A FLIGHT CONTROL SYSTEM FOR AERIAL ROBOTS: ALGORITHMS AND EXPERIMENTS

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

The Department of EECS, University of California, Berkeley,
Berkeley CA 94720-1774, USA.
{*hcshim, jin, sastry*}*@eecs.berkeley.edu*

Abstract: This paper presents a flight control system designed as on-board intelligence for rotorcraft-based unmanned aerial vehicles (RUAVs). This hierarchical flight control system, endowed with autonomy resembling sense-reason-act processes of intelligent agents in nature, gradually refines given abstract mission commands into real-time control signals for each vehicle. A tracking control layer is designed on the identified vehicle dynamics and integrated with a trajectory generator for logistical action planning. The proposed structure has been implemented on radio-controlled helicopters and validated in a variety of experiments. Results from way-point navigation, a probabilistic pursuit-evasion game and vision-based tracking of a moving target show the promising potential of intelligent flying robots.

Keywords: aircraft control, intelligent control, control system synthesis, predictive control, nonlinear systems, real-time systems, air traffic control

1. INTRODUCTION

Deployment of intelligent robots has been made possible through technological advances, and there is little doubt that the world of the future will be filled with intelligent robots employed to autonomously perform tasks, or embedded in systems all around us, extending our capabilities to perceive, reason and act, and substituting human efforts in applications where human operation is dangerous, inefficient and/or impossible. Subscribing to this idea, Rotorcraft-based unmanned aerial vehicles (RUAVs) deserve special interests, due to their flight capabilities. The unique lift generation mechanism of a rotorcraft enables hover, vertical take-off/landing, pirouette, and sideslip, which cannot be achieved by a fixed-wing aircraft. These versatile flight modes are often desired for high-fidelity detection, location and tracking of targets.

While the last decade has witnessed remarkable progresses in RUAV research including modeling (Mettler *et al.*, 1999), control theory (Shim *et al.*, 1998; Corban *et al.*, 1998; Bendotti and Morris, 1995) and avionic systems (Gavrilets *et al.*, 2000), current technology is still far from achieving most real-world solutions. The BErkeley AeRobot (BEAR) research project has been directed toward improving the performance of RUAVs as members of a networked intelligence consisting of multiple robotic vehicles with heterogeneous capabilities. As a benchmark problem that addresses many issues in multi-robot systems, a probabilistic pursuit-

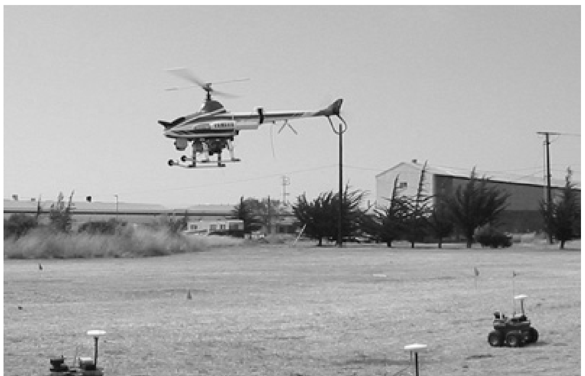
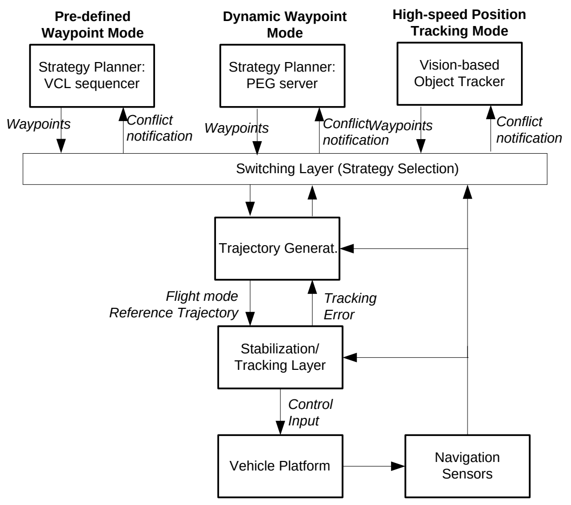


Fig. 1. A Berkeley RUAV in an autonomous mission with Unmanned Ground Vehicles

[1] The authors would like to thank Hoam Chung and Cory Sharp. This research was supported by the ONR grants N00014-97-1-0946 and N00014-00-1-0621, and ARO MURI grant DAAH04-96-1-0341.



Pre-defined Waypoint Mode
Dynamic Waypoint Mode
High-speed Position Tracking Mode
Strategy Planner: VCL sequencer
Strategy Planner: PEG server
Vision-based Object Tracker
Waypoints
Conflict notification
Waypoints
Conflict notification
Waypoints
Conflict notification
Switching Layer (Strategy Selection)
Trajectory Generat.
Flight mode Reference Trajectory
Tracking Error
Stabilization/ Tracking Layer
Control Input
Vehicle Platform
Navigation Sensors

Fig. 2. Multi-functional hierarchical flight management system of Berkeley RUAVs

evasion game has been studied, in which a team of vehicles pursue a team of evading vehicles and concurrently build a map in an unknown environment. Algorithms and experimental results on real games between two teams of ground robots in a centralized setting are presented in (Kim *et al.*, 2001).

In order to employ RUAVs in a wider range of applications, a decentralized framework should be considered for the reliable and efficient operations. Therefore, it is essential that each flight control system be endowed with well-suited autonomy, i.e., capabilities to independently sense, reason, plan and act in coordination with other robots or environments. This paper presents the synthesis of a hierarchical flight management system (FMS) for RUAVs that provides autonomy while allowing coordination among team members.

Section 2 presents an overview of a hierarchical flight control system for RUAVs. Section 3 describes the identification and regulation of vehicle dynamics, and trajectory generation. In Section 4, the proposed FMS is applied to three examples: pre-programmed waypoint navigation, dynamic way-point navigation in a pursuit-evasion game, and high-speed target tracking control. Section 5 concludes the paper.

2. FLIGHT MANAGEMENT SYSTEM FOR INTELLIGENT UNMANNED AERIAL VEHICLES

An "intelligent agent" continuously (1) perceives dynamically changing conditions in its environment, (2) reasons to interpret perceived information, to solve problems and to determine appropriate action, and (3) acts appropriately to affect conditions in its environment. Based on these attributes, this section describes each layer in a hierarchical flight management system shown in Fig. 2.

2.1 *Sensing*

Dynamically changing conditions in the environment and the vehicle states are perceived by various onboard 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 GPS. Additional sensors such as ultrasonic sensors and laser range-finders are used to acquire the environment-specific information such as relative distance from the ground surface, or to detect the objects around the host vehicle. Contact switches are installed on the landing gear of the helicopter primarily to assist automatic take-off/landing. A computer vision system (Sharp *et al.*, 2001) is used to detect the objects of interest based on their colors or shapes.

2.2 *Reasoning & Coordination*

Data sensed by sensor-suites should be properly interpreted by a strategy planning layer. Fig. 2 shows three types of strategy planners to be implemented for each experiment in Section 4. The appropriate strategy planner is selected by a switching layer for a given mission.

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 4.2. The strategy planner then 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 (Kim *et al.*, 2001), or, the class of policies to search through is limited (Ng and Jordan, 2000). Algorithms are typically run on real-time operating systems to satisfy hard real-time constraints.

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

While the autonomy of each vehicle is important, intervention of human intelligence is often necessary due to contingencies or mission characteristics. 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-UAV interface. The human-to-console interface, implemented as a graphic-user-interface (GUI), receives human commands and displays the information downloaded from the UAV. The console-to-UAV interface sends the commands in a proper data structure to the UAV controller and receives the UAV status.

2.3 *Action*

One of the most essential capabilities of a UAV is to autonomously guide itself through the requested trajectories or way-points, with minimal supervision by human operators. Each vehicle platform should be equipped with stabilizing controllers that take input saturation and state constraints into consideration, as will be described in 3.3. Action-sensing coordination occurs at a very fast rate in order to cope with contingencies, for example, such as detection and avoidance of collisions.

3. VEHICLE-LEVEL CONTROL & TRAJECTORY COORDINATION

This section describes the components at the vehicle-level of the hierarchy for autonomous flight: dynamic model identification, control and trajectory generation.

3.1 *Vehicle Platform Construction*

Berkeley RUAVs are built with off-the-shelf radio-controlled helicopters of various sizes and payloads. An industrial radio-controlled helicopter, Yamaha R-50, is equipped with on-board navigation computers and sensors for the experiments described in this paper. An optical engine RPM sensor regulates the engine at a constant speed in order to maintain the vehicle dynamic response close to the nominal operating point at which the dynamic model is acquired. The FMS software, implemented on a 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 (Shim, 2000).

3.2 *Dynamic Model Identification*

Acquisition of high-fidelity models of the target UAVs is a critical step in the design of a high-performance flight management system. Multi-input multi-output (MIMO), nonlinear characteristics and severe disturbance must be accounted for to acquire precise models. A lumped parameter method models a helicopter as a combination of main rotor, tail rotor, fuselage and stabilizer fins. Since a servorotor mechanism, augmented to the main rotor system in order to increase a response time delay and damping, dominates the main rotor dynamics, it should be properly reflected in the template model. In this research, prediction-error method (Ljung, 1997) is applied to the collected data using the parametric model proposed by (Mettler *et al.*, 1999), resulting in a six degree-of-freedom rigid body model with the first-order servorotor dynamics:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= f_c(\mathbf{x}(t),\mathbf{u}(t)), \quad \mathbf{x}(0) \text{ given} \qquad (1)\\ \mathbf{x} &= [\mathbf{x}^K,\mathbf{x}^D] \in \mathbb{R}^{n_x}\\ \mathbf{x}^K &= [x^S, y^S, z^S, \phi, \theta, \psi]\\ \mathbf{x}^D &= [\dot{x}^B, \dot{y}^B, \dot{\phi}, \dot{\theta}, a_{1s}, b_{1s}, \dot{z}^B, \dot{\psi}, r_{fb}]\\ \mathbf{u} &= [u_{a1s}, u_{b1s}, u_{\theta_M}, u_{\theta_T}] \in \mathbb{R}^{n_u},\end{aligned}$$

where S and B denote spatial and body coordinate respectively, and ϕ, θ, and ψ denote roll, pitch, and yaw, respectively. The transformation between spatial and body coordinates are given by

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

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

3.3 *Stabilization & Tracking Control*

The unstable RUAV dynamics needs proper stabilization using feedback control by the on-board real-time controller. A multi-loop single-input single-output (SISO) controller (Shim *et al.*, 2000) demonstrated stable responses with 0.5 m accuracy in the x and y directions, 0.1 m in the altitude, and 3° in the heading, when employed for hover and slow motion. In order to account for nonlinear nature and input/state saturation over the flight envelope including agile maneuvers, a nonlinear model predictive control is currently being applied. Only for controller design purposes, Eqn. (1) is discretized to

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k). \qquad (2)$$

and a cost function for tracking is defined by

$$J \triangleq \phi(\tilde{\mathbf{y}}_N) + \sum_{k=0}^{N-1} L(\mathbf{x}_k, \tilde{\mathbf{y}}_k, \mathbf{u}_k) \qquad (3)$$

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 $\mathbf{y}$. By introducing a sequence of Lagrange multiplier vectors $\{\lambda_k \in \mathbb{R}^{n_x}\}_{k=1}^N$, Eqn. (3) can be written as

$$\begin{aligned}J = \phi(\tilde{\mathbf{y}}_N) + \sum_{k=0}^{N-1} & L(\mathbf{x}_k, \tilde{\mathbf{y}}_k, \mathbf{u}_k) \qquad (4)\\ &+ \lambda_{k+1}^T[f(\mathbf{x}_k, \mathbf{u}_k) - \mathbf{x}_{k+1}].\end{aligned}$$

With the Hamiltonian function

$$H_k = L(\mathbf{x}_k, \tilde{\mathbf{y}}_k, \mathbf{u}_k) + \lambda_{k+1}^T f(\mathbf{x}_k, \mathbf{u}_k), \qquad (5)$$

Eqn. (4) can be written as

$$J = \phi(\mathbf{x}_N) - \lambda_N^T \mathbf{x}_N + \sum_{k=1}^{N-1}[H_k - \lambda_k^T \mathbf{x}_k] + H_0.$$

In order to compute $\{\mathbf{u}_k\}_0^{N-1}$ that minimize J, take a look at

$$
\begin{aligned}
dJ &= \left[\frac{\partial \phi}{\partial \mathbf{x}_N} - \lambda_N^T\right] d\mathbf{x}_N \\
&+ \sum_{k=1}^{N-1}\left[\left\{\frac{\partial H_k}{\partial \mathbf{x}_k} - \lambda_k^T\right\} d\mathbf{x}_k + \frac{\partial H_k}{\partial \tilde{\mathbf{y}}_k} d\tilde{\mathbf{y}}_k + \frac{\partial H_k}{\partial \mathbf{u}_k} d\mathbf{u}_k\right] \\
&+ \frac{\partial H_0}{\partial \mathbf{x}_0} d\mathbf{x}_0 + \frac{\partial H_k}{\partial \tilde{\mathbf{y}}_0} d\tilde{\mathbf{y}}_0 + \frac{\partial H_k}{\partial \mathbf{u}_0} d\mathbf{u}_0. \qquad (6)
\end{aligned}
$$

By choosing

$$
\lambda_N^T = \frac{\partial \phi}{\partial \mathbf{x}_N} = -\tilde{\mathbf{y}}_N^T \mathbf{P}_0 \mathbf{C} \qquad (7)
$$

$$
\begin{aligned}
\lambda_k^T &= \frac{\partial H_k}{\partial \mathbf{x}_k} + \frac{\partial H_k}{\partial \tilde{\mathbf{y}}_k}\frac{\partial \tilde{\mathbf{y}}_k}{\partial \mathbf{x}_k} \\
&= \mathbf{x}_k^T \mathbf{S} + \lambda_{k+1}^T \frac{\partial f_k}{\partial \mathbf{x}_k} - \tilde{\mathbf{y}}^T \mathbf{Q} \mathbf{C}, \qquad (8)
\end{aligned}
$$

Eqn. (6) is simplified to

$$
dJ = \sum_{k=0}^{N-1} \frac{\partial H_k}{\partial \mathbf{u}_k} d\mathbf{u}_k + \lambda_0^T d\mathbf{x}_0, \qquad (9)
$$

and

$$
\frac{\partial H_k}{\partial \mathbf{u}_k} = \mathbf{u}_k^T \mathbf{R} + \lambda_{k+1}^T \frac{\partial f_k}{\partial \mathbf{u}_k}. \qquad (10)
$$

With a candidate input sequence $\{\mathbf{u}_k\}_0^{N-1}$ and a given $\mathbf{x}_0$, on-line optimizations can be achieved by the following process presented in (Sutton and Bitmead, 2000):

```
while $|\Delta J| > \varepsilon$ do
    for $k = 1, \cdots, N$
        compute $\{\mathbf{x}_k\}_1^N$ using (2)
    end
    for $k = N, \cdots, 1$
        compute $\lambda_k$ using (7) and (8)
    end
    for $k = 1, \cdots, N$
        compute $\frac{\partial H_k}{\partial \mathbf{u}_k}$ using (10)
    end
    if $\Delta J \le 0$
        $\mathbf{u}_{k+1} := \mathbf{u}_k + \Delta_k \frac{\partial H_k}{\partial \mathbf{u}_k}$ for $k = 0, \cdots, N-1$
    else reduce $\Delta_k$ .
```

The details on this tracking control scheme are reported in (Kim *et al.*, 2002). The associated parameters are tuned to generate an appropriate control input for each flight mode, and resulting controllers will be integrated with the trajectory coordination layer.

3.4 *Trajectory Generation*

A trajectory generation layer is responsible for generating a desired trajectory or a sequence of flight modes and triggering the proper control law in the stabilization/tracking layer to execute it.

The trajectory generation layer employs a framework called Vehicle Control Language (VCL). VCL is implemented as a script language that decomposes a given mission into a sequence of flight modes or way-points with the provided command set, as will be shown in Section 4.1. Using rapidly reprogrammable, easily transmitted VCL codes, it is possible to isolate the strategic planner and the stabilization layer. By abstracting away the details of sensing and control of each agent, the unified interoperability for high-level planning across heterogeneous platforms is achieved. Yet by considering the dynamics of each vehicle in high-level planning, the overall system can achieve real-time performance. A VCL module consists of the user interface on the ground station, the language interpreter, and the sequencer on the FMS. For a given flight pattern, the VCL code may be generated using a graphic user interface, or manually and uploaded as an ASCII file to the flight computer for a sequential execution.

4. EXPERIMENTS

In this section, the performance of the proposed hierarchical FMS is evaluated in a series of test flights of three distinct scenarios: (1) way-point navigation using a batch (or preprogrammed) VCL mode, (2) a pursuit-evasion game employing a dynamic VCL mode, and (3) high-speed tracking of a moving target assisted by the on-board vision computer.

4.1 *Way-point Navigation: Batch VCL Mode*

In this mode, the VCL execution module assumes the highest hierarchy in the guidance of the RUAV. A lawn-mowing pattern as shown in Fig. 3 is used as a sample trajectory and the corresponding VCL codes are generated in the strategy planner as a text file. The flight mode, way-point, and other optional parameters are extracted in each line of VCL code and then sent to the trajectory coordination layer. Upon receiving a new VCL command, it activates a suitable control module for the current flight mode associated with the target way-point and other options. The real-time control outputs generated by the stabilization/ tracking layer are sent to the actuators on the host RUAV. The navigation measurements are reported to all the layers for feedback control and other supervisory tasks.

4.2 *Pursuit-Evasion Game: Dynamic VCL Mode*

This experiment evaluates the performance of the FMS in a probabilistic pursuit-evasion game (PEG) (Kim *et al.*, 2001). 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 it is in the pursuer’s visibility region. The initial locations of evaders are unknown a priori. At each discrete time instant, the group of pursuers, consisting of RUAVs and/or unmanned ground vehicles (UGVs), is required to go to the requested way-points and take measurements of their own locations and of any evaders within their visibility regions using sensor-suites. This measurement is used to decide the pursuers’ next action that minimizes the capture time. From the pursuers’ point of view, this PEG is modeled as a POMDP, i.e., a tuple $\langle \mathcal{S}, \mathcal{A}, T, \mathcal{Z}, O, R \rangle$ [2] :

[2] Random variables are indicated in bold type according to the usual convention.

```
TakeoffTo(0,0,-5)rel;
Hover(0,0,0)rel heading=270deg duration=10s;
FlyTo(0,-5,0)rel vel=0.5m/s stopover autoheading;
Hover(0,0,0)rel heading=0deg duration=10s;
FlyTo(5,0,0)rel vel=0.5m/s stopover autoheading;
Hover(0,0,0)rel heading=270deg duration=10s;
FlyTo(0,-5,0)rel vel=0.5m/s stopover autoheading;
Hover(0,0,0)rel heading=180deg duration=10s;
FlyTo(-5,0,0)rel vel=0.5m/s stopover autoheading;
Hover(0,0,0)rel heading=270deg duration=10s;
FlyTo(0,5,0)rel vel=0.5m/s stopover autoheading;
Hover(0,0,0)rel heading=0deg duration=10s;
FlyTo(5,0,0)rel vel=0.5m/s stopover autoheading;
Hover(0,0,0)rel heading=270deg duration=10s;
Land;
```

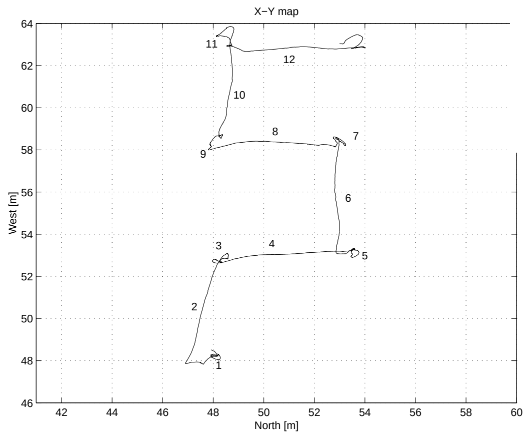


Fig. 3. A VCL code for lawn-mowing pattern and flight experiment result

- $\mathcal{S}$ is a finite set of states of the world, i.e., the configurations of the pursuers and evaders in the given field;
- $\mathcal{A}$ is a finite set of actions;
- $T : \mathcal{S} \times \mathcal{A} \to \mathbf{PD}(\mathcal{S})$ is a transition function. $T(s', s, a_t) = \mathrm{P}(\mathbf{s}(t+1) = s' \mid \mathbf{s}(t) = s, \mathbf{a}(t) = a_t)$ is the probability of landing in the state $s' \in S$ under the action $a \in A$ from the state $s \in S$;
- $\mathcal{Z}$ is a finite set of observations the pursuer can experience of its world;
- $O : \mathcal{S} \times \mathcal{A} \to \mathbf{PD}(\mathcal{Z})$ is the observation function. $O(z_t, s', a_{t-1}) = \mathrm{P}(\mathbf{z}(t) = z_t \mid \mathbf{s}(t) = s', \mathbf{a}(t-1) = a_{t-1})$ is the probability of making observation z given that the pursuer took action a_t and landed in state s';
- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{Z} \to \mathbb{R}$ is a reward function. $r(s, a_t, z_t) =$ 1 if s corresponds to the evader-captured configuration and 0 otherwise.

The pursuers' belief state, $\eta^t_{(s)} \triangleq \mathrm{P}(\mathbf{s}(t) = s \mid \mathbf{A}_{t-1} = A_{t-1}, \mathbf{Z}_t = Z_t)$ denote the conditional probability that the world is in state s given $\eta^0_{(s)} \triangleq \mathrm{P}(\mathbf{s}_0 = s)$, and the action and observation histories, i.e., $A_{t-1} \triangleq \{a_0, \cdots, a_{t-1}\}$, and $Z_t \triangleq \{z_0, \cdots, z_t\}$. Given that the pursuer observes z_{t+1} after applying a_t, the recursive belief state dynamics can be obtained by applying Bayes' rule

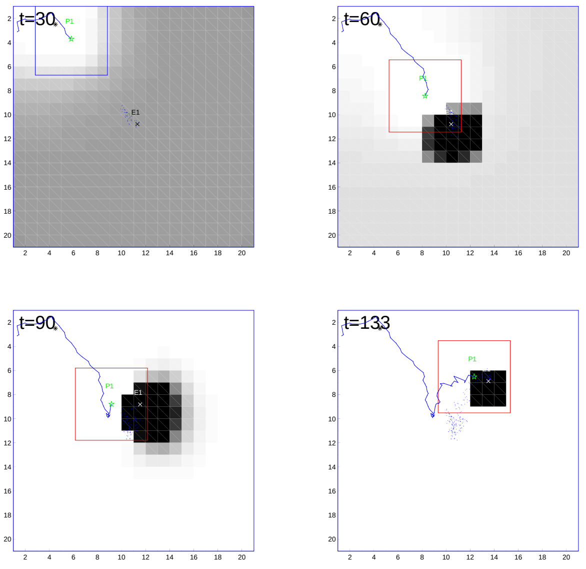


Fig. 4. Snapshots of 1 vs. 1 Pursuit-Evasion Game (P: Pursuer RUAV, E: Evader UGV)

$$\begin{aligned}\eta^{t+1}_{(s')} &= \frac{\mathrm{P}(\mathbf{s}_{t+1}{=}s', \mathbf{A}_t{=}A_t, \mathbf{Z}_{t+1}{=}Z_{t+1})}{\sum_{s' \in \mathcal{S}} \mathrm{P}(\mathbf{s}_{t+1}{=}s', \mathbf{A}_t{=}A_t, \mathbf{Z}_{t+1}{=}Z_{t+1})} \\ &= \frac{\mathrm{P}(x', A_{t-1}, Z_t, a_t, z_{t+1})}{\sum_{x' \in \mathcal{X}} \mathrm{P}(x', A_{t-1}, Z_t, a_t, z_{t+1})} \\ &= \frac{\mathrm{P}(z_{t+1} | s', a_t) \sum_{s \in \mathcal{S}} \mathrm{P}(s', s, A_{t-1}, Z_t, a_t)}{\sum_{s' \in \mathcal{S}} \mathrm{P}(s', A_{t-1}, Z_t, a_t, z_{t+1})} \\ &= \frac{O(z_{t+1}, s', a_t) \sum_{s \in \mathcal{S}} T(s', s, a_t) \eta^t_{(s)}}{\sum_{s' \in \mathcal{S}} O(z_{t+1}, s', a_t) \sum_{s \in \mathcal{S}} T(s', s, a_t) \eta^t_{(s)}},\end{aligned}$$

whose denominator can be treated as a normalizing factor, independent of s'. The strategic planner implements a variety of computationally-efficient suboptimal policies, including a greedy policy with respect to $\eta^{t+1}(s')$, under which the location in the pursuer's one-step reachability region with the highest probability of containing the evader at the next step is selected as the way-point for the pursuers. This information is sent to the pursuers in a VCL code via wireless communication and processed by the on-board VCL execution module. Fig. 4 shows a PEG of one greedy aerial pursuer vs. one greedy ground evader in a 20m x 20m field. The number of participating agents can be easily changed. The setup of one aerial pursuer is shown so that the load of RUAV is maximized. Along with the trajectories for the RUAV pursuer and the UGV evader, the evolution of the probabilistic map is shown as the gray-scale background and the square represents the visibility region of RUAV. The RUAV pursuer catches the evader in 133 seconds. This experiment shows that the proposed control law and dynamic VCL are well-suited in a hierarchical control structure for the PEG.

4.3 *High-Speed Position Tracking*

In this scenario, an RUAV is required to track a moving ground object, after the RUAV detects it. The vision computer estimates the relative position of the ground target by extracting a special feature of a marker detected by a camera (Sharp *et al.*, 2001). The high-rate position-tracking request, e.g., at 3 Hz in this case, activates a specially-tuned way-point navigator.

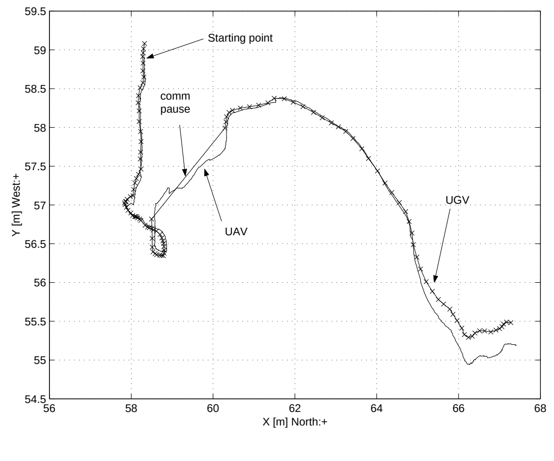


Fig. 5. Tracking control of ground target

In Fig. 5, the trajectories of the RUAV and UGV are shown. The FMS shows satisfactory tracking performance with a small error attributed to wind gusts. In the middle of the experiment, it was noticed that the vision computer ceased sending the reference trajectory for about 8 seconds. The FMS demonstrates its fail-safe feature in this faulty situation by following an *expected* trajectory of targets until next command is received.

5. CONCLUSION

This paper has shown the design/implementation procedure and effectiveness of a hierarchical structure for an RUAV flight control system. The experimental results validate the satisfactory performance of the multi-functional flight management system constructed on Berkeley RUAVs in the three examples considered in this paper: way-point navigation, pursuit-evasion game and tracking of a moving target. Further research effort will be exercised to expand the capability of the flight management system with rich strategy planning logics and increased robustness in order to narrow down the gap between current RUAVs and intelligent flying robots.

6. REFERENCES

Bendotti, P. and J. C. Morris (1995). Robust control for a model helicopter. In: *Proc. of the American Control Conference*. pp. 682–687.

Corban, J. E., A. J. Calise and J. V. R. Prasad (1998). Implementation of adaptive nonlinear control for flight test on an unmanned helicopter. In: *Proc. of 37th IEEE Conference on Decision and Control*. pp. 3641–3646.

Gavrilets, V., A. Shterenberg, M. Dehaleh and E. Feron (2000). Avionics system for a small unmanned helicopter performing aggressive maneuvers. In: *Digital Avionics Systems Conference*.

Kim, H. J., D. H. Shim and S. Sastry (2002). A flight management system for intelligent unmanned aerial vehicles with nonlinear model predictive control. Submitted to *21st American Control Conference*.

Kim, H. J., R. Vidal, D. H. Shim and S. Sastry (2001). A hierarchical approach to probabilistic pursuit-evasion games with unmanned ground and aerial vehicles. To appear in *Proc. of 40th IEEE Conference on Decision and Control*.

Ljung, L. J. (1997). *Matlab System Identification Toolbox User's Guide*.

Mettler, B., M. B. Tischler and T. Kanade (1999). System identification of small-size unmanned helicopter dynamics. In: *American Helicopter Society 55th Forum*.

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

Sharp, C. S., O. Shakernia and S. S. Sastry (2001). A vision system for landing an unmanned aerial vehicle. In: *IEEE International Conference on Robotics and Automation*. pp. 1720–1727.

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

Shim, D. H., H. J. Kim and S. Sastry (2000). Hierarchical control system synthesis for rotorcraft-based unmanned aerial vehicles. In: *AIAA Guidance, Navigation and Control Conference*.

Shim, H., T. J. Koo, F. Hoffmann and S. Sastry (1998). A comprehensive study of control design for an autonomous helicopter. In: *Proc. of 37th IEEE Conference on Decision and Control*. pp. 3653–3658.

Sutton, G. J. and R. R. Bitmead (2000). Computational implementation of nmpc to nonlinear submarine. In: *Nonlinear Model Predictive Control* (F. Allgöwer and A. Zheng, Eds.). Vol. 26. pp. 461–471. Birkhäuser.