PERGAMON

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Control Engineering Practice 11 (2003) 1389–1400

CONTROL ENGINEERING PRACTICE

www.elsevier.com/locate/conengprac

A flight control system for aerial robots: algorithms and experiments

H. Jin Kim*, David H. Shim

EECS Department, University of California, 459 Cory Hall, Berkeley, CA 94720, USA

Received 31 October 2002; accepted 8 April 2003

Abstract

This paper presents a hierarchical flight control system for unmanned aerial vehicles. The proposed system executes high-level mission objectives by progressively substantiating them into machine-level commands. The acquired information from various sensors is propagated back to the higher layers for reactive decision making. Each vehicle is connected via standardized wireless communication protocol for scalable multi-agent coordination. The proposed system has been successfully implemented on a number of small helicopters and validated in various applications. Results from waypoint navigation, a probabilistic pursuit-evasion game and vision-based target tracking demonstrate the potential of the proposed approach toward intelligent flying robots.
© 2003 Elsevier Science Ltd. All rights reserved.

Keywords: Control system synthesis; PID control; Predictive control; Nonlinear systems; Intelligent control; Real-time systems

1. Introduction

Deployment of intelligent robots has been made possible through technological advances in various fields such as artificial intelligence, robotics, wireless communication, and control theories. There is little doubt that intelligent robots will be employed to autonomously perform tasks, or embedded in many systems, and extend our capabilities to perceive, reason and act, or substitute human efforts in applications where human operation is dangerous, inefficient and/or impossible. Subscribing to this idea, BErkeley AeRobot (BEAR) project aims to organize multiple autonomous agents into integrated and intelligent systems with reduced cognition and control complexity, fault-tolerance, adaptivity to changes in task and environment, modularity and scalability to perform complex missions efficiently.

Rotorcraft-based unmanned aerial vehicles (RUAVs) have unique flight capabilities such as hover, vertical take-off/landing, pirouette, and side-slip, which cannot be achieved by conventional fixed-wing aircraft. These versatile flight modes are useful for various situations including reconnaissance, ground target tracking, and operations with limited launching space such as a shipdeck or in cases that require frequent landings and take-offs (Fig. 1).

The last decade has witnessed remarkable progress in RUAV research including modeling (Mettler, Tischler, & Kanade, 1999), control theory (Shim, Koo, Hoffmann, & Sastry, 1998; Corban, Calise, & Prasad, 1998; La Civita, Papageorgiou, Messner, & Kanade, 2002) and avionics (Gavrilets, Shterenberg, Dehaleh, & Feron, 2000). However, the current status still falls short of implementing solutions to most real-world applications and exploiting the full capabilities of the rotorcraft. The BEAR research project has been directed toward improving the performance of RUAVs as members of a networked intelligent team consisting of multiple heterogeneous robotic vehicles. To achieve this objective, 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.

The dynamics of an RUAV is identified by applying a parametric identification method to the collected flight data. This paper presents two control approaches: a multi-loop proportional–integral–differential controller

*Corresponding author. Tel.: +1-510-642-3843; fax: +1-510-643-2356.

E-mail addresses: jin@eecs.berkeley.edu (H.J. Kim), hcshim@eecs.berkeley.edu (D.H. Shim).

0967-0661/03/$ - see front matter © 2003 Elsevier Science Ltd. All rights reserved.
doi:10.1016/S0967-0661(03)00100-X

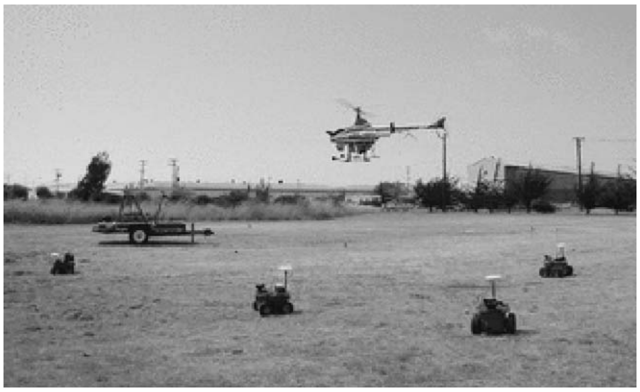

Fig. 1. A Berkeley RUAV in an autonomous flight with ground robots.

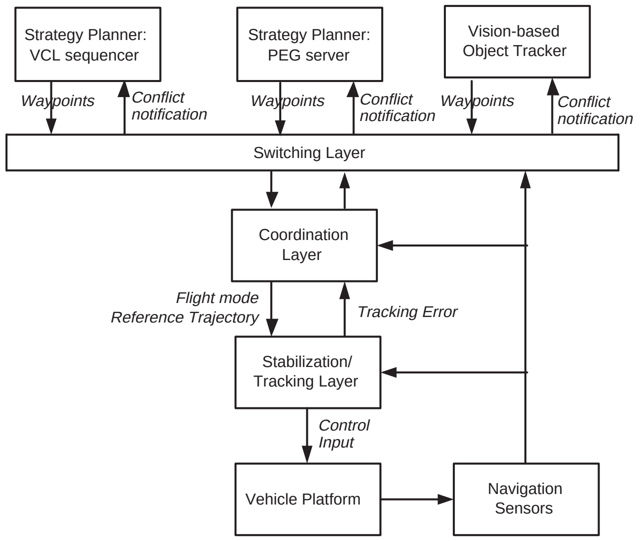

Fig. 2. Multi-functional hierarchical flight management system implemented on Berkeley RUAVs.

and a nonlinear model predictive tracking controller. The former has been successfully validated in various scenarios including those presented in this paper. The latter is a relatively new approach, which is very effective in addressing nonlinearity, coupling, input and state saturations.

The low-level vehicle stabilization layer is connected to the higher-level strategy planner using vehicle control language (VCL), a script language interface for autonomous agents as well as human operators to command the host vehicle. Each autonomous agent is a part of a wireless communication network, by which complex tasks may be performed in a coordinated manner.

As benchmark problems, the following scenarios are considered: waypoint navigation, pursuit-evasion, ground target tracking, and vision-based landing. These scenarios exemplify one or more functionalities of the hierarchical multi-agent system. In waypoint navigation, the functionality of the guidance layer using the VCL framework is highlighted. The pursuit-evasion addresses probabilistic reasoning for strategy planning, multi-agent coordination over a wireless network, dynamic VCL operation, and vision-based sensing. The ground target tracking and vision-based landing experiments high-speed position tracking control, target recognition and tracking technology of the onboard vision processing unit as a strategy planner.

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 aforementioned examples. Section 5 summarizes our results.

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 the hierarchical flight management system shown in Fig. 2.

2.1. Sensing

Dynamically changing conditions in the environment and vehicle states are perceived by various onboard sensors. Motion-related information, which is vital for vehicle control and high-level operation, is measured by the onboard navigation sensors such asinertial navigation system (INS) and global positioning system (GPS). Additional sensors such as ultrasonic sensors and laser range-finders are used to acquire the environment-specific information including relative distance from the ground surface, or to detect the objects in the vicinity of the host vehicle. A computer vision system (Sharp, Shakernia, & Sastry, 2001) is used to detect objects of interest based on their color or shape.

2.2. Reasoning and coordination

Fig. 2 shows three types of strategy planners to be implemented for each experiment in Section 4. The appropriate strategy planner for a given mission is selected by a switching layer.

When the current state of the world is not fully measurable, 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, Vidal, Shim, & Sastry, 2001), or, the class of policies to search through is limited (Ng & Jordan, 2000). Algorithms are typically run on real-time operating systems to satisfy hard real-time constraints.

The strategy planner also manages communication networks. Evolved from a simple telemetry for data up/down link, the communication plays a vital role in the real-time coordination and reconfiguration of multiple agents in dynamic environment as a tightly coordinated, reconfigurable, distributed networked intelligence. Moreover, it is desirable to have the support of a high quality-of-service (QoS) wireless communication system with minimal latency, in the presence of ambient noise or signal jamming for secure operation.

2.3. Action

Finally, the UAV is instructed to move to the strategic locations that are computed by the decision making process. In doing so, the UAV should be able to autonomously guide itself through the reference trajectories or waypoints. Each vehicle platform is equipped with stabilizing controllers as will be described in Section 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 and trajectory coordination

This section describes the components at the vehicle-level of the hierarchy for autonomous flight: the vehicle platform (Section 3.1), dynamic model identification (Section 3.2), control and trajectory generation using multi-loop PID (Section 3.3) and a nonlinear model predictive method (Section 3.4).

3.1. Vehicle platform

A UAV is tightly integrated with mechanical and electronic components, including an airframe, navigation sensors, computers, batteries and other onboard sensors, aimed at performing autonomous tasks with minimal intervention by a remote human operator. Berkeley RUAVs are constructed on off-the-shelf radio-controlled helicopters of various sizes and payloads. In the experiments described below, a radio-controlled helicopter, Yamaha R-50 is used. The onboard components are categorized into the followings: (1) flight control computer, (2) navigational sensors, (3) communication module, and (4) onboard power system (Fig. 3).

The onboard flight computer is central to the guidance, navigation, and control of the host vehicle. It is in charge of real-time vehicle control, sensor integration, and inter-agent communication. The flight management software is implemented in the QNX^{TM} real-time operating system. The input to the servo control system is computed at 50 Hz using the flight control algorithms described in Section 3.3.

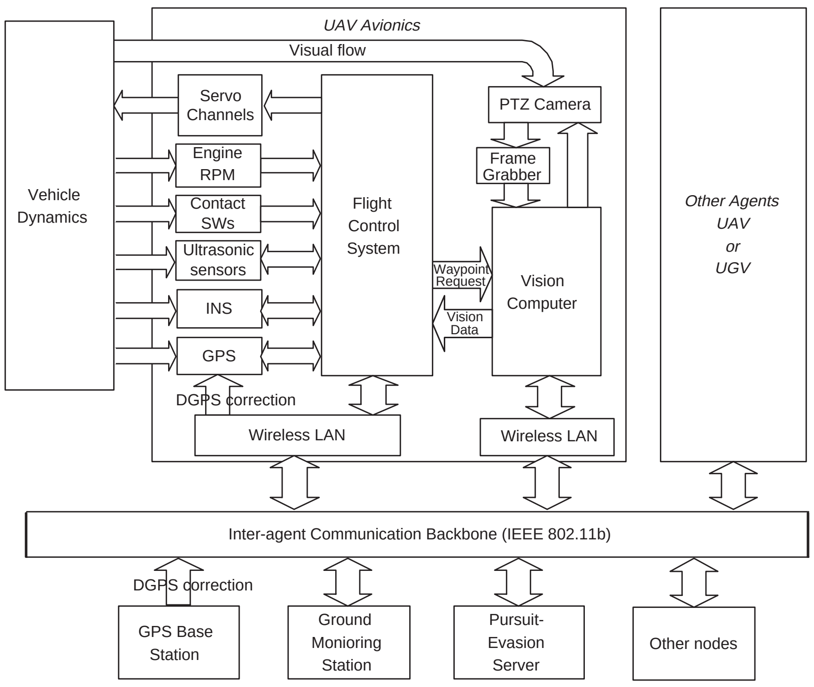

Fig. 3. Berkeley RUAV platform architecture.

The navigation system is built around INS and GPS. INS provides position, velocity, attitude angles and rates at an arbitrarily high rate. A drawback of INS is the unbounded error growing rapidly over time. This can be effectively corrected by an external position sensor such as GPS. Due to the complementary natures of INS and GPS, a combination of these sensors has become a standard configuration for UAVs. In order to acquire the environment-specific information such as the relative distance from the ground or nearby objects, laser range finders, ultrasonic sensors, and vision sensors are used as well.

Berkeley RUAVs based on the Yamaha R-50 are equipped with an onboard vision processing unit (VPU) and a camera mounted on a pan-tilt platform. The VPU tracks markers of special pattern and estimate the relative motion between the camera and the target. For autonomous take-off and landing, a vision-based sensing estimates the relative distance and inclination angles to the marker on the landing spot. The VPU estimate is adjusted with navigation data from the flight computer via a serial link (see Sharp et al. (2001) for the detail on the vision system).

Wireless network is used to achieve the remote accessibility and connectivity among multiple agents. The information flow on the communication link is defined in a standardized message format, which enables the interoperability of heterogeneous agents, i.e., aerial or ground-based agents. This feature is highlighted in the pursuit-evasion example (Section 4.2).

Detailed description on the theoretical and the practical issues in designing and building an RUAV are described by Shim (2000).

3.2. Helicopter dynamics

A helicopter is a highly nonlinear multi-input multi-output (MIMO) system, which is exposed to severe disturbance such as its own rotor wake and wind gusts. The modeling of the helicopter deserves a devoted coverage and the detailed explanation of the dynamic models, from which the proposed control laws are designed, is found in Shim (2000).

The overall dynamics of a RUAV are modeled as a set of nonlinear differential equations, which is divided into the kinematics (Eqs. (1) and (2)) and the system-specific dynamics (Eq. (3)):

$$[\dot{x}^S, \dot{y}^S, \dot{z}^S]^{\mathrm{T}} = R^{B\to S}[\dot{x}^B, \dot{y}^B, \dot{z}^B]^{\mathrm{T}}, \tag{1}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\phi\\ \theta\\ \psi\end{bmatrix} = \begin{bmatrix}1 & \sin\phi\tan\theta & \cos\phi\tan\theta\\ 0 & \cos\phi & -\sin\phi\\ 0 & \sin\phi\cos\theta & \cos\phi\cos\theta\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix}, \tag{2}$$

$$\dot{\mathbf{x}}^D(t) = f_c(\mathbf{x}^D(t), \mathbf{u}(t)), \tag{3}$$

where

$$\mathbf{x} = [\mathbf{x}^K, \mathbf{x}^D]^{\mathrm{T}} \in \mathbb{R}^{n_x},$$
$$\mathbf{x}^K = [x^S, y^S, z^S, \phi, \theta, \psi]^{\mathrm{T}},$$
$$\mathbf{x}^D = [u, v, w, p, q, r, a_{1s}, b_{1s}, r_{fb}]^{\mathrm{T}},$$
$$\mathbf{u} = [u_{a1s}, u_{b1s}, u_{\theta_M}, u_{r_{ref}}]^{\mathrm{T}} \in \mathbb{R}^{n_u}.$$

Here S and B denote spatial and body coordinate. $\dot{x}^B$, $\dot{y}^B$, and $\dot{z}^B$ (u, v, and w respectively, will be used for notational simplicity) denote velocity with respect to the body-coordinate frame. ϕ, θ, and ψ denote roll, pitch, and yaw, and p, q, and r are their rates, respectively. The parameters a_{1s} and b_{1s} are longitudinal and lateral flapping angles, and r_{fb} is the feedback gyro system state (Mettler et al., 1999). The dynamic model (Eq. (3)) has four inputs. u_{a1s} and u_{b1s} control lateral and longitudinal cyclic pitch, respectively. The cyclic pitch changes the individual pitch of each rotor blade during a cycle of revolution to vary the direction of the thrust vector. u_{θ_M} is the servo input for the main rotor collective pitch. The collective control changes the pitch of all blades and hence changes the magnitude of the thrust vector. $u_{r_{ref}}$ controls the magnitude and direction of the tail rotor thrust, which counteracts the anti-torque of the main rotor and thus controls the heading angle. Due to the complexity and the uncertainty inherent to aerodynamic systems, the dynamic model was identified as a whole by applying a parametric identification algorithm to a set of test flight data. A test pilot gives frequency sweeping signals to the instrumented RUAV in longitudinal, lateral, heave and yaw channels in turn, while maintaining the overall stability of the vehicle. The vehicle response is measured by the navigation sensors and downloaded to the ground station via a wireless link. The recorded measurement is conditioned and then processed by prediction error method, a time-domain parametric identification method (Ljung, 1997). The resulted model for Eq. (3) is a linear time-invariant system with states and inputs defined above. Fig. 4 compares the state variables predicted by the identified model, which shows a satisfactory match with the true flight data.

3.3. Stabilization and tracking using multi-loop controller

Based on the identified model in Section 3.2, a stabilizing control law is designed. In the first approach, multiple single-input, single-output (SISO) control loops are designed around the four inputs of longitudinal/lateral cyclic pitches and main/tail collective pitches. This approach has obvious advantages in terms of a simpler structure, straightforward design process, and low computing load. On the other hand, it does not provide a systematic way to account for uncertainty, disturbance, and saturation. Moreover, it has very limited means to alleviate the coupling among channels.

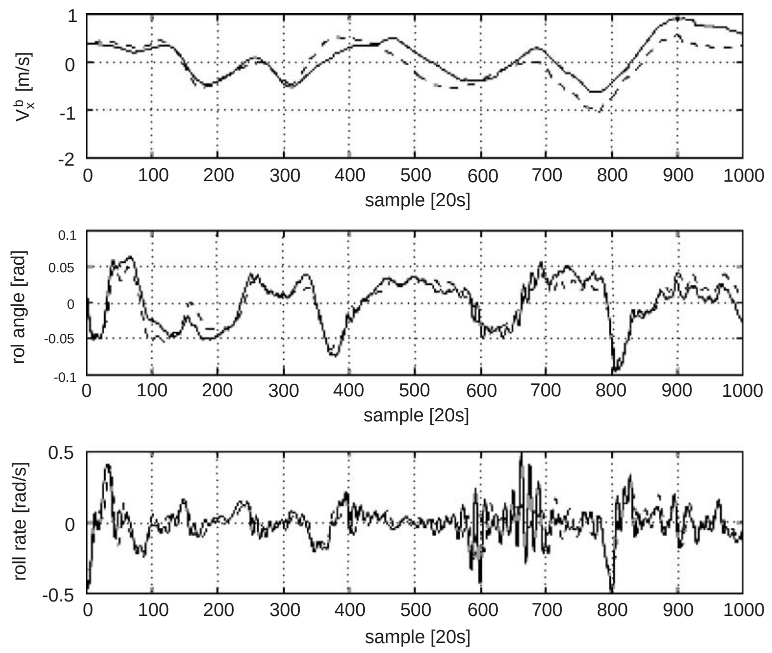

Fig. 4. Identification result: experimental data (solid) vs. prediction by the identified model (dotted). From top, $\dot{x}^B$, ϕ (roll), and p (roll rate), respectively.

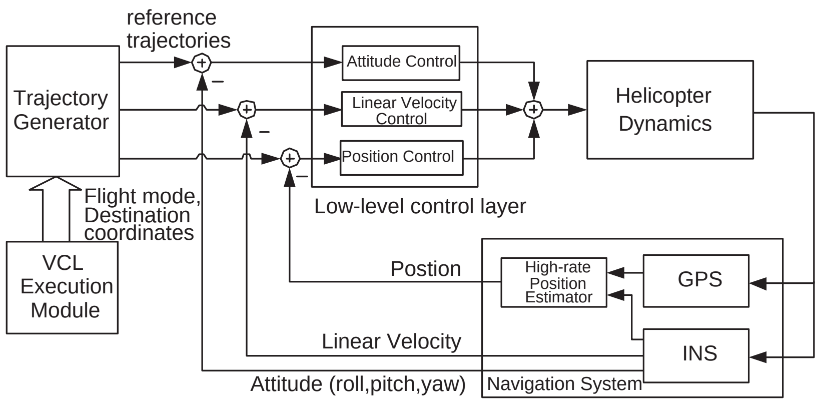

Fig. 5. Multi-loop controller architecture.

The proposed controller consists of three loops: (1) innermost attitude controller, (2) mid-loop linear velocity controller, and (3) outer loop position controller (Fig. 5).

The attitude controller feeds back only the deviation of the roll and pitch angles from the trim condition (nonzero angle needed to maintain an equilibrium), not the noisy angular rates p and q measured by rate gyros. This approach yields a controller that is simpler and more robust to mechanical vibration. The adequate angular feedback gains for roll and pitch channels are determined to have acceptable response speed and damping ratio.

The translational velocity dynamics of small helicopters are unstable with the attitude feedback only. They should be stabilized with velocity feedback, which is determined by a combination of root locus and step response methods.

For hover control, the position control loops in x-, y-, and z-axis are added on top of the linear velocity and attitude feedback. The position control involves internal coordinate transformation to compensate the heading change. The position gains are found by applying the similar methods described above to the augmented RUAV dynamics with velocity and attitude feedback. Finally, integral actions are added to eliminate steady-state errors and trim mismatch.

The vertical and heading dynamics are inherently stable due to the interaction between the inflow and the induced lift. The vertical response is improved by artificial damping using negative velocity feedback. For yaw tracking, the heading error and its integral are fed back on top of the built-in rate gyro system.

In summary, the multi-loop PID (MLPID) control law is given as the following simple equation:

$$\begin{aligned} u_{a1s} &= -K_\phi \phi - K_v v - K_y\ e_y s - K_{Iy} \int e_y s\, \mathrm{d}t, \\ u_{b1s} &= -K_\theta \theta - K_u u - K_x\ e_x s - K_{Ix} \int e_x s\, \mathrm{d}t, \\ u_{\theta_M} &= -K_w w - K_z\ e_z s - K_{Iz} \int e_z s\, \mathrm{d}t, \\ u_{r_{ref}} &= -K_\psi \psi - K_{I\psi} \int e_\psi\, \mathrm{d}t, \end{aligned} \quad (4)$$

where $e_x s$, $e_y s$, and $e_z s$ denote the position error, and e_ψ denotes the heading error.

Fig. 6 shows the experiment result of hovering controller tested on R-50 UAV. The RUAV showed a stable and accurate regulation response with (± 0.3, ± 0.4, ± 0.1 m, $\pm 2^\circ$) accuracy in (x, y, z, ψ)-axis. Roll, pitch, translational velocity in x and y directions are regulated very well altogether.

3.4. Stabilization and tracking using nonlinear model predictive controller

In the previous section, we have shown that the conventional multi-loop control performs reasonably well. In order to improve the tracking performance for complex trajectories by taking into account of nonlinear characteristics, coupling among modes, and input/state saturation, we also consider a nonlinear model predictive controller (NMPTC) as a tracking layer.

At each sample time, a NMPTC computes a finite control sequence, which minimizes a cost function, typically a weighted quadratic sum of states and inputs over a finite horizon. We used a discretized internal model obtained from a partially nonlinear continuous-time model (with nonlinear force terms and full nonlinear kinematic equations).

As for the internal model, Eq. (2) is discretized to

$$\begin{aligned} \mathbf{x}_{k+1} &= f(\mathbf{x}_k, \mathbf{u}_k) \triangleq f_d(\mathbf{x}_k) + B_d \mathbf{u}_k, \\ f_d(\mathbf{x}_k) &\triangleq \mathbf{x}_k + T_s f_c(\mathbf{x}_k), \\ B_d &\triangleq T_s B_c, \end{aligned} \quad (5)$$

where T_s is the sampling time. For tracking, we define a cost function

$$J = \phi(\tilde{\mathbf{y}}_N) + \sum_{k=0}^{N-1} L(\mathbf{x}_k, \tilde{\mathbf{y}}_k, \mathbf{u}_k), \quad (6)$$

$$\phi(\tilde{\mathbf{y}}_N) \triangleq \frac{1}{2} \tilde{\mathbf{y}}_N^{\mathrm{T}} P_0 \tilde{\mathbf{y}}_N, \quad (7)$$

$$L(\mathbf{x}_k, \tilde{\mathbf{y}}_k, \mathbf{u}_k) \triangleq \frac{1}{2} \tilde{\mathbf{y}}_K^{\mathrm{T}} Q \tilde{\mathbf{y}}_k + \frac{1}{2} \mathbf{x}_k^{\mathrm{T}} S \mathbf{x}_k + \frac{1}{2} \mathbf{u}_k^{\mathrm{T}} R \mathbf{u}_k, \quad (8)$$

where $\tilde{\mathbf{y}} \triangleq \mathbf{y}_d - \mathbf{y}$, $\mathbf{y} = C\mathbf{x} \in \mathbb{R}^{n_y}$, $\mathbf{y}_d$ is the desired trajectory, and S is introduced to bound the state variables that do not directly appear in $\mathbf{y}$. By introducing a sequence of Lagrange multiplier vectors $\{\lambda_k \in \mathbb{R}^{n_x}\}_{k=1}^N$, Eq. (6) can be written as

$$J = \phi(\tilde{\mathbf{y}}_N) + \sum_{k=0}^{N-1} L(\mathbf{x}_k, \tilde{\mathbf{y}}_k, \mathbf{u}_k) + \lambda_{k+1}^{\mathrm{T}} [f(\mathbf{x}_k, \mathbf{u}_k) - \mathbf{x}_{k+1}]. \quad (9)$$

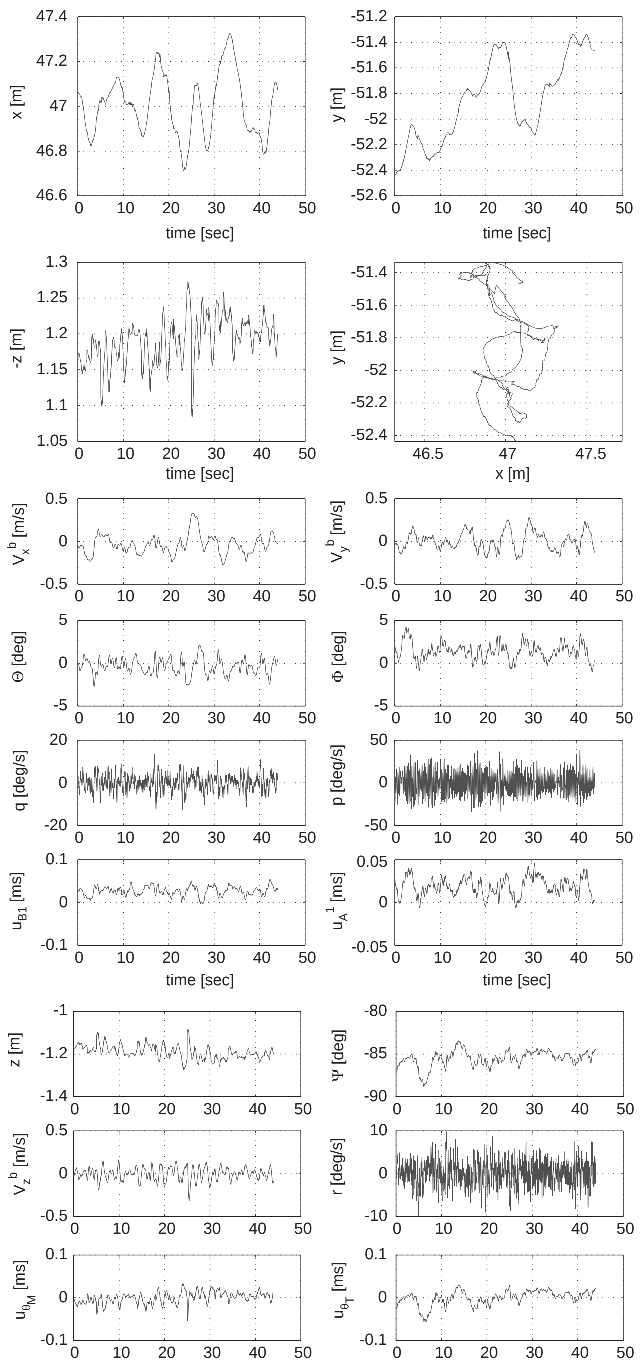

Fig. 6. Experiment result of autonomous hover: the plots on the left column, starting from top, show x^S, y^S, $\dot{x}^B$, θ, q, u_{b1s}, z^S, $\dot{z}^B$, u_{0_M}, and the plots on the left column from top show y^S, x^S vs. y^S, $\dot{y}^B$, ϕ, p, u_{a1s}, ψ, r, $u_{r_{ref}}$, respectively.

By defining the Hamiltonian function as

$$H_k = L(\mathbf{x}_k, \tilde{\mathbf{y}}_k, \mathbf{u}_k) + \lambda_{k+1}^{\mathrm{T}} f(\mathbf{x}_k, \mathbf{u}_k). \tag{10}$$

Eq. (6) can be written as

$$J = \phi(\mathbf{x}_N) - \lambda_N^{\mathrm{T}}\mathbf{x}_N + \sum_{k=1}^{N-1} [H_k - \lambda_k^{\mathrm{T}}\mathbf{x}_k] + H_0. \tag{11}$$

Since we want to choose $\{\mathbf{u}_k\}_0^{N-1}$ that minimizes J, we take a look at

$$\mathrm{d}J = \left[\frac{\partial\phi}{\partial\mathbf{x}_N} - \lambda_N^{\mathrm{T}}\right]\mathrm{d}\mathbf{x}_N + \frac{\partial H_0}{\partial\mathbf{x}_0}\mathrm{d}\mathbf{x}_0 + \frac{\partial H_k}{\partial\tilde{\mathbf{y}}_0}\mathrm{d}\tilde{\mathbf{y}}_0 + \frac{\partial H_k}{\partial\mathbf{u}_0}\mathrm{d}\mathbf{u}_0 + \sum_{k=1}^{N-1}\left[\left\{\frac{\partial H_k}{\partial\mathbf{x}_k} - \lambda_k^{\mathrm{T}}\right\}\mathrm{d}\mathbf{x}_k + \frac{\partial H_k}{\partial\tilde{\mathbf{y}}_k}\mathrm{d}\tilde{\mathbf{y}}_k + \frac{\partial H_k}{\partial\mathbf{u}_k}\mathrm{d}\mathbf{u}_k\right].$$

Choosing

$$\lambda_N^{\mathrm{T}} = \frac{\partial\phi}{\partial\mathbf{x}_N} = -\tilde{\mathbf{y}}_N^{\mathrm{T}} P_0 C, \tag{12}$$

$$\begin{aligned}\lambda_k^{\mathrm{T}} &= \frac{\partial H_k}{\partial\mathbf{x}_k} + \frac{\partial H_k}{\partial\tilde{\mathbf{y}}_k}\frac{\partial\tilde{\mathbf{y}}_k}{\partial\mathbf{x}_k} \\ &= \mathbf{x}_k^{\mathrm{T}} S + \lambda_{k+1}^{\mathrm{T}}\frac{\partial f_k}{\partial\mathbf{x}_k} - \tilde{\mathbf{y}}^{\mathrm{T}} QC\end{aligned} \tag{13}$$

yields

$$\mathrm{d}J = \sum_{k=0}^{N-1}\frac{\partial H_k}{\partial\mathbf{u}_k}\mathrm{d}\mathbf{u}_k + \lambda_0^{\mathrm{T}}\,\mathrm{d}\mathbf{x}_0 \tag{14}$$

and

$$\frac{\partial H_k}{\partial\mathbf{u}_k} = \mathbf{u}_k^{\mathrm{T}} R + \lambda_{k+1}^{\mathrm{T}}\frac{\partial f_k}{\partial\mathbf{u}_k}. \tag{15}$$

With an initial value of the input sequence $\{\mathbf{u}_k^{(0)}\}_0^{N-1}$ obtained using a MLPID controller and a given $\mathbf{x}_0$, $\{\mathbf{x}_k\}_1^N$ are first computed using (5). Then, for $k = N, \ldots, 1$, λ_k are computed recursively using (12) and (13), and for $k = 1, \ldots, N, (\partial H_k/\partial\mathbf{u}_k)$ are computed using (15) and used for the gradient descent. By initializing $\mathbf{u}_k$ at the beginning of the optimization at each time step with the $\mathbf{u}_k$ of the previous time sample, the iteration count reduces significantly.

3.4.1. Trajectory generation and tracking under input/state constraints

To generate physically realizable trajectories, input constraints are enforced by projecting each $\mathbf{u}_k$ into the constraint set. In our helicopter model, this corresponds to $[u_{a1s}, u_{b1s}, u_{\theta_M}, u_{\theta_T}] \in [-1, 1]^4$. State constraints are also incorporated as an additional penalty in the cost function J:

$$S(\mathbf{x}_k) \triangleq \sum_{l=1}^{n_x} S_{ik}\max(0, |x_{i,l}(k)| - x_{i,l}^{\mathrm{sat}})^2. \tag{16}$$

3.4.2. Performance of MPC

Here we evaluate the performance of the nonlinear model predictive tracking controller designed above for a spiral ascent profile shown in Fig. 7 under the impact of model uncertainty. For comparative study, MLPID is put into the same scenario. An additional constraint is imposed on the heading of an RUAV so that its nose pointing toward the center of the spiral trajectory, i.e.,

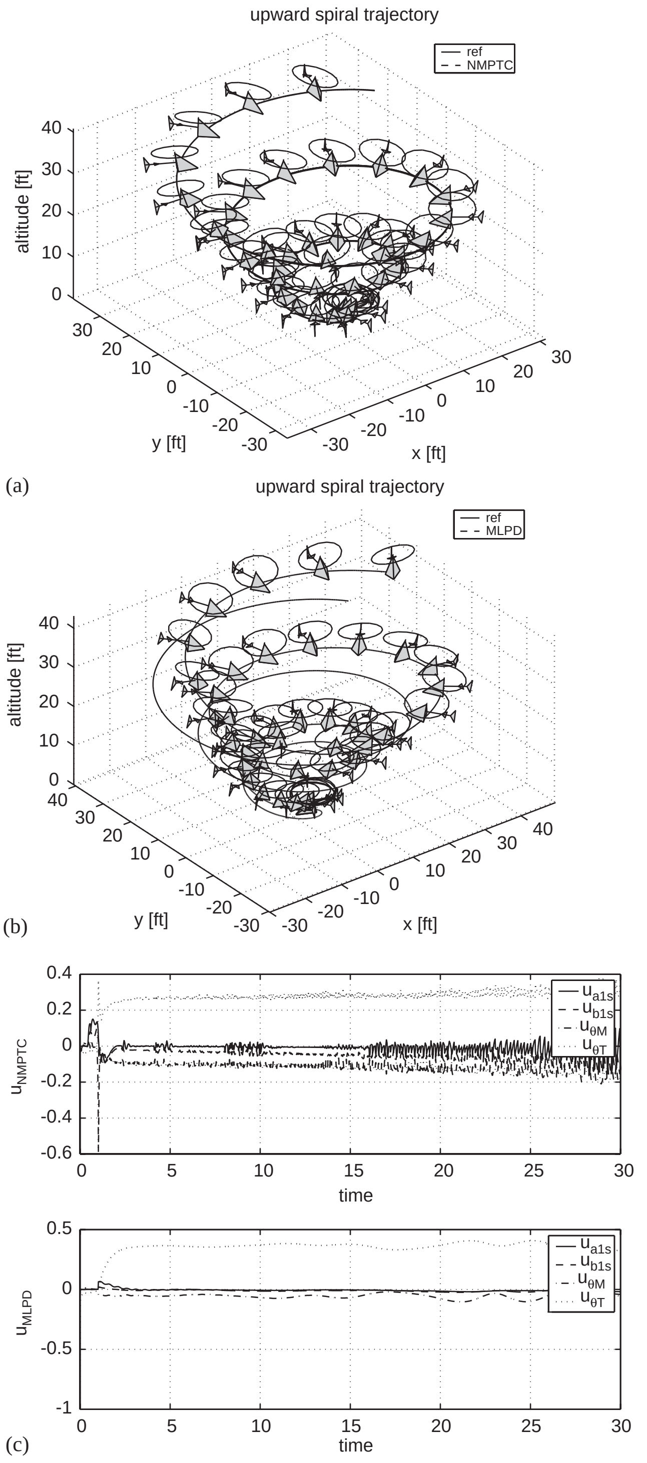

Fig. 7. Tracking of a spiral ascent trajectory in the presence of 20% model uncertainty: Trajectory using (a) a nonlinear model predictive controller (NMPTC), (b) a multi-loop PID (MLPID) controller, and (c) control inputs using NMPTC and MLPID, respectively.

$[x_d^S, y_d^S, z_d^S, \psi_d] = [R(t)\cos(2\pi/10)t,\ R(t)\sin(2\pi/10)t,\ -(4\pi/10)t,\ \pi + (2\pi/10)t]$, for $0 \leqslant t \leqslant 30$ s, $R(t) = 5 + (t/2)$ ft. This particular trajectory is chosen to differentiate the capability of controllers to handle the nonlinear kinematics as well as the multivariable coupling in the system dynamics. In this comparative study, we introduced up to 20% perturbation to all system parameters except for the gravity terms and evaluated the tracking performance for the spiral ascent profile identical to that used in the previous case. In Fig. 7(a), an RUAV controlled by the NMPTC follows the given spiral trajectory and desired heading ψ_d. As shown in Fig. 7(b), when the RUAV is controlled by MLPID controller, the deviation from the spiral trajectory increases as time elapses. Fig. 7(c) presents the tracking error of $x^S(t)$, $y^S(t)$, $z^S(t)$ and $\psi(t)$ under the two controllers. The controller output $\mathbf{u}(t)$ of both cases are shown in Fig. 7(d). The failure of the linear controller to follow complex trajectory is attributed to its deficiency to handle the coupling as well as the nonlinear kinematics of rigid-body motion.

The proposed algorithm has been implemented in C language and tested during the simulation. It has been shown that the C implementation could be reasonably solved in real time on a Intel Pentium III class CPU. Using a dual-computer architecture, the NMPTC algorithm is solved in real-time on a secondary flight computer, while the primary flight computer handles the hard real-time control (Shim, 2000).

3.5. Trajectory generation

A trajectory generation layer is responsible for generating a desired trajectory or a sequence of flight modes and enacting the proper control law in the stabilization/tracking layer. Each helicopter flight from take-off to land can be described as a sequence of flight modes as shown in the diagram in Fig. 8. In this research, a framework called vehicle control language (VCL) is proposed. VCL is implemented as a script language or a binary data format that describes a given mission as a sequence of flight segments, which are associated with the target waypoint, flight mode, and other attributes, as will be shown in Sections 4.1 and 4.2. 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. The VCL code may be generated for the entire flight as a batch file, or command by command for a dynamic operation mode.

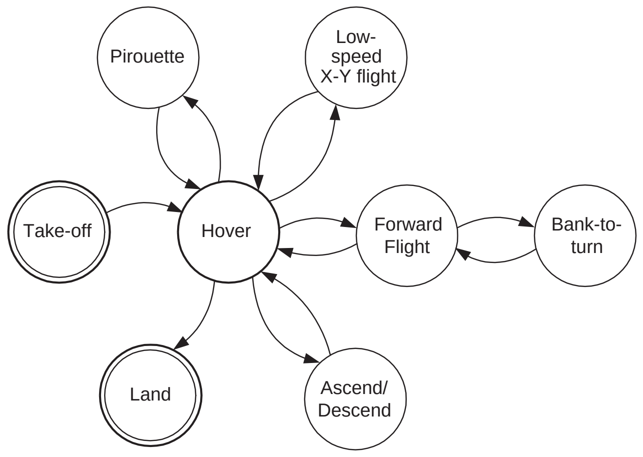

Fig. 8. State transition diagram for flight modes.

4. Experiments with different strategy planners

In this section, the performance of the proposed hierarchical FMS is evaluated in a series of test flights of three distinct scenarios: (1) waypoint 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 onboard vision computer.

4.1. Waypoint navigation: Batch VCL mode

In this mode, the VCL execution module assumes the highest hierarchy in the guidance of the RUAV. For example, when a lawn-mowing pattern followed by a series of waypoints with fixed heading as shown in Fig. 9 is a desired trajectory, the corresponding VCL codes are generated in the strategy planner into a data file. The flight mode, waypoint, 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 control law for the optimal flight mode associated with the target waypoint 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 waypoints 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 $\langle \mathscr{S}, \mathscr{A}, T, \mathscr{Z}, O, R \rangle$:[1]

- $\mathscr{S}$ is a finite set of states of the world, i.e., the configurations of the pursuers and evaders in the given field;
- $\mathscr{A}$ is a finite set of actions, i.e., movement to adjacent cells or stay in the same position;
- $T : \mathscr{S} \times \mathscr{A} \to \mathbf{PD}(\mathscr{S})$ is a transition function. $T(s', s, a_t) = P(\mathbf{s}(t+1) = s' \mid \mathbf{s}(t) = s, \mathbf{a}(t) = a_t)$ is the probability of landing in the state $s' \in S$ under the action $a \in A$ from the state $s \in S$, thus the actuation model is reflected here;
- $\mathscr{Z}$ is a finite set of observations the pursuer can experience of its world, i.e. the location of pursuers themselves and evaders and obstacles within the visibility region;
- $O : \mathscr{S} \times \mathscr{A} \to \mathbf{PD}(\mathscr{Z})$ is the observation function. $O(z_t, s', a_{t-1}) = P(\mathbf{z}(t) = z_t \mid \mathbf{s}(t) = s', \ \mathbf{a}(t-1) = a_{t-1})$ is the probability of making observation z given that the pursuer took action a_t and landed in state s', which reflects the sensing capability;
- $R : \mathscr{S} \times \mathscr{A} \times \mathscr{Z} \to \mathbb{R}$ 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, $\eta^t_{(s)} \triangleq P(\mathbf{s}(t) = s \mid \mathbf{A}_{t-1} = A_{t-1}, \mathbf{Z}_t = Z_t)$ denotes the conditional probability that the world is in state s given $\eta^0_{(s)} \triangleq P(\mathbf{s}_0 = s)$, and the action and observation histories, i.e., $A_{t-1} \triangleq \{a_0, \ldots, a_{t-1}\}$, and $Z_t \triangleq \{z_0, \ldots, z_t\}$. Given that the pursuer observes z_{t+1} after applying a_t, the recursive belief state dynamics can be obtained by applying Bayes' rule:

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

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

```
0:  Takeoff To (0,0,-5)rel;
1:  Hover (0,0,0) rel heading = 270deg duration=7sec;
2:  Fly To (0,-5,0) rel vel = 0.5m/s stopover autoheading;
3:  Hover (0,0,0) rel heading = 0deg duration=7sec;
4:  Fly To (5,0,0) rel vel = 0.5mps stopover autoheading;
5:  Hover (0,0,0) rel heading = 90deg duration=7sec;
6:  Fly To (0,5,0) rel vel = 0.5mps stopover autoheading;
7:  Hover (0,0,0) rel heading =180deg duration=7sec;
8:  Fly To (-5,0,0) rel stopover autoheading;
9:  Hover (0,0,0) rel heading =-90deg duration=7sec;
10:  Fly To (0,-5,0) rel vel = 0.5mps stopover autoheading;
11:  Hover (0,0,0) rel heading =-90deg duration=7sec;
12:  Fly To (0,5,0) rel vel = 0.5mps stopover autoheading;
13:  Hover (0,0,0) rel heading =-90deg duration=7sec;
14:  Move To (0,-3,0) rel vel = 0.5m/s heading=180deg;
15:  Move To (0,3,0)rel vel = 0.5m/s heading=180deg;
16:  Move To (2,0,0) rel vel = 0.8m/s heading=180deg;
17:  Move To (-3,0,0) rel vel = 0.3m/s heading=180deg;
18:  Move To (4,4,0) rel vel = 0.2m/s heading=180deg;
19:  Move To (-1,-2,0) rel vel = 0.2m/s heading=180deg;
20:  Move To (-3,1,0) rel vel = 0.2m/s heading=180deg;
21:  Move To (3,-3,0) rel vel = 0.2m/s heading=180deg;
22:  Hover (0,0,0) relheading =180deg duration=7sec;
23:  Land;
```

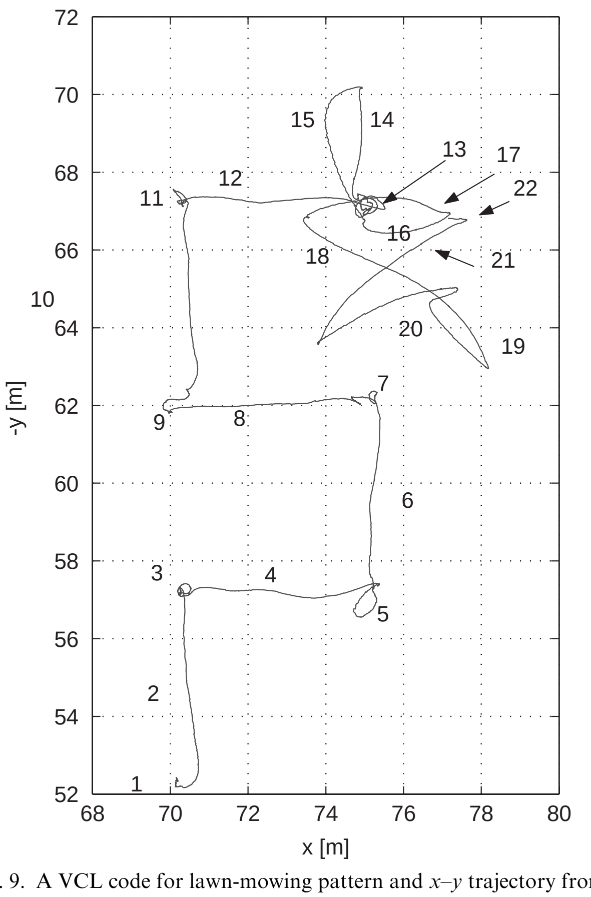

Fig. 9. A VCL code for lawn-mowing pattern and x–y trajectory from the flight experiment.

whose denominator can be treated as a normalizing factor, independent of s'. The strategic planner implements a variety of computationally efficient sub-optimal 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 waypoint for the pursuers (See Kim et al. (2001) for detail on algorithms and experimental results). This position command is sent to the pursuers in the dynamic VCL format over the wireless communication and processed by the VCL execution layer in the flight computer (Fig. 10).

Fig. 11 shows a PEG of one greedy aerial pursuer vs. one ground evader in a 20 m $\times$ 20 m field. The role and the number of participating agents can be easily changed in the scalable architecture. The setup of one aerial pursuer is shown so that the load on the 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 s. This experiment shows that the proposed control law and dynamic VCL are well-suited in a hierarchical control structure for the PEG.

4.3. Target tracking

In this scenario, an RUAV is required to track a moving ground object. The vision computer estimates the relative position of the ground target by extracting a special feature of a marker (Sharp et al., 2001).

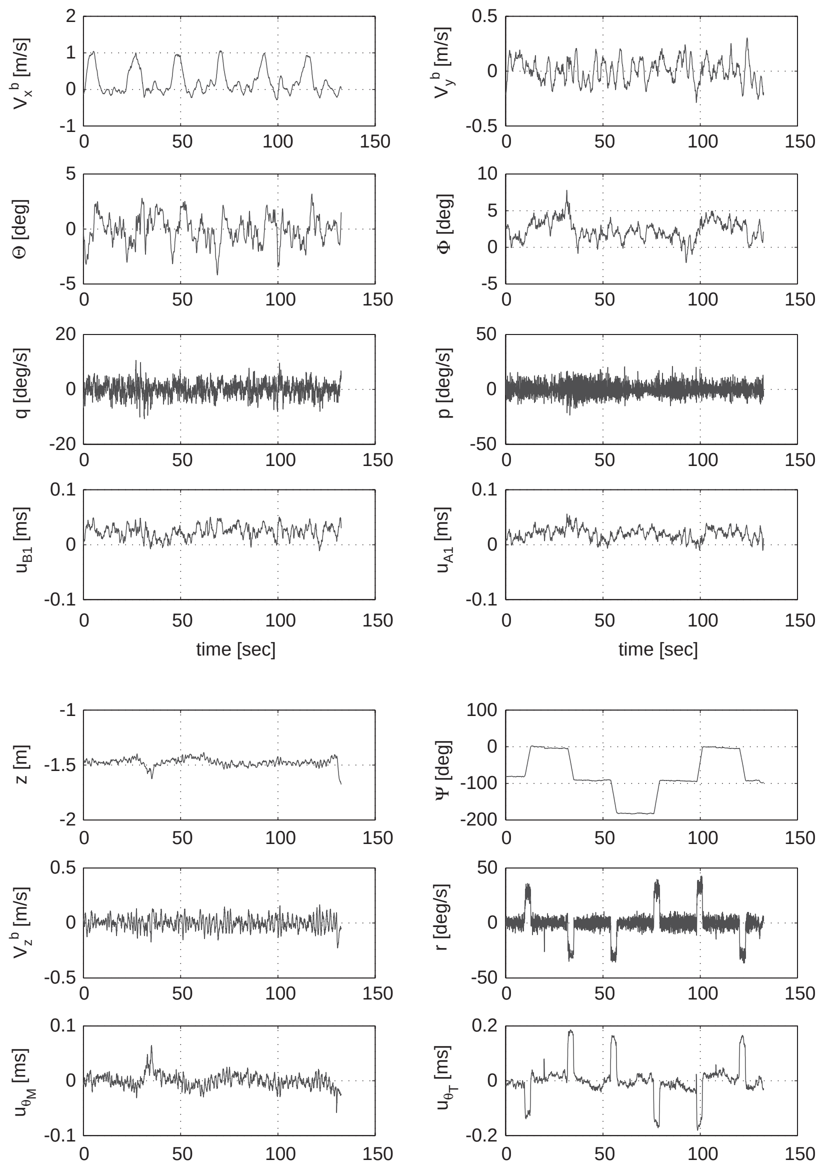

Fig. 10. Control inputs and state variables during a maneuver shown in Fig. 9: the plots on the left column from top shows $\dot{x}^B$, θ, q, u_{b1s}, z^S, $\dot{z}^B$, u_{θ_M}, and the plots on the left column from top shows $\dot{y}^B$, ϕ, p, u_{a1s}, ψ, r, $u_{r_{ref}}$, respectively.

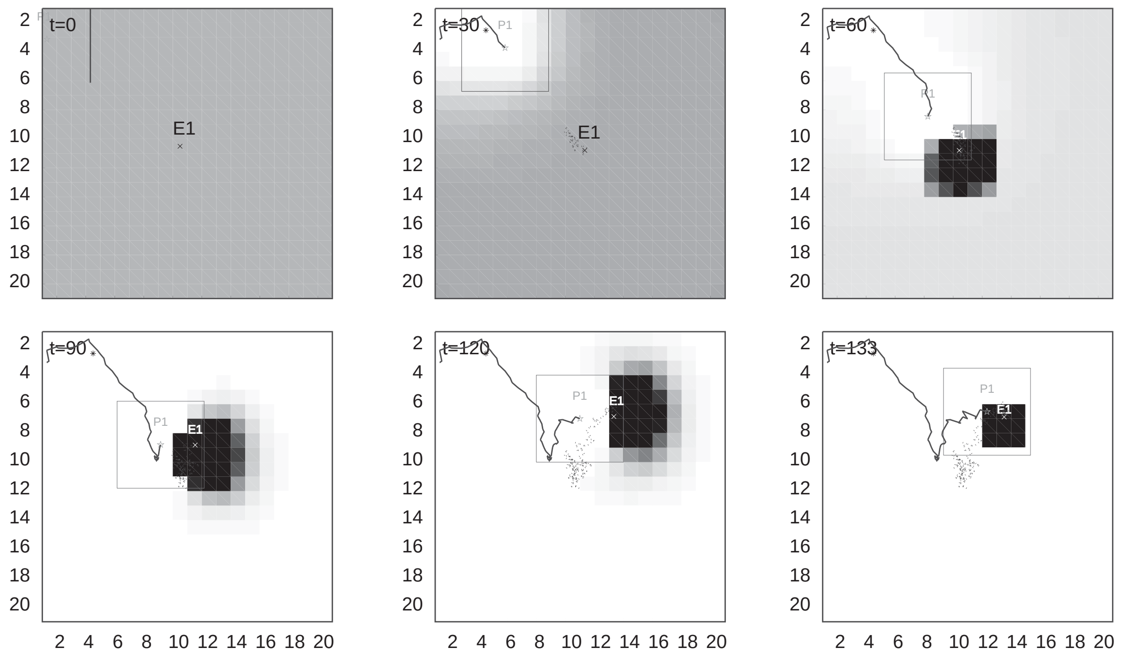

Fig. 11. Snapshots of 1 vs. 1 pursuit-evasion game: trajectory of Pursuer RUAV (P) and Evader UGV (E). t denotes time in second.

High-rate position-tracking requests at 3 Hz are sent to the VCL execution layer in the dynamic VCL format. In Fig. 12, 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 s. The FMS demonstrates its fail-safe feature in this faulty situation by following an *expected* trajectory of targets until the next command is received.

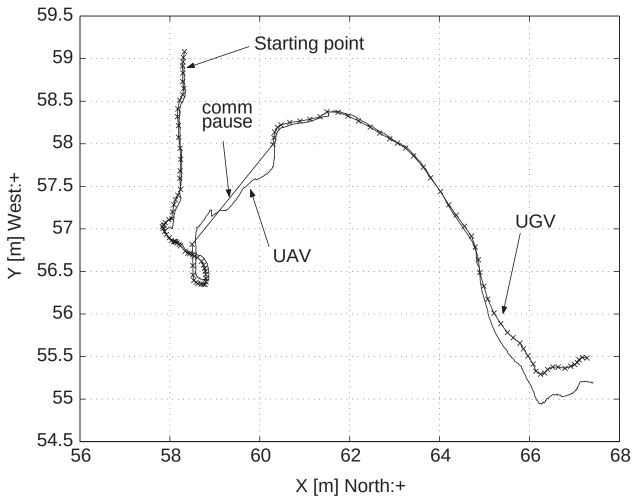

Fig. 12. x–y trajectory of UAV and UGV in tracking experiment.

4.4. Vision-based landing

The landing of a helicopter on a shipdeck poses a significant workload on the pilot, especially when the ship and the helicopter are exposed to a hostile weather. A common solution of winching down the helicopter using a steel cable may be lethal to the shipdeck crew because of the huge amount of the static electricity built up during the flight. The landing algorithm using the vision-based tracking can be very useful not only for the landing of manned helicopters but also for the automatic retrieval of RUAVs. In this experiment, the RUAV is requested to descend onto the shipdeck and then track the deck motion estimated by vision as a precedent step for final touchdown. The experiment is conducted with a motion simulator, which reproduces the shipdeck motion on the sea using the Stuart platform (Fig. 3(a)). The VPU estimates the location of the marker as in Section 4.3, supervises the landing sequence, and sends the reference trajectory $(x(t), y(t), z(t))$ to the flight control computer at a higher rate of 10 Hz for more accurate tracking. Fig. 13(b) shows the landing approach procedure that consists of the following phases: (1) relocation to the initial position above the landing pad, (2) initial descent, (3) acquisition of the marker, (4) descent to the point about 30 cm above the deck, and (5) tracking of the deck motion. In Fig. 13(c), the actual trajectory (solid) of the RUAV is shown with the reference trajectories (dashed) for about 3 min. This experiment demonstrates the successful hierarchical coordination of the flight computer and the VPU through the dynamic VCL interface.

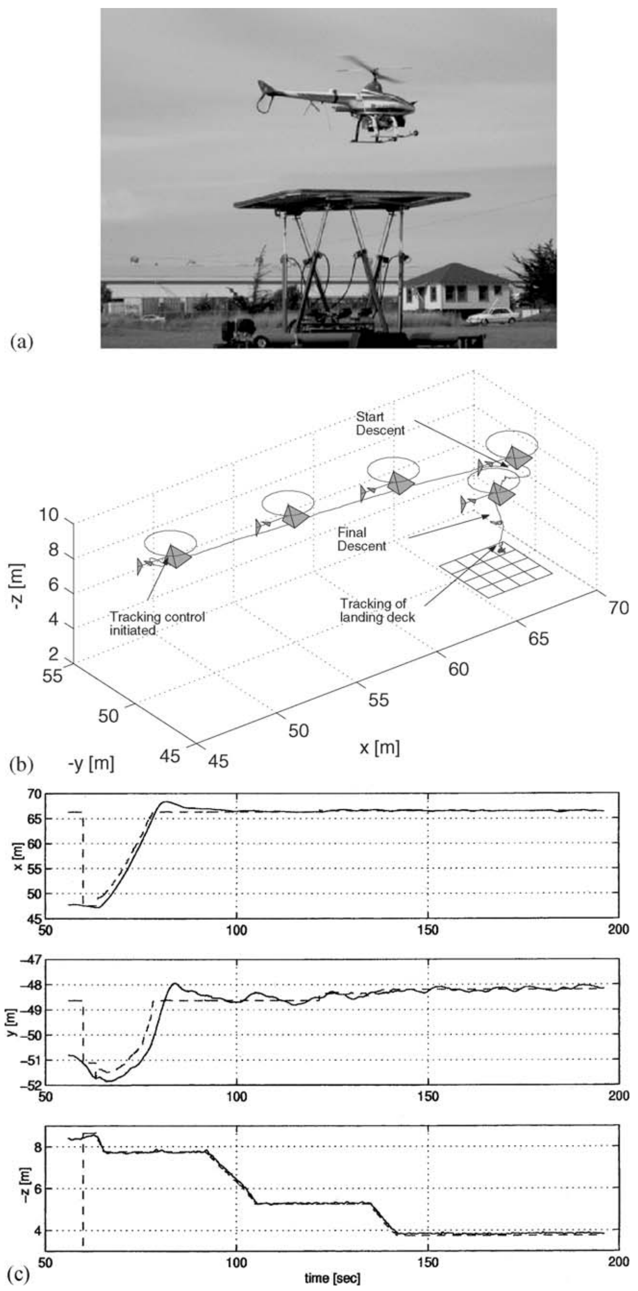

Fig. 13. (a) An RUAV landing on a deck, (b) landing procedure, and (c) reference (dashed) vs. actual (solid) trajectory.

5. Conclusion

This paper presented a hierarchical RUAV flight control system. The vehicle dynamics are identified as a linear model from the test flight data. The tracking control layer is designed using the following two methods: multi-loop PID control and nonlinear model predictive control. The performance of PID controller has been validated in experiments that require a tracking trajectories of moderate difficulty. The nonlinear model predictive control has shown an outstanding tracking performance in the presence of strong coupling and control input saturation at the expense of heavier computation load. The proposed multi-functional flight management system was tested in the following examples: waypoint navigation, pursuit-evasion, tracking of a moving targets and autonomous landing. Further research effort will be made to expand the capability of the flight management system with rich strategy planning logics, increased robustness, and the wider flight envelope, hence narrowing down the gap between current RUAVs and highly maneuverable flying robots with intelligence.

Acknowledgements

The authors would like to thank Hoam Chung, Tullio Celano, Cory Sharp, Omid Shakernia and Peter Ray for their contributions to the experiments. This research was supported by the ONR grant N00014-00-1-0621, and ARO MURI grant DAAD19-02-1-0383.

References

Corban, J. E., Calise, A. J., & Prasad, J. V. R. (1998). Implementation of adaptive nonlinear control for flight test on an unmanned helicopter. In *Proceedings of the 37th IEEE conference on decision and control*. Orlando, FL. (pp. 3641–3646).

Gavrilets, V., Shterenberg, A., Dehaleh, M., & Feron, E. (2000). Avionics system for a small unmanned helicopter performing aggressive maneuvers. In *Digital avionics systems conference*. Philadelphia, PA.

Kim, H. J., Vidal, R., Shim, D. H., & Sastry, S. (2001). A hierarchical approach to probabilistic pursuit-evasion games with unmanned ground and aerial vehicles. In *Proceedings of the 40th IEEE conference on decision and control*. Orlando, FL.

La Civita, M., Papageorgiou, G., Messner, W. C., & Kanade, T. (2002). Design and flight testing of a high-bandwidth H_∞ loop shaping controller for a robotic helicopter. In *Proceedings of the AIAA guidance, navigation, and control conference* number AIAA-2002-4836. Montery, CA.

Ljung, L. J. (1997). *Matlab system identification toolbox user's guide*.

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

Ng, A. Y., & Jordan, M. (2000). PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings of the 17th international conference on uncertainty in artificial intelligence*. Stanford, CA.

Sharp, C. S., Shakernia, O., & Sastry, S. S. (2001). A vision system for landing an unmanned aerial vehicle. In *IEEE international conference on robotics and automation*. Seoul, Korea. (pp. 1720–1727).

Shim, D. H. (2000). *Hierarchical control system synthesis for rotorcraft-based unmanned aerial vehicles*. Ph.D. thesis. University of California, Berkeley.

Shim, H., Koo, T. J., Hoffmann, F., & Sastry, S. (1998). A comprehensive study of control design for an autonomous helicopter. In *Proceedings of the 37th IEEE conference on decision and control*. Tampa, FL. (pp. 3653–3658).