JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 2, January 2005

Autonomous Vehicle Technologies for Small Fixed-Wing UAVs

Randal Beard,[*] Derek Kingston,[†] Morgan Quigley,[‡] Deryl Snyder,[§] Reed Christiansen,[†] Walt Johnson,[†]
Timothy McLain,[**] and Michael A. Goodrich[††]
Brigham Young University, Provo, Utah 84602

The objective of this paper is to describe the design and implementation of a small semi-autonomous fixed-wing unmanned air vehicle. In particular we describe the hardware and software architectures used in the design. We also describe a low weight, low cost autopilot developed at Brigham Young University and the algorithms associated with the autopilot. Novel PDA and voice interfaces to the UAV are described. In addition, we overview our approach to real-time path planning, trajectory generation, and trajectory tracking. The paper is augmented with movie files that demonstrate the functionality of the UAV and its control software.

I. Introduction

Recent advances in communications, solid state devices, and battery technology have made small, low-cost fixed wing unmanned air vehicles a feasible solution for many applications in the scientific, civil and military sectors. With the use of on-board cameras this technology can provide important information for low-altitude and high-resolution applications such as scientific data gathering, surveillance for law enforcement and homeland security, precision agriculture, forest fire monitoring, geological survey, and military reconnaissance.

The fixed wing UAVs that are currently used by the military (e.g., Predator, Global Hawk) are large, expensive, special purpose vehicles with limited autonomy. At the other end of the spectrum are small UAVs (less than six foot wingspan), and micro air vehicles (MAV) (less than one foot wingspan), which are primarily being developed by universities and research laboratories. The design challenges for small and micro UAVs are significantly different than those that must be addressed for larger UAVs. In particular, successful deployment of small UAVs requires a strong lightweight vehicle platform, a low-power, lightweight autopilot, intuitive, easy-to-use human interfaces, as well as increased autonomy including path planning, trajectory generation and tracking algorithms.

The MAGICC lab at Brigham Young University has recently developed and successfully flight tested a fleet of small UAVs and the associated autopilot, user interface, and autonomous vehicle technologies. The objective of this paper is described our approach to small UAVs and our solution to the design challenges that we have faced. The paper is organized as follows. In Sec. 2 we describe the high level hardware and software architectures. In Sec. 3 we describe one of our small UAVs. In Sec. 4 we describe the design of the autopilot hardware, and in Sec. 5 we describe the ground station. A PDA and voice interface is described in Sec. 6. Path planning, trajectory generation and trajectory tracking are described in Secs. 7-9 respectively. Given the on-line nature of this journal, we have embedded several movie files and included internal and external hyperlinks within the document.

Presented as Paper 2003-6559 at the AIAA 2nd Unmanned Unlimited Systems, Technologies, and Operations-Aerospace, Land, and Sea Conference and Workshop & Exhibit, San Diego, CA; received 17 February 2004, revision received 30 September 2004, accepted for publication 18 October 2004. Copyright © 2005 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 1542-9423/04 $10.00 in correspondence with the CCC.

[*]Associate Professor, Electrical and Computer Engineering 459 CB; beard@ee.byu.edu. AIAA Member.
[†]Graduate Research Assistant, Electrical and Computer Engineering 459 CB.
[‡]Graduate Research Assistant, Computer Science 3361 TMCB.
[§]Assistant Professor, Mechanical Engineering 435 CTB.
[**]Associate Professor, Mechanical Engineering 435 CTB.
[††]Associate Professor, Computer Science 3361 TMCB.

II. System Architecture

This section describes our system architecture. A flow chart of the hardware architecture is shown in Fig. 1. As shown in the figure, there are three RF communication channels. The first is a 900 MHz bidirectional link between the autopilot on the UAV and the laptop ground station. The laptop uplinks trajectory commands to the UAV which in turn downlinks telemetry information. Data is communicated on this link at about 20Hz. The second RF channel is a 2.4 GHz video downlink from the camera on-board the UAV to a video display on the ground. Video data is in standard NTSC format and can be displayed on a video monitor or converted to digital format via a framegrabber on the laptop. The third RF link is a 70 MHz RC uplink to the UAV which is used to bypass the autopilot. The bypass capability is used as a fail-safe mode that allows new algorithms to be rapidly tested and debugged. As shown in Fig. 1, the ground station includes an 802.11b wireless connection to PDA and voice control devices to provide an effective user interface to the UAV. The autopilot will be described in more detail in Sec. 4, the ground station will be described in Sec. 5, and the PDA and voice interfaces will be described in Sec. 6. The hardware architecture shown in Fig. 1 provides a flexible experimental testbed to explore a variety of algorithms that enable autonomous and semi-autonomous behaviors for the UAVs.

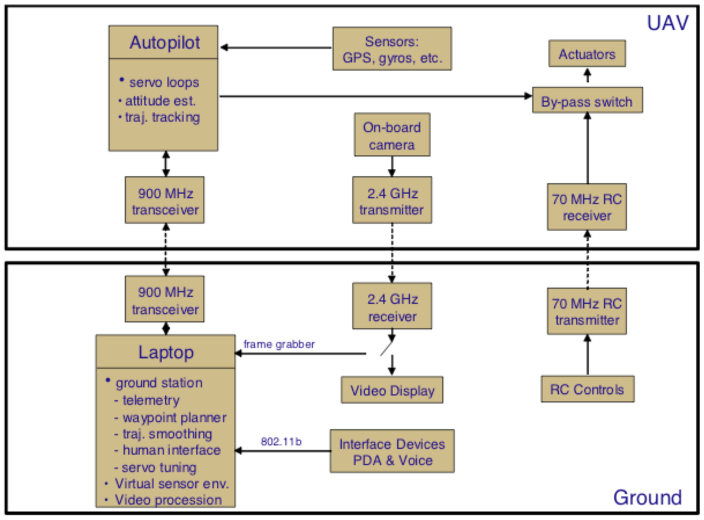


Fig. 1 Flowchart of the hardware architecture.

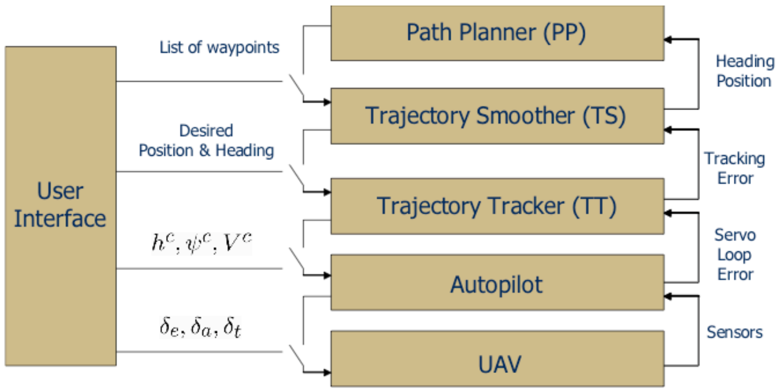


Fig. 2 Guidance and control software architecture.

A schematic of the guidance and control software architecture is shown in Fig. 2. In particular, guidance and control algorithms are divided into four distinct hierarchical layers: path planning, trajectory smoothing, trajectory tracking, and the autopilot. In this paper, paths refer to a series of waypoints that are *not* time-stamped, while trajectories will refer to time-stamped curves which specify the desired inertial location of the UAV at a specified time. As shown in Fig. 2, the top level of the control architecture is a path planner (PP). It is assumed that the PP knows the location of the UAV, the target, and the location of a set of threats. The PP generates a path

$$\mathscr{P} = \{v, \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N\}\}$$

where $v \in [v_{\min}, v_{\max}]$ is a feasible velocity and $\{\mathbf{w}_i\}$ is a series of waypoints which define straight-line segments along which the UAV attempts to fly. Note that at the PP level, the path is compactly represented by $\mathscr{P}$. Higher level decision making algorithms reason and make decisions according to the data represented in $\mathscr{P}$.

The Trajectory Smoother (TS) transforms, in real-time, the waypoint trajectory into a time parameterized curve which defines the desired inertial position of the UAV at each time instant. The output of the TS is the desired trajectory $\mathbf{z}^d(t) = \left(z_x^d(t), z_y^d(t)\right)^T$ at time t.

The Trajectory Tracker (TT) transforms $\mathbf{z}^d(t)$ into a desired velocity command V^d, a desired altitude command h^d, and a desired heading command ψ^d. The autopilot receives these commands and controls the elevator, δ_e, and aileron, δ_a, deflections and the throttle setting δ_t.

Recognizing that it will be useful for human operators to interact with the UAV at different autonomy levels, careful attention has been given to the human interface. As shown in Fig. 2, the human can interact with the UAV at the stick-and-throttle level, the autopilot command level, the time-parameterized trajectory level, or at the waypoint path planning level.

III. UAV Hardware

The BYU MAGICC lab operates a small fleet of miniature UAVs for flight tests and demonstrations. These aircraft include commercially available R/C models, in-house flying-wing aircraft constructed of EPP foam, and other specialty aircraft. One such UAV, the BATCAM, developed by the Munitions Directorate of the Air Force Research Laboratory (AFRL/MN) has provided an excellent test bed for the hardware and control algorithms under development. BATCAM is a proof-of-concept aircraft envisioned by the Air Force to be used for personal reconnaissance and bomb damage information (BDI) missions. Depending upon the operational scenario, supervised to full autonomy is required, including takeoff and landing capabilities. This UAV, shown in flight in Fig. 3, weighs less than a pound, has a wingspan of only 21 inches, and can fly with full autonomy for approximately 30 minutes. Real-time video from multiple onboard cameras is transmitted to a PC-based ground station. Due to its small size and the limited control authority provided by control surfaces on the V-tail only, this platform presents a formidable controls challenge and has driven several aspects of the autopilot design.

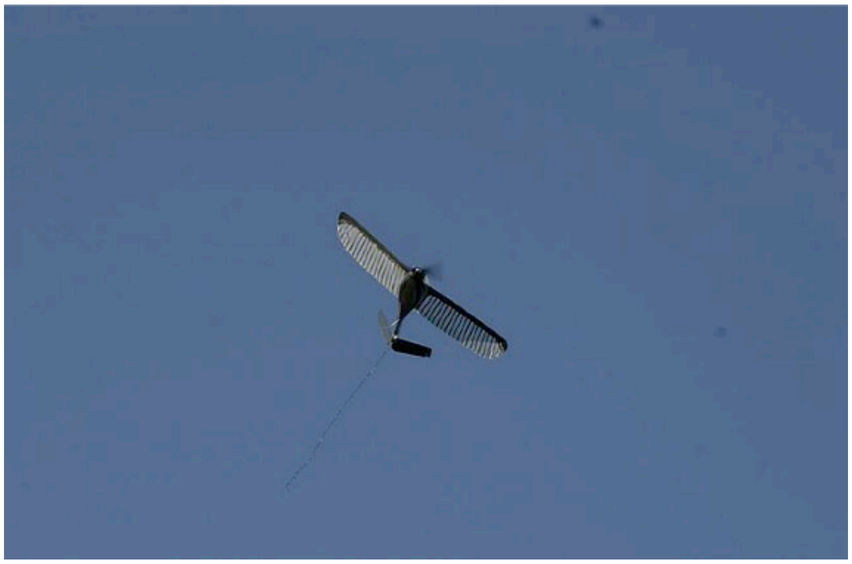


Fig. 3 BATCAM mini-UAV in flight.

The breakdown of the overall mass among the four major subsystems (airframe, propulsion, GNC, and payload) is shown in Fig. 4. From this figure, it is seen that the majority of the mass budget is allocated to the propulsion system. This is expected considering that the subsystem includes the battery required to power all electrical systems onboard the aircraft. It is interesting to note that the payload capacity is only 11% of the total mass. Although it is not uncommon to see this low level of payload capacity in mini/micro air vehicles,[1, 2] it is more severe in the case of BATCAM due to the extra mass required by the GNC subsystem required to obtain full autonomy. The final item to note is that the airframe, comprising 28% of the total mass, is rather heavy compared to other aircraft in this class. A large design emphasis on ruggedness accounts for the extra weight in the airframe: BATCAM is designed to survive 4 severe impacts with buildings, trees, and the ground.

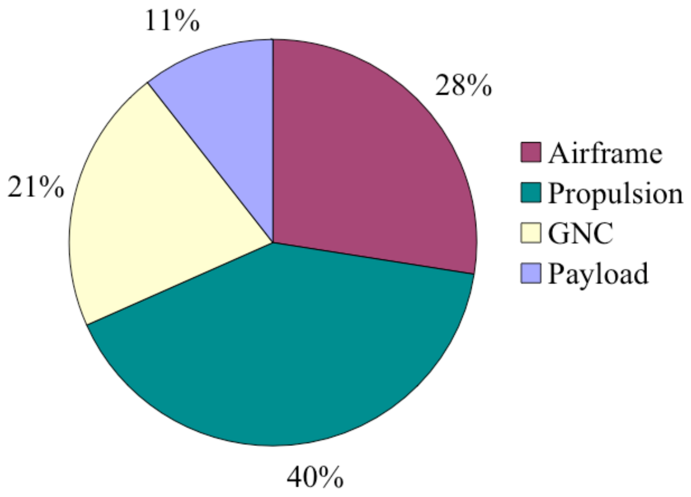


Fig. 4 Subsystem mass breakdown for the BATCAM mini-UAV.

The airframe is constructed primarily of carbon fiber, resulting in a lightweight yet extremely rugged structure. The properties of carbon fiber are advantageous to the fabrication of small UAVs in that the composite material forms complex shapes at relatively low cost, and yet can exhibit incredible strength and stiffness. The wing is fabricated with a carbon fiber skeleton, containing a main spar running the length of the span with thin battens attached, and nonbreathable rip-stop nylon skin. This construction technology, developed in part at the University of Florida,[3] results in a flexible wing design providing two features key to the success of this small UAV. First, the elastic properties of the wing battens, constructed of unidirectional carbon fiber strips, allow passive adaptive washout, improving stability and gust response. Much like the feathers of a bird, the battens near the wing root are stiffened with an increased number of layers of carbon fiber, while the tip battens remain thin to promote flexibility. The second feature is the ability to fold the wings. The main spar of the wing uses a bidirectional carbon fiber mesh with the fibers oriented ±45 deg relative to the span. The fabrication technique, combined with the geometry of the airfoil camber, results in a structure that is stiff in the upward direction and flexible in the downward direction, allowing the wings to be rolled underneath the fuselage. This wing design, along with a non-rigid V-tail design, allows the BATCAM to be stored, fully assembled and ready to fly, inside a tube less than six inches in diameter.

The propulsion system consists completely of low-cost commercial-off-the-shelf products. Lithium-polymer batteries provide a high energy-density power source for the brushless electric motor and all electronics. With any aircraft of this type, the power budget is of great importance. Drawing only 250mA, the autopilot system (including GPS receiver and communications transceiver) described in Sec. 2 consumes less than 10% of the total energy required for a typical flight.

IV. Autopilot Design

In this section we briefly describe the autopilot design as well as our technique for attitude estimation. The autopilot board, which was designed and built at BYU, is shown in Fig. 5. The CPU on the autopilot is a 29 MHz Rabbit microcontroller with 512K Flash and 512K RAM. The autopilot has four servo channels, two 16 channel, 12 bit analog-to-digital converters, four serial ports, and five analog inputs. On-board sensors include three-axis rate gyros with a range of 300 degrees per second, three axis accelerometers with range of two *g*s, an absolute pressure sensor capable of measuring altitude to within eight feet, a differential pressure sensor capable of measuring airspeed to within 0.36 feet per second, and a standard GPS receiver. The autopilot package weighs 2.25 ounces including the GPS antenna. The size of the autopilot is roughly 3.5 inches by 2 inches.

Following standard procedures[4-6] we assume that the longitudinal and lateral dynamics of the UAV are decoupled and design longitudinal and lateral autopilots independently. As shown in Fig. 6, the inputs to the longitudinal autopilot are commanded altitude, h^c and commanded velocity, V^c. The outputs are the elevator deflection, δ_e, and the motor voltage, δ_t. The Altitude Hold autopilot converts altitude error into a commanded pitch angle θ^c. The Pitch Attitude Hold autopilot converts pitch attitude error into a commanded pitch rate q^c. The Pitch Rate Hold autopilot converts pitch rate error to elevator command. The Velocity Hold autopilot converts velocity error to throttle command.

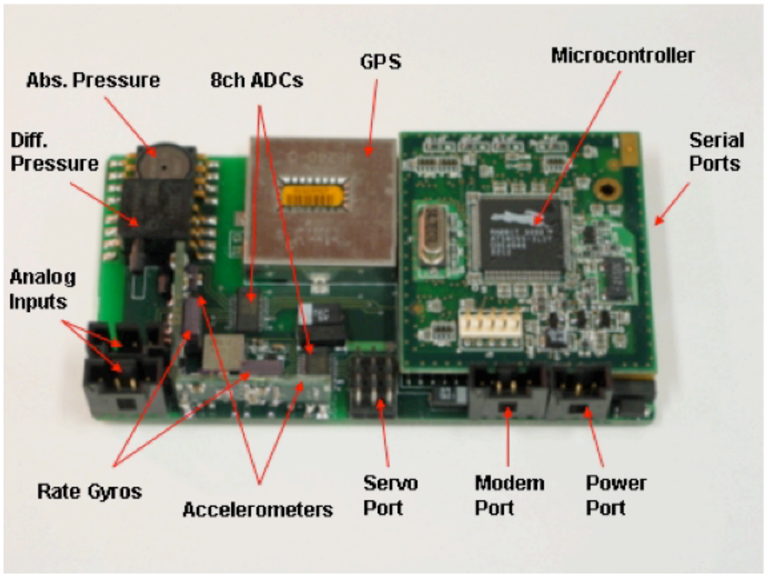


Fig. 5 BYU autopilot hardware.

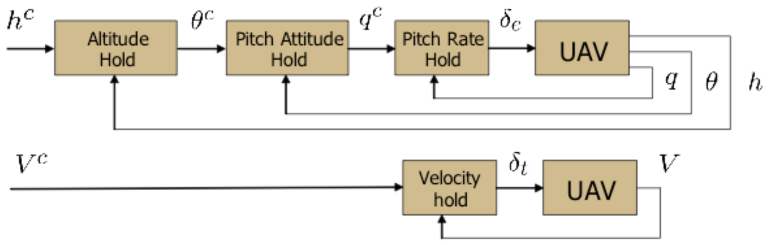


Fig. 6 Autopilot for longitudinal motion.

The lateral autopilot is shown in Fig. 7. The input command to the lateral autopilot is the commanded heading, ψ^c. The output is the aileron command δ_a. The Heading Hold autopilot converts heading error to roll attitude command, ϕ^c. The Roll Attitude Hold autopilot converts roll angle error to roll rate command, p^c. The Roll Rate Hold autopilot converts the roll rate error to aileron command, δ_a. Each autopilot mode is realized with a PID controller augmented with output saturation and integrator anti-windup.

Unfortunately, roll and pitch angles are not directly measurable with low-cost sensors. In addition, heading angle is measured with GPS at very low data rates. To compensate, the autopilot has been augmented with an Extended Kalman Filter (EKF) to estimate roll, pitch, and heading angle.

The EKF uses rate gyro information to do the time update and the accelerometers in combination with the airspeed sensor to do the measurement update. The EKF time update equations are as follows

$$\dot{\phi} = p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \tag{1}$$

$$\dot{\theta} = q\cos\phi - r\sin\phi \tag{2}$$

$$\dot{\psi} = q\sin\phi\sec\theta + r\cos\phi\sec\theta \tag{3}$$

where p, q, and r are roll, pitch, and yaw rates as measured by on-board rate gyros. A simple Euler approximation is used to convert to a discrete update equation.

Part of the complication of estimating roll and pitch angles is the lack of a good sensor to measure them directly. However, roll and pitch angles can be approximated using accelerometer measurements. The basic idea is to use an estimate of the direction of the gravity vector to extract roll and pitch angles. This can be done from the following set of equations:

$$\dot{u} = -g\sin\theta + A_x + vr - wq \tag{4}$$

$$\dot{v} = g\sin\phi\cos\theta + A_y + wp - ur \tag{5}$$

$$\dot{w} = g\cos\phi\cos\theta + A_z + uq - vp \tag{6}$$

where A_* are the accelerometer measurements, and u, v, w are the body velocities of the UAV. Because u, v, and w are not measured, some simplifying assumptions need to be made. First, we assume that $\dot{u}, \dot{v}$, and $\dot{w}$ are zero. This is true over short periods of time as well as in steady-state flight. The next assumption is to notice that in steady-state flight (coordinated turn or level flight) $v \approx 0$. The last assumption is that u and w are proportional to the airspeed (V_p). Through analysis of flight data on our UAVs the following have been determined to be decent measurements of roll and pitch

$$\theta_m = \sin^{-1}\left(\frac{A_x - 0.2V_p q}{g}\right), \tag{7}$$

$$\phi_m = \sin^{-1}\left(\frac{-A_y + 0.4V_p r}{g\cos\theta_m}\right), \tag{8}$$

These measurements, along with a GPS heading measurement, are used to do the EKF measurement update. Figure 8 shows the filter output (blue) versus the measured values (green). This was generated from real data gathered during flight tests.

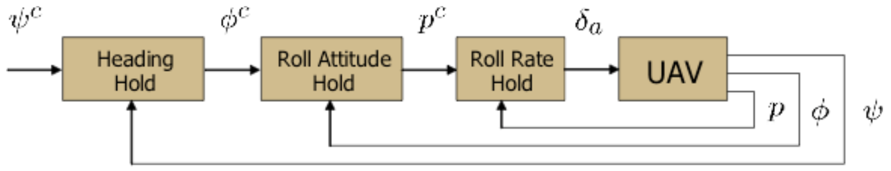


Fig. 7 Autopilot for lateral motion.

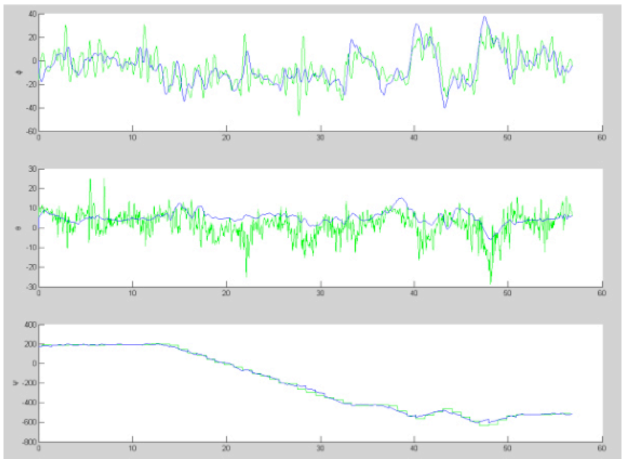


Fig. 8 EKF (blue) vs measured (green) of roll, pitch, and heading.

V. Ground Station

Essential to development, debugging, and visualization is the Ground Station which allows easy interface to everything on the UAV; from raw analog-to-digital sensor readings to the current PID values in the control loops. Status 8 packets are sent to the Ground Station at 5 Hz from the UAV over a 900 MHz wireless link to indicate the state of the UAV and its controllers. This allows for real-time plotting of position (map and waypoints), altitude, airspeed, etc. It also allows the user to be aware of GPS status, battery voltage, and communication link status. The ground station default window is shown in Fig. 9.

The Ground Station was designed primarily to help tune the autopilot. To this end, a user-configurable data-logging tool was added. The Data-logger commands the autopilot to store the state of the UAV for a specified period of time. When the log is completed, it is transmitted back to the Ground Station for viewing. This is especially helpful to see what the UAV was planning and commanding during maneuvers. Essentially every variable in the autopilot can be data-logged, allowing the user to reconstruct a flight as the autopilot viewed it. This capability has been used to debug the autopilot, build Kalman filters, and develop waypoint navigation capability. A user-configurable data log interface is shown in Fig. 10.

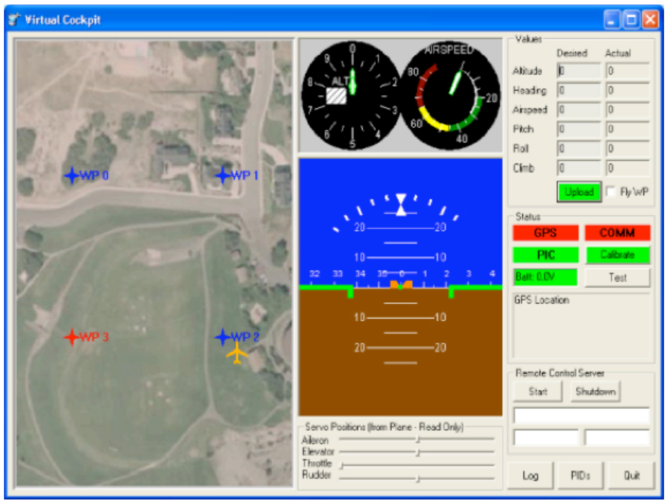
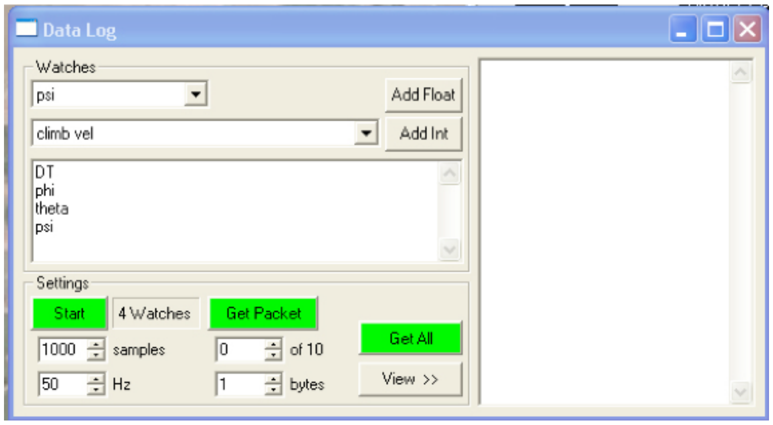


Fig. 9 Virtual cockpit interface.



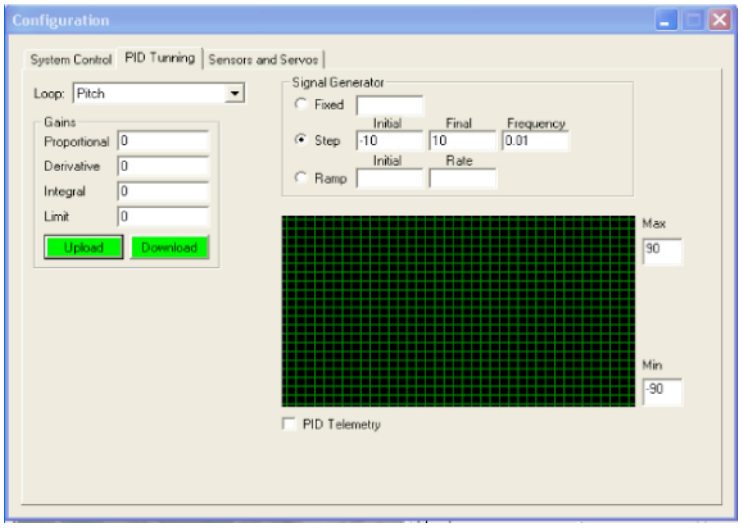
Watches: psi; climb vel

DT
phi
theta
psi

4 Watches
1000 samples
50 Hz
0 of 10
1 bytes

Fig. 10 User-configurable data log interface.

Because the autopilot control system consists of PID loops, tuning the PID values is very important to the performance of the UAV. To facilitate this, a real-time PID tuning and graphing utility is integrated into the Ground Station. It allows the user to request and set PID values on any loop while also providing the capability to command steps, ramps, and square waves to the different loops. These commands are plotted next to the performance of the UAV in real-time. Using the graphical information, the user can easily adjust values to tune the loops to desired specifications. Our autopilot has been flown on a number of different platforms, each requiring the PID loops to be re-tuned. Using the Ground Station has accelerated the tuning process, so that the autopilot can be tuned for a new UAV in less than 15 minutes. The PID tuning window is shown in Fig. 11.



Loop: Pitch

Proportional 0
Derivative 0
Integral 0
Limit 0

Step: Initial -10, Final 10, Frequency 0.01

Ramp: Initial, Rate

Max 90

Min -90

Fig. 11 Ground-station: PID tuning.

VI. PDA Interface

To facilitate simple and intuitive operation of the UAV, we have developed PDA and voice interfaces which are based on our previous work in human-robot interaction.[7-9]

The interface between the base station and the PDA and voice interface is shown in Fig. 12. The base station instantiates a server accepting wireless 802.11b connections. The base station server accepts a simple set of commands that include reference velocity, altitude, heading, roll angle, and climb rates. The server can provide telemetry data obtained directly from the autopilot. The communication packets are formatted in ASCII, thus enabling the use of a wide variety of external user interface devices. The PDA is required to request telemetry packets from the server which ensures that the data transmission rate does not exceed the rate at which the client can process and display the data.

The PDA interface to the UAV is shown in Fig. 13. The left side of the display compactly displays both roll attitude as well as relative altitude. The “wing-view” display is created from the vantage point of an observer behind the UAV who is looking through an abstracted cross-section of the UAV’s main wing in the direction that the UAV is flying. This kind of representation is simple to draw, which is a necessary condition when dealing with the limited graphics hardware of a typical PDA.

Despite its simplistic appearance, the “wing-view” presents a persuasive abstract visualization of the instantaneous relationship between the UAV and the world. The exact altitude is intentionally not shown; the designer of the system specifies a “high” and a “low” altitude for the given application, and the user need not be concerned with the exact altitude target that he or she is sending. The altitude constraints are previously created using the designer’s knowledge of both the requirements of the application and the capabilities of the UAV.

The wing-view display draws three “control handles” on the abstract representation of the UAV, signified by dotted white boxes. The handle in the center of the wing gives the user control of the UAV’s target altitude. Using the PDA stylus, the user simply drags the center of the UAV to the desired target altitude. The handle on each wingtip allows the user to simply drag the wingtip to create the desired roll angle. The interface will not allow the user to create a roll angle beyond the constraints set by the designer.

It will take the UAV some time to meet the user’s commands, particularly if the user has desired a significant change in altitude. When the desired flight characteristics are significantly different from the UAV’s current telemetry data, a “target UAV” is drawn with a red dotted line, which contrasts sharply with the heavy blue line of the “real UAV,” and the control handles are plotted on this target UAV. This approach allows for the user to receive instantaneous feedback from the display, and confirms that the user’s commands have been received. The target UAV is plotted as the stylus moves, without waiting for the real UAV to receive the new target, which further improves the interface response time. Multi-threading allows the real-time telemetry (the “real UAV”) to continue to be received and plotted on the display while the user is dragging the target plane. When the actual UAV states approach those of the target UAV, the target UAV disappears and the control handles return to the real UAV.

In addition to altitude and roll angle commands, the user can also specify the heading and velocity of the UAV. As shown in Fig. 13, the interface to these commands is intuitive and simple to render on the PDA. As with altitude, a precise scale for the velocity gauge is unnecessary and is intentionally not supplied. Each UAV airframe will have a different range of acceptable velocities; using an absolute velocity scale would require users to learn more than necessary about the performance characteristics and stall speed of each particular UAV. The heading and velocity gauges also follow the principles of direct manipulation: the user can drag the velocity or heading “indicator needle” to the desired position, and the UAV will then seek to match the user input.

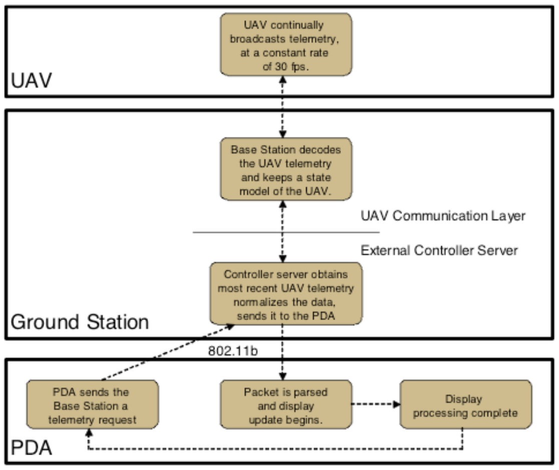

UAV continually broadcasts telemetry, at a constant rate of 30 fps.
UAV
Base Station decodes the UAV telemetry and keeps a state model of the UAV.
UAV Communication Layer
External Controller Server
Controller server obtains most recent UAV telemetry normalizes the data, sends it to the PDA
Ground Station
802.11b
PDA sends the Base Station a telemetry request
Packet is parsed and display update begins.
Display processing complete
PDA

Fig. 12 Telemetry and visualization timing.

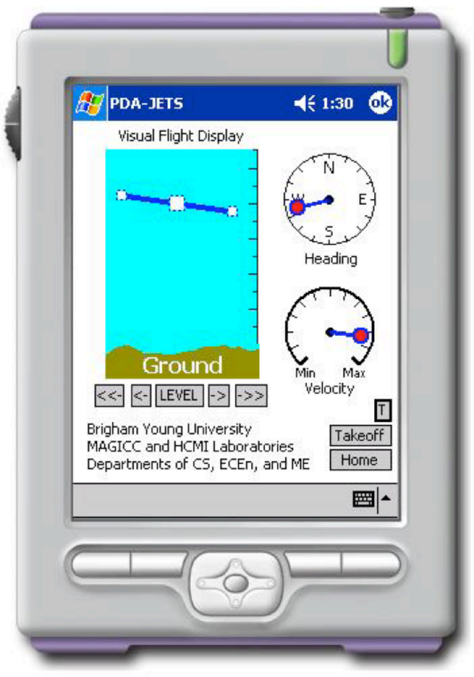


Fig. 13 Complete PDA screen.

The roll angle and heading commands are competing objectives. Therefore, it is necessary for the interface to determine whether to use the autopilot's "heading hold" mode, or the "roll hold" mode, and the interface must present a clear visualization of this decision. In our implementation, the most recently issued user command is taken to imply the desired control mode of the UAV. If the user had previously supplied a target roll angle, but then drags the compass needle to signify a target heading, the interface implicitly switches control modes and commands the autopilot to switch to heading-hold mode. The target roll angle is leveled out on the wing-view display to inform the user of this control mode change.

Takeoff and Home buttons are also part of the UAV interface shown in Fig. 13. An automated takeoff is achieved by specifying a target altitude, zero roll angle, and a medium velocity. These parameters are automatically set when the "takeoff" button is pressed. The takeoff mode allows a single user to tap the "takeoff" button, hand-launch the UAV, and then use the PDA for flight control. The Home button commands the UAV to return to the GPS location from which it was launched. A .wmv movie demonstrating the PDA interface is shown by clicking here.

In addition to the PDA interface, we have implemented a voice activated user interface that can recognize commands such as "Turn Left," "Climb," "Speed up," "Go North," etc., using a grammar of twenty commands. A speech recognition agent listens for these commands, and when a successful recognition is made, a speech synthesis agent offers instantaneous feedback by simply stating the command in the present progressive tense: "Turning Left," "Climbing," "Speeding up," or "Going North." A command packet is then created and sent to the base station server via the open 802.11b connection, which subsequently translates it into a condensed binary format and uploads it to the UAV.

Our current implementation of the voice controller runs on a separate laptop. The UAV operator plugs a headset microphone into the laptop, closes the laptop lid, tucks the laptop under an arm or in a backpack, and is free to roam inside the 100-yard range of the wireless connection with the base station. If the application required it, the entire base station could also be placed in a backpack, giving the UAV operator complete freedom of movement within the range of radio communication with the UAV. A .wmv movie demonstrating the voice interface is shown by clicking here.

VII. Waypoint Path Planning

For many anticipated military and civil applications, the capability for a UAV to plan its route as it flies is important. Reconnaissance, exploration, and search and rescue missions all require the ability to respond to sensed information and to navigate on-the-fly.

In the flight control architecture shown in Fig. 2, the coarsest level of route planning is carried out by the path planner (PP). Our approach uses a modified Voronoi diagram to generate possible paths. The Voronoi graph provides a method for creating waypoint paths through threats or obstacles in the airspace. The Voronoi diagram is then searched via Eppstein's k-best paths algorithm,[11] which is a computationally efficient search algorithm. Similar path planners have been previously reported in Refs. 12-14.

We model threats and obstacles in two different ways: as points to be avoided and as polygonal regions that cannot be entered. With threats specified as points, construction of the Voronoi graph is straightforward using existing algorithms. For an area with n point threats, the Voronoi graph will consist of n convex cells, each containing one threat. Every location within a cell is closer to its associated threat than to any other. By using threat locations to construct the graph, the resulting graph edges form a set of lines that are equidistant from the closest threats. In this way, the edges of the graph maximize the distance from the closest threats. Figure 14 shows an example of a Voronoi graph constructed from point threats.

Construction of a Voronoi graph for obstacles modeled as polygons is an extension of the point-threat method. In this case, the graph is constructed using the vertices of the polygons that form the periphery of the obstacles. This initial graph will have numerous edges that cross through the obstacles. To eliminate infeasible edges, a pruning operation is performed. Using a line intersection algorithm, those edges that intersect the edges of the polygon are removed from the graph. Figure 15 shows an initial polygon based graph and the final graph after pruning.

Finding good paths through the graph requires the definition of a cost metric associated with traveling along each edge. We employ two metrics: path length and threat exposure. A weighted sum of these two costs provides a means for finding safe, but reasonably short paths. Although the highest priority is usually threat avoidance, the length of the path must be considered to prevent safe, but meandering paths from being chosen. Once a metric is defined, the graph is searched using an Eppstein search.[11] This is a computationally efficient search with the ability to return k best paths through the graph.

The path planner shown in Fig. 2 produces straight-line waypoint paths. The paths are not time-stamped and are not kinematically feasible for the UAVs. The task of the Trajectory Smoother is to transform the waypoint paths into time-stamped, kinematically feasible trajectories that can be followed by the UAV.[14]

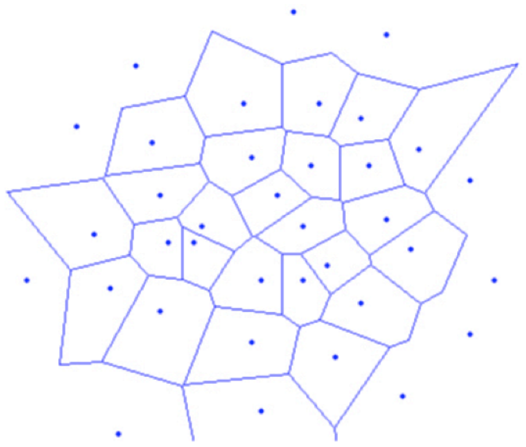


Fig. 14 Voronoi graph with point threats.

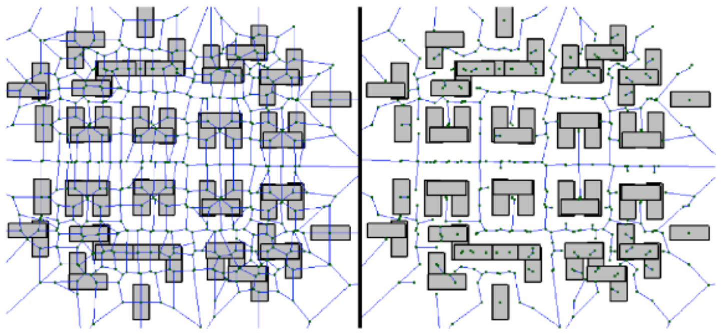


Fig. 15 Voronoi graph before and after pruning with polygon threats.

VIII. Trajectory Smoother

This section describes our approach to the Trajectory Smoother shown in Fig. 2. A complete description is given in Refs. 15-17.

The Trajectory Smoother (TS) translates a straight-line path into a feasible trajectory for a UAV with velocity and heading rate constraints. Our particular implementation of trajectory smoothing also has some nice theoretical properties including time-optimal waypoint following.[15-16]

We start by assuming that an auto-piloted UAV is modeled by the kinematics equations

$$\begin{aligned}\dot{x} &= v^c \cos(\psi)\\ \dot{y} &= v^c \sin(\psi)\\ \dot{\psi} &= \omega^c,\end{aligned} \tag{9}$$

where (x,y) is the inertial position of the UAV, ψ is its heading, v^c is the commanded linear speed, and ω^c is the commanded heading rate. The dynamics of the UAV impose input constraints of the form

$$\begin{aligned}0 < v_{\min} \le v^c \le v_{\max}\\ -\omega_{\max} \le \omega^c \le \omega_{\max}\end{aligned}. \tag{10}$$

The fundamental idea behind feasible, time-extremal trajectory generation is to impose on the Trajectory Smoother a mathematical structure similar to the kinematics of the UAV. Accordingly, the structure of the Trajectory Smoother is given by

$$\begin{aligned}\dot{x}_r &= v_r \cos(\psi_r)\\ \dot{y}_r &= v_r \sin(\psi_r)\\ \dot{\psi}_r &= \omega_r,\end{aligned} \tag{11}$$

with the constraints that v_r and ω_r are piecewise continuous in time and satisfy the constraints

$$\begin{aligned}v_{\min} + \varepsilon_v \le v_r \le v_{\max} - \varepsilon_v\\ -\omega_{\max} + \varepsilon_\omega \le \omega_r \le \omega_{\max} - \varepsilon_\omega,\end{aligned} \tag{12}$$

where ε_v and ε_ω are positive control parameters that will be exploited in Sec. 9 to develop a trajectory tracking algorithm that satisfies the original constraints in Eq. (10). To simplify notation, let $c = \omega_{\max} - \varepsilon_\omega$. With the velocity fixed at $\hat{v}$, the minimum turn radius is defined as $R = \hat{v}/c$. The minimum turn radius dictates the local reachability region which is shown in Fig. 16.

As shown in Refs. 15 and 17, for a trajectory to be time-optimal, it will be a sequence of straight-line path segments combined with arcs along the minimum radius circles (i.e. along the edges of the local reachability region). By postulating that ω_r follows a bang-bang control strategy during transitions from one path segment to the next, Refs. 15, and 17 showed that a k-trajectory is time-extremal, where a κ-trajectory is defined as follows.

Definition 8.1 *As shown in Fig. 17, a κ-trajectory is defined as the trajectory that is constructed by following the line segment $\overline{\mathbf{w}_{i-1}\mathbf{w}_i}$ until intersecting $\mathcal{C}_i$, which is followed until $\mathcal{C}_{\mathbf{p}(\kappa)}$ is intersected, which is followed until intersecting $\mathcal{C}_{i+1}$ which is followed until the line segment $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$ is intersected.*

Note that different values of κ can be selected to satisfy different requirements. For example, κ can be chosen to guarantee that the UAV explicitly passes over each waypoint, or κ can be found by a simple bi-section search to make the trajectory have the same length as the original straight-line path,[17] thus facilitating timing-critical trajectory generation problems.

The TS implements κ-trajectories to follow waypoint paths. In evaluating the real-time nature of the TS, we chose to require trajectories to have path length equal to the initial straight line waypoint paths. The computational complexity to find ω_r is dominated by finding circle-line and circle-circle intersections. Since the TS also propagates the state of the system in response to the input ω_r, Eq. (11) must be solved in real-time. This is done via a fourth order Runge-Kutta algorithm.[18] In this manner, the output of the TS corresponds in time to the evolution of the UAV kinematics and ensures a time-optimal kinematically feasible trajectory.

Hardware implementation of the TS has shown the real-time capability of this approach. On a 1.8 GHz Intel Pentium 4 processor, one iteration of the TS took on average 36 μ-seconds. At this speed, the TS could run at 25 kHz - well above the dynamic range of a small UAV. On the autopilot discussed in Sec. 4, one iteration of the TS requires 16 a maximum of 47 milliseconds. The low computational demand allows the TS to be run in real-time at approximately 20 Hz.

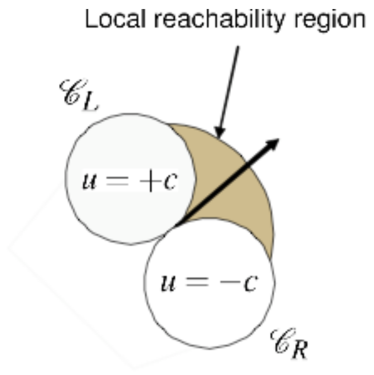


Fig. 16 Local reachability region of the TS.

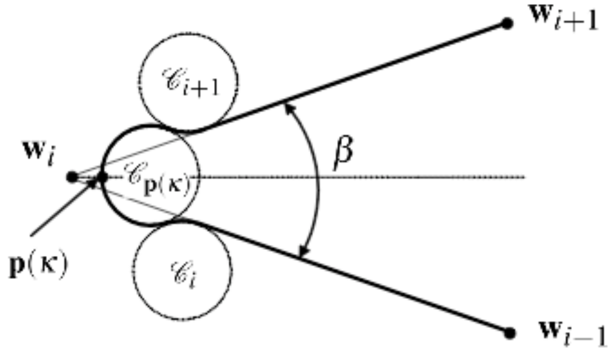


Fig. 17 A dynamically feasible κ-trajectory.

IX. Trajectory Tracker

This section gives a brief overview of our trajectory tracker. A complete description is contained in Refs. 19 and 20. We will assume that the UAV/autopilot combination is adequately modeled by the kinematic equations given in Eq. (9), with the associated constraints given in Eq. (10). As explained in Sec. 8, the trajectory generator produces a reference trajectory that satisfies Equation (11) with input constraints given by Eq. (12), where ε_v and ε_ω are control parameters that will be exploited in this section to facilitate asymptotic tracking. The trajectory tracking problem is complicated by the nonholonomic nature of Eq. (9) and the positive input constraint on the commanded speed v^c. The first step in the design of the trajectory tracker is to transform the tracking errors to the UAV body frame as follows:

$$\begin{bmatrix} x_e \\ y_e \\ \psi_e \end{bmatrix} = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \psi_r - \psi \end{bmatrix}. \tag{13}$$

Accordingly, the tracking error model can be represented as

$$\begin{aligned}\dot{x}_e &= \omega^c y_e - v^c + v_r \cos(\psi_e)\\ \dot{y}_e &= \omega^c x_e - v^c + v_r \sin(\psi_e)\ .\\ \dot{\psi} &= \omega_r - \omega^c\end{aligned} \tag{14}$$

Following Ref. 21, Eq. (14) can be simplified to

$$\begin{aligned}\dot{x}_0 &= u_0\\ \dot{x}_1 &= (\omega_r - u_0)x_2 + v_r \sin(x_0)\\ \dot{x}_2 &= -(\omega_r - u_0)\, x_1 + u_1\end{aligned} \tag{15}$$

where

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \psi_e \\ y_e \\ -x_e \end{bmatrix} \tag{16}$$

and

$$\begin{bmatrix} u_0 \\ u_1 \end{bmatrix} = \begin{matrix} \omega_r - \omega_c \\ v^c - v_r \cos(x_0) \end{matrix}.$$

The input constraints under the transformation are given by

$$\begin{gathered} -\varepsilon_\omega \le u_0 \le \varepsilon_\omega \\ v_{\min} - v_r \cos(x_0) \le u_1 \le v_{\max} - v_r \cos(x_0) \end{gathered}. \tag{17}$$

Obviously, Eqs. (13) and (16) are invertible transformations, which means that is equivalent to (x_0, x_1, x_2) = (0, 0, 0) and (x_e, y_r, ψ_r) = (x, y, ψ). Therefore, the original tracking control objective is converted to a stabilization objective, that is, our goal is to find feasible control inputs u_0 and u_1 to stabilize x_0, x_1, and x_2.

Note from Eq. (15) that when both x_0 and x_2 go to zero, that x_1 becomes uncontrollable. To avoid this situation we introduce another change of variables. Let

$$\bar{x}_0 = m x_0 + \frac{x_1}{\pi_1} \tag{18}$$

where $m > 0$ and $\pi_1 \overset{\Delta}{=} \sqrt{x_1^2 + x_2^2 + 1}$. Accordingly, $x_0 = \frac{\bar{x}_0}{m} - \frac{x_1}{m\pi 1}$. Obviously, $(\bar{x}_0, x_1, x_2)$ is equivalent to (x_0, x_1, x_2) = (0, 0, 0). Therefore it is sufficient to find control inputs u_0 and u_1 to stabilize $\bar{x}_0, x_1, \text{and}\, x_2$. With the same input constraints of Eq. (17), Eq. (15) can be rewritten as

$$\begin{aligned}\dot{\bar{x}}_0 &= \left(m - \frac{x_2}{\pi_1}\right)u_0 + \frac{x_2}{\pi_1}\omega_r\\ &\quad + \frac{1 + x_2^2}{\pi_1^3} v_r \sin\left(\frac{\bar{x}_0}{m} - \frac{x_1}{m\pi_1}\right) - \frac{x_1 x_2}{\pi_1^3} u_1\ .\\ \dot{x}_1 &= (\omega_r - u_0)x_2 + v_r \sin\left(\frac{\bar{x}_0}{m} - \frac{x_1}{m\pi_1}\right)\\ \dot{x}_2 &= -(\omega_r - u_0)x_1 + u_1\end{aligned} \tag{19}$$

In Refs. 19, and 20 we have shown that if

$$u_0 = \begin{cases} -\eta_0 \bar{x}_0, & |\eta_0 \bar{x}_0| \le \varepsilon_\omega \\ -\text{sign}(\bar{x}_0)\varepsilon_\omega, & |\eta_0 \bar{x}_0| > \varepsilon_\omega \end{cases} \tag{20}$$

$$u_1 = \begin{cases} \underline{v} & -\eta_1 x_2 < \underline{v} \\ -\eta_1 x_2, & \underline{v} \le -\eta_1 x_2 \le \bar{v}, \\ \bar{v}, & -\eta_1 x_2 > \bar{v} \end{cases} \tag{21}$$

where $\underline{v} = v_{\min} - v_r \cos\left(\frac{\bar{x}_0}{m} - \frac{x_1}{m\pi_1}\right)$ and η_0 and η_1 are sufficiently large (made precise in Refs. 19 and 20), then the tracking error goes to zero asymptotically.

Note the computational simplicity of Eqs. (20)-(21). We have found that the tracker can be efficiently implemented on the autopilot hardware discussed in Sec. 3. Figure 18 shows a reference trajectory of the UAV in green and the actual UAV trajectory in blue. Figure 19 plots the tracking errors verses time and demonstrates the asymptotically stable characteristics of the trajectory tracker.

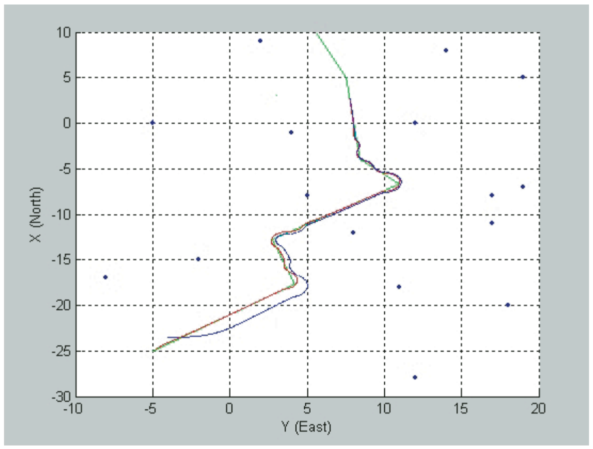


Fig. 18 The simulation scenario: waypoint path (green), smoothed reference trajectory (red), and actual trajectory (blue).

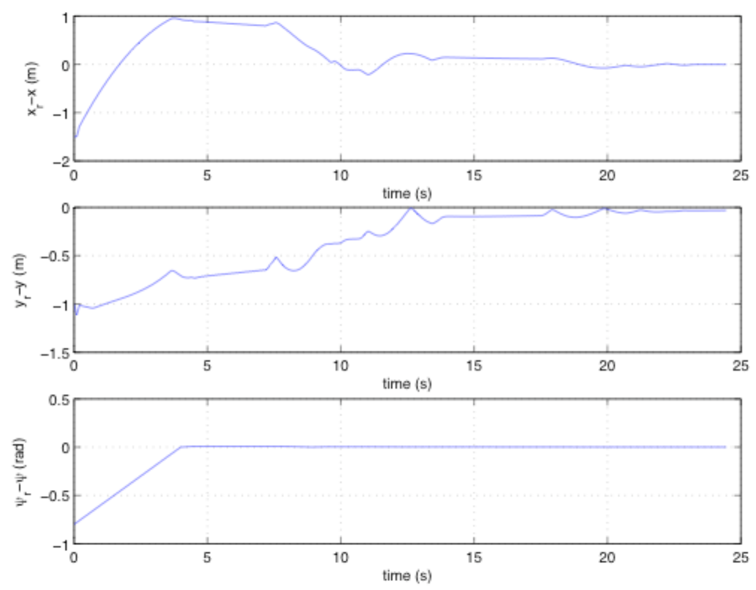


Fig. 19 The trajectory tracking errors expressed in the inertial frame.

X. Conclusion

This paper has described an approach to autonomous navigation, guidance and control for small fixed-wing UAVs. In particular, we have described miniature UAV hardware, low-cost, lightweight autopilot technologies, a novel PDA and voice interface, and a computationally efficient approach to path planning, trajectory generation and tracking. A video of a flight test scenario using the Tactical MAV described in Sec. 3 can be seen by clicking here. The video clip shows the operator hand launching the UAV, followed by an automated climb-out routine. Once the UAV has reached an operating altitude, the operator directs the UAV via waypoint and heading commands. The video clip also contains video footage captured by the UAV, and completes by showing an automated landing.

Acknowledgments

This work was funded by AFOSR grants F49620-01-1-0091 and F49620-02-C-0094, and by DARPA grant NBCH1020013. The authors acknowledge the efforts of David Hubbard in developing the ground station described in this paper.

References

[1]Grasmeyer, J. M., and Keennon, T. M., "Development of the Black Widow Micro Air Vehicle*," Proceedings of the AIAA 39th Aerospace Sciences Meeting and Exhibit*, AIAA Paper No. 2001-0127, Jan. 2001.

[2]Mueller, T. J., and DeLaurier, J. D., "Aerodynamics of Small Vehicles," *Annual Review of Fluid Mechanics*, Vol. 35, 2003, pp. 89–111.

[3]Fju, P. G., Jenkins, D. A., Ettinger, S., Lian, Y., and Shyy, W., "Flexible-Wing-Based Micro Air Vehicles*," Proceedings of the AIAA 40th Aerospace Sciences Meeting and Exhibit*, AIAA Paper No. 2002-0705, Jan. 2002.

[4]Nelson, R. C., *Flight Stability and Automatic Control*, 2nd ed., McGraw-Hill, Boston, 1998.

[5]Rauw, M., *FDC 1.2 - A SIMULINK Toolbox for Flight Dynamics and Control Analysis*, Feb. 1998; available online at http://www.mathworks.com (cited Nov. 2004).

[6]Roskam, J., *Airplane Flight Dynamics and Automatic Flight Controls*, Parts I & II, DARcorporation, Lawrence, KS, 1998.

[7]Goodrich, M. A., and Boer, E. R., "Designing Human-Centered Automation: Tradeoffs in Collision Avoidance System Design*," IEEE Transactions on Intelligent Transportation Systems*, Vol. 1, No. 1, March 2000.

[8]Goodrich, M. A., and Boer, E. R., "Model-Based Human-Centered Task Automation: A Case Study in ACC Design," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, Vol. 33, No. 3, May 2003, pp. 325-336.

[9]Crandall, J. W., and Goodrich, M. A., "Characterizing Efficiency of Human Robot Interaction: A Case Study of Shared-Control Teleoperation," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

[10]Sedgewick, R., *Algorithms*, Addison-Wesley, 2nd ed., 1988.

[11]Eppstein, D., "Finding the k shortest paths," *SIAM Journal of Computing*, Vol. 28, No. 2, 1999, pp. 652–673.

[12]McLain, T., and Beard, R., "Cooperative Rendezvous of Multiple Unmanned Air Vehicles," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, AIAA Paper 2000-4369, Aug. 2000.

[13]Chandler, P., Rasumussen, S., and Pachter, M., "UAV Cooperative Path Planning," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, AIAA Paper No. AIAA-2000-4370, Aug. 2000.

[14]Beard, R. W., McLain, T. W., Goodrich, M., and Anderson, E. P., "Coordinated Target Assignment and Intercept for Unmanned Air Vehicles," *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 6, Dec. 2002, pp. 911–922.

[15]Anderson, E. P. and Beard, R.W., "An Algorithmic Implementation of Constrained Extremal Control for UAVs," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, AIAA Paper No. 2002-4470, Aug. 2002.

[16]Anderson, E. P., Beard, R. W., and McLain, T. W., "Real Time Dynamic Trajectory Smoothing for Uninhabited Aerial Vehicles," *IEEE Transactions on Control Systems Technology*, in review.

[17]Anderson, E. P., "Constrained Extremal Trajectories and Unmanned Air Vehicle Trajectory Generation," Master's thesis, Brigham Young Univ., Provo, UT, April 2002; available online at http://www.ee.byu.edu/ee/robotics/ publications/ thesis/ErikAnderson.ps (cited Nov. 2004).

[18]Burden, R. L., and Faires, J. D., *Numerical Analysis*, 4th ed, PWS-KENT Publishing Company, Boston, 1988.

[19]Ren, W., and Beard, R. W., "CLF-based Tracking Control for UAV Kinematic Models with Saturation Constraints," *Proceedings of the IEEE Conference on Decision and Control*, Dec. 2003, pp. 3924-3929.

[20]Ren, W., and Beard, R. W., "Trajectory Tracking for Unmanned Air Vehicles with Velocity and Heading Rate Constraints," *IEEE Transactions on Control Systems Technology*, Vol. 12, No. 5, Sept. 2004, pp. 706-716.

[21]Lee, T.-C., Song, K.-T., Lee, C.-H., and Teng, C.-C., "Tracking Control of Unicycle-Modeled Mobile Robots Using a Saturation Feedback Controller," *IEEE Transactions on Robotics and Automation*, Vol. 9, No. 2, March 2001, pp. 305–318.