

Hybrid Modeling and Experimental Cooperative Control of Multiple Unmanned Aerial Vehicles*

Selcuk Bayraktar[†], Georgios E. Fainekos[‡] and George J. Pappas[‡]

Technical Report MS-CIS-04-32
Department of Computer and Information Science
University of Pennsylvania

December 10, 2004

*Research is partially supported by the Army Research Office MURI Grant DAAD 19-02-01-0383 and NSF ITR 0324977

[†]Laboratory for Information and Decision Systems, MIT, E-mail: selcuk@mit.edu

[‡]GRASP Laboratory, University of Pennsylvania, Philadelphia, PA 19104, USA. E-mail: {fainekos,pappasg}@grasp.cis.upenn.edu

Abstract

Recent years have seen rapidly growing interest in the development of networks of multiple unmanned aerial vehicles (U.A.V.s), as aerial sensor networks for the purpose of coordinated monitoring, surveillance, and rapid emergency response. This has triggered a great deal of research in higher levels of planning and control, including collaborative sensing and exploration, synchronized motion planning, and formation or cooperative control. In this paper, we describe our recently developed experimental testbed at the University of Pennsylvania, which consists of multiple, fixed-wing UAVs. We describe the system architecture, software and hardware components, and overall system integration. We then derive high-fidelity models that are validated with hardware-in-the-loop simulations and actual experiments. Our models are hybrid, capturing not only the physical dynamics of the aircraft, but also the mode switching logic that supervises lower level controllers. We conclude with a description of cooperative control experiments involving two fixed-wing UAVs.

Keywords: *Hybrid modeling, multiple UAVs, experimental formation flying.*

1 Introduction

The control systems community has historically been motivated by challenging problems in the aerospace industry. Modern aerospace applications call for increasing levels of autonomy from decreasing (in size) aerial and space vehicles, which are frequently networked.

This modern view of future aerospace vehicles, whether civilian or military, has directed a substantial amount of recent research efforts in the areas of interconnected systems, cooperative control, formation control, control with communication constraints, and the relationship between dynamic network topologies and control. The references cited in [1]–[3] can serve as a partial survey of much of the related literature.

The developments on the theoretical front have been followed with similar achievements on the experimental side. However, most experimental results have focused on single aerial vehicles. In particular, we have witnessed autonomous or aggressive control of helicopters [4]–[7], vision-based landing of helicopters [8], blimps [9] and formation flight of blimps [10] as well as fixed wing planes [11]–[14].

Even though the theoretical autonomous formation flying (AFF) problem is well studied [1]–[3], [16]–[19], the experimental control of multiple fixed-wing aircraft is in its infancy. To the authors’ best knowledge, the only published experimental results concerning autonomous formation flying of autonomous aerial vehicles (UAVs) are [20]–[24]. In [22] and [23], the authors investigate the benefits of precise AFF to the reduction of fuel consumption. In [24], the author attempts AFF for aerial autonomous refueling purposes and/or for docking a smaller UAV to a mother UAV. Other experiments with UAVs that do not include formation flying and which concern collision avoidance and UAV coordination can be found in [25] and [26].

In this paper which comes as an extension to [20], we describe in more detail the experimental testbed of the GRASP lab at the University of Pennsylvania, which consists of multiple, fixed-wing UAVs (Fig. 1 and Fig. 2). Our testbed has been rapidly developed by integrating of-the-shelf solutions for lower levels of control. This has allowed us to bypass the typically long development of (by now mature) low level controllers, and has enabled algorithm development and experimentation at more unexplored higher levels of UAV planning and operation, which include multi-UAV planning and control. The success of this approach has been demonstrated with successful AFF experiments and with successful cooperation experiments between an unmanned ground vehicle (UGV) and a PennUAV [27] and [28].

In Section 2, we describe the system, software, and hardware architecture of our recently developed testbed. Section 3 develops a higher-level hybrid model of each UAV. Our hybrid models capture both high level abstractions of the aircraft dynamics, as well as mode switching logic that supervises the switching among lower level controllers. The hybrid models derived are used in high level algorithm development. The algorithms are then tested on high fidelity (hardware-in-the-loop) multi-UAV simulation environment, described in Section 4, which contains accurate aerodynamic models of our UAVs. In Section 5, we present

the algorithm used for the formation flight experiment and in Section 6, we conclude this report with the actual cooperative control experiments performed in August 2003 at Fort Benning, Georgia.



Figure 1: PennUAVs: Two Piper J3 cub model airplanes

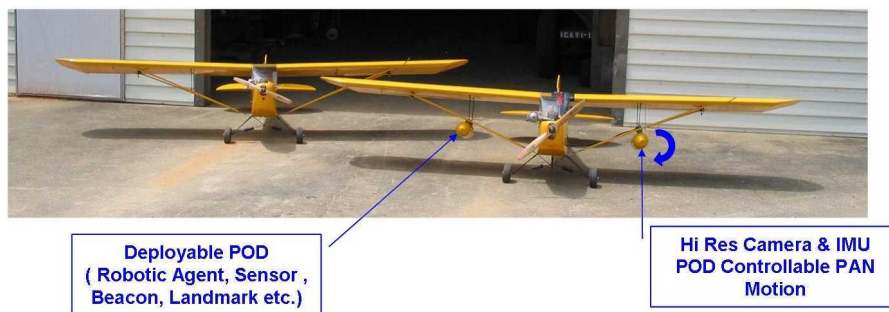


Figure 2: PennUAVs: External Payloads (POD)

2 System Description and Architecture

Fig. 1 and Fig. 2 illustrate the Unmanned Aerial Vehicles (UAVs) recently developed at the GRASP Laboratory of the University of Pennsylvania. Each UAV consists of an airframe and engine, avionics package, onboard laptop and additional sensing payload. We briefly describe the basic components of our UAVs, as well as the overall system architecture.

2.1 UAV Airframe and Payload

The airframe of each UAV is a quarter scale Piper Cub J3 model airplane with a wing span of 104 inches (~ 2.7 m) (see Fig. 1). The powerful glow fuel engine has a power rating of 3.5 HP, resulting in a maximum cruise speed of 60 knots ($\sim 30m/s$) at altitudes up to 5000 feet (~ 1500 m) and a flight duration of 15 - 20 minutes.

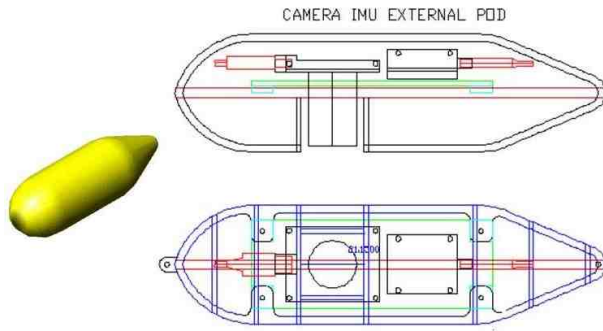


Figure 3: The design of the Camera-IMU Pod. CAD drawing showing the placement of the IMU and the high resolution camera.

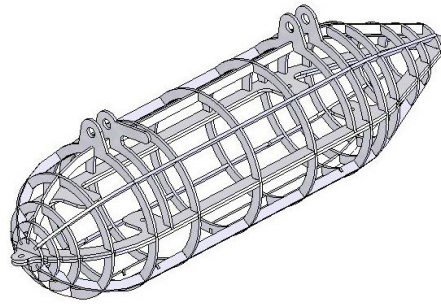


Figure 4: The modular design of the Pod allows for fast production time using CNC machines and for easy assembly.

The airframe-engine combination enables having significant scientific payload on board. Fig. 2 shows pods that have been installed underneath each side of the wing containing high resolution cameras, IMUs, as well as deployable sensors, beacons, and landmarks. More precisely, each UAV can carry the following internal and external payloads:

- Onboard Laptop PC Dell X200
- IMU 3DM-G from MicroStrain. Includes three angular rate gyros with three orthogonal DC accelerometers, three orthogonal magnetometers, multiplexer, 12 bit A/D converter and embedded microcontroller to provide the three orientation angles (pitch, roll, yaw) in dynamic and static environments.
- External GPS Navistar GPS receiver from CMC electronics. 10 Hz raw data output.
- Camera DragonFly IEEE-1394 1024 × 768 at 15 fps from *Point Grey Research*.
- Custom designed camera-IMU Pod (see Fig. 3) includes the IMU and the camera mounted on the same plate. The plate is soft mounted on four points inside the pod. Furthermore, the pan motion of the pod can be controlled through an external user PWM port on the avionics.
- Custom designed deployable Pod could be used to carry sensors, beacons, landmarks or even robotic agents. It can be deployed with a RC servo connected to external user PWM port on the avionics. In the future, each UAV can also serve as a mothership for smaller, lighter micro-UAVs.

2.2 UAV Avionics and Ground Station

Each UAV is controlled by a highly integrated, user customizable Piccolo avionics board which is manufactured by *CloudCap Technologies* [29]–[31]. The avionics board comes

equipped with the core autopilot, a sensor suite which includes GPS, Inertial Measurement Unit consisting of three gyros, three accelerometers and two pressure ports one for barometric altitude one for true airspeed. A 40 Mhz embedded Motorola MPC 555 Power PC receives the state information from all sensors and runs core autopilot loops at a rate of 20 Hz commanding the elevator, ailerons, rudder and throttle actuators as well as external user payload ports.

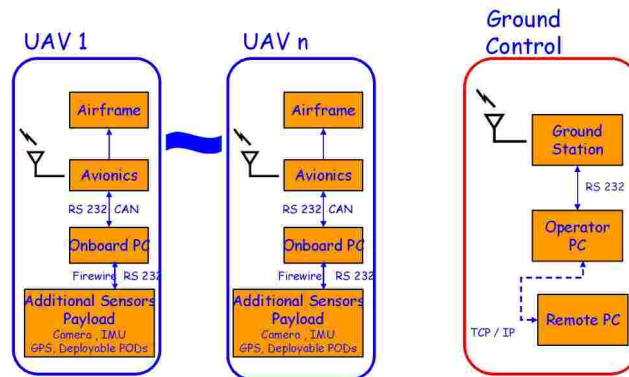


Figure 5: Multi-UAV and Ground Station Functional Architecture

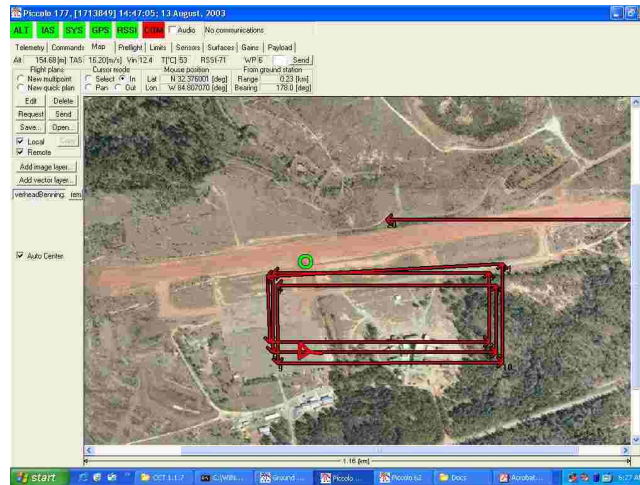


Figure 6: Ground Station Operator Interface showing flight plan and actual UAV position (August 2003, Fort Benning, Georgia).

2.3 System Architecture

Each UAV continuously communicates with the ground station. The communication occurs at 1 Hz and the range of the communication can reach up to 6 miles. The ground station performs differential GPS corrections, and updates the flight plan, which is a sequence of

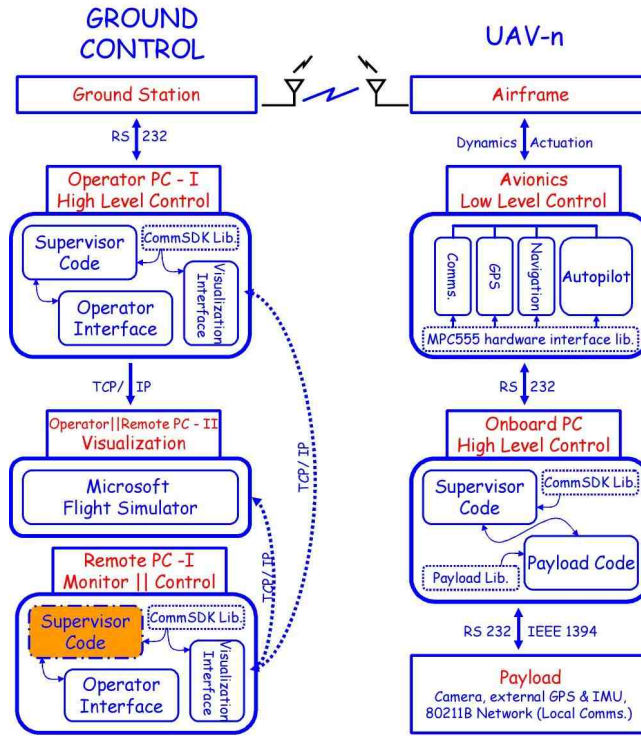


Figure 7: Software Components and Software Architecture.

three dimensional way-points connected by straight lines. The UAVs can also be commanded in a similar way from a supervisory controller (residing on-board the UAV laptop), allowing further decentralization in the physical layer of the architecture (see Fig. 5).

The ground station can concurrently monitor up to 10 UAVs. Direct communication between UAVs can be emulated through the ground or using the local communication channel on the UAVs (80211b - wireless network card). The ground station has a friendly operator interface program (shown in Fig. 6) which allows the operator to monitor flight progress, obtain telemetry data or dynamically change the flight plans using geo-referenced maps. Furthermore the operator interface program can act as a server and enable multiple instances of the same software to communicate over a TCP/IP connection. This allows us to monitor or command and control the experiment in real-time, remotely. For an overview of the software architecture see Fig. 7. Also with the custom interface we developed, the experiment can be visualized in real time or offline in *Microsoft Flight Simulator* with realistic scenery.

2.4 Development Philosophy

The integration of off-the-shelf components for the airframe, engine and avionics has resulted in rapid development and deployment for each UAV. It should be noted that the assembly of an operational UAV is a matter of 6–8 days maximum. This has also allowed us to

quickly experiment with traditionally less-researched areas of coordinated motion planning and sensing while quickly developing the necessary infrastructure in the more established areas of sensing and control.

3 High-Level Hybrid UAV Model

Our goal in this section, and one of the main goals of this paper, is to derive a hybrid model of the Piccolo-controlled dynamics, coupled with the mode (or waypoint) switching logic. Fig. 8 shows the main components that need to be modeled, which include the aircraft dynamics (which are controlled by the Piccolo avionics board), a sensor model, as well as the mode switching logic. We describe each component individually and we model the integrated system using hybrid models.

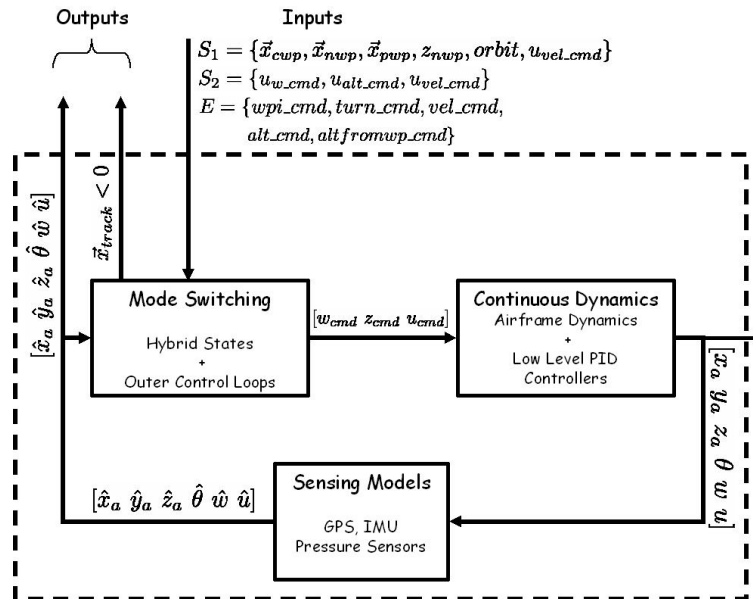


Figure 8: Hybrid UAV Control Loop.

3.1 Piccolo-Controlled UAV Dynamics

The autopilot in the Piccolo avionics is responsible for the low level control, stabilization and, also, the flight plan navigation and tracking of the waypoints. The autopilot consists of seven PID loops and a turn compensator (see [29] and Table 1). The inner control loops regulate, among other things, airspeed, altitude, turn rate. The low-level inner control loops allows us to abstract away lower level, detailed dynamics. The longitudinal dynamics of the system, the altitude and velocity states are modeled as decoupled first order differential equations with saturation constraints. The simplified equations that model the (Piccolo-)controlled physical continuous UAV dynamics are as follows:

Autopilot loops	Inputs	Outputs
Airspeed	Airspeed from dynamic pressure port	Drives elevator control surface to maintain a commanded dynamic pressure (airspeed)
Altitude	Barometric Altitude from static pressure port	Drives throttle to maintain commanded altitude
Turn Rate	Yaw rate from yaw gyro	Aileron, rudder control surfaces to maintain a commanded turn rate
Tracker (Line Segment)	Position and track from GPS sensor	Drives the turn rate command of the turn rate loop to achieve desired track targets
Roll Damper (or Turn Rate)	Estimated Roll Angle	Uses aileron control surfaces to damp roll disturbances and can be used as an alternative turn rate control
Pitch Damper	Estimated pitch through pitch gyro	Uses elevator control surface to regulate pitch oscillations
Yaw Damper	Yaw rate from yaw gyro	Drives rudder control surface to regulate yaw oscillations
Turn Compensator	Commanded turn rate from tracker loop or user and pitch rate from pitch gyro	Uses elevator control surface to maintain the target airspeed command during entering or exiting turns.

Table 1: Autopilot loops

$$\begin{aligned}
\dot{x}_a &= u \cos \theta \\
\dot{y}_a &= u \sin \theta \\
\dot{u} &= 1/\tau_u(-u + u_{cmd}), \underline{u} \leq \dot{u} \leq \bar{u} \\
\dot{\theta} &= \omega
\end{aligned} \tag{1}$$

$$\begin{aligned}
\dot{\omega} &= 1/\tau_\omega(-\omega + \omega_{cmd}) \\
\dot{z}_a &= 1/\tau_z(-z_a + z_{cmd}), \underline{z} \leq \dot{z} \leq \bar{z}
\end{aligned} \tag{2}$$

where (x_a, y_a, z_a) is the position of the UAV in the space with respect to some world frame, u is the UAV velocity, ω is the turn rate of the UAV, and θ is the angle defined within $-\pi \leq \theta \leq \pi$. External inputs include u_{cmd} , ω_{cmd} , and z_{cmd} where the subscript cmd defines controlled inputs. The time constants are τ_u , τ_ω , and τ_z respectively. Note that the inner control loops result in abstracted dynamics that decouple the planar dynamics from the altitude dynamics. As wind sensors are available on board, the above model can be easily extended to include (and compensate for) the effect of wind.

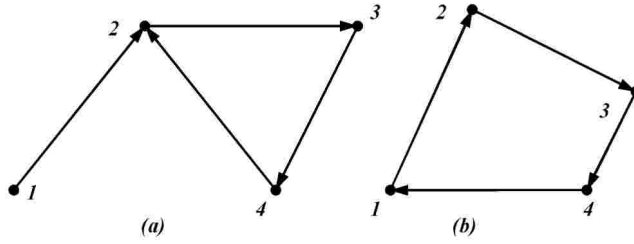


Figure 9: Valid flight plans. (a) In this flight plan, WP 1 does not have a preceding waypoint. (b) Usual form of flight plans for autonomous flight.

3.2 Sensing model

The sensors due to way the control laws and sensing are implemented in the avionics are modeled as a first order hold (with delay or not) where T_s is the sampling period, $k = \lfloor t/T_s \rfloor$ (where $\lfloor \cdot \rfloor$ is the floor function), τ_{dx} , τ_{dy} , $\tau_{d\theta}$ are the delays and

$$\begin{aligned}
 \hat{x}_a(t) &= x_a(kT_s - \tau_{dx}) \\
 \hat{y}_a(t) &= y_a(kT_s - \tau_{dy}) \\
 \hat{u}(t) &= u(kT_s) \\
 \hat{w}(t) &= w(kT_s) \\
 \hat{\theta}(t) &= \theta(kT_s - \tau_{d\theta})
 \end{aligned} \tag{3}$$

$$\hat{z}_a(t) = z_a(kT_s) \tag{4}$$

3.3 Piccolo-High Level Controllers

The main objective of the high level controller is control the UAV to fly a flight plan, which is a sequence of waypoints. The avionics has the ability to store up to 100 waypoints. Each waypoint consists of latitude, longitude, and altitude of the waypoint. The flight plan can be traversed at various desired airspeeds. Note that the flight plans must be closed, that is at some point a next waypoint entry of a waypoint must point to a waypoint that it is already in the flight plan (see Fig. 9). Also note that the waypoints of the flight plan that is being executed cannot be altered. Furthermore, the controller has the capability of circling around waypoints (holding pattern), if desired.

The high level controller completes the flight plan by using one airspeed controller, one altitude controller, and two lateral controllers, one focusing on line segment tracking, and one on circling around waypoints (see [32]). These controllers are orchestrated by the waypoint switching logic, which determines which waypoint segment is active. We now describe the controllers and the waypoint switching logic.

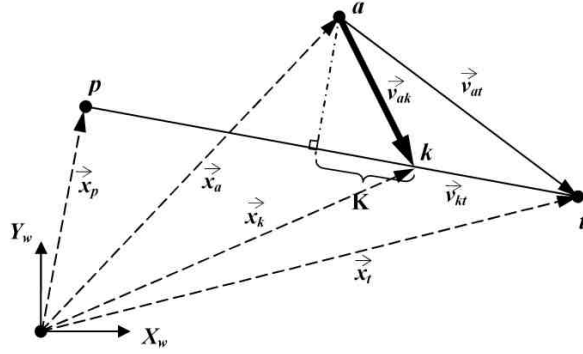


Figure 10: Inertial frame model. X_w, Y_w is the world coordinates system. p is the previous waypoint, t is the target waypoint and a the current position of the aircraft.

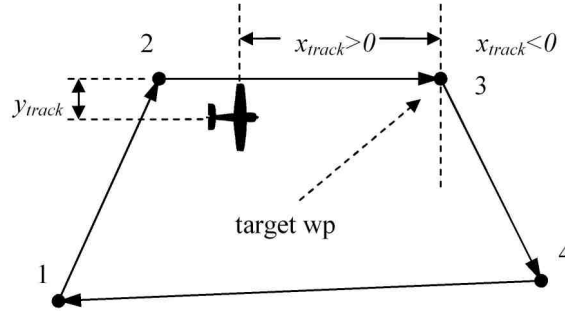


Figure 11: Tracking a flight plan. The flight plan consists of four waypoints 1,2,3,4. The UAV is currently tracking line segment 2–3

Lateral Controller - Line segment tracker

The lateral control law is trying to drive to zero the angle between the actual heading of the UAV and the desired heading. The desired heading is given by the position of the aircraft \vec{x}_a , the line segment to be tracked \vec{v}_{kt} and the track convergence parameter K . In order to determine the desired heading of the UAV, we just need to express the vector \vec{v}_{ak} in terms of vectors \vec{x}_t and \vec{x}_p (see Fig. 10).

$$\vec{v}_{ak} = \vec{x}_t - l_{\vec{v}_{kt}} \frac{\vec{x}_t - \vec{x}_p}{\|\vec{x}_t - \vec{x}_p\|} - \vec{x}_a \quad (5)$$

$$l_{\vec{v}_{kt}} = \vec{v}_{at} \cdot \frac{(\vec{x}_t - \vec{x}_p)}{\|\vec{x}_t - \vec{x}_p\|} - K \quad (6)$$

Where \vec{x}_p is the position of the preceding waypoint and \vec{x}_t is the position of the target waypoint. Even though the actual coordinates of each WP as well as the position of the UAV lie in \mathbb{R}^3 , for the lateral control law we consider only the xy plane parallel to earth's

plane. Hence, from now on all the vectors will be in \mathbb{R}^2 , i.e. $\vec{x}_p, \vec{x}_t, \vec{x}_a = (\hat{x}_a, \hat{y}_a) \in \mathbb{R}^2$, unless we explicitly state otherwise. Finally, K is the track convergence parameter ($K > 0$) and it should be chosen in such a way such that it representative of the nominal vehicle turn radius [32] and [33].

The position x_{track} of the UAV along the track (see Fig. 11) is given by the following equation:

$$\vec{x}_{track} = \frac{(\vec{x}_t - \vec{x}_p)^T (\vec{x}_t - \vec{x}_a)}{\|\vec{x}_t - \vec{x}_p\|} \quad (7)$$

The desired angle θ_{des} for the UAV is:

$$\theta_{des} = atan2(y_{ak}, x_{ak}) \quad (8)$$

where x_{ak} and y_{ak} are the components of the vector $\vec{v}_{ak} = x_{ak}, y_{ak}$ (5). The error to be driven to zero is:

$$\theta_e = \theta_{des} - \hat{\theta} \quad (9)$$

which is modified by the following relations in order to keep the error in the range $[0, 2\pi)$:

$$\begin{aligned} \text{if } \theta_e > \pi & \quad \text{then } \theta_e = \theta_{des} - \hat{\theta} - 2\pi \\ \text{if } -\pi \leq \theta_e \leq \pi & \quad \text{then } \theta_e = \theta_{des} - \hat{\theta} \\ \text{if } \theta_e < -\pi & \quad \text{then } \theta_e = \theta_{des} - \hat{\theta} + 2\pi \end{aligned} \quad (10)$$

The following PID control is used for the command input. The actual implementation of this filter is in discrete time with integrator antiwindup nonlinearities however for making it intuitively obvious we present it here in continuous form.

$$\omega_{cmd} = K_p \theta_e + K_d \dot{\theta}_e + K_i \int \theta_e dt \quad (11)$$

Moreover the controller bounds the bank angle, which also induces bounds on the control input depending on velocity (u) by:

$$|\omega_{cmd}| \leq \frac{g \tan \phi_{limit}}{u} \quad (12)$$

where ϕ_{limit} is the bank limit and g gravitational acceleration.

Lateral Controller - Circle Track

When the specified waypoint is a circle waypoint the UAV circles around the waypoint at a constant radius defined by the parameter K [32] until a new waypoint index change command arrives. In this mode the outer track loop simply projects the current position of the UAV radially on to the circle and calculates the tangent to the circle at that point. Then by simply

using the line track control law the UAV flies the line segment defined by the projected point and the tangent vector. The \vec{x}_p and \vec{x}_t vectors get updated periodically with frequency $1/T_s$.

$$\vec{v}_{radial} = \frac{\vec{\hat{x}}_a - \vec{x}_{wp}}{\|\vec{\hat{x}}_a - \vec{x}_{wp}\|} \quad (13)$$

where \vec{x}_{wp} is the position vector of the waypoint to be orbited.

$$\vec{x}_p = \vec{x}_{wp} + K\vec{v}_{radial} \quad (14)$$

where K is the track convergence parameter.

$$\vec{x}_t = \vec{x}_p + \begin{pmatrix} -v_y \\ v_x \end{pmatrix}_{radial} \quad (15)$$

3.4 Waypoint Switching Logic

The controllers described above are subject to supervisory switching logic. The switching logic arises in two forms. First, the system needs to decide whether the waypoint has been reached or not, and then reset the controllers with information from the next waypoint in the sequence. Second, the user can dynamically update the waypoints and the UAV switches to the assigned waypoint (WP). In this case the UAV makes decisions as to whether it should fly the old flight plan or switch to the new flight plan.

If the UAV is already executing a flight plan then it switches to the next waypoint when $x_{track} < 0$ (see Fig. 11). For example assume that the UAV tracks the line segment (2–3) – that is when the waypoint index (WPI) has the value 3 and the preceding waypoint is 2 – then when it flies over the target WP 3 x_{track} becomes less than 0 and the waypoint index (WPI) is updated to 4.

If the user or the supervisory control orders a new waypoint then the UAV does not always fly directly to the new waypoint but decides its flight plan according to the following logic (see Fig. 12):

1. If $0 < x_{track} < \|\vec{v}_{pt}\|$ where \vec{v}_{pt} is the vector between the positions of the preceding waypoint \vec{x}_p and the target waypoint \vec{x}_t , then the UAV tracks v_{pt} (region 1).
2. If the UAV is at a position where $x_{track} > \|\vec{v}_{pt}\|$, then it flies directly to the new waypoint using as preceding waypoint its current position (region 2).
3. If the UAV is at a position where $x_{track} < 0$, then it flies directly to the new waypoint using as preceding waypoint its current position (region 3).
4. If the new waypoint does not have a preceding waypoint, then the UAV flies directly to the new waypoint using as preceding waypoint its current position. This is case (a) in Fig. 9. Note: this case does not occur in autonomous fly as in this case all the waypoints have preceding waypoints.

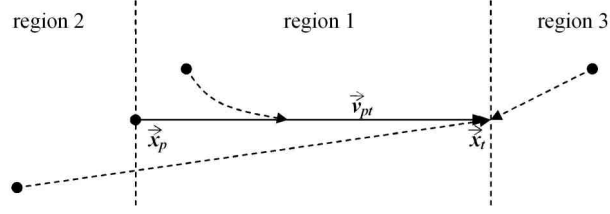


Figure 12: Regions used in waypoint switching logic

In our modeling of the waypoint list storage unit, we use an 100×6 array where we store the following information in each column: (1) x position, (2) y position, (3) z position, (4) next waypoint, (5) preceding waypoint and (6) orbit. The xyz coordinates of the UAV are according to some fixed world frame.

The waypoints are numbered from 0 to 99 and are used according to the following table: 0 to 89 free for the definition of flight plans 90 to 94 reserved for internal use (heading of the UAV) 95 to 99 reserved for the lost communications flight plan A flight plan can include from 2 up to 90 waypoints, hence the avionics can store up to 45 flight plans.

3.5 Hybrid Modeling

The hybrid model for the UAV that we present in this section captures both the continuous physical dynamics of the UAV and the switching logic that supervises the lower level controllers. The hybrid model, which is presented in Fig. 13 and Fig. 14, uses the formalism developed in [34]. It consists of two concurrent but decoupled hybrid systems with inputs and outputs, one modeling the lateral and one the altitude dynamics. The lateral dynamics model consists of three (discrete) modes of operation consisting of the line and circle tracking controller described previously, as well as a turn rate controller which allows the user to directly specify a turn rate. The altitude controller has two modes of operation, one when it receives altitude setpoints from the waypoint list, and one when it directly receives them from the user. We assume that the transitions are activated and completed concurrently for each transition system.

For our model, we will use the following notation. Let $X_1 = [x_a, y_a, u, \theta, \omega]^T$ and $U_1 = [u_{cmd}, \omega_{cmd}]^T$, then $\dot{X}_1 = F_1(X_1, U_1)$ represents the set of equations (1). In the same way, let $X_2 = z_a$, $U_2 = z_{cmd}$ and $\dot{X}_2 = F_2(X_2, U_2)$ for (2). We denote the set of equations (3) by $\hat{X}_1(t) = G_1(X_1(t))$. Let equations (5) to (12) be represented by $\omega_{cmd} = g_1(\vec{x}_t, \vec{x}_p, \vec{x}_a, \hat{\theta})$, and equations (13) to (15) be represented by $[\vec{x}_t, \vec{x}_p]^T = G_2(\vec{x}_a, \vec{x}_{wp})$.

Some additional notation is the following. G_{ij} is the guard of the transition from the state i (j is a counter for the transitions) and R_{ij} are the corresponding reset equations. *orbit* is a binary variable denoting whether the UAV is going to orbit around the next target WP (NWP) with coordinates \vec{x}_{nwp} . \vec{x}_{cwp} are the coordinates of the current target WP (CWP)

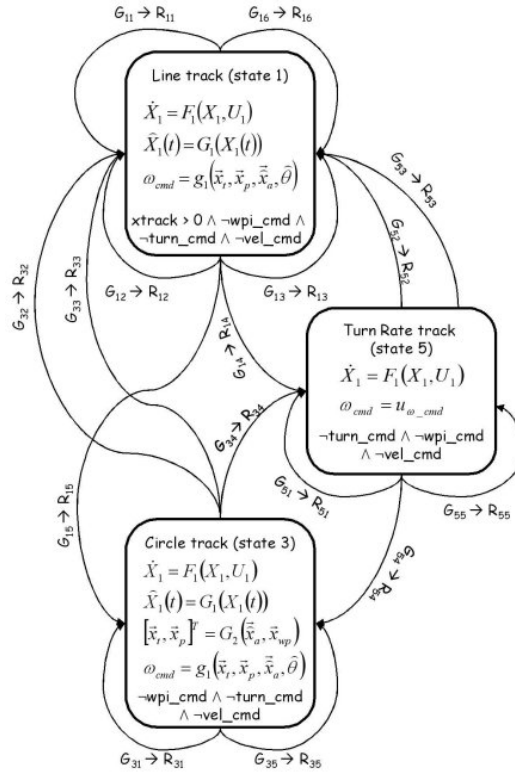


Figure 13: Hybrid model of lateral dynamics

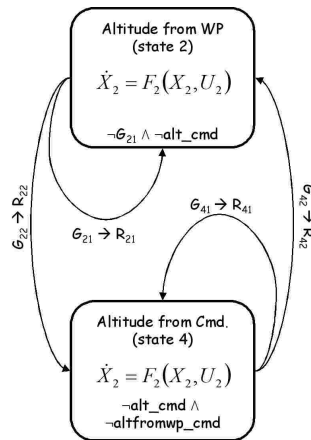


Figure 14: Hybrid model of altitude dynamics

and \vec{x}_{pwp} are the coordinates of the previous WP (PWP).

There is a variety of events that trigger transitions in this hybrid model:

- *wpi_cmd*: command the UAV to track a new WP
- *turn_cmd*: command the UAV to start turning with the specified turn rate u_{ω_cmd}
- *alt_cmd*: UAV changes its altitude to u_{alt_cmd}
- *vel_cmd*: UAV changes its velocity to u_{vel_cmd}
- *altfromWP_cmd*: command the UAV to use the altitude from the current WP.

Switching from one mode of operation to another can be directly triggered using the above discrete commands. In addition, switching also occurs when some predicates (or guards) of the states and inputs are true. In the current framework, we maintain a set S with the following variables: $S = \{\vec{x}_{cwp}, \vec{x}_{nwp}, \vec{x}_{pwp}, orbit, z_{nwp}\}$, where z_{nwp} is the altitude of the NWP. This list represents all the information that is needed by the UAV in order to track a line segment. It is actually an abstraction of the waypoint list (see section 3.4) used by the avionics. The set S is updated either after a transition has been taken or when a new WP index command (*wpi_cmd*) is received. Note that this update could be modeled as an extra state.

In particular, waypoint switching is based on a function that calculates the position x_{track} . Let this function to be $f_{xtrack} : \mathbb{R}^6 \rightarrow \mathbb{R}$ (Note: all the vectors are in \mathbb{R}^2) with:

$$f_{xtrack}(\vec{x}_a, \vec{x}_t, \vec{x}_p) = \begin{cases} \frac{(\vec{x}_t - \vec{x}_p)^T (\vec{x}_t - \vec{x}_a)}{\|\vec{x}_t - \vec{x}_p\|} & \text{if } x_p \text{ is defined} \\ -1 & \text{otherwise} \end{cases} \quad (16)$$

In order to determine in which region the UAV is (see Fig. 12 and section 3.4), we have to know the length of the line segment to be tracked, i.e. the length of the vector \vec{v}_{pt} . Let the function $f_{\vec{v}_{pt}} : \mathbb{R}^4 \rightarrow \mathbb{R}^+ \cup \{-1\}$ with:

$$f_{\vec{v}_{pt}}(\vec{x}_t, \vec{x}_p) = \begin{cases} \|\vec{x}_t - \vec{x}_p\| & \text{if } x_p \text{ is defined} \\ -1 & \text{otherwise} \end{cases} \quad (17)$$

Note that \vec{x}_p can be undefined (see case 4 in section 3.4).

3.6 Guards and reset equations

In this section, we are going to present in detail all the guards and the reset equations of the transitions in our hybrid model.

State 1 - Line Track Switching Conditions and Reset Maps

- The UAV has reached the switching boundary of the current WP \longrightarrow Switch to the next WP in the flight plan.

Guard:

$$G_{11} = (f_{xtrack}(\vec{\hat{x}}_a, \vec{\hat{x}}_t, \vec{x}_p) \leq 0) \wedge \neg orbit$$

Reset map:

$$R_{11} = \{\vec{x}_p := \vec{x}_{cwp}; \vec{x}_t := \vec{x}_{nwp}\}$$

- Change WP index command event (*wpi_cmd*) triggered (and) we are in region 1 (see Fig. 11) with respect to the new line segment \longrightarrow Switch to the commanded WP.

Guard:

$$G_{12} = wpi_cmd \wedge \neg orbit \wedge (0 < f_{xtrack}(\vec{\hat{x}}_a, \vec{x}_{cwp}, \vec{x}_{pwp}) \leq f_{\vec{v}_{pt}}(\vec{x}_{cwp}, \vec{x}_{pwp}))$$

Reset map:

$$R_{12} = \{\vec{x}_p := \vec{x}_{pwp}, \vec{x}_t := \vec{x}_{cwp}\}$$

- Change WP index command event (*wpi_cmd*) triggered (and) we are outside region 1 with respect to the new line segment \longrightarrow Switch to commanded WP.

Guard:

$$G_{13} = wpi_cmd \wedge \neg orbit \wedge \neg(0 < f_{xtrack}(\vec{\hat{x}}_a, \vec{x}_{cwp}, \vec{x}_{pwp}) \leq f_{\vec{v}_{pt}}(\vec{x}_{cwp}, \vec{x}_{pwp}))$$

Reset map:

$$R_{13} = \{\vec{x}_p := \vec{\hat{x}}_a, \vec{x}_t := \vec{x}_{cwp}\}$$

- Turn rate command event (*turn_cmd*) triggered \longrightarrow Switch to state "Turn rate track" (state 5).

Guard:

$$G_{14} = turn_cmd$$

Reset map:

$$R_{14} = \{\omega_{cmd} := u_{\omega_cmd}\}$$

- Change WP index command event (*wpi_cmd*) triggered and the specified WP is an orbit \longrightarrow Switch to "Circle track" (state 3) to commanded WP.

Guard:

$$G_{15} = wpi_cmd \wedge orbit$$

Reset map:

$$R_{15} = \{\vec{x}_{wp} := \vec{x}_{cwp}\}$$

- Velocity command event triggered \longrightarrow Set velocity input to commanded velocity.

Guard:

$$G_{16} = vel_cmd$$

Reset map:

$$R_{16} = \{u_{cmd} := u_{vel_cmd}\}$$

State 2 - Altitude from WP

- WP index changed either due to a command or to a switching condition \longrightarrow Set altitude command to WP's altitude.

Guard:

$$G_{21} = (G_{11} \vee G_{12} \vee G_{13} \vee G_{15} \vee G_{31} \vee G_{32} \vee G_{33})$$

Reset map:

$$R_{21} = \{z_{cmd} := z_{nwp}\}$$

- Altitude command event (*alt_cmd*) triggered \longrightarrow Switch to state "Altitude from Command" (state 4).

Guard:

$$G_{22} = alt_cmd$$

Reset map:

$$R_{22} = \{z_{cmd} := u_{alt_cmd}\}$$

State 3 - Circle track

- Change WP index command event (*wpi_cmd*) triggered (and) the specified WP is an orbit \longrightarrow Remain in "Circle track" (state 3).

Guard:

$$G_{31} = wpi_cmd \wedge orbit$$

Reset map:

$$R_{31} = \{\vec{x}_{wp} := \vec{x}_{cwp}\}$$

- Change WP index command event triggered (and) the specified WP is not an orbit (and) we are in region 1 with respect to the new line segment \longrightarrow Switch to "Line track" (state 1) and set \vec{x}_t to commanded WP.

Guard:

$$G_{32} = wpi_cmd \wedge \neg orbit \wedge (0 < f_{xtrack}(\vec{x}_a, \vec{x}_{cwp}, \vec{x}_{pwp}) \leq f_{\vec{v}_{pt}}(\vec{x}_{cwp}, \vec{x}_{pwp}))$$

Reset map:

$$R_{32} = \{\vec{x}_p := \vec{x}_{pwp}; \vec{x}_t := \vec{x}_{cwp}\}$$

- Change WP index command event triggered (and) the specified WP is not an orbit (and) we are outside region 1 with respect to the new line segment \longrightarrow Switch to "Line track" (state 1) and set \vec{x}_t to commanded WP.

Guard:

$$G_{33} = wpi_cmd \wedge \neg orbit \wedge \neg(0 < f_{xtrack}(\hat{x}_a, \vec{x}_{cwp}, \vec{x}_{pwp}) \leq f_{\vec{v}_{pt}}(\vec{x}_{cwp}, \vec{x}_{pwp}))$$

Reset equations R_{33} :

$$R_{33} = \{\vec{x}_p := \hat{x}_a; \vec{x}_t := \vec{x}_{cwp}\}$$

- Turn rate command triggered \longrightarrow Switch to state "turn rate track" (state 5).

Guard:

$$G_{34} = turn_cmd$$

Reset map:

$$R_{34} = \{\omega_{cmd} := u_{\omega_cmd}\}$$

- Velocity command event triggered \longrightarrow Set velocity input to commanded velocity.

Guard:

$$G_{35} = vel_cmd$$

Reset map:

$$R_{35} = \{u_{cmd} := u_{vel_cmd}\}$$

State 4 - Altitude from command

- Altitude command event triggered \longrightarrow Update altitude command.

Guard:

$$G_{41} = alt_cmd$$

Reset map:

$$R_{41} = \{z_{cmd} := u_{alt_cmd}\}$$

- Use the altitude from the CWP when an $altfromWP_cmd$ command event is triggered \longrightarrow Switch to "Altitude from WP" (state 2).

Guard:

$$G_{42} = altfromWP_cmd$$

Reset map:

$$R_{42} = \{z_{cmd} := z_{cwp}\}$$

State 5 - Turn Rate track

- Turn rate command event $turn_cmd$ triggered \longrightarrow Assign new turn rate command.

Guard G_{51} :

$$G_{51} = turn_cmd$$

Reset map R_{51} :

$$R_{51} = \{\omega_{cmd} := u_{\omega_cmd}\}$$

- Change WP index command event triggered (and) the specified WP is not an orbit (and) we are in region 1 with respect to the new line segment \longrightarrow Switch to "Line track" (state 1) and set \vec{x}_t to commanded WP.

Guard:

$$G_{52} = wpi_cmd \wedge \neg orbit \wedge (0 < f_{xtrack}(\vec{x}_a, \vec{x}_{cwp}, \vec{x}_{pwp}) \leq f_{\vec{v}_{pt}}(\vec{x}_{cwp}, \vec{x}_{pwp}))$$

Reset map:

$$R_{52} = \{\vec{x}_p := \vec{x}_{pwp}; \vec{x}_t := \vec{x}_{cwp}\}$$

- Change WP index command event triggered (and) the specified WP is not an orbit (and) we are outside region 1 with respect to the new line segment \longrightarrow Switch to "Line track" (state 1) and set \vec{x}_t to commanded WP.

Guard:

$$G_{53} = wpi_cmd \wedge \neg orbit \wedge \neg(0 < f_{xtrack}(\vec{x}_a, \vec{x}_{cwp}, \vec{x}_{pwp}) \leq f_{\vec{v}_{pt}}(\vec{x}_{cwp}, \vec{x}_{pwp}))$$

Reset map:

$$R_{53} = \{\vec{x}_p := \vec{x}_a; \vec{x}_t := \vec{x}_{cwp}\}$$

- Change WP index command event triggered (and) the specified WP is an orbit \longrightarrow Switch to "Circle track" (state 3) to commanded WP.

Guard:

$$G_{54} = wpi_cmd \wedge orbit$$

Reset map:

$$R_{54} = \{\vec{x}_{wp} := \vec{x}_{cwp}\}$$

- Velocity command event triggered \longrightarrow Set velocity input to commanded velocity.

Guard:

$$G_{55} = vel_cmd$$

Reset map:

$$R_{55} = \{u_{cmd} := u_{vel_cmd}\}$$

4 High Fidelity Simulation

The hybrid model developed in the previous section will serve as a modeling abstraction in developing high-level algorithms involving multiple UAVs. In order to implement and verify the performance of algorithms designed using the hybrid model described in the previous section, the designed algorithms are first executed on very detailed simulation models (MATLAB simulation toolbox developed in house). Furthermore, in order to fully utilize the hardware-in-the-loop (HIL) simulator (see [35]) offered with the Piccolo developer’s kit and thus attain accurate simulations, it is mandatory to develop a high fidelity dynamics model for the Piper J3 cub model airplane. This dynamics model takes into account aerodynamics, propulsion and inertia effects.

The high fidelity hardware-in-the-loop simulator for the Piccolo avionics increases our confidence in the experimental performance of the algorithms as it accurately reflects and predicts the system behavior during the actual flight experiments.

4.1 Aerodynamic Model

The lack of publicly available aerodynamic data for the Piper J3 cub model airplane as well as the modifications done on the airframe by the PennUAV team necessitate the estimation of the aerodynamic parameters of the model airplane. For this purpose, the vortex panel method with boundary layer analysis was employed. By using the panel method for this purpose one should expect overestimation of the lift and underestimation of the drag. CFD tools (such as solution of Navier-Stokes equations) or wind-tunnel tests were too time consuming for the scope and the needs of this research – at least at his point.

The software toolbox used for the aerodynamic analysis is presented in [36] and [37]. This software package also provides tools for safe store release prediction, calculation of the ballistic trajectory of the store by solving the 6-degree freedom problem and determination of the final impact point. Thus, it makes possible the accurate deployment of any external store of the airplane (i.e the deployable pods see Fig. 2).

The paneling for the Piper J3 cub model airplane is presented in Fig. 15. The half model (symmetric) consists of 1167 panels and the total computation time is less than $5min$ on a Pentium 4. All the tests were run at nominal speed $\sim 15m/sec$ at sea level standard (SLS) conditions (i.e. a subsonic low Reynolds number environment; at these conditions $Re \sim 4.2 \cdot 10^5$). The paneling for the Piper was designed from the blue prints of the model airplane taking into consideration the modifications that we have performed on the frame. We should mention that the wing’s airfoil is not a standard airfoil, but it can be approximated very well by a NACA 4412. For simplicity, we chose to ignore the effects of the propeller even though the software toolbox supports such phenomena. In Fig. 16, we present the c_L and c_D coefficients used in the hardware in the loop simulator. Note that these results include the combined effects of the fuselage and the wing¹.

¹For historical reasons, we would like to point the reader to [38]

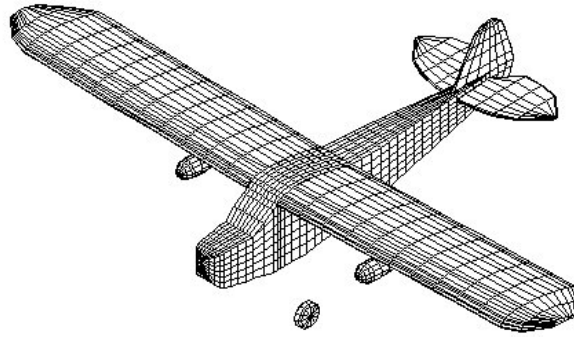


Figure 15: The paneling for the Piper J3 cub

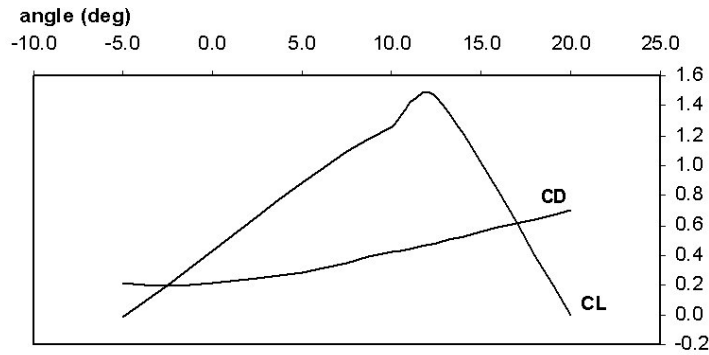


Figure 16: The calculated and modified lift c_L and drag c_D coefficients used in the HIL simulator ($Re \sim 4.2 \cdot 10^5$). These results include the effects of the fuselage and the wing. Note that the wing has an incidence of 2.44° .

4.2 Propulsion Model

This model consists of the engine and the propeller model and it is quite simplistic but comprehensive enough for the HIL simulator. For this part, data from an engine similar to the one actually employed are used.

4.3 Inertia Model

The simple design of the Piper J3 cub made possible the rapid development of a 3D-solid CAD model (see Fig. 17). The model includes all skeletal structure of the airframe with ribs and panels and also all other significant components of the UAV namely PODs, actuators, onboard PC, avionics box and enclosure, landing gear, fuel, fuel tank, engine and propeller. From such a detailed model one can easily extract a very accurate estimate of the inertia coefficients. Furthermore, the center of gravity of the CAD model was compared with the

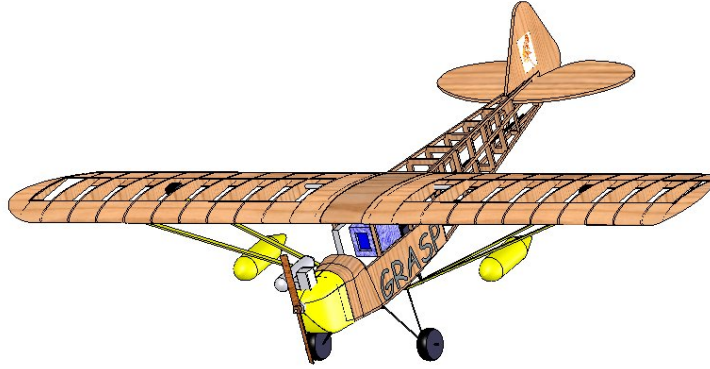


Figure 17: 3D-solid CAD model of the Piper J3 cub

real airframe for verification and was found to be very accurate.

4.4 Visualization

The results of the high-fidelity HIL simulator can be realistically visualized using *Microsoft Flight Simulator*, as shown in Fig. 18. In addition to animating simulated trajectories, we can also use the Microsoft Flight Simulator for playing back actual multi-UAV experiments, which is the focus of the next section.

5 Formation Flight Algorithm

Our formation flying experiments consist of a leader–follower scheme (see Fig. 19). A simple formation control algorithm that requires transmission of minimal information is used. The leader flies a predetermined path and the follower tries to keep the specified formation using as only information the position of the leader. The actual algorithm consists of a feed-forward outer loop over the autopilot of the follower. The algorithm is actually using the turn rate track (state 5) of our hybrid model. We consider only the planar formation control problem assuming that both UAV fly at predetermined heights. We chose to ignore the altitude dynamics even though the extension to the 3D formation problem is straightforward by adding one additional outer loop that accesses state 4 of our hybrid model.

Let the subscript L to denote the leader and F to denote the follower, then $\vec{r}_i = (x_i, y_i) \in \mathbb{R}^2$ is the position vector with respect to the world frame and \vec{v}_i is the ground speed of UAV i as it is given by the GPS. The distance between the two UAVs is:

$$d = \sqrt{(x_L - x_F)^2 + (y_L - y_F)^2} \quad (18)$$

Let $R(\theta)$ denote the rotation matrix:

Center of mass (<i>mm</i>)	
X	287.59
Y	-3.07
Z	10.83
Mass	$\approx 10 \text{ kgr}$

Table 2: The mass properties of the PennUAV at full configuration. The origin of the coordinate system is at the engine mount point. The X axis points to the tail of the aircraft and the Z axis points upwards.

$I_{xx} = 1200705083.35$	$I_{xy} = -9381009.86$	$I_{xz} = 80104396.63$
$I_{yx} = -9381009.86$	$I_{yy} = 938853512.70$	$I_{yz} = 6646052.59$
$I_{zx} = 80104396.63$	$I_{zy} = 6646052.59$	$I_{zz} = 1913020741.95$

Table 3: The moments of inertia for the PennUAV at full configuration (in $gr \cdot mm^2$). The origin of the coordinate system is at the engine mount point. The X axis points to the tail of the aircraft and the Z axis points upwards. The moments of inertia are given with respect to the center of gravity of the aircraft and they are aligned with the coordinate system.

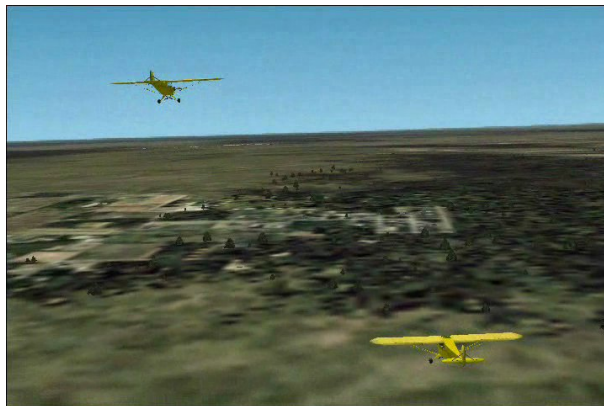


Figure 18: *MS Flight Simulator* visualizing formation control experiment Fort Benning Army Base in Georgia.

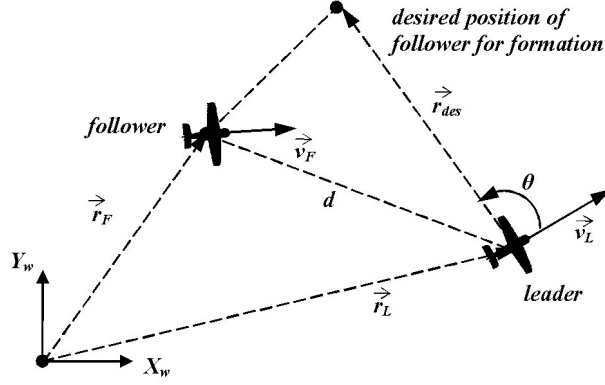


Figure 19: The leader–follower formation flight

$$R(\theta) = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

Assume that we want the follower to be at distance d_{des} and angle θ with respect to the heading h_L of the leader. Then the desired position of the follower would be:

$$\vec{r}_{des} = d_{des}R(-\theta)\vec{h}_L \quad (19)$$

where $\vec{h}_L = \vec{v}_L/\|\vec{v}_L\|$. Let $K_p \in \mathbb{R}$ be the parameter of the potential field, then the velocity given by the potential field is:

$$\vec{v}_p = K_p(\vec{r}_L + \vec{r}_{des} - \vec{r}_F) \quad (20)$$

and the desired velocity:

$$\vec{v}_{des} = \vec{v}_p + \vec{h}_L\vec{v}_{L,TAS} \quad (21)$$

where $\vec{v}_{L,TAS}$ is the true air speed of the leader. Hence, the angle between the current heading of the follower and the desired heading is:

$$\phi = \text{atan2}(v_{y,des}, v_{x,des}) - \text{atan2}(v_{y,F}, v_{x,F}) \quad (22)$$

where $\phi \in (-\pi, \pi]$. Then the inputs for the turn rate track would be:

$$\omega_F = K_\omega\phi \quad (23)$$

$$\vec{v}_{F,TAS} = \|\vec{v}_{des}\| \quad (24)$$

where $K_\omega \in \mathbb{R}$ a parameter.

6 Cooperative Control Experiments

6.1 Aerial Surveillance, Mapping And Exploration

In this experiment, a single UAV operating in autonomous mode executed a predetermined flight plan at a test site at the army base in Fort Benning, Georgia. The UAV flew multiple passes at different altitudes over the target site. The high resolution camera was visually surveying the ground, while the 100 Hz IMU and the 10 Hz GPS collected real time data for further accuracy and redundancy in generating mosaic maps of the environment. Fig. 20 shows the generated mosaic map from images taken at different altitudes.

6.2 Formation Flights

A variety of formation control experiments were performed using two fixed wing UAVs. In our cooperative control experiments, two UAVs flew in formation in a decentralized manner, using a leader follower architecture. The leader UAV was given a pre-determined flight plan. The trajectory of the follower UAV was updated once per second in real time through the ground station to keep the follower at a fixed distance offset from the leader. The formation control law we used is inspired by the control law obtained in [3]. Fig. 21 shows the trajectories of the UAVs and the predetermined flight plan that was assigned to the leader. Marks on the trajectories represent the respective positions of the UAVs every 7 seconds. The winds were around 4 m/s relatively significant with respect to airspeed ($\sim 15m/s$) in the x direction (east to west) at the time of the experiment. The tracking of the leader UAV, average deviation from course on the upwind leg is much better from the downwind leg due to the way the lateral control laws were implemented. Another detail to note is that the tracking behaviour of the system is dominated by the low sampling frequency (1Hz) of the outer lateral control loop which is responsible for generating turn rate commands. Thus, if it is desirable to achieve tighter formation, the sampling frequency of the lateral control loop has to be increased together with higher performance GPS and IMU sensor fusion.

A variant of the previous experiment was recently conducted where the follower UAV was flying 50 meters directly above the leader, monitoring the UAV below with a high resolution camera. Videos of all experiments described in this section can be seen in [21] (also available on line at www.grasp.upenn.edu/uav).

7 Conclusions

In this paper, we have described the multi-UAV testbed developed at the University of Pennsylvania, developed a hybrid model that captures both abstracted UAV dynamics as well as mode-switching logic, and reported on cooperative control experiments involving two Penn UAVs. The hybrid model described in this paper, which has been validated in both hardware-in-the-loop simulations and actual experiments, will serve as modeling abstraction and will be utilized in applying hybrid control methods in order to solve challenging problems involving



Figure 20: Aerial Surveillance Experiment – Generated Mosaic Image (Aug. 14th 2003, Fort Benning, Georgia)

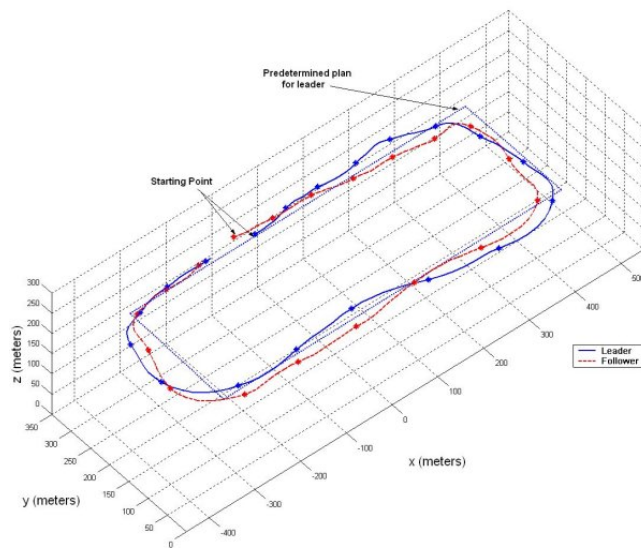


Figure 21: Autonomous Formation Experiment 1 (Aug. 14th 2003, Fort Benning, Georgia,



Figure 22: Autonomous Formation Experiment 1 (Aug. 14th 2003, Fort Benning, Georgia,

multiple UAVs, such as synchronous or asynchronous formations, coordinated search and rescue, air-ground integration, and rapid emergency response.

Acknowledgements The authors would like to thank George Asteris, Ben Grocholsky, James F. Keller, Vijay Kumar and Camillo J. Taylor.

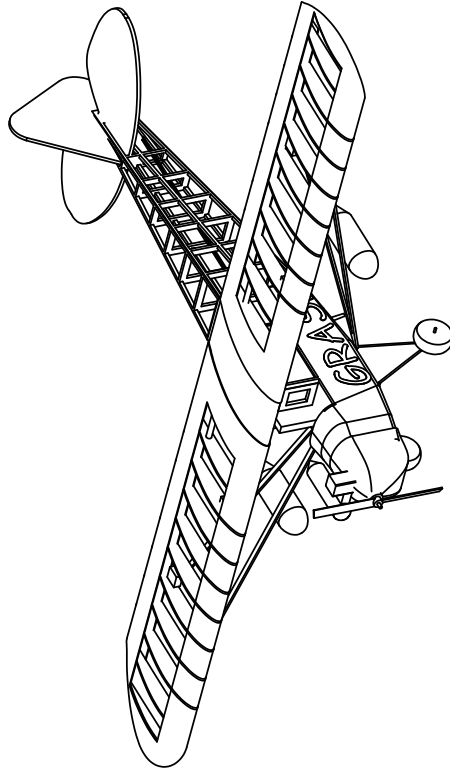
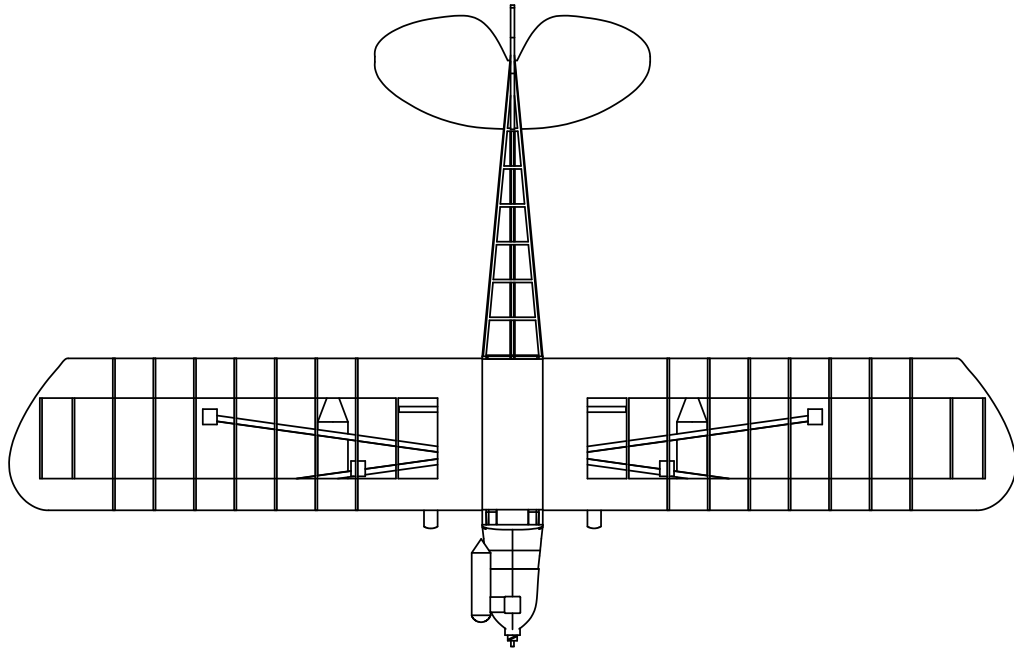
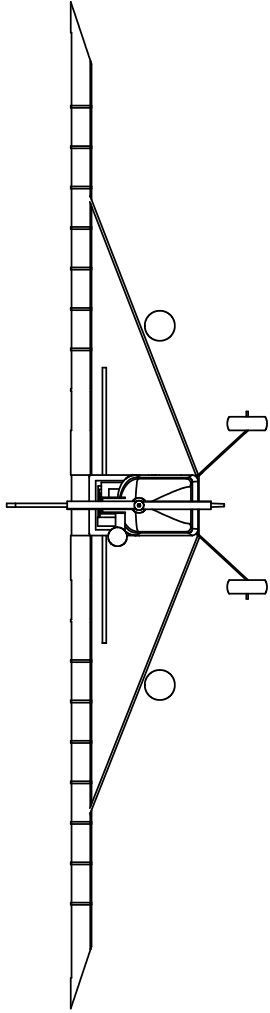
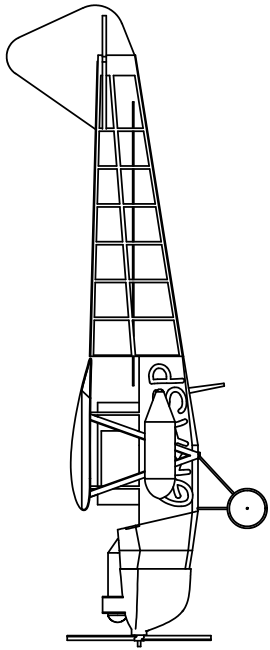
References

- [1] H. Tanner, A. Jadbabaie, and G.J. Pappas, "Stable flocking of mobile agents, Part I : Fixed Topology", *Proceedings of the 42nd IEEE Conference on Decision and Control*, Maui, Hawaii, December 2003
- [2] H. Tanner, A. Jadbabaie, and G.J. Pappas, "Stable flocking of mobile agents, Part II : Dynamic Topology", *Proceedings of the 42nd IEEE Conference on Decision and Control*, Maui, Hawaii, December 2003
- [3] H. Tanner, A. Jadbabaie, and G.J. Pappas, "Flocking in fixed and switching networks", *Automatica*, 2003, Submitted.
- [4] V. Gavrillets, A. Shterenberg, M. Dehaleh and E. Feron, "Avionics System for a Small Unmanned Helicopter Performing Aggressive Maneuvers", in *19th Digital Avionics Systems Conference Proceedings.*, 7-13 Oct. 2000, vol.1, pp:1E2/1 - 1E2/ .
- [5] E.N. Johnson, D.P. Schrage, "The Georgia Tech Unmanned Aerial Research Vehicle: GTMax", in *Proceedings of the AIAA Guidance, Navigation, and Control Conference 2003*, AIAA Paper 2003-5741.
- [6] S. Saripalli, D. J. Naffin, and G. S. Sukhatme, "Autonomous Flying Vehicle Research at the University of Southern California", in *Multi-Robot Systems: From Swarms to*

- Intelligent Automata, Proceedings of the First International Workshop on Multi-Robot Systems*, A. Schultz and L. E. Parker, Kluwer Academic Publishers, 2002, pp. 73-82.
- [7] O. Amidi, T. Kanade, and R. Miller. "Autonomous Helicopter Research at Carnegie Mellon Robotics Institute", in *Proceedings of Heli Japan 98*, Gifu, Japan, 1998.
- [8] H.J. Kim, D.H. Shim, S. Sastry, "Flying robots: modeling, control and decision making", in *Proceedings of International Conference on Robotics and Automation, ICRA '02*, 11-15 May 2002, vol.1
- [9] J. Kim, J. Keller, V. Kumar, "Design and Verification of Controllers for Airships", *IEEE IROS*, Las Vegas, NV, Oct, 2003
- [10] E. Olsen, C.W. Park, and J. How, "3D Formation Flight using Differential Carrier-phase GPS Sensors", *The J. of The Institute Of Navigation*, Spring, 1999, Vol. 146, No. 1, pp. 3548.
- [11] J. Evans, G. Inalhan, J.S. Jang, R. Teo, C.J. Tomlin, "DragonFly: a versatile UAV platform for the advancement of aircraft navigation and control", in *20th Conference Digital Avionics Systems*, 14-18 Oct. 2001, Pages:1C3/1 - 1C3/12 vol.1.
- [12] J. Lee, R. Huang, A. Vaughn, X. Xiao, J.K. Hedrick, M. Zennaro, R. Sengupta, "Strategies of Path-Planning for a UAV to Track a Ground Vehicle", in *The Second Annual Symposium on Autonomous Intelligent Networks and Systems*, Menlo Park, CA, June 2003.
- [13] T. Schouwenaars, J. P. How, and E. Feron, "Decentralized Cooperative Trajectory Planning of Multiple Aircraft with Hard Safety Guarantees", presented at *the AIAA Guidance, Navigation and Control Conference*, Aug 2004. AIAA-2004-5141.
- [14] Mark E. Campbell and Jarurat Ousingsawat, On-line Estimation and Path Planning for Multiple Vehicles in an Uncertain Environment, *Proceedings of the 2002 AIAA Conference on Guidance, Navigation and Control*, Monterey, CA, 2002
- [15] D. Kingston, R. Beard, T. McLain, M. Larsen, W. Ren, "Autonomous Vehicle Technologies for Small Fixed Wing UAVs", *AIAA 2nd Unmanned Unlimited Systems, Technologies, and Operations—Aerospace, Land, and Sea Conference and Workshop & Exhibit*, San Diego, CA, September, 2003, Paper no. AIAA-2003-6559.
- [16] Fierro, R., Belta, C., Desai, J.P. and Kumar, V., "On Controlling Aircraft Formations", *Proceedings of the 40th IEEE Conference on Decision and Control*, Orlando, FL, Dec. 2001, vol. 2, pp. 1065-70.
- [17] C. Schumacher, "Nonlinear control of multiple UAVs in formation flight", in *Proc. of the AIAA GNC Conference*, August 2000.

- [18] S. Singh, M. Pachter, P. Chandler, S. Banda, S. Rasmussen, and C. Schumacher, "Input-output invertibility and sliding mode control for close formation flying", in *Proc. of the AIAA GNC Conference*, August 2000.
- [19] F. Giulletti, L. Pollini, and M. Innocenti, "Autonomous formation flight", *IEEE Control Systems Magazine*, vol. 20, pp. 4552, December 2000.
- [20] S. Bayraktar, G.E. Fainekos, and G.J. Pappas, "Experimental Cooperative Control of Fixed-Wing Unmanned Aerial Vehicles", Proceedings of the 43rd *IEEE Conference on Decision and Control*, The Bahamas, December 2004.
- [21] S. Bayraktar and G. J. Pappas, "Experimenting with Multiple Unmanned Aerial Vehicles", in *Video Proceedings of the 2004 International Conference on Robotics and Automation*, New Orleans, April 2004.
- [22] Eugene Lavretsky, "F/A-18 Autonomous Formation Flight Control System Design", AIAA Paper 2002-4757, AIAA Guidance, Navigation, and Control Conference and Exhibit, August 2002, Monterey, California.
- [23] C.E. Hanson, J. Ryan, M.J. Allen, and S.R. Jacobson, "An overview of flight test results for a formation flight autopilot", NASA/TM-2002-210729, August 2002.
- [24] Sanghyuk Park, "Avionics and Control System Development for Mid-Air Rendezvous of Two Unmanned Aerial Vehicles", Ph.D. Thesis, MIT, February 2004.
- [25] Derek R. Nelson, Timothy W. McLain, Reed S. Christiansen, Randal W. Beard, David Johansen, "Initial Experiments in Cooperative Control of Unmanned Air Vehicles", AIAA 3rd Unmanned Unlimited Systems Conference and Workshop, Chicago, IL, September, 2004, Paper no. AIAA-2004-6533
- [26] Rodney Teo, Jung Soon Jang, and Claire J. Tomlin, "Automated Multiple UAV Flight-the Stanford DragonFly UAV Program", Proceedings of the 43rd *IEEE Conference on Decision and Control*, The Bahamas, December 2004.
- [27] Ben Grocholsky, Selcuk Bayraktar, Vijay Kumar, Camillo J. Taylor, and George J. Pappas, "Synergies in Feature Localization by Air-Ground Robot Teams", Proceedings of the *9th International Symposium on Experimental Robotics 2004*, Singapore, June 2004.
- [28] B. Grocholsky, S. Bayraktar, V. Kumar and G. Pappas, "UAV and UGV Collaboration for Active Ground Feature Search and Localization", AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop and Exhibit, Chicago, Illinois, Sep. 20-23, 2004.
- [29] B. Vaglienti, R. Hoag and M. Niculescu, "Piccolo system user guide", [<http://www.cloudcaptech.com/downloads.htm>], *Cloud Cap Technology*, Version 1.2.2, July 1, 2004

- [30] B. Vaglienti and R. Hoag, Communications for the Piccolo avionics, [<http://www.cloudcaptech.com/downloads.htm>], *Cloud Cap Technology*, Version 1.1.7, May 9, 2003.
- [31] Communications Standard Developer Kit, [<http://www.cloudcaptech.com/downloads.htm>], *Cloud Cap Technology*, Version 1.2.17, Oct 31, 2003.
- [32] B. Vaglienti, "Lateral track control law for Piccolo", [<http://www.cloudcaptech.com/>], *Cloud Cap Technology*, March 12 2003.
- [33] M. Niculescu, "Lateral track control law for Aerosonde UAV", *39th AIAA Aerospace Sciences Meeting and Exhibit*, Paper 2001-0016, January 2001.
- [34] J. Lygeros, "Hierarchical Hybrid Control of Large Scale Systems", Ph.D Thesis, *Department of EECS, University of California at Berkeley*, 1996.
- [35] B. Vaglienti and M. Niculescu, "Hardware in the loop simulator for the Piccolo avionics", [<http://www.cloudcaptech.com/downloads.htm>], *Cloud Cap Technology*, June 18, 2004
- [36] E. G. Fenekos, A. E. Filios, D. P. Margaritis, A. P. Fragias, D. G. Papanikas, "On the calculation of steady flow about isolated helicopter fuselage", *4th GRACM Congress on Computational Mechanics*, Vol. IV, pp. 1302-1308, 27-29 June 2002, Patras, Greece.
- [37] N. Tachos, A. Fragias, E. Fenekos, D. Margaritis and D. Papanikas, "Aerodynamic noise prediction of a horizontal axis wind turbine rotor", In proceedings of *The Science of making Torque from Wind*, Delft University of Technology, 19-21 April 2004.
- [38] F. L. Thompson and P. H. Keister, "Lift and drag characteristics of a cabin monoplane determined in flight", NACA Technical Note No. 362, Washington, January, 1931.

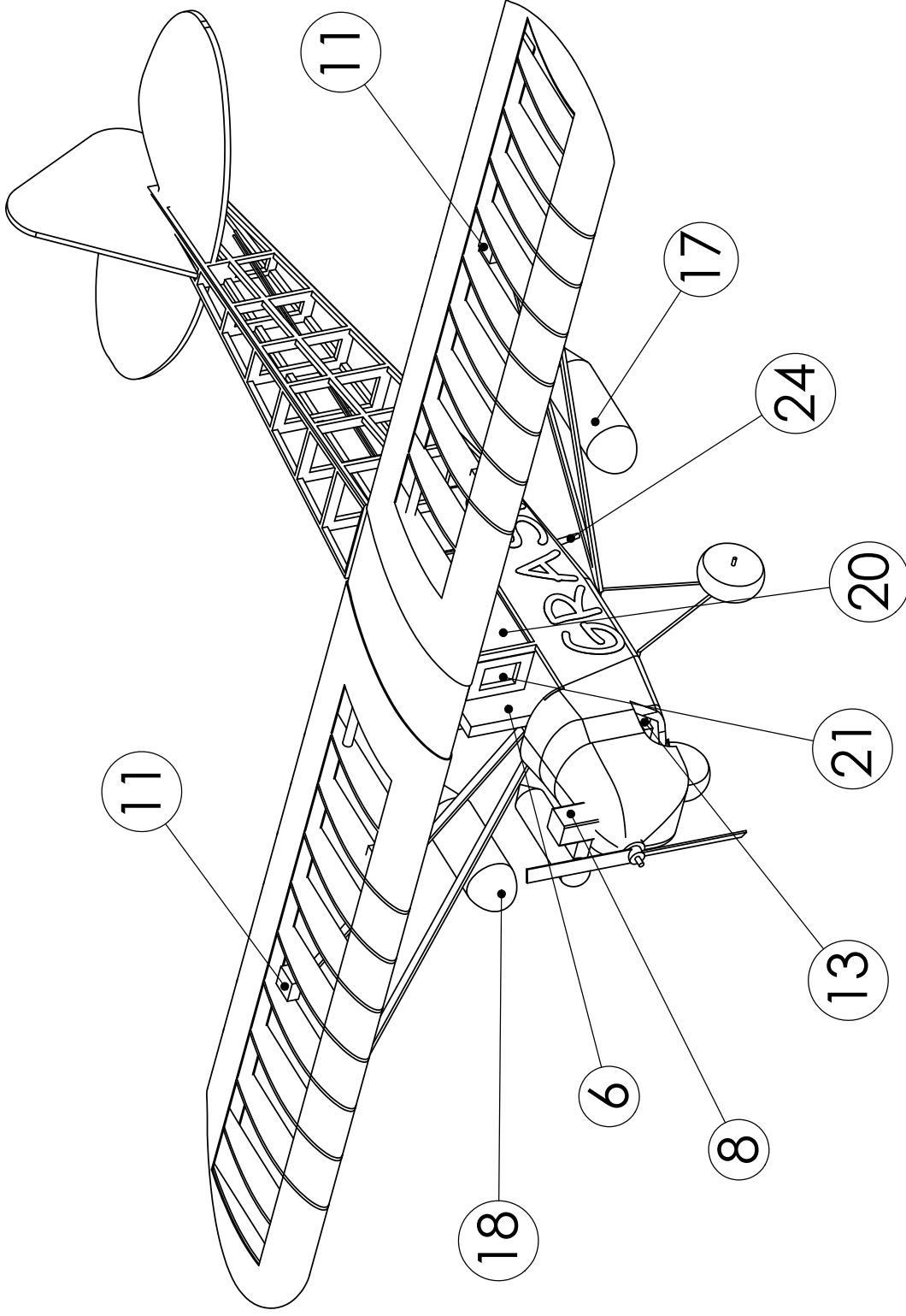


1:20

Piper J3 Cub

04.11.28

GRASP Lab
University of Pennsylvania



ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
6	Laptop	PC Dell X200	1
8	Engine	glow fuel engine 3.5HP	1
11	Servo		7
13	Battery		3
17	Pod	camera DragonFly and IMU 3DM-G	1
18	Pod	deployable pod	1
20	Avionics enclosure	custom made	1
21	Avionics	Piccolo avionics	1
24	Antenna		1

1:10

Piper J3 Cub

04.11.28

GRASP Lab
University of Pennsylvania