixed rectangle in the image. This visibility region is trapezoidal for pursuers with perspective projection cameras, and is a square for pursuers with omni-directional cameras. The trapezoidal and rectangular visibility regions are depicted in Figures 3 and 4.

3 Architecture Implementation

This section briefly describes a hierarchical control architecture for multi-agent coordination and its implementation on our fleet of UGVs and UAVs. Our architecture design was inspired by Automated Highway Systems [6], Air Traffic Management Systems [7], and Flight Management Systems [4].

3.1 System Architecture

As shown in Figure 2, we employ a hierarchical architecture which divides the control of each vehicle into different layers of abstraction. The architecture is modular and scalable, allowing one to design a complex large scale system by integrating simpler components. The abstraction allows a unified framework for high-level intelligent planning across heterogeneous platforms of unmanned vehicles.

The hierarchical system architecture consists of:

- **High level**
  - *Strategy Planner* implements high-level intelligent control for the vehicles, *i.e.* the pursuit policy computation described in Section 2.3.
  - *Map Builder* gathers sensor information from each vehicle and implements the map building described in Section 2.2.
- **Low level**
  - *Tactical Planner* converts strategic plans into a sequence of way-points or flight modes.
  - *Trajectory Planner* produces a realizable and safe trajectory based on a dynamic model of the vehicle and the specified way-points.
  - *Regulation Layer* performs real-time control to guide the vehicle along the specified trajectory.

3.2 Implementation of Low Level

Each UAV and UGV has two on-board computers: The tactical planner is implemented on one computer while the trajectory planner and regulation layers are implemented on the other. The UAVs and UGVs share many components for sensing and communication, such as IEEE 802.11b wireless LAN connectivity, differential GPS, a PC104 Pentium 233MHz-based PC running Linux, and a color-tracking vision system. Each of these components is described in detail in [11].

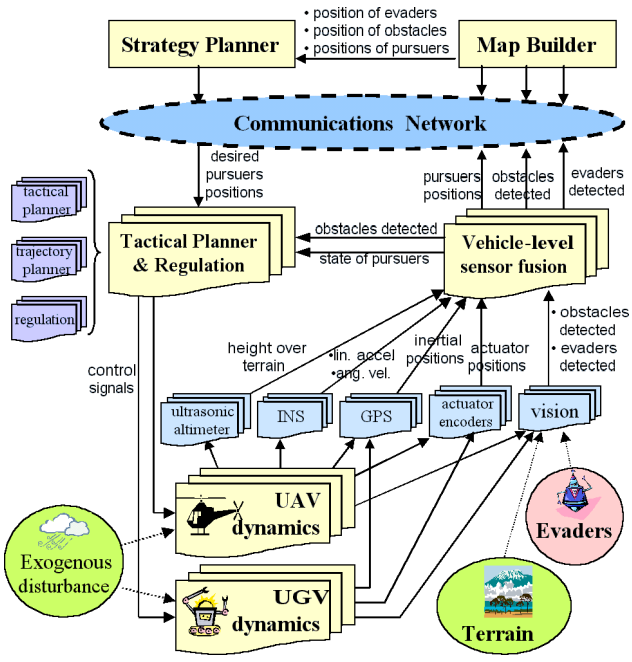

Figure 2: System Architecture

Our UAV fleet consists of custom-designed UAVs based on Yamaha R-50 and R-MAX industrial helicopters. The interface between the strategy planner and the lower level regulation was implemented through a framework called Vehicle Control Language (VCL). VCL is a script language that specifies a sequence of flight-modes and/or way-points. The implementation details of VCL and our UAV test-bed appear in [9, 10].

Our UGV fleet consists of ActivMedia Pioneer 2-AT all-terrain ground robots. The implementation details of the tactical/trajectory planner and regulation layer for the UGV's are provided in [11].

3.3 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. Further, we use a TCP interface to connect the MATLAB-based strategy planner and map builder with the UAVs and UGVs through the wireless LAN. The interface is implemented as a MEX function, which opens a TCP socket and communicates with a remote process. We enforce Simulink to run in real-time by having the TCP socket block program execution.

With this unified platform we are able to seamlessly combine experiments and simulations. In simulation mode, the strategy planner sends control commands in VCL to the UAV simulator obtained from system identification [10] and over TCP 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 based on 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, and visibility region and locations of obstacles and evaders from the vision system.

4 Simulation and Experimental Results

Here we present the results from simulations and experiments performed using the unified simulation/experimentation platform. We also study the effects of the pursuers' physical limitations (speed, visibility region, etc.) on their behavior under different pursuit policies, considering the capture time as a performance measure[1].

4.1 Simulation Results

Table 1 presents mean capture time of ten PEG simulations between 3 pursuers and 1 evader with random initial conditions. Experiments 1–4 evaluate the performance of the two pursuit policies against a randomly moving evader for two types of visibility regions: A sonar-based omni-directional view S_{p_k}[2] and a camera-based trapezoidal view T_{p_k}[3]. Experiments 5–8 evaluate the performance of the global maximum policy for different speeds and levels of intelligence of the evader.

Table 1: Simulation Results

Exp	Purs. Policy	Purs. Speed	Evad. Policy	Evad. Speed	Visib. Region	Capt. Time
1	Greedy	0.3	Rand	0.3	Omni	$279s$
2	Greedy	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	$168s$

4.2 Experimental Results

Table 2 presents results of real PEG 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.

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

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

[3] $T_{p_k}(t) := \triangle(x_{p_k}(t), \angle 45°, 7m) \backslash \triangle(x_{p_k}(t), \angle 45°, 1m)$, where $\triangle(x, \angle\theta, h)$ is an isosceles triangle with vertex x, height h and angle θ.

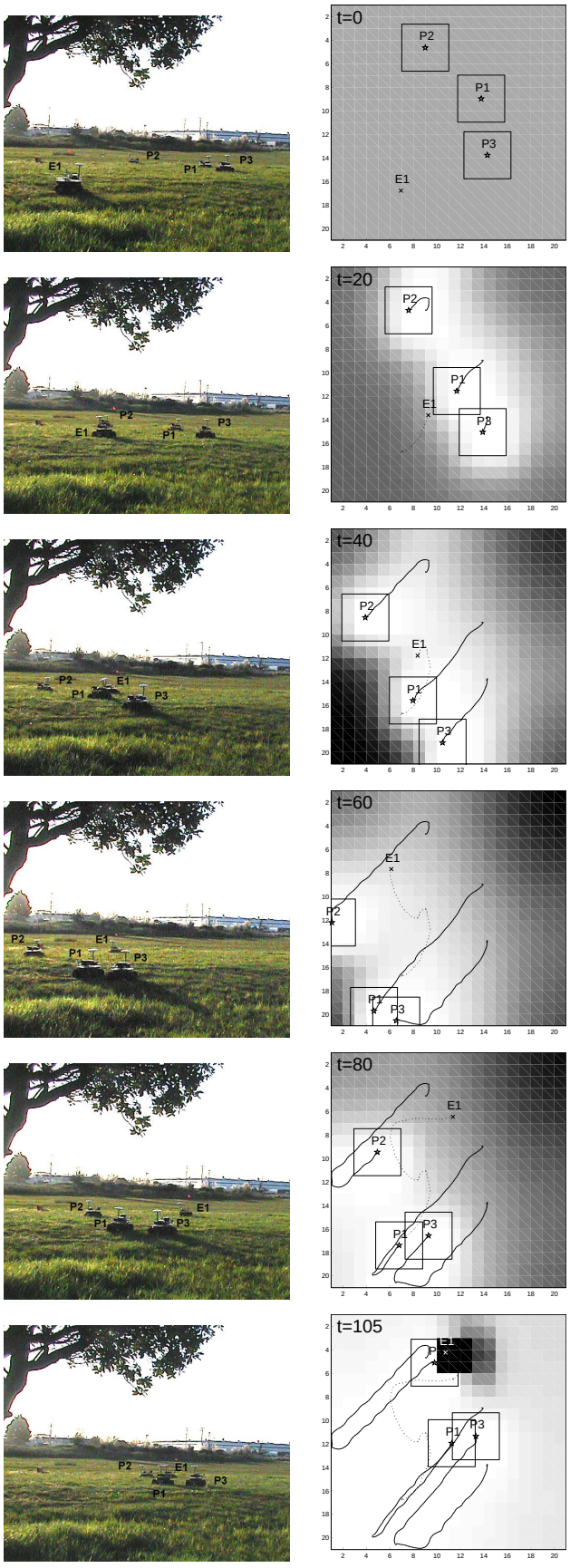

Figure 3: Showing a PEG between 3 UGV pursuers and 1 UGV evader. The pursuers move at 0.3 m/s using the global-max policy; the evader moves randomly at 0.1 m/s.

Table 2: 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	$37s$

4.3 Discussion

Capture Time vs. Pursuit Policy: Experiments 1–4 in Table 1 show that the global-max policy generally outperforms the greedy policy. This is because for each pursuer, the direction to the cell with *locally* maximum probability of containing an evader changes much more quickly than the direction to the cell with *globally* maximum probability. Thus, greedy pursuers spend more time changing heading than global-max pursuers, which effectively reduces their translational speed.

Capture Time vs. Visibility Region: Experiments 1–4 in Table 1 and 1–3 in Table 2 show that, regardless of the policy, pursuers with trapezoidal visibility regions outperform those with omni-directional visibility regions. At a given instant, the trapezoidal and omni-directional visibility regions cover approximately the same number of cells. However, a heading change allows a pursuer with a trapezoidal view to cover many more *new* cells than a pursuer with an omni-directional view.

Capture Time vs. Evasion Policy: Experiments 5–8 in Table 1 evaluate the global-max pursuit policy against an evader following either a random or global-min evasion policy. The pursuers always assume a randomly moving evader, instead of a computationally expensive one-step Nash solution. As expexted, it takes longer to capture an intelligent evader than a randomly moving evader. Further, for a fast evader it takes 300% more time to capture an intelligent one than a randomly moving one, while for a slow evader it takes only 64% more time. Finally, we can see that the global-max pursuit policy is robust with respect to changes in the evasion policy.

Capture Time vs. Evader Speed: Experiments 5 and 6 in Table 1 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.

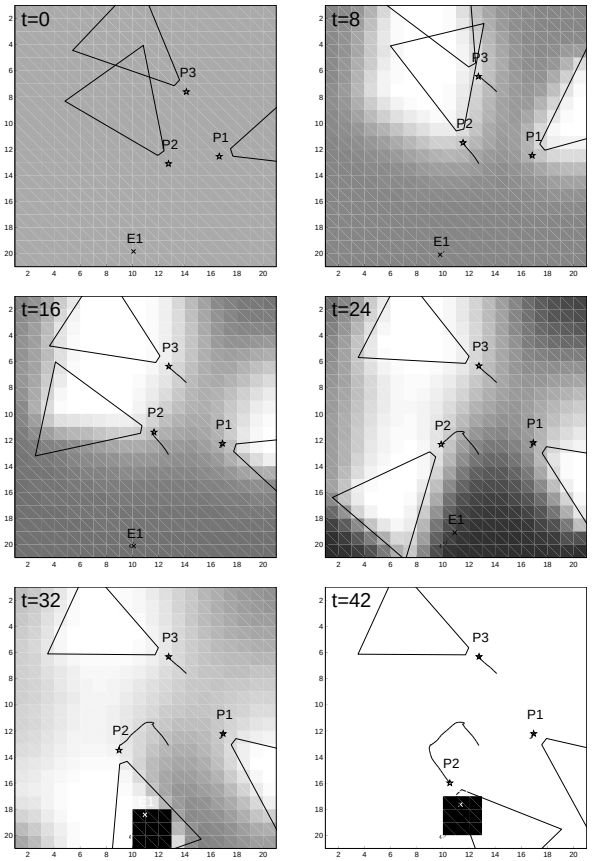

Figure 4: Showing snapshots of an actual PEG between 3 UGV pursuers and 1 UGV evader. The pursuers move at 0.3 m/s and use the global-max pursuit policy with simulated trapezoidal view. The evader moves at 0.1 m/s.

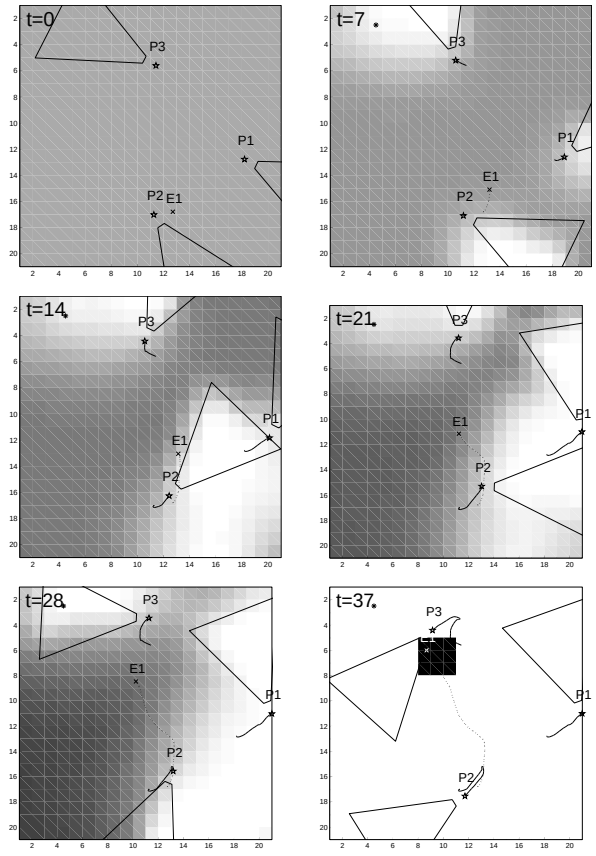

Figure 5: Showing snapshots of an actual PEG between 3 UGV pursuers and 1 UGV evader. The pursuers move at 0.3 m/s and use the global max pursuit policy with simulated trapezoidal view. The evader moves at 0.5 m/s.

5 Conclusions

We presented a framework for real-time control of multiple unmanned ground and aerial vehicles. The implementation was based on a distributed hierarchical architecture that is modular, scalable, and applicable to heterogeneous unmanned vehicles. The architecture was successfully applied to a scenario in which one team pursues another while concurrently building a map of the environment. We presented experimental results evaluating the performance of different pursuit policies with respect to speed, intelligence of the evaders and sensing capabilities of the pursuers. Our result shows that the global-max policy outperforms the greedy policy in realistic situations in which the vehicle dynamics and vision-based detection are incorporated.

References

[1] X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment I: The rectilinear case. *Journal of the ACM*, 45(2):215–245, March 1998.

[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] J. Hespanha, M. Prandini, and S. Sastry. Probabilistic pursuit-evasion games: a one-step Nash approach. In *Proc. of 39th IEEE CDC*, pages 2272–2277, Dec. 2000.

[4] T. Koo, F. Hoffmann, D.H. Shim, B. Sinopoli, and S. Sastry. Hybrid control of an autonomous helicopter. In *IFAC Workshop on Motion Control*, pages 285–290, 1998.

[5] 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.

[6] J. Lygeros, D.N. Godbole, and S. Sastry. Verified hybrid controllers for automated vehicles. *IEEE Transactions on Automatic Control*, 43(4):522–539, April 1998.

[7] 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*, pages 2405–2410, Dec. 1997.

[8] S. Rashid and H.J. Kim. Multiple-agent probabilistic pursuit-evasion games in 2.5D. Technical Report UCB/ERL M99/34, UC Berkeley, 1999.

[9] D.H. Shim. *Hierarchical Flight Control System Synthesis for Rotorcraft-Based Unmanned Aerial Vehicles*. PhD thesis, UC Berkeley, 2000.

[10] D.H. Shim, H.J. Kim, and S. Sastry. Hierarchical control system synthesis for rotorcraft-based unmanned aerial vehicles. In *Proc. of AIAA Conference on Guidance, Navigation and Control*, Denver, 2000.

[11] R. Vidal, S. Rashid, C. Sharp, O. Shakernia, H.J. Kim, and S. Sastry. Pursuit-evasion games with unmanned ground and aerial vehicles. In *Proc. of IEEE Conference on Robotics and Automation*, pages 2948–2955, 2001.