Proceedings of the 2002 IEEE
International Conference on Robotics & Automation
Washington, DC • May 2002

FLYING ROBOTS: Modeling, Control and Decision Making

H. Jin Kim David H. Shim Shankar Sastry [1]

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

Abstract

This paper presents a flight management system (FMS) implemented as on-board intelligence for rotorcraft-based unmanned aerial vehicles (RUAVs), in order to gradually refine given abstract mission commands into real-time control signals for each vehicle. A strategy planner uses the probabilistic decision making algorithms to determine suboptimal action at each time step. A graphical interface on ground station enables human intervention. We derive nonlinear dynamics model upon which we design a tracking control layer using nonlinear model predictive control and integrate with a trajectory generator for logistical action planning. The proposed structure has been implemented on Berkeley RUAVs and validated in probabilistic pursuit-evasion games to show the possibility of intelligent flying robots.

1 Introduction

Rotorcraft-based aerial vehicles are very promising platform for applications of intelligent unmanned vehicles due to their versatile maneuvers that cannot be achieved by other type of vehicles. Much of our work as part of the BErkeley AeRobot (BEAR) research project has been directed toward improving the performance of RUAVs to be employed in real-world applications. We have been studying probabilistic pursuit-evasion games (PEGs), where a team of aerial and ground-based vehicles pursue a team of evading vehicles while concurrently building a map in an unknown environment. In [1], we presented an experimental setup for pursuit-evasion games with unmanned aerial/ground vehicles (Figure 1), addressed physical issues in centralized heterogeneous multi-robot systems and presented algorithms and experimental results on PEGs.

This paper presents the synthesis of a flight management system (FMS) for RUAVs that endows RUAVs with autonomy to independently sense, reason, plan and act in coordination with other agents or environments and accessibility by human operators if necessary so that they can be employed in a versatile, resilient decentralized system. At the abstract level, we cast the pursuit-evasion problem in partially observable Markov decision process framework. By employing a policy search method, we obtained a scalable policy with the far better performance than myopic policies. At the physical level, we studied nonlinear model predictive planning and control that combines the trajectory generation and tracking control problem into a single problem. Our algorithm generates the control law for multiple unmanned vehicles, while explicitly dealing with their multi-input multi-output nonlinear dynamics, input saturation and state constraints.

[1]The authors would like to thank Andrew Ng. This research was supported by the ONR grants N00014-00-1-0621, ARO MURI grant DAAH04-96-1-0341, and DARPA MICA F33615-01-C-3150.

The remaining of this paper is as follows: Section 2 presents overview of flight management system for RUAVs. Section 3 describes the design of the tracking control system, highlighting the nonlinear model predictive approach. In Section 4, the proposed FMS is applied to pursuit-evasion games. Section 5 concludes the paper.

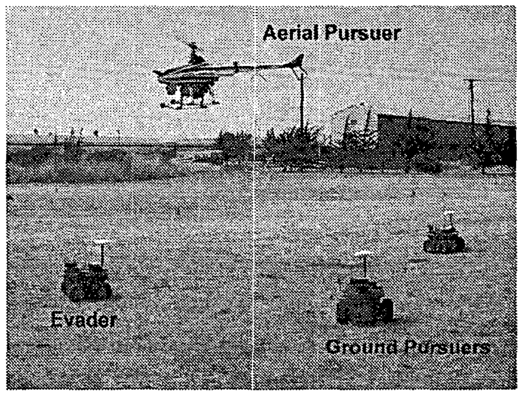

Figure 1: Berkeley RUAV in an autonomous search operation with unmanned ground vehicles

2 Flight Management System for Intelligent Unmanned Aerial Vehicles

This section describes implementation of each component in our flight management system shown in Figure 2.

2.1 Sensing

Dynamically changing conditions in the environment and the vehicle states are perceived by various on-board sensors. The precise guidance of the host vehicle of much smaller size demands more accurate navigation sensors. GPS-based INS is employed as a central navigation sensor-suite in order to correct the unbounded error of strap-down INS by supplementing a high-accuracy differential GPS. An additional Kalman filter is used in order to generate position estimates at a higher rate for more accurate position control including hover. Localizing sensors such as ultrasonic sensors and laser range-finders supplement the navigation sensor unit for the acquisition of the environment-specific information such as relative distance from the ground surface and for the detection of nearby objects around the host vehicle. A computer vision system [2] with a PTZ camera is used to detect the target objects or estimate the relative position and attitude with the help of INS/GPS.

2.2 Reasoning & Coordination

Data sensed by sensor suites should be properly interpreted by a strategy planner implemented on a flight control computer. When this information is not enough to identify the current state of the world, the world is modeled as a partially observable Markov decision process (POMDP), as described later in Section 4. The strategy planner (either centrally or on each vehicle) updates each agent's *belief (information) state*, i.e., probability distribution over the state space of the world, given measurement and action histories, and generates a policy, i.e, a mapping from the agent's belief state to its action set. Search of the optimal policy is computationally intractable in most problems, thus usually sub-optimal policies are implemented [1], or, the search for an (approximately) optimal policy is performed over a restricted class of policies [3].

Since there may be simple-structured policies with satisfactory performance, direct policy search methods have attracted particular interest. In the policy search framework, we want to find a good policy $\pi \in \Pi$, in a fixed class Π of policies. For a given POMDP and policy class Π, define $\pi^* \triangleq \arg\sup_{\pi\in\Pi} V_{\Xi}(\pi)$, where $V_{\Xi}(\pi) \triangleq \mathbf{E}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right]$ is the expected sum of rewards when using the policy π starting from the initial state, γ is a discount factor and r_t denotes the reward received at time t. The goal is then to find a policy $\hat{\pi} \in \Pi$ such that $V_{\Xi}(\hat{\pi})$ is close to $V_{\Xi}(\pi^*)$.

In [3], the algorithm to draw samples according to the initial-state distribution and reuse the same samples to evaluate every $\pi \in \Pi$ was presented. Given a POMDP with a finite action space, the number of samples needed to obtain a policy with the value close to $\sup_{\pi\in\Pi} V_{\Xi}(\pi)$ is a polynomial independent of the size of the state space or on the complexity of the transition distribution, as long as the policy class Π is simple. We will apply this algorithm to PEG in Section 4.

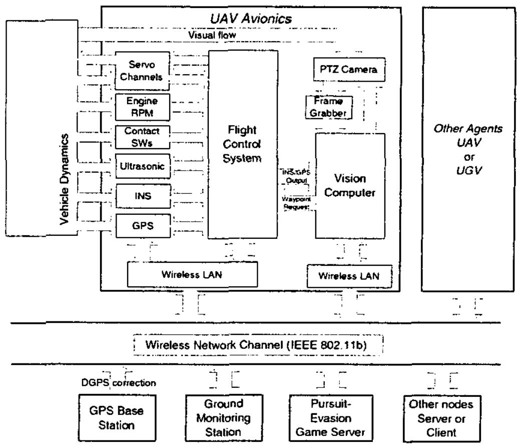

Figure 2: Flight management system implemented for multi-agent scenarios

The strategy planner also manages the configuration of the communication network. The role of communication in the FMS for RUAVs is more critical than in conventional FMSs for manned vehicles, because RUAVs should report the vehicle status and accept external commands typically at a faster rate than human voice communication. Moreover, the support of high quality-of-service (QoS) wireless communication system is desirable in order for multiple RUAVs to function as a tightly coordinated, reconfigurable, distributed networked intelligence.

2.3 Action

One of the most essential capabilities of an RUAV is to autonomously guide itself through the requested trajectories or way-points, in an autonomous manner. Each vehicle platform needs a flight controller that generates real-time control signals from the way-points requested by higher-level planners. Such a controller should be able to stabilize and follow the given trajectory in the presence of input saturation, state constraints and strong disturbance, as will be described in 3.2. Action-sensing coordination occurs at a very fast rate in order to cope with contingencies, for example, such as detection and avoidance of collisions.

2.4 Incorporating Human Intervention

While the autonomy of each vehicle is important, intervention of human intelligence is often necessary due to contingencies or mission characteristics. Human inputs in the form of information about the state of the world can be incorporated as *a priori* knowledge or transition rules. In the POMDP framework, this affects only the belief state, not the procedure of computing optimal actions. Human commands in the form of mission objectives can be expressed as a change to the reward function, and the importance of objective is specified by changing the magnitude of the rewards. Open-control architecture allows each strategic planner to accept incoming requests from human operators for mixed initiative planning through human-to-console and console-to-RUAV interface. The human-to-console interface, implemented as a graphic-user-interface (GUI) shown in Figure 3, receives human commands and displays the information downloaded from the RUAV. The console-to-RUAV interface sends the commands in a proper data structure to the RUAV controller and receives the RUAV status.

3 Flight Control and Trajectory Generation

This section describes the configuration of RUAV platforms and the design of control and trajectory generation layer at the vehicle-level of the hierarchy for autonomous flight.

3.1 Vehicle Platform and Dynamics

Berkeley RUAVs are built on commercial off-the-shelf (COTS) radio-controlled helicopters of various sizes and payloads. The vehicle platform is equipped with on-board navigation computers and sensors previously shown in Figure 2. The flight control software, implemented on QNX^{TM} real-time operation system, manages sensors, vehicle control, and communication. More detailed theoretical and practical issues in building an RUAV are described in [4].

As shown in Figure 4, we model an RUAV as a six degree-of-freedom rigid body augmented with the servorotor and gyroscope dynamics:

$$\begin{aligned}
\dot{\mathbf{x}}(t) &= f_c(\mathbf{x}(t), \mathbf{u}(t)), \qquad (1)\\
\mathbf{x} &= [\mathbf{x}^K, \mathbf{x}^D] \in \mathbb{R}^{15}\\
\mathbf{x}^K &= [x^S, y^S, z^S, \phi, \theta, \psi]\\
\mathbf{x}^D &= [u, v, p, q, a_{1s}, b_{1s}, w, r, r_{fb}]\\
\mathbf{u} &= [u_{a1s}, u_{b1s}, u_{\theta_M}, u_{r_{ref}}] \in \mathbb{R}^4,
\end{aligned}$$

where S denotes the spatial coordinate, and $[u, v, w]$ is the tangential velocities in the body coordinate frame respectively. The transformation between spatial and body coordinates are given by

$$[\dot{x}^S, \dot{y}^S, \dot{z}^S]^T = \mathbf{R}^{B\to S}[u, v, w]^T,$$

where $\mathbf{R}^{B\to S} \in \mathbf{SO}(3)$ is the rotational matrix of the body axis relative to the spatial axis, represented by ZYX Euler angles $[\phi, \theta, \psi]$;

$$\mathbf{R}^{B\to S} = \begin{bmatrix} c\psi c\theta & s\psi c\theta & -s\theta \\ -s\psi c\phi + c\psi s\theta s\phi & c\psi c\phi + s\psi s\theta s\phi & c\theta s\phi \\ s\psi c\phi + c\psi s\theta c\phi & -c\psi s\phi + s\psi s\theta c\phi & c\theta c\phi \end{bmatrix}. \qquad (2)$$

The variables ϕ, θ, and ψ denote roll, pitch, and yaw, respectively. p, q, and r denote the angular rates in roll, pitch, and yaw direction in the body coordinate frame, respectively;

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \qquad (3)$$

The parameters a_{1s} and b_{1s} are longitudinal and lateral flapping angles, and r_{fb} is the feedback gyro system state [5]. The input $\mathbf{u}$ consists of inputs to the lateral cyclic pitch, longitudinal cyclic pitch, main rotor collective pitch, and tail rotor collective pitch.

Assuming small velocity, small blade flapping, and constant rotor stiffness terms, and applying Newton-Euler Equation, we obtain

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} -\frac{1}{m}T_M a_{1s} - g\sin\theta + vr - wq \\ -\frac{1}{m}(T_M b_{1s} + T_T) + g\sin\phi - ur + wp \\ -\frac{1}{m}T_M + g + uq - vp \\ -\frac{1}{I_{xx}}[\{c_0 + T_M c_1\}b_{1s} + Q_M a_{1s} + T_M c_2 + T_T c_3] + \frac{I_{yy} - I_{zz}}{I_{xx}}qr \\ \frac{1}{I_{yy}}[\{c_0 + T_M c_1\}a_{1s} - Q_M b_{1s} + T_M c_4 - Q_T] + \frac{I_{zz} - I_{xx}}{I_{yy}}pr \\ \frac{1}{I_{zz}}(-Q_M + T_M b_{1s} c_4 + T_T c_5) + \frac{I_{xx} - I_{yy}}{I_{zz}}pq \end{bmatrix},$$

where c_i's are constants, m and I_{**}'s are the mass and inertia of RUAV. The forces vertical to each rotor surface, T_M, T_T, and anti-torques, Q_M, Q_T, are modelled as a linear function of $u_{\theta_M}, u, v, w, r, r_{fb}$. Finally first-order servorotor and feedback gyroscope dynamics (i.e., mappings from u_{a1s}, u_{b1s} to a_{1s}, b_{1s}, and from $u_{r_{ref}}$ to r_{fb}) are augmented to yield Equation (1).

3.2 Vehicle Stabilization & Control

In order to address multi-input multi-output, nonlinear nature and input/state saturation over the flight envelope, we design a nonlinear model predictive tracking controller (NMPTC) on Equation (4), which is discretized from (1);

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \qquad (4)$$

and a cost function for tracking is defined by

$$J \triangleq \phi(\tilde{\mathbf{y}}_N) + \sum_{k=0}^{N-1} L(\mathbf{x}_k, \tilde{\mathbf{y}}_k, \mathbf{u}_k) \qquad (5)$$

with

$$\begin{aligned}
\phi &\triangleq \frac{1}{2}\tilde{\mathbf{y}}_N^T \mathbf{P}_0 \tilde{\mathbf{y}}_N\\
L &\triangleq \frac{1}{2}\tilde{\mathbf{y}}_k^T \mathbf{Q} \tilde{\mathbf{y}}_k + \frac{1}{2}\mathbf{x}_k^T \mathbf{S} \mathbf{x}_k + \frac{1}{2}\mathbf{u}_k^T \mathbf{R} \mathbf{u}_k
\end{aligned}$$

where $\tilde{\mathbf{y}} \triangleq \mathbf{y}_d - \mathbf{y}$, $\mathbf{y} = \mathbf{C}\mathbf{x} \in \mathbb{R}^{n_y}$, $\mathbf{y}_d$ is the desired trajectory, and $\mathbf{S}$ is introduced to bound the state variables that do not directly appear in $\mathbf{y}$.

The details on this tracking controller design and online optimization using Lagrange multiplier and gradient-descent methods are reported in [6].

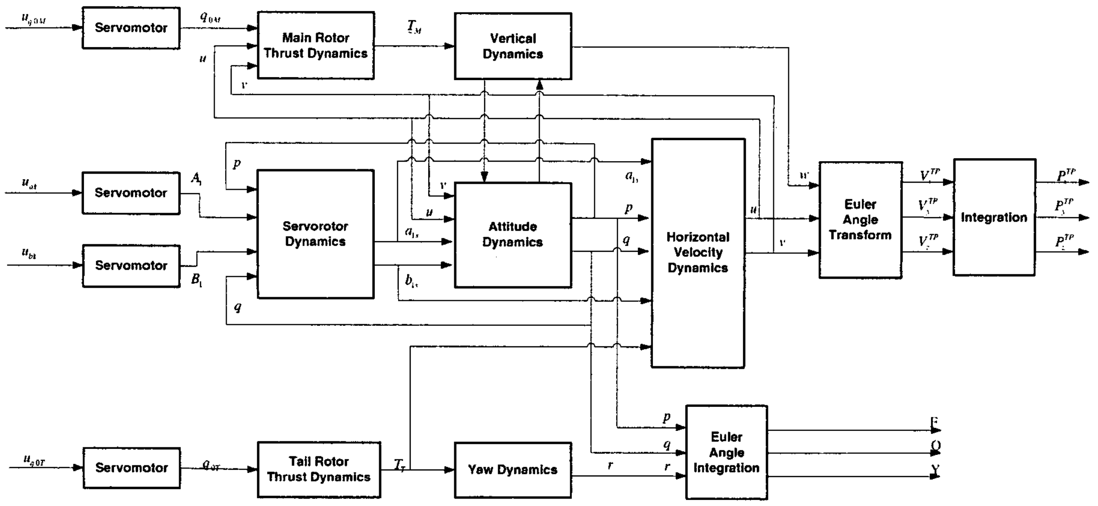

Figure 4: Block diagram of RUAV dynamics

3.3 Trajectory Generation

Trajectory generation layer, as a coordinator between the stabilization/tracking layer and the strategic planner, is responsible for refining reference trajectories and triggering the proper control law of the stabilization/tracking layer in order to execute each of these flight modes in a pre-programmed sequence or dynamically upon request. In designing such a way-point navigator, we employ the vehicle control language (VCL) which allows external systems as well as ground operators to request flight sequences or trajectories using the provided flight command set in Figure 5. Via rapidly reprogrammable, easily transmitted VCL codes, we obtain the isolation between the strategic planner and the stabilization layer. By abstracting away the details of sensing and control of each agent, we gain the interoperability of a unified framework for high-level planning across heterogeneous platforms. A VCL module consists of the user interface part on the ground station, the language interpreter, and the sequencer on the RUAV flight control computer.

In the context of batch VCL mode, a given flight is decomposed into a sequence of flight modes such as hover, forward flight, bank-to-turn, etc. A set of VCL commands is sent to the VCL execution module residing in the flight computer as a static command file or dynamic command set, over communication channels such as wireless Ethernet or RS-232 serial link.

When we incorporate a potential navigation function with the NMPTC framework and use the high-speed tracking VCL mode, multiple unmanned aerial vehicles can replan their trajectories on-the-fly when a collision is imminent and generate safe trajectories, while minimizing the tracking error from the original trajectory command. By considering low-level dynamics including input saturation and state constraints from the trajectory planning step, this approach removes the feasibility issues, i.e., generates physically realizable trajectories [6].

4 Experiments

In this section, we evaluate the effectiveness of the proposed hierarchical FMS in a pursuit-evasion game (PEG) assisted by the onboard vision computer.

This experiment evaluates the reasoning-action coordination and the performance of dynamic VCL in a pursuit-evasion game (PEG) [1]. The goal of pursuers is to "capture" evaders in a given grid-field. An evader is considered as captured when it is located within a certain range (e.g., 1.5 m) from a pursuer and in the pursuer's visibility region. The initial locations of evaders are unknown a priori. At each time step, the group of pursuers are required to go to the requested way-points and take measurements of their own locations and of any evaders within their visibility regions using the sensor suites. This measurement is used to build probabilistic maps of the possible locations of evaders and decide the pursuers' next action that minimizes the capture time. From the pursuers' point of view, this PEG is modeled as a POMDP.

The policy search framework [3] provides a natural way of specifying human insight in the process of constructing the pursuit policy class. Under the same pursuit-evasion game setup described in [1], we construct the pursuit policy class Π so that each policy $\pi \in \Pi$ is characterized by a set of parameters $\theta_\pi = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \beta, \gamma_2, \gamma_3\}$, which we will explain below.

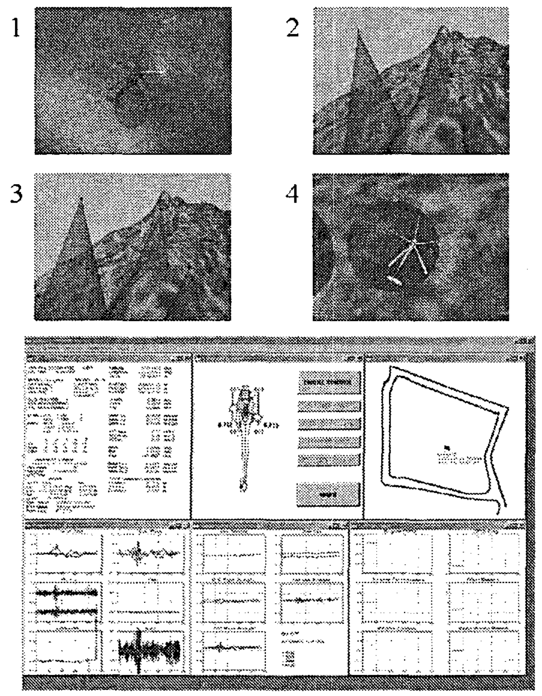

Figure 3: Graphical human-to-console interface on ground station of Berkeley testbed accepts human commands and displays the information received from multiple RUAVs.

At each t, the pursuers recursively update the evader map $p_e(\mathbf{x}_e(t+1) = x \mid Z^t)$, i.e., the posterior probability that the evader position $\mathbf{x}_e$ at time $t+1$ is x given Z^t, the pursuers' collected observation history upto time t. Then the pursuer k located at x_{p_k} chooses the action u_k^t based on the value $\{f_1, f_2, f_3\}$ as follows:

$$\begin{aligned} f_1 &= \log\left(\sum_{i=1, i\neq k}^{n_p} \mathrm{d}(\hat{x}_{p_k}^{t+1}, x_{p_i}^t) + \alpha_1 + 1\right) \\ f_2 &= \sum_{\tau=1}^{T} \sum_{x \in \mathcal{V}_{p_k}(\hat{x}_{p_k,fg}^{t+\tau})} \gamma_2^\tau p_e(\mathbf{x}_e(t+\tau) = x \mid Z^t) \\ f_3 &= \sum_{x \in \mathcal{R}(x_{p_k}^t, u_k^t)} \frac{\gamma_3^{\mathrm{d}(\hat{x}_{p_k}^{t+1}, x)}}{n_{\mathrm{d}(\hat{x}_{p_k}^{t+1}, x)}} p_e(\mathbf{x}_e(t+1) = x \mid Z^t)\,. \end{aligned}$$

Here $\hat{x}_{p_k}^{t+1}$ is the pursuer's position at the next instance if the action $u_k^t \in \{STAY, N, NE, E, SE, S, SW, W, NW\}$ were taken and performed accurately, thus the quantity f_1 represents the insight to maximize the summed distance from the other pursuers.

The set $\mathcal{V}_{p_k}(x)$ denotes the pursuer's visibility region of the pursuer when located at x. $\hat{x}_{p_k,fg}^{t+\tau}$ denotes the pursuer's position after taking the frontier-greedy action, i.e., the action that maximizes the probability summed over

```
TakeoffTo ⟨coord⟩ {abs,rel}
```
: perform autonomous take-off to a certain target point

```
Hover ⟨coord⟩ {abs,rel} {heading=⟨heading⟩ {deg,rad}}
⟨duration⟩ {sec,min}
```
: hover with a given heading angle for a given time

```
FlyTo ⟨coord⟩ {abs,rel}
{vel ⟨velocity⟩ {mps,kmps,fps,knots,mph}}
{passby,stopover} {autoheading,
heading=⟨heading⟩ {deg,rad}}
```
: cruise to a certain way-point, stopping over or passing by

```
MoveTo ⟨coord⟩ {abs,rel}
{vel ⟨velocity⟩ {mps,kmps,fps,knots,mph}}
{autoheading, heading=⟨heading⟩ {deg,rad}}
```
: move to a certain way-point to stop-over with a fixed heading

```
BankToTurn ⟨heading change⟩ {deg,rad}
{{radius} ⟨radius⟩{m,ft}}
{{vel} ⟨velocity⟩ {mps,kmps,fps,knots,mph}}
```
: perform bank-to-turn during cruise

```
Land
```
: command the vehicle to land

Figure 5: Vehicle Control Language Syntax

the newly observed cells at each step (see the shaded regions in Figure 6(a)), which is analogous to the "frontier" concept in map-building literature [7]. Thus, f_2 represents the probability of capturing the evader, summed over the visibility region following the path taken by frontier-greedy actions from time $t+1$ until $t+T$, after taking the action u_k^t, witha discount factor γ^2. This reflects the heuristic to maximize the probability to find the evader over the horizon T.

For $\tau = 1, 2, \ldots, n_\tau \triangleq |\{x \in \mathcal{R}(x_{p_k}^t, u_k^t) : \mathrm{d}(x, \hat{x}_{p_k}^{t+1}) = \tau\}|$ denotes the number of cells that can be reached within τ steps, where $\mathrm{d}(\cdot,\cdot)$ is the distance function defined by the pursuer's action set, and $\mathcal{R}(x_{p_k}^t, u_k^t)$ denotes the region in $\mathcal{X}$ exemplified in Figure 6(b). f_3 denotes the current evader map value summed over the quadrant, depending on the proceeding direction generated by action u_k^t, thus prioritizing the action sequence with a smaller heading change.

For each u_k^t in the pursuer's action set, we compute the value

$$\phi(u_k^t) = \alpha_2 f_1 + \alpha_3 \tfrac{f_2}{1-\gamma_2} + \alpha_4 \sqrt{\tfrac{f_2}{1-\gamma_2}} + \alpha_5 \tfrac{f_3}{1-\gamma_3} + \alpha_6 \sqrt{\tfrac{f_3}{1-\gamma_3}}\,, \quad (6)$$

and assign the probability of taking action u_k^t according to the Boltzman distribution

$$\mathrm{P}(\mathbf{u}_k^t = u_k^t) = \frac{\exp(\beta\phi(u_k^t))}{\sum_{u_k^t} \exp(\beta\phi(u_k^t))}\,. \quad (7)$$

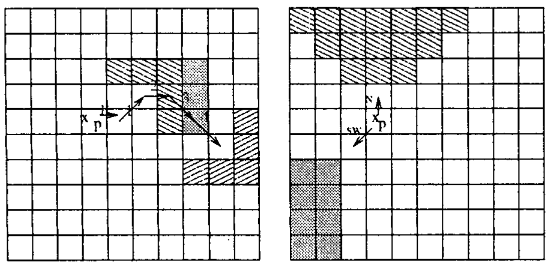

Figure 6: (a) The value f_2 represents the discounted sum of evader map value over the pursuer's visibility region along the path that maximizes frontier values. The evader map over the shaded region will be summed when $u_k^t = E$ and the frontire-greedy action sequence is $\{NE, E, SE, SE\}$ for $T = 4$. (b) explains f_3 for the actions $u_k^t = N$ and SW. Each shaded region represents $R(x_{p_k}^t, N)$ and $R(x_{p_k}^t, SW)$, respectively.

grid size	greedy	policy search
10 × 10	(7.3,4.8)	(5.1,2.7)
20 × 20	(42.3,19.2)	(12.3,4.3)

Table 1: Mean and standard deviation of the capture time over 100 runs

Then the parameters θ_π are optimized (i.e., approximately optimal $\hat{\pi} \in \Pi$ is searched) using standard optimization techniques. This policy computation algorithm is run in real-time using blocking socket of TCP/IP communication and the incoming VCL commands are processed by the on-board VCL execution module as previously described.

Note that $\alpha_2 = 0, \gamma_2 = 0, \alpha_4 = 0, \alpha_5 = 0, \alpha_6 = 0, T = 0$ yields a stochastic policy that is greedy with respect to p_e, i.e., that chooses the action to move to the cell with the maximum probability of capturing the evader at the next time instant. Table 1 compares the performance of this greedy policy with two policies with 1000 samples, $H = 40$, $T = 5$, $n_p = 2$. In a 10 × 10 grid, it took 7.3 steps for stochastic greedy pursuers to capture one randomly moving evader, while the pursuit policy optimized in the predescribed pursuit class needed 5.1 steps. The performance difference of these policies increases as the grid size increases. For example, in a 20 × 20 grid, the stochastic greedy policy takes 42.3 steps, while the optimized policy takes only 12.3 steps. The other notable point is a very large standard deviation under the greedy policy, which we attribute to the well-known shortcoming of greedy policy: the high possibility of getting in a trap while failing to explore for long-term optimality.

5 Conclusion

This paper presented a hierarchical flight management system designed for intelligent RUAVs. We described nonlinear RUAV dynamics model upon which we design a tracking controller, and addressed how to generate feasible trajectories for RUAVs. The experimental results validate the satisfactory performance of the multi-functional flight management system constructed on Berkeley RUAVs. At the strategy planning level, we cast the pursuit-evasion game in partially observable Markov decision process framework. In applying policy search methods to find pursuit policies, we incorporated the insight in constructing a policy class, and obtained a sub-optimal policy with the far better performance than myopic policies. Future research effort will be focused on expanding the capability of the flight control system with rich strategy planning logics that also consider limited resources or communication network, and improving robustness of current RUAV flight management systems.

References

[1] R. Vidal, O. Shakernia, H. J. Kim, H. Shim, and S. Sastry, "Multi-agent probabilistic pursuit-evasion games with unmanned ground and aerial vehicles," accepted for publication in *IEEE Transactions on Robotics and Automation.*

[2] C. S. Sharp, O. Shakernia, and S. S. Sastry, "A vision system for landing an unmanned aerial vehicle," in *IEEE International Conference on Robotics and Automation*, Seoul, Korea, 2001, pp. 1720–1727.

[3] A. Y. Ng and M. Jordan, "PEGASUS: A policy search method for large MDPs and POMDPs," in *Proc. of 17th International Conference on Uncertainty in Artificial Intelligence*, 2000.

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

[5] B. Mettler, M. B. Tischler, and T. Kanade, "System identification of small-size unmanned helicopter dynamics," in *American Helicopter Society 55th Forum*, Montreal, Quebec, Canada, May 1999.

[6] H. J. Kim, D. H. Shim, and S. Sastry, "A flight management system for intelligent unmanned aerial vehicles with nonlinear model predictive control," to appear in *21st American Control Conference*, 2002.

[7] B. Yamauchi, A. C. Schultz, and W. Adams, "Mobile robot exploration and map-building with continuous localization," in *Proc. of IEEE Conference on Robotics and Automation*, Leuven, Belgium, 1998, pp. 3715–3720.