

Fundamentals of Small Unmanned Aircraft Flight

Jeffrey D. Barton

B

ecause of advances in sensor, processing, and battery technologies over the past decade, unmanned aircraft systems (UASs) have become both smaller and more affordable. In particular, readily available low-weight, low-power, low-cost sensors based on microelectromechanical systems technology have facilitated the development of UAS autopilots for military and civil use, academic research, and recreation. The purpose of this article is to explore the fundamentals of state estimation and flight control for small fixed-wing UASs. The article covers the common sensors and sensor configurations used on small UASs for state estimation and, in general terms, the algorithms used to control UAS flight.

INTRODUCTION

Small unmanned aircraft systems (UASs) have become a mainstay in current military operations, providing combat troops and decision makers with vital intelligence, surveillance, and reconnaissance (ISR). A UAS, also dubbed an unmanned aerial vehicle (UAV) or drone, is a reusable aircraft that typically uses onboard sensors and processing to estimate its current kinematic state and automatically control its flight. UASs come in many different shapes and sizes and have been used in a variety of military and civilian applications including ISR, search and rescue, and atmospheric research.^{1,2} The purpose of this article is to explore the fundamentals of state estimation and flight control on small (<20 lb) fixed-wing UASs. Although dealing primarily with small fixed-wing aircraft, much of the information discussed in this article can be applied to other sizes and types of UASs.

Leveraging advances in sensor, processing, and battery technologies over the past decade, small UASs are a combination of sophisticated electronics and hobby remote-controlled (R/C) airplane components. The small UAS airframe might be specially designed or converted from a hobby R/C airplane, and it might use a battery-powered electric motor or a gas engine for powered flight. Typical small fixed-wing UASs, such as those shown in Fig. 1, can be hand-launched or launched with the assistance of a bungee or pneumatic launch device and might fly for 20–90 min at flight speeds between 10 and 50 m/s. Although capable of flying at much higher altitudes, small UASs will generally be flown between 30 and 400 m above ground level. The purpose of a UAS, of course, is to fly a payload, with the most common payload being a fixed or gimbaled video camera. Using

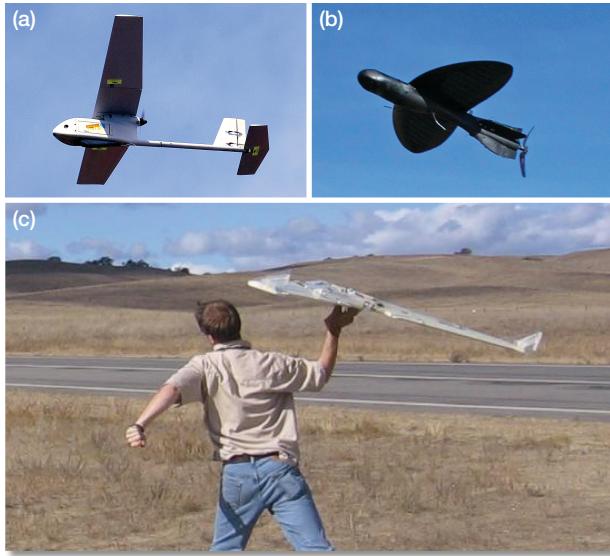


Figure 1. Small fixed-wing UASs can be specially designed, like the AeroVironment Raven (a) and the Prioria Maveric (b), or they can use an R/C hobby airframe such as in the flying wing used for UAS research at APL (c). (Image of the Prioria Maveric in b is reprinted with permission from Prioria Robotics © 2011.)

a radio communications link between the ground control station and the aircraft, an operator can send flight commands to the UAS and receive telemetry and video imagery in return.

The brain of the UAS is called the flight computer or autopilot. The autopilot, either solely or in combination with other avionics, uses onboard sensors to estimate its current position and orientation, perform flight control by translating flight commands into airframe actuator commands, and in some cases perform payload control (e.g., pointing a gimbaled camera). In addition to proprietary autopilots developed for specific applications, there are numerous commercially available autopilots,³ such as those shown in Fig. 2, that can be integrated into almost any appropriately sized, flyable airframe. These commercial autopilots provide waypoint navigation, in-flight rerouting, telemetry downlink, automatic takeoff and landing, gimbaled camera pointing control, user-friendly ground station software, and robust fail-safe features. The technology used in UAS autopilots has become so available that worldwide communities of do-it-yourself UAS hobbyists have emerged, providing open-source software and instructions to build functional UAS autopilots from low-cost components (see Ref. 4 for a community forum for the open-source ArduPilot series of autopilots, Ref. 5 for the open-

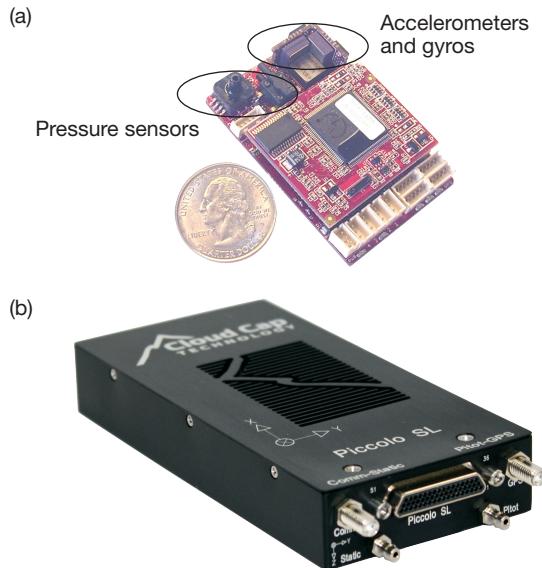


Figure 2. Commercially available autopilots with built-in micro-electromechanical systems (MEMS) gyros, accelerometers, and pressure sensors. (a) Procerus Technologies Kestrel Autopilot 2.4. (Reprinted with permission from Procerus Technologies © 2004–2011. All rights reserved.) (b) Cloud Cap Technology Piccolo SL Autopilot. (Reprinted with permission from Cloud Cap Technology © 2011.)

source Paparazzi series of autopilots, and Ref. 6 for the open-source GluonPilot autopilot). Although lacking some of the advanced capabilities of their commercial counterparts, these open-source autopilots provide fully functioning waypoint navigation and are supported by energetic and innovative online communities.

As the name implies, a UAS is a system of components, where the aircraft with its avionics, payload, and radio transponder is only a portion of that system. As a system, the UAS also includes the ground control station and its accompanying user-interface software, antennas, and radio transponders for both aircraft control and payload downlink. Although components such as radio communications, user interface, payload, and onboard power are of critical importance in a UAS, they

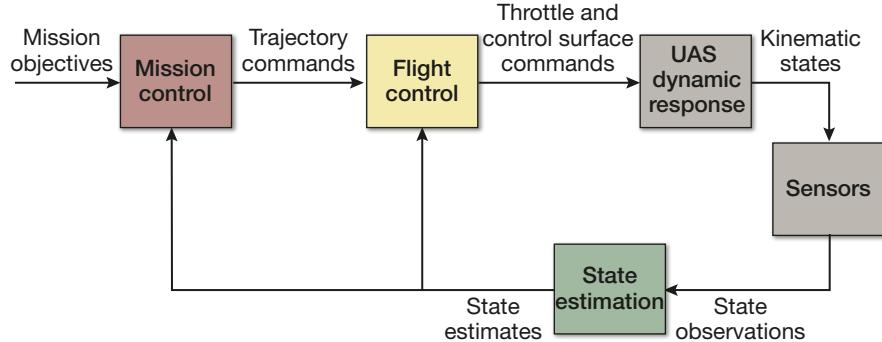


Figure 3. Block diagram representing the basic elements in controlling small UAS flight.

will not be discussed in any detail here.

Figure 3 shows a simplified block diagram for a generic UAS, including state estimation, flight control algorithms, and a mission controller. State estimation consists of fusing a combination of state-observing sensors [e.g., global positioning system (GPS) and gyros] and forming an estimate of the vehicle kinematic state (e.g., position and attitude). The flight control algorithms automatically manipulate the throttle and aerodynamic control surface actuators to achieve a set of trajectory commands using the kinematic state estimate as feedback. The trajectory commands are generated by the mission controller, which could be a human operator or a set of algorithms that convert mission objectives into trajectory commands.

In the following sections of this article, we explore the state estimation, flight control, and mission control components of a small fixed-wing UAS. The next section, which is also the bulk of the article, is entitled *State Estimation* and focuses on typical sensors and algorithms used on small UASs to observe and estimate the requisite kinematic states. The resulting state estimates are used in the common feedback control architectures described in the *Flight Control* section to achieve specified trajectory commands. Finally, the *Mission Control* section briefly describes some mission control mechanisms that can be used in a UAS, from a human operator to a set of algorithms providing full mission autonomy, to generate trajectory and payload commands. Note that the state estimation and flight control algorithms described herein do not represent the algorithms used in any specific autopilot. Rather, this article describes typical algorithms in a general context. For the interested reader, MATLAB code implementing some of the described UAS state estimation methods can be found at <http://www.jhuapl.edu/ott/Technologies/Copyright/SuasCode.asp>.

STATE ESTIMATION

A UAS uses a set of sensors in conjunction with appropriate onboard algorithms to measure and/or estimate translational states (position and velocity) and rota-

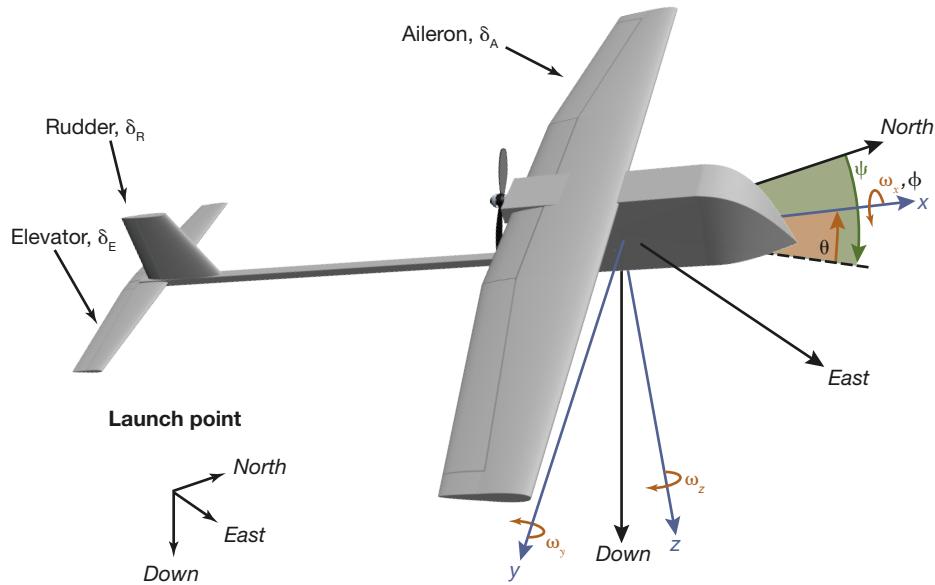


Figure 4. Representation of inertial North–East–Down coordinates, body x–y–z coordinates, and the aerodynamic control channels of elevator, aileron, and rudder. The Euler angles representing the rotation from North–East–Down coordinates to body x–y–z coordinates are yaw (ψ), pitch (θ), and roll (ϕ).

tional states (attitude and attitude rate). For the purpose of describing small UAS state estimation sensors and algorithms, it is necessary to define both a body-fixed coordinate frame and an inertial (nonmoving) coordinate frame. For a small UAS, we can generally ignore Earth rotation and curvature effects; hence it is sufficient to use a North–East–Down Cartesian coordinate frame centered at the launch position as an inertial frame. The North–East–Down and UAS body x–y–z coordinate frames used in this article are represented in Fig. 4.

Translational State Sensing

Global Positioning System

A GPS antenna and receiver are typically used to provide the UAS position and velocity in global coordinates. Specifically, an off-the-shelf GPS receiver can usually provide to the UAS its latitude, longitude, and altitude above mean sea level (MSL), in addition to its three-dimensional inertial velocity vector. Converting these measurements to local Cartesian coordinates, the GPS receiver provides the following state observations:

$P_{n,\text{GPS}}$: North position

$P_{e,\text{GPS}}$: East position

$h_{\text{MSL,GPS}}$: Height above MSL

$V_{n,\text{GPS}}$: North inertial velocity

$V_{e,\text{GPS}}$: East inertial velocity

$V_{d,\text{GPS}}$: Downward inertial velocity.

The North–East–Down GPS position and velocities can be written more efficiently in vector form, $\bar{P}_{\text{GPS}}^{\text{ned}} = [P_{n,\text{GPS}} \ P_{e,\text{GPS}} \ -h_{\text{MSL,GPS}}]^T$ and $\bar{V}_{\text{GPS}}^{\text{ned}} = [V_{n,\text{GPS}} \ V_{e,\text{GPS}} \ V_{d,\text{GPS}}]^T$. (Note that vectors in this article are printed with an overbar. The superscript provides the coordinate frame in which the vector is expressed.) The horizontal component of the GPS velocity vector can also be expressed as a GPS ground speed ($V_{H,\text{GPS}}$) and course heading (χ_{GPS}):

$$V_{H,\text{GPS}} = \sqrt{V_{e,\text{GPS}}^2 + V_{n,\text{GPS}}^2}, \quad (1a)$$

$$\chi_{\text{GPS}} = \tan^{-1}\left(\frac{V_{e,\text{GPS}}}{V_{n,\text{GPS}}}\right). \quad (1b)$$

As a rule of thumb, a standard nondifferential GPS receiver can provide position to an accuracy of 5–10 m at update rates between 1 and 10 Hz and generally has a response lag on the order of its update rate.⁷ (If needed, significantly improved position accuracies can be acquired using a differential GPS system, where the GPS receiver on board the UAS is aided by a nearby stationary receiver to better estimate atmospheric signal propagation effects.) Horizontally, GPS position accuracy is generally sufficient for UAS flight control and sensor pointing. Vertically, however, 5–10 m of error is oftentimes undesirable. Moreover, the GPS position accuracy deteriorates as fewer satellites are in view. Thus, to avoid premature flight termination (i.e., crashing), if the GPS fix is lost, most UASs use a static pressure sensor to estimate altitude. In this manner, the UAS’s vertical control system can still maintain a safe flight altitude even in the absence of GPS.

Pressure Sensors

A static pressure sensor is used to estimate UAS “barometric height” by comparing the current pressure measurement with the pressure at some reference height. Specifically, the change in static pressure, ΔP_s , is related to the change in height, Δh , by gravity, g , and air density, ρ . But, air density is a function of pressure, P_s , and air temperature, T_{air} :

$$\frac{\Delta P_s}{\Delta h} = -gp = -g\frac{M}{RT_{\text{air}}}P_s. \quad (2)$$

In Eq. 2, M is the molecular mass of air and R is the universal gas constant. For a constant temperature, Eq. 2 can be treated as a differential equation yielding a solution for pressure as a function of the change in height from a reference and the static pressure at the reference height, $P_{s,\text{ref}}$:

$$P_s = P_{s,\text{ref}} e^{-\frac{Mg}{RT_{\text{air}}}(h-h_{\text{ref}})}. \quad (3)$$

Thus, if the static pressure at the launch point, $P_{s,\text{launch}}$, is recorded by the UAS, the height above the

launch point (ALP) as estimated from the barometric pressure sensor is:

$$h_{\text{ALP,baro}} = -\frac{RT_{\text{air}}}{Mg} \ln(P_s/P_{s,\text{launch}}) + \varepsilon_h, \quad (4)$$

where ε_h represents sensing errors. By using a barometric pressure measurement, the UAS has a robust and stable means of estimating its altitude above the launch point. Variations in the local barometric pressure, however, can change over time and locale. So, if left uncorrected during lengthy flights, the barometric altitude estimate can drift significantly (e.g., 10 m over an hour). For this reason, some more advanced autopilots will periodically recalibrate the pressure reference using either the onboard GPS altitude estimate or a separate static pressure sensor at the ground station.

A second pressure sensor connected to a pitot tube sticking out into the airstream can be used to measure the axial airspeed of the UAS, which is the UAS speed relative to wind. The total pressure measured through the pitot tube, P_{pitot} , is the summation of the static pressure, P_s , and the dynamic pressure, P_{dyn} . Dynamic pressure is proportional to air density, ρ , and increases with the square of the airspeed, V_{air} :

$$P_{\text{dyn}} = P_{\text{pitot}} - P_s = \frac{1}{2}\rho V_{\text{air}}^2. \quad (5)$$

Thus, the wind-relative airspeed can be estimated by comparing the onboard pitot and static port pressure readings as in Eq. 6, where ε_V represents sensing errors:

$$V_{\text{air,pitot}} = \sqrt{2(P_{\text{pitot}} - P_s)/\rho} + \varepsilon_V. \quad (6)$$

Wind Estimation

A small UAS flying at relatively slow airspeeds (e.g., 10–20 m/s) will be heavily influenced by the movement of the air mass around it. In fact, flying in wind speeds up to 50% of the UAS airspeed is not uncommon. (The author has even caused a UAS to fly backward because the wind speed exceeded the UAS’s commanded airspeed.) As a result, in-flight estimation of the local wind vector can be very useful. Neglecting gusts and assuming that winds are slowly varying, a UAS can estimate the wind vector by comparing GPS inertial velocity measurements with pitot-tube airspeed measurements.⁸ Note the wind triangle in Fig. 5 relating the inertial velocity vector, $\bar{V}_{\text{GPS}}^{\text{ned}}$, with the wind vector, \bar{V}_w^{ned} , and the wind-relative airspeed vector, $\bar{V}_{\text{air}}^{\text{ned}}$. In the horizontal plane, the inertial velocity vector has a magnitude of V_H and a heading χ . Similarly, the wind vector has a magnitude V_w and a heading χ_w . Then, the horizontal component of airspeed, $V_{H,\text{air}}$, can be found using the law of cosines:

$$V_{H,\text{air}}^2 = V_H^2 + V_w^2 - 2V_H V_w \cos(\chi_w - \chi). \quad (7)$$

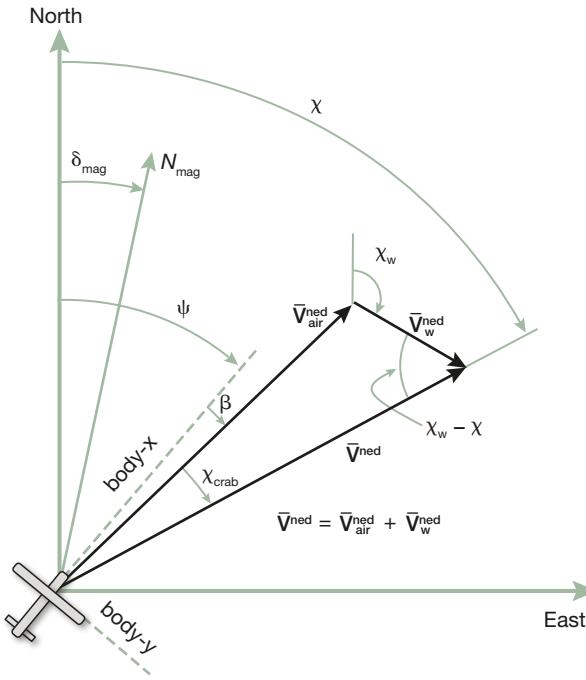


Figure 5. Representation of angular and vector relationships in the horizontal (East–North) plane. The UAS body- x axis is oriented at an angle of ψ (yaw) relative to true north. Similarly, the inertial velocity vector, \bar{V}^{ned} , and the wind vector, \bar{V}_w^{ned} , are oriented at angles χ and χ_w , respectively, relative to true north. $\bar{V}_{\text{air}}^{\text{ned}}$ is the wind-relative airspeed vector, which is related to the body- x direction and the inertial velocity vector by sideslip (β) and crab angle (χ_{crab}), respectively. Finally, magnetic north (N_{mag}) is deviated from true north by the angle δ_{mag} .

In Eq. 7, the inertial velocity components V_H and χ are measured by GPS as $V_{H,\text{GPS}}$ and χ_{GPS} , respectively. Furthermore, the horizontal component of airspeed can be approximated by rotating the pitot-measured axial airspeed by a pitch estimate, $\hat{\theta}$: $V_{H,\text{air}} \approx V_{\text{air,pitot}} \cos \hat{\theta}$. (Note that a “hat” denotes an estimated value in this article.) As a result, Eq. 7 provides an instantaneous equation relating wind speed and heading (V_w and χ_w) with otherwise measured or estimated quantities. Although Eq. 7 by itself is underdetermined (one equation with two unknowns), a sequence of N temporally separated realizations can yield N independent equations provided that χ_{GPS} changes sufficiently. Thus, using a least squares or iterative method, the complete horizontal wind vector (V_w and χ_w) becomes observable and can be measured whenever the UAS is turning.

Rotational State Sensing

Vehicle orientation estimates are critical for flight control and are often required for sensor pointing. The UAS orientation is generally denoted by three ordered Euler angles, namely yaw, pitch, and roll. These Euler angles define the transformation from the inertial

North–East–Down frame to the UAS body x – y – z frame, which is shown in Fig. 4. Although simple and intuitively appealing, the Euler angle representation suffers from nonunique orientation representations and a singularity as pitch approaches $\pm 90^\circ$. As a result, attitude computations within the autopilot are typically performed using some combination of direction cosine matrices (DCMs) and quaternions. DCMs, which are used throughout this article, are introduced in Box 1. Quaternions, though highly useful, are not used in this article.

Together, the pitch and roll Euler angles define the orientation of the UAS relative to horizontal, and thus both can be measured in similar manners. Most often, pitch and roll are estimated on small UASs using either an inertial measurement unit (IMU) consisting of accelerometers and gyros or a set of “horizon-sensing” heat radiation measurement devices called thermopiles.

BOX 1. DIRECTION COSINE MATRICES

A DCM is an orthonormal matrix representing a rotation from one coordinate frame to another. A DCM written in this article as C_a^b represents a rotation from the a coordinate frame to the b coordinate frame. Thus, a vector in the a coordinate frame, \bar{v}^a , can be expressed in the b coordinate frame via premultiplication: $\bar{v}^b = C_a^b \bar{v}^a$. Two important characteristics of DCMs are that rotations can be cascaded (e.g., $C_a^c = C_b^c C_a^b$) and that the transpose of a DCM is equivalent to the reverse coordinate transformation (e.g., $C_b^a = [C_a^b]^T$).

A common coordinate transformation in this article is the rotation from the North–East–Down coordinate frame to the body x – y – z coordinate frame via the ordered Euler angles yaw (ψ), pitch (θ), and roll (ϕ). Specifically, the first rotation is ψ about z , which is denoted here as $C_z(\psi)$. The second rotation is $C_y(\theta)$, or θ about y . Finally, the third rotation is $C_x(\phi)$, or ϕ about x . These three single-axis rotations are written as:

$$C_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{bmatrix},$$

$$C_y(\theta) = \begin{bmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{bmatrix},$$

$$C_z(\psi) = \begin{bmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Thus, the transformation from the North–East–Down coordinate frame to the body x – y – z coordinate frame is written as a cascade of the three single-axis rotations above, which can be solved using standard matrix multiplication: $C_{\text{ned}}^b = C_x(\phi)C_y(\theta)C_z(\psi)$.

Yaw is not independently observable from either an IMU or a set of thermopiles. As a result, yaw is generally estimated using either a magnetometer measurement or by combining an estimate of the wind-induced crab angle with the GPS-sensed course angle. These common attitude estimation methods are discussed in more detail below.

Gyros

An IMU-based attitude estimator makes use of gyros and accelerometers to estimate the body attitude.^{9–12} A gyro, which is also referred to as a gyroscope or gyrocompass, measures the angular rate about a specific axis. An IMU-based attitude estimator will generally have a triad of orthogonal gyros to measure the angular rates about each body axis. The angular rates about the body axes x , y , and z are denoted ω_x , ω_y , and ω_z , respectively, and are combined as the vector $\bar{\omega}^b = [\omega_x \ \omega_y \ \omega_z]^T$. The relationship between body rates and Euler angle rates is given by:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin \phi \tan \theta & \cos \phi \tan \theta \\ 0 & \cos \theta & -\sin \phi \\ 0 & \sin \phi \sec \theta & \cos \phi \sec \theta \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \quad (8)$$

If perfect noise- and bias-free gyros were used and the initial body orientation were known, the UAS orientation could simply be determined by integrating the Euler angle rates as computed from the gyro measurements. In practice, however, every gyro has some amount of measurement error. Typically, a small UAS will use gyros based on MEMS technology, which have the advantages of being low power, low weight, and low cost. The accompanying disadvantages, of course, are relatively high levels of measurement noise and biases. Ignoring nonlinear error sources (like scale factor), the measurements from the triad of gyros can be represented as:

$$\bar{\omega}_{\text{gyro}}^b = \begin{bmatrix} \omega_{x,\text{gyro}} \\ \omega_{y,\text{gyro}} \\ \omega_{z,\text{gyro}} \end{bmatrix} = \bar{\omega}^b + \begin{bmatrix} b_{\omega_x} + \eta_{\omega_x} \\ b_{\omega_y} + \eta_{\omega_y} \\ b_{\omega_z} + \eta_{\omega_z} \end{bmatrix}, \quad (9)$$

where the b and η terms are biases and zero-mean noise, respectively. Directly integrating these gyro-measured body rates will yield significant attitude drift due to the low-frequency biases.

Accelerometers

Pitch and roll drift resulting from gyro biases can be effectively mitigated with the aid of accelerometers. Accelerometers measure what is referred to as specific force (\bar{f}^b), which is actually the total acceleration relative to free fall. Assuming that the IMU is near the UAS center of gravity and neglecting Earth rotation effects,

the specific force measured by an ideal triad of accelerometers aligned with body coordinates would measure

$$\bar{f}^b = \dot{\bar{V}}^b + \bar{\omega}^b \times \bar{V}^b - \bar{g}^b, \quad (10)$$

where \bar{V}^b is the inertial velocity vector in body coordinates, $\dot{\bar{V}}^b$ is the translational acceleration of the UAS with respect to body coordinates, $\bar{\omega}^b \times \bar{V}^b$ is the rotational (or centripetal) acceleration, and \bar{g}^b is the gravity vector in body coordinates, which acts in the downward direction. If a steady-state horizontal turn is assumed, then the velocity vector is inertially rotating (hence the centripetal acceleration) but the three velocity vector components relative to the body frame are constant, resulting in zero translational acceleration, $\dot{\bar{V}}^b$. Furthermore, the gravity vector has a magnitude of g and can be expressed in body coordinates using pitch (θ) and roll (ϕ). Thus, in a steady-state turn the specific force measured by ideal accelerometers would be:

$$\bar{f}_{\text{steady state}}^b = \bar{\omega}^b \times \bar{V}^b - g \begin{bmatrix} -\sin \theta \\ \sin \phi \cos \theta \\ \cos \phi \cos \theta \end{bmatrix}. \quad (11)$$

Using Eq. 11, it can be seen that steady-state estimates of roll and pitch can be derived given an accelerometer specific force measurement, a gyro body rate measurement ($\bar{\omega}_{\text{gyro}}^b \approx \bar{\omega}^b$), and an estimate of the inertial velocity in body coordinates, \bar{V}^b . Unfortunately, pure accelerometer-based estimates of pitch and roll are generally not sufficient for a fixed-wing UAS because they do not capture higher-frequency dynamics. Moreover, MEMS-based accelerometers can be rather noisy, further reducing the accuracy of the estimate. Here, the accelerometer specific force measurement is represented with biases and noise (nonlinear errors are ignored):

$$\bar{f}_{\text{accel}}^b = \begin{bmatrix} f_{x,\text{accel}} \\ f_{y,\text{accel}} \\ f_{z,\text{accel}} \end{bmatrix} = \bar{f}^b + \begin{bmatrix} b_{a_x} + \eta_{a_x} \\ b_{a_y} + \eta_{a_y} \\ b_{a_z} + \eta_{a_z} \end{bmatrix}. \quad (12)$$

Fortunately, the low-frequency attitude estimation capability of MEMS accelerometers complements the high-frequency attitude estimation capability of the MEMS gyros. A number of different algorithmic methods have been used successfully to combine these estimates, as will be discussed later.

Magnetometer

Accelerometers can adequately aid the gyro pitch and roll estimates, but they provide no observability of the body yaw angle. The most direct method for estimating the body yaw angle is to use a magnetometer. Both 2-D and 3-D MEMS-based magnetometers are available for small UASs, but 3-D magnetometers provide observabil-

ity of the local magnetic field pointing vector regardless of vehicle orientation. The magnetic declination (δ_{mag}), which is the bearing difference between true north and magnetic north (see Fig. 5), can be determined at the UAS position using available databases (see Ref. 13 for an online magnetic declination model). Assuming proper magnetometer calibration, a 3-D magnetometer provides a unit-vector measurement, $[M_x \ M_y \ M_z]^T$, of magnetic north in body coordinates, as in Eq. 13, where $\bar{\epsilon}_M^b$ represents measurement errors. Solving Eq. 13 for ψ , a magnetometer-based estimate of yaw (ψ_{mag}) can be formed as in Eq. 14. (Note that ψ_{mag} is a measure of the total yaw angle from *true* north, as opposed to *magnetic* north.)

$$\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = C_{\text{ned}}^b C_z(-\delta_{\text{mag}}) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \bar{\epsilon}_M^b. \quad (13)$$

$$\psi_{\text{mag}} = \tan^{-1}\left(\frac{-M_y \cos \phi + M_z \sin \phi}{M_x \cos \theta + (M_y \sin \phi + M_z \cos \phi) \sin \theta}\right) + \delta_{\text{mag}}. \quad (14)$$

In reality, the magnetic field measured aboard a UAS is heavily influenced by local perturbations generated from the avionics, particularly from an electric propulsion system. As a result, many small UAS developers will forego the use of a magnetometer completely. If a magnetometer is used, it must be carefully positioned far from onboard electromagnetic devices, which can be difficult on a smaller platform. Even then, a magnetometer calibration process is needed to estimate and remove any significant scale factors and offsets in the magnetometer unit vector measurement.¹⁴

Yaw Estimation

If the relatively direct measurement of body yaw angle from a magnetometer is not available, the UAS must use a less direct estimation means. As shown in the horizontal representation in Fig. 5 and provided in Eq. 15, the true body yaw angle can be expressed as a combination of inertial velocity vector course (χ), aerodynamic sideslip (β), and the wind-induced crab angle (χ_{crab}):

$$\psi = \chi - \chi_{\text{crab}} - \beta. \quad (15)$$

An inertial velocity vector course measurement is provided by the GPS receiver as χ_{GPS} (see Eq. 1b). Furthermore, aerodynamic sideslip for predominantly left-right axisymmetric UAS airframes can be assumed to be negligible ($\beta \approx 0$), although its magnitude is heavily dependent on the airframe design and lateral dynamics. Thus, recognizing the wind triangle vector relationship in Eq. 16 and assuming that estimates of the horizontal wind magnitude (V_w) and direction (χ_w) are available, then the crab angle can be approximated as in Eq. 17, and a resulting body yaw estimate, $\psi_{\text{windCorrected}}$, can be determined as in Eq. 18.⁸ Because of the inherent filtering

in both the GPS measurement and the wind estimate, $\psi_{\text{windCorrected}}$ will generally lag the true yaw angle ψ :

$$\bar{V}_{\text{air}}^{\text{ned}} = \bar{V}^{\text{ned}} - \bar{V}_w^{\text{ned}} = \begin{bmatrix} V_n - V_w \cos \chi_w \\ V_e - V_w \sin \chi_w \\ V_d - 0 \end{bmatrix}, \quad (16)$$

$$\chi_{\text{crab}} \approx \chi_{\text{GPS}} - \tan^{-1}\left(\frac{V_{e,\text{GPS}} - V_w \sin \chi_w}{V_{n,\text{GPS}} - V_w \cos \chi_w}\right), \quad (17)$$

$$\psi_{\text{windCorrected}} = \chi_{\text{GPS}} - \chi_{\text{crab}}. \quad (18)$$

As discussed above, the UAS yaw angle (ψ) can be directly measured if a magnetometer is available ($\psi \approx \psi_{\text{mag}}$), or it can be estimated if an axial pitot tube is available ($\psi \approx \psi_{\text{windCorrected}}$). As will be shown shortly, this yaw estimate can be used directly, or it can be used as a stable low-frequency complement to the biased angular rate gyro measurements. If neither a magnetometer nor a pitot tube is available on the UAS, yaw can be grossly estimated simply as the inertial course angle provided by GPS, $\psi \approx \chi_{\text{GPS}}$. Doing so, however, will degrade flight performance in significant winds and may limit the ability of an operator to geolocate a target as viewed by an onboard camera.

Thermopiles

IMU-based navigators on UASs using MEMS sensors have been proven reliable but at the cost of algorithmic complexity. A simpler, yet still effective, method used in some low-cost UASs is to directly measure pitch and roll angles using horizon-sensing thermopiles.^{4,5,15} A thermopile, as commonly used on UASs, is a heat-radiation measurement device sensing wavelengths from 5.5 to 14.5 μm . In essence, a single thermopile directionally measures the average near-field and far-field temperatures over a large field of view (e.g., 90°). Noting that the sky temperature is generally less than the ground temperature, two diametrically opposed thermopiles oriented vertically will measure the temperature difference between the sky and the ground. In the same manner, if a UAS has a thermopile oriented out toward each wing, as shown in Fig. 6, the wing pointed more toward the sky will yield a lower temperature. After scaling this temperature difference by the total ground-to-sky temperature difference, the result is roughly proportional to the roll angle, ϕ . In this manner, a UAS with two pairs of diametrically opposed thermopiles in the body x - y plane can estimate roll, ϕ_{IR} , and pitch, θ_{IR} , by measuring the scaled temperature differences between the thermopiles. A third pair of thermopiles along the body z axis can be used to provide the total ground-to-sky temperature difference for scaling. (Otherwise, a preflight calibration process is necessary.) Although not generally as accurate as an IMU-based attitude estimator, thermopiles are a

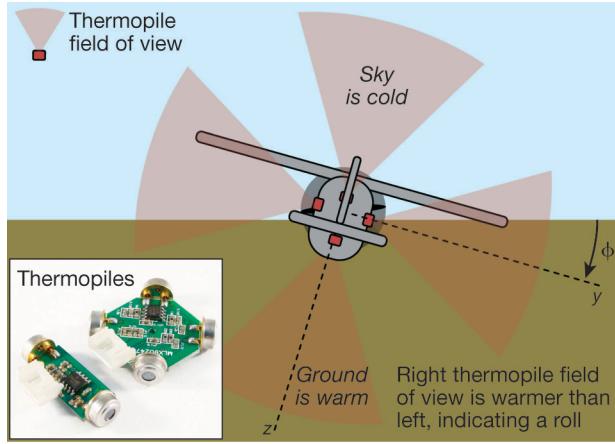


Figure 6. Heat-sensing thermopiles (inset) directly measure pitch and roll on a UAS by leveraging the temperature difference between the ground and the sky. (Image of thermopiles in inset is reprinted with permission from www.3DRobotics.com © 2011.)

simple and reliable means of directly measuring pitch and roll, provided the UAS has unobstructed vantages of the sky, ground, and horizon. For example, flying below the tree line might be problematic because the horizon is obstructed. Flying within clouds or fog, on the other hand, would be disastrous because every direction would have a near-uniform temperature. Despite those limitations, thermopiles have been very popular as a low-cost and simple attitude estimation sensor.

Common UAS Sensor Configurations

Multiple different configurations of the described sensors have been successfully used on UASs, but the three most common configurations are shown in Figs. 7, 8, and 11 and are described in the following paragraphs. All three configurations use GPS and pressure sensors for position and velocity. The differences between the three methods lie in the means of estimating the rotational states. Note that the low-cost sensors used on

small UASs are generally noisy and may require low-pass filtering, but that detail is not discussed here. For the interested reader, MATLAB code implementing the described IMU-based UAS state estimation methods can be found at <http://www.jhuapl.edu/ott/Technologies/Copyright/SuasCode.asp>.

GPS with Thermopile-Based Attitude Estimation

By far the simplest of the described UAS state estimation methods algorithmically, the configuration in Fig. 7 uses thermopiles to directly estimate the vehicle pitch ($\hat{\theta} = \theta_{IR}$) and roll ($\hat{\phi} = \phi_{IR}$) angles. If a magnetometer is available and reliable, it can be used to estimate body yaw ($\hat{\psi} = \psi_{mag}$). Otherwise, body yaw is estimated using either the raw GPS course ($\hat{\psi} = \chi_{GPS}$) or by correcting the GPS course angle with the estimated wind-induced crab angle ($\hat{\psi} = \psi_{windCorrected}$). An autopilot using thermopile-based attitude estimation will typically rely directly on position and velocity measurements from the GPS sensor, without further filtering. A ramification of using thermopiles is that the UAS must fly above large horizon obstructions and cannot fly in fog or clouds. (Most UASs should avoid high-moisture environments anyway.) Although relatively simple to implement, this sensor configuration has proven to be fairly robust¹⁶ in valid operating environments and is by definition drift-free. Because of its simplicity of design and low sensor cost, this configuration has been used extensively, though certainly not exclusively, within the do-it-yourself and open-source UAS communities.^{4,5}

GPS with IMU-Based Attitude and Heading Reference System

In the UAS sensor configuration shown in Fig. 8, the attitude angles are estimated with the aid of an IMU, and translational states (position and velocity) are acquired directly from GPS and pressure sensors. The attitude estimation component of the configuration in Fig. 8 is often a stand-alone component called an attitude and heading reference system (AHRS).

An AHRS uses an IMU consisting of a triad each of gyros and accelerometers, along with velocity and yaw references, to estimate the body yaw, pitch, and roll angles. In essence, an AHRS uses stable low-bandwidth attitude observations to estimate biases in the high-bandwidth angular rate gyros, then integrates the de-biased gyro measurements to form a complete attitude estimate.

The low-bandwidth yaw observation, or reference, can

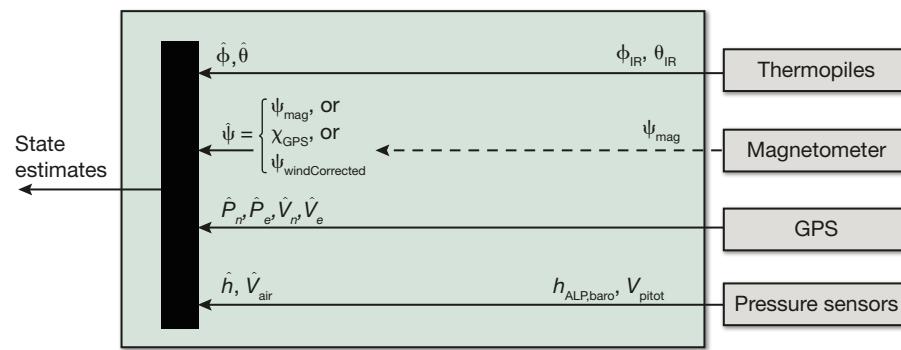


Figure 7. State estimation: GPS with thermopile-based attitude estimation. All the requisite position and attitude estimates are acquired directly from the available onboard sensors: GPS, pressure sensors, thermopiles, and possibly a magnetometer. If a magnetometer is not used, the body yaw angle can be estimated from the GPS course or a wind-corrected GPS course.

be a direct measurement of yaw from a magnetometer ($\psi_{\text{ref}} = \psi_{\text{mag}}$), a gross estimate using the GPS course ($\psi_{\text{ref}} = \chi_{\text{GPS}}$), or a crab-angle-corrected GPS course ($\psi_{\text{ref}} = \psi_{\text{windCorrected}}$). The low-bandwidth pitch and roll observations used for bias estimation are provided by using the accelerometers to estimate the IMU orientation relative to gravity. Note that the accelerometers actually measure a combination of gravity, translational acceleration, and centripetal (or rotational) acceleration. For an AHRS, translational acceleration is typically assumed to be negligible and centripetal acceleration is accounted for using the gyro measurements and a velocity reference, $\bar{\mathbf{V}}^b_{\text{ref}}$. The velocity reference is oftentimes estimated grossly from GPS ($\bar{\mathbf{V}}^b_{\text{ref}} = [V_{\text{H,GPS}} \ 0 \ 0]^T$) or the sensed airspeed ($\bar{\mathbf{V}}^b_{\text{ref}} = [V_{\text{pitot}} \ 0 \ 0]^T$).

Algorithmically, a UAS or AHRS developer has many options for fusing the sensors shown in Fig. 8 to estimate gyro biases and generate attitude state estimates ($\hat{\phi}, \hat{\theta}, \hat{\psi}$). A traditional approach is to use an extended Kalman filter (EKF) to estimate a set of time-evolving states $\hat{\mathbf{x}}$ based on multiple observations, $\bar{\mathbf{z}}$, of those states over time. To implement an EKF, it is imperative to mathematically model both the dynamics of $\hat{\mathbf{x}}$ and the measurements $\bar{\mathbf{z}}$, as in Eq. 19:

$$\frac{d}{dt} \hat{\mathbf{x}} = f(\hat{\mathbf{x}}, \dots) + \bar{\mathbf{w}}(t), \quad (19a)$$

$$\bar{\mathbf{z}} = h(\hat{\mathbf{x}}, \dots) + \bar{\mathbf{r}}(t). \quad (19b)$$

The functions $f(\hat{\mathbf{x}}, \dots)$ and $h(\hat{\mathbf{x}}, \dots)$ are the nonlin-

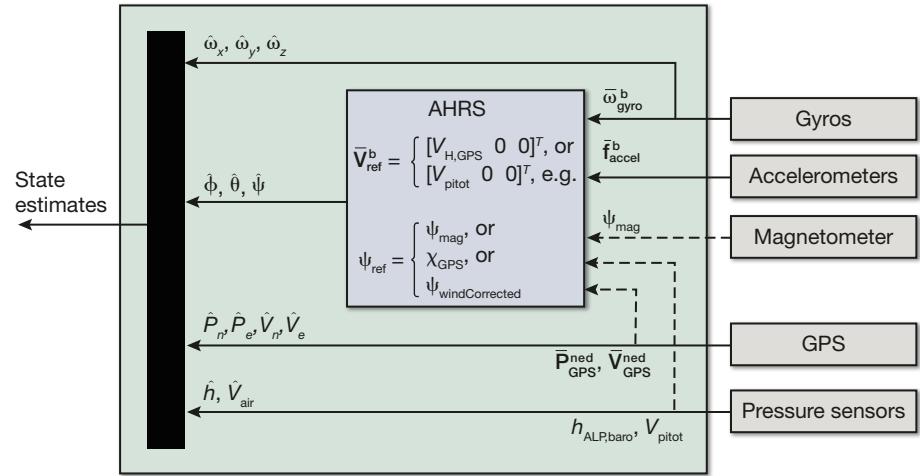


Figure 8. State estimation: GPS with an IMU-based AHRS. Position and velocity estimates are acquired from GPS and pressure sensors. The orientation is acquired by fusing gyro and accelerometer measurements with yaw and velocity references, generally via an EKF or a feedback filter.

ear dynamics and measurement models, respectively. $\bar{\mathbf{w}}(t)$ and $\bar{\mathbf{r}}(t)$ are the non-deterministic influences, such as unmodeled state propagation effects and sensor measurement noise, that affect the system state dynamics and measurements, respectively. In general, an EKF can successfully track the states $\hat{\mathbf{x}}$, within a bounded error, if the measurement vector $\bar{\mathbf{z}}$ provides sufficient observability of the states $\hat{\mathbf{x}}$; the nonlinear relationships $f(\hat{\mathbf{x}}, \dots)$ and $h(\hat{\mathbf{x}}, \dots)$ are relatively linear about the states $\hat{\mathbf{x}}$; and any unmodeled effects can be sufficiently captured by Gaussian processes $\bar{\mathbf{w}}(t)$ and $\bar{\mathbf{r}}(t)$. To fully implement an EKF, a designer must quantitatively define the characteristics of $\bar{\mathbf{w}}(t)$ and $\bar{\mathbf{r}}(t)$. For our purposes, it is sufficient merely to denote the nonlinear mathematical relationships defining the state dynamics, $f(\hat{\mathbf{x}}, \dots)$, and the measurement, $h(\hat{\mathbf{x}}, \dots)$. A more thorough description of the EKF mechanics can be found in Ref. 17.

An example implementation of the AHRS algorithms in Fig. 8 is to use an EKF to simultaneously estimate the UAS attitude and the gyro biases. Thus, using the state vector $\hat{\mathbf{x}} = [\hat{\phi} \ \hat{\theta} \ \hat{\psi} \ \hat{b}_{\omega_x} \ \hat{b}_{\omega_y} \ \hat{b}_{\omega_z}]^T$ and Eqs. 8, 9, and 11, a viable set of the nonlinear relationships necessary to implement an AHRS EKF is:

$$\frac{d}{dt} \hat{\mathbf{x}} = \begin{bmatrix} \dot{\hat{\phi}} \\ \dot{\hat{\theta}} \\ \dot{\hat{\psi}} \\ \dot{\hat{b}_{\omega_x}} \\ \dot{\hat{b}_{\omega_y}} \\ \dot{\hat{b}_{\omega_z}} \end{bmatrix} = \begin{bmatrix} 1 & \sin \hat{\phi} \tan \hat{\theta} & \cos \hat{\phi} \tan \hat{\theta} & [\omega_{x,\text{gyro}} - \hat{b}_{\omega_x}] \\ 0 & \cos \hat{\phi} & -\sin \hat{\phi} & [\omega_{y,\text{gyro}} - \hat{b}_{\omega_y}] \\ 0 & \sin \hat{\phi} \sec \hat{\theta} & \cos \hat{\phi} \sec \hat{\theta} & [\omega_{z,\text{gyro}} - \hat{b}_{\omega_z}] \\ \hline 0 & 0 & 0 & 0 \end{bmatrix}, \quad (20)$$

$$\bar{\mathbf{z}} = \begin{bmatrix} f_{x,\text{accel}} \\ f_{y,\text{accel}} \\ f_{z,\text{accel}} \\ \hline \psi_{\text{ref}} \end{bmatrix} = \begin{bmatrix} [\omega_{x,\text{gyro}} - \hat{b}_{\omega_x}] \\ [\omega_{y,\text{gyro}} - \hat{b}_{\omega_y}] \\ [\omega_{z,\text{gyro}} - \hat{b}_{\omega_z}] \\ \hline \psi \end{bmatrix} \times \bar{\mathbf{V}}^b_{\text{ref}} - g \begin{bmatrix} -\sin \hat{\theta} \\ \sin \hat{\phi} \cos \hat{\theta} \\ \cos \hat{\phi} \cos \hat{\theta} \end{bmatrix}. \quad (21)$$

An experienced designer can use the relationships in Eq. 20 and Eq. 21, along with expected measurement noise and vehicle dynamic characteristics, to implement

an EKF that successfully estimates the vehicle attitude and gyro biases. Some advantages of using an EKF are that a designer can use available information, like expected measurement noise levels, to provide a more robust solution, and that the EKF inherently provides a measure of state-estimation error, provided the estimate converges. Some disadvantages of using an EKF are that it is (relatively) computationally complex; it requires some *a priori* knowledge of sensor error characteristics; and it can appear algorithmically daunting for an inexperienced designer. Note that Euler angles were used in this EKF representation because they are intuitively appealing. A similar EKF derived with quaternions would be more robust (see MATLAB code). In particular, it would avoid the previously described Euler angle singularity problem as pitch approaches 90°. Note also that if a 3-D magnetometer is used, the measurement expressed in Eq. 21 could alternatively use the raw magnetometer unit-vector measurement as in Eq. 13.

A computationally simpler alternative to an EKF for the AHRS algorithms in Fig. 8 is to use a feedback controller to correct for gyro drift.^{4,10,11} In the AHRS feedback controller shown in Fig. 9, a compensator, typically based on proportional–integral control, is used to estimate gyro biases based on a body-frame attitude error vector, \bar{e}^b . \bar{e}^b is derived by noting that if the output attitude estimate were correct and other error sources were negligible, then the estimated North and

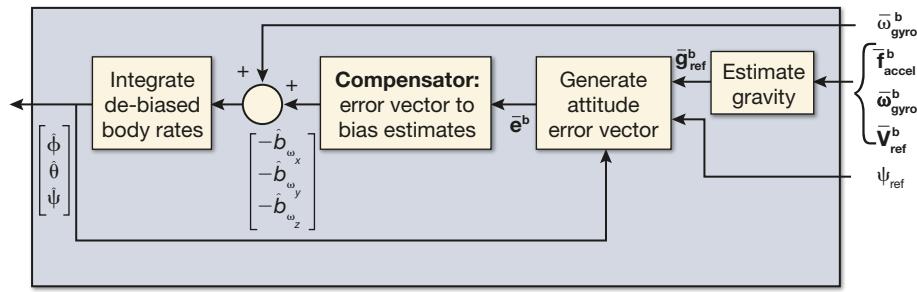


Figure 9. An AHRS feedback controller estimates the body orientation by fusing high-bandwidth gyro angular rate measurements with low-bandwidth attitude references. The yaw reference comes from a magnetometer or a GPS course-based estimate. Pitch and roll references are acquired from an estimate of the gravity vector via centripetally corrected accelerometer measurements. Essentially, this feedback controller uses a compensator to estimate the gyro biases by regulating the error between the estimated orientation and the orientation expressed by the low-bandwidth attitude references.

Down directions would align with those observed from a yaw reference (ψ_{ref}) and a body-frame gravity vector reference (\bar{g}_{ref}^b). The gravity vector reference is derived from the centripetally corrected accelerometer measurements:

$$\bar{g}_{\text{ref}}^b = \bar{\omega}_{\text{gyro}}^b \times \bar{V}_{\text{ref}}^b - \bar{f}_{\text{accel}}^b \quad (22)$$

Using vector geometry and Fig. 10, the error \bar{e}^b can be expressed as in Eq. 23d, which is formed using the current attitude estimates ($\hat{\phi}$, $\hat{\theta}$, $\hat{\psi}$) and the gravity and yaw references.

$$\hat{C}_{\text{ned}}^b = C_x(\hat{\phi})C_y(\hat{\theta})C_z(\hat{\psi}) \quad (23a)$$

$$\bar{e}_{\psi}^b = \left(C_x(\hat{\phi})C_y(\hat{\theta})C_z(\psi_{\text{ref}}) \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right) \times \left(\hat{C}_{\text{ned}}^b \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right) \quad (23b)$$

$$\bar{e}_g^b = \left(\frac{\bar{g}_{\text{ref}}^b}{|\bar{g}_{\text{ref}}^b|} \right) \times \left(\hat{C}_{\text{ned}}^b \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right) \quad (23c)$$

$$\bar{e}^b = \bar{e}_{\psi}^b + \bar{e}_g^b \quad (23d)$$

In Eq. 23, \bar{e}_{ψ}^b is the rotation vector expressed in body coordinates between the observed North direction defined by ψ_{ref} and the AHRS-estimated North direction. (Note that the cross product, \times , between two vectors yields a vector orthogonal to both with a magnitude proportional to the sine of the angle between them.) Similarly, \bar{e}_g^b is the rotation vector in body coordinates between the centripetally corrected gravity direction (Down) estimated from the accelerometers and the AHRS-estimated Down direction. As a result, the combined error vector \bar{e}^b expresses the angular error and rotation axis between the observed (or reference) North–East–Down coordinate frame and the AHRS-estimated North–East–Down coordinate frame. This feedback error vector is filtered via a compensator to generate the estimated gyro biases, $[\hat{b}_{\omega_x} \hat{b}_{\omega_y} \hat{b}_{\omega_z}]^T$. Using these estimated biases, the de-biased gyro measurements are converted to Euler angle rates, as in Eq. 8, and

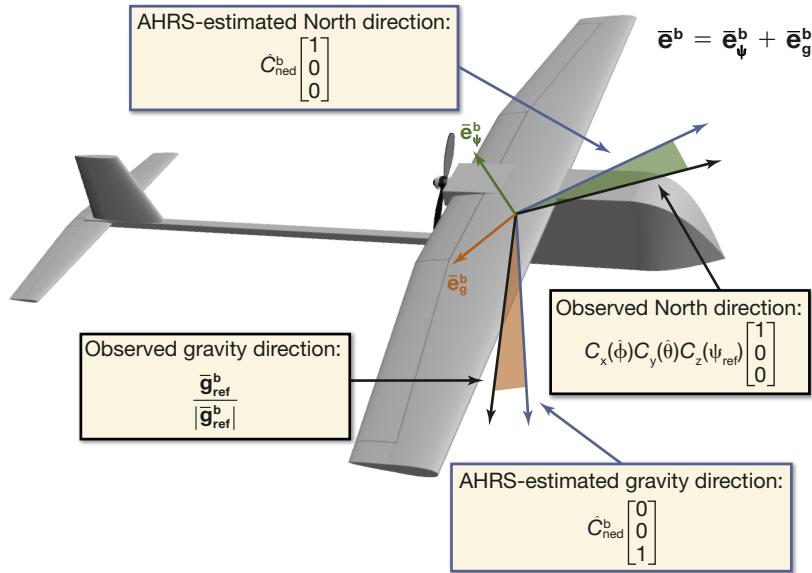


Figure 10. In the AHRS feedback controller, the body-frame attitude error vector, $\bar{\mathbf{e}}^b$, is formed from the summation of the gravity error vector, $\bar{\mathbf{e}}_g^b$, and the yaw error vector, $\bar{\mathbf{e}}_\psi^b$. Note that each error vector is perpendicular to both the observed and estimated vector directions. The shaded regions represent the angular difference between the observed and estimated vectors.

integrated to form the attitude estimates. As with the EKF description, this feedback controller AHRS algorithm was described using Euler angles merely because of their intuitive appeal. The attitude states could be similarly integrated as either quaternions or a DCM for increased robustness at all orientations (see MATLAB code). As an aside, the feedback controller in Fig. 9 is actually a classical complementary filter¹¹ that combines the low-bandwidth attitudes derived from the reference inputs with the high-bandwidth attitudes integrated from the gyros. If a proportional–integral controller is used, the integral gain determines the crossover frequency, and the proportional gain trades the damping and high-frequency noise suppression. In practice, separate proportional–integral gains for $\bar{\mathbf{e}}_\psi^b$ and $\bar{\mathbf{e}}_g^b$ compensators might be used to account for the differing bandwidth characteristics of the magnetometer and accelerometers.

Full GPS Inertial Navigation System

A more computationally complex option for UAS state estimation is to combine the IMU and GPS measurements (along with other available measurements, if applicable) into a complete inertial navigation system (INS)^{9,18,19} as shown in Fig. 11, where ψ_{ref} is acquired in the same manner as with an AHRS. The primary distinction between a GPS/INS and a GPS/AHRS (Fig. 8) is that the inherently low-pass-filtered translational state measurements from GPS are augmented with the higher-bandwidth acceleration and body rate measurements from the IMU. Doing so also provides observability into acceleration biases and enhanced dead-reckoning capabilities, where translational states are predicted during short GPS dropouts. A GPS/INS on a UAS is typically accomplished using an EKF. A relatively simplified example realization of a GPS/INS EKF for a UAS is provided in Eqs. 24 and 25, where the realization is specified by the nonlinear state dynamics and measurement models. The states used for the described realization are attitude, position, velocity, gyro biases, and accelerometer biases. The mathematical derivations of Eqs. 24 and 25 can be inferred from Eqs. 8, 9, 10, and 12:

$$\frac{d\hat{\mathbf{x}}}{dt} = \begin{bmatrix} \dot{\hat{\phi}} \\ \dot{\hat{\theta}} \\ \dot{\hat{\psi}} \\ \dot{\hat{P}_n} \\ \dot{\hat{P}_e} \\ \dot{\hat{h}_{MSL}} \\ \dot{\hat{V}_n} \\ \dot{\hat{V}_e} \\ \dot{\hat{V}_d} \\ \dot{\hat{b}_{\omega_x}} \\ \dot{\hat{b}_{\omega_y}} \\ \dot{\hat{b}_{\omega_z}} \\ \dot{\hat{b}_{a_x}} \\ \dot{\hat{b}_{a_y}} \\ \dot{\hat{b}_{a_z}} \end{bmatrix} = \begin{bmatrix} 1 & \sin \hat{\phi} \tan \hat{\theta} & \cos \hat{\phi} \tan \hat{\theta} \\ 0 & \cos \hat{\phi} & -\sin \hat{\phi} \\ 0 & \sin \hat{\phi} \sec \hat{\theta} & \cos \hat{\phi} \sec \hat{\theta} \end{bmatrix} \begin{bmatrix} \omega_{x,gyro} - \hat{b}_{\omega_x} \\ \omega_{y,gyro} - \hat{b}_{\omega_y} \\ \omega_{z,gyro} - \hat{b}_{\omega_z} \end{bmatrix} \\ - \begin{bmatrix} \hat{V}_n \\ \hat{V}_e \\ -\hat{V}_d \end{bmatrix} \\ \hat{C}_b^{ned} \begin{bmatrix} f_{x,accel} - \hat{b}_{a_x} \\ f_{y,accel} - \hat{b}_{a_y} \\ f_{z,accel} - \hat{b}_{a_z} \end{bmatrix}, \quad (24)$$

$$\bar{\mathbf{z}} = \begin{bmatrix} \Psi_{ref} \\ P_{n,GPS} \\ P_{e,GPS} \\ h_{MSL,GPS} \\ V_{n,GPS} \\ V_{e,GPS} \\ V_{d,GPS} \end{bmatrix} = \begin{bmatrix} \hat{\psi} \\ \hat{P}_n \\ \hat{P}_e \\ \hat{h}_{MSL} \\ \hat{V}_n \\ \hat{V}_e \\ \hat{V}_d \end{bmatrix}. \quad (25)$$

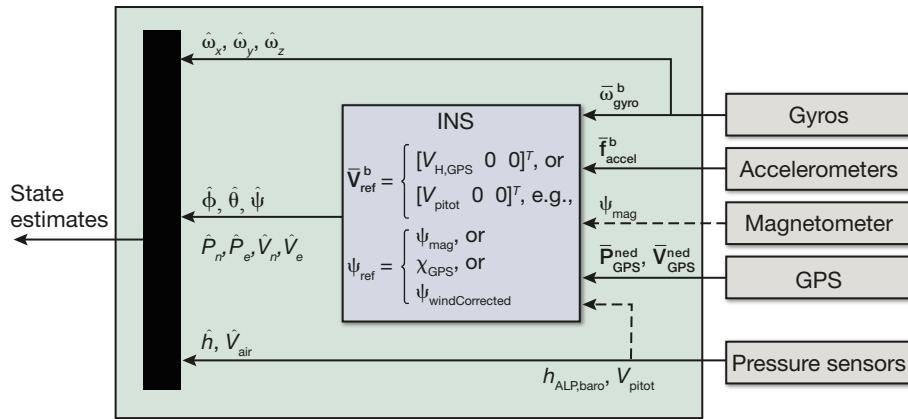


Figure 11. State estimation: full GPS/INS. An EKF is used to fuse gyros, accelerometers, and possibly a magnetometer with GPS measurements to estimate both rotational and translational states. Note that the pressure sensors are still used directly for height and airspeed.

Note that \hat{C}_b^{ned} in the state dynamics model is a function of the estimated attitude states. Note also that the described EKF provides an altitude estimate, \hat{h}_{MSL} , which is essentially a smoothed GPS altitude. Most small UASs will use the barometric altitude measurement in lieu of a GPS altitude measurement for flight control, but the state is necessary in this EKF in order to gain additional pitch and roll observability through the altitude rate dynamics. The described GPS/INS EKF implementation is merely an example. Both simpler and more complex implementations can be realized, including a more robust quaternion-based filter (see MATLAB code). Finally, as with the AHRS EKF, if a 3-D magnetometer is used, then the measurement expressed in Eq. 25 could alternatively use the raw magnetometer unit-vector measurement (see Eq. 13).

All of the sensor configurations shown in Figs. 7, 8, and 11 used some combination of GPS, IMU, magnetometer, thermopile, and pressure sensor measurements. These are by far the most common navigation sensors used on UASs today. They are by no means the only sensors being used, however. UAS technology developers, whether professional, academic, or amateur, are constantly devising new means of using existing sensor technologies for UAS applications. Detailing these novel sensing technologies is beyond the scope of this article, but a list of such sensors that have been demonstrated on small UASs or are under active research includes vision sensors, optic

flow sensors, scanning laser rangefinders, active acoustic arrays, and radio frequency receivers for positioning using non-GPS signals. The primary motivations behind the development of these other sensor technologies are navigating in cluttered and/or GPS-denied environments and autonomous sensing-and-avoidance of other air vehicles and obstacles.

FLIGHT CONTROL

Referring back to Fig. 3, the UAS flight control algorithms compare flight commands with state estimates and generate flight control channel commands. The four flight control channels typically discussed on an air vehicle are throttle, elevator, aileron, and rudder, though not every air vehicle will use all channels. The throttle channel (T) modulates the thrust (via propeller speed for example) and hence affects translational force. The elevator (δ_E), aileron (δ_A), and rudder (δ_R) channels, as shown in Fig. 4, affect pitching, rolling, and yawing rotational moments, respectively. The specific combination of motor and moveable surface commands that enact these channel commands depends on the air vehicle.

Most fixed-wing small UAS airframes fly via coordinated turns, wherein the air vehicle turns by rolling. As such, many fixed-wing UASs stabilize and steer purely using the elevator and aileron channels. For example, the flying wing airframe in Fig. 1 has only two moveable aerodynamic surfaces, one on each wing. Elevator commands (δ_E) are achieved by moving both surfaces up or down in conjunction. Aileron commands (δ_A) are achieved by moving the two surfaces in opposite directions, one up and one down. Control of the rudder channel (δ_R) is not explicitly necessary on this airframe because the fixed vertical stabilizers at the wing tips induce a stabilizing yawing moment. Other fixed-wing airframes, such as the Raven in Fig. 1, use a horizontal surface for elevator control and a vertical rudder surface for horizontal steering. This rudder motion induces a yawing moment, which tends to tip the wing in the turn direction, thus achieving coordinated turns and roll control via a rudder. As described, roll control can be achieved using either the aileron channel or the rudder channel, depending on the aircraft design.

The flight control algorithms are typically decoupled into lateral control algorithms for achieving route or steering commands and longitudinal control algorithms for achiev-

ing altitude and speed commands. The vast majority of small UASs use classical control methods wherein a flight command (e.g., a desired airspeed) is achieved by automatically adjusting a control signal (e.g., throttle) to minimize the feedback error between the flight command and the achieved value. The most common method is proportional–integral–derivative (PID) control, where the control signal is formed as a linear combination of the feedback error, the integral of that error, and the derivative of that error. Oftentimes, PID control might be combined with a feed-forward component and an expected “trim” value. As an example, Fig. 12 shows a possible implementation of controlling airspeed using throttle. Desired response characteristics, such as a suitable response time and minimal airspeed oscillations, can be achieved by adjusting (or “tuning”) the scalar PID gains K_P , K_I , and K_D . To speed up the response to airspeed command changes, the feedback PID control in the example is further augmented with a “trim” or steady-state throttle value and a feed-forward gain, K_{FF} , directly from the airspeed. The resulting control signal is then limited to achievable throttle values. Note that K_P , K_I , K_D , and K_{FF} are all scalar gains, though some autopilots might use gain scheduling where the gains change in different flight modes or regimes. Similar methods are often used in all of the lateral and longitudinal control components described in the following sections.

Lateral Flight Control

Lateral trajectory commands are provided to the autopilot as desired trajectory segments such as a combination of waypoint-defined paths to follow and desired loiter circles. As shown in Fig. 13, a UAS will typically either fly directly toward each waypoint or use control logic to minimize the crosstrack position and heading error along the waypoint-defined trajectory. Figure 14

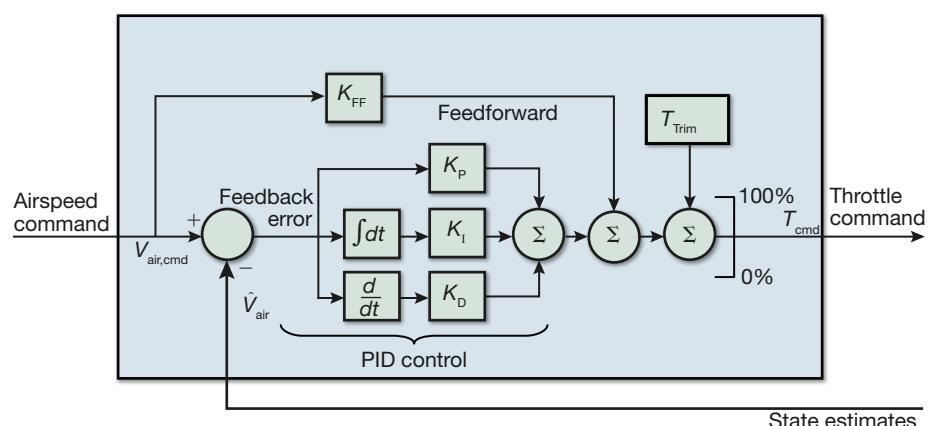


Figure 12. Airspeed to throttle control (example). Classical feedback control methods are often used in small UAS flight control. As an example, this figure shows an augmented PID compensator controlling throttle to achieve a desired airspeed.

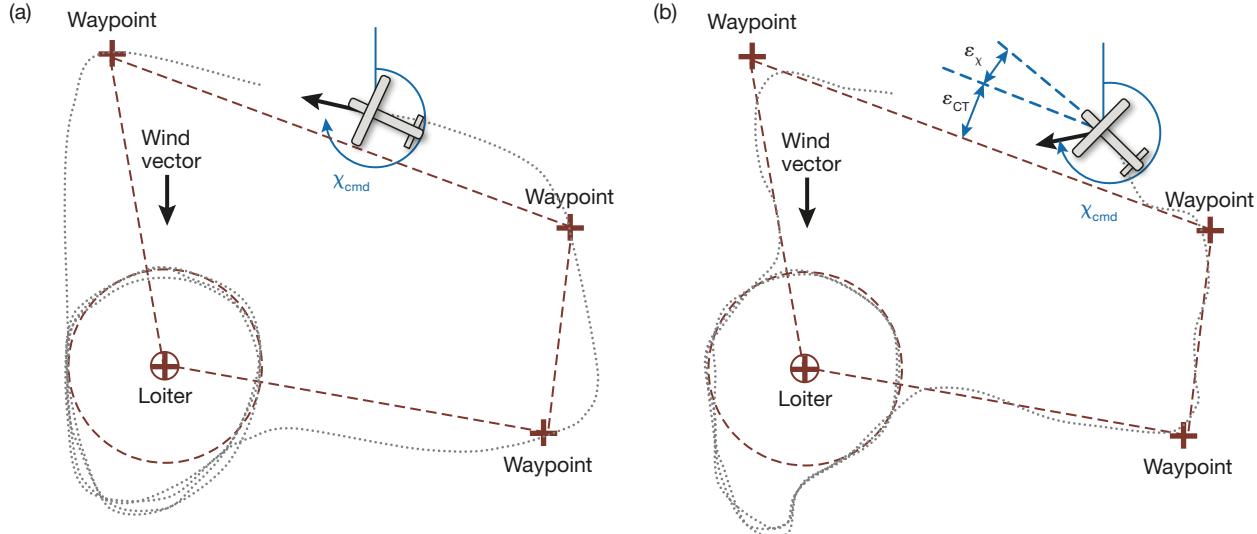


Figure 13. A lateral trajectory for a UAS is generally specified by a sequence of waypoints and possibly loiter circles, where the UAS might be commanded to either fly directly toward each waypoint (a) or fly along the trajectory segment between each waypoint (b). When the UAS is commanded to fly directly toward each waypoint, the course command (χ_{cmd}) is simply the bearing to the next waypoint. When the UAS is commanded to fly along the trajectory, the course command is a function of crosstrack error (ϵ_{CT}) and heading error (ϵ_x). Analogous methods can be used to achieve a desired flight around a loiter circle. Regardless of the method, small UAS flight trajectories can be heavily influenced by winds.

shows a block diagram representing typical lateral control algorithms in a small UAS. The horizontal trajectory control algorithms convert trajectory errors (e.g., bearing-to-waypoint and crosstrack errors) into a heading command (χ_{cmd}). This heading command is in turn compared with the estimated UAS heading (\hat{x}) to generate a roll command (ϕ_{cmd}). The roll command feeds the horizontal stability control algorithms, which generate aileron ($\delta_{A,cmd}$) or rudder ($\delta_{R,cmd}$) commands using roll ($\hat{\phi}$) and possibly roll-rate ($\dot{\omega}_x$) feedback. The described lateral control structure provides a method for autonomous waypoint and loiter navigation. Given a sequence of waypoint and loiter commands, the UAS will automatically control its flight to achieve the desired route.

Most autopilots will also have a remote piloting mode during which a user can directly command either heading (χ_{cmd}) or roll (ϕ_{cmd}) from a remote control device at the ground control station. In this mode, altitude and airspeed are automatically maintained with the closed-loop longitudinal algorithms. Remotely piloting a UAS is often the most effective manner of achieving a certain ISR mission objective, at the cost of requiring continuous operator attention and uninterrupted communications.

Longitudinal Flight Control

In longitudinal control, the combination of elevator and throttle are used to control altitude and airspeed. In

a simple autopilot, the control algorithms for altitude and airspeed may be decoupled, though they are certainly coupled in the airframe response. Specifically, a change in altitude at a fixed throttle command will induce a change in airspeed because of the conservation of energy. Similarly, a change in airspeed at a fixed pitch angle will induce a change in height because the amount of lift produced by the wings changes with airspeed. Figures 15 and 16 show two typical implementations of simple longitudinal flight control

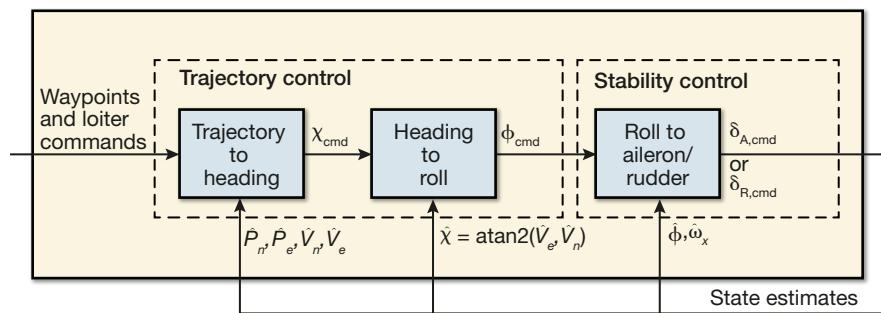


Figure 14. Lateral flight control uses state estimate feedbacks to convert waypoint and loiter commands into aileron or rudder commands. The inner-loop stability control portion achieves the desired roll angle using the control surface commands. The outer trajectory control portion derives a course command and subsequently a roll command to achieve the desired trajectory. (Note: this figure assumes that the UAS flies via coordinated turns, which is generally the case for a fixed-wing UAS.)

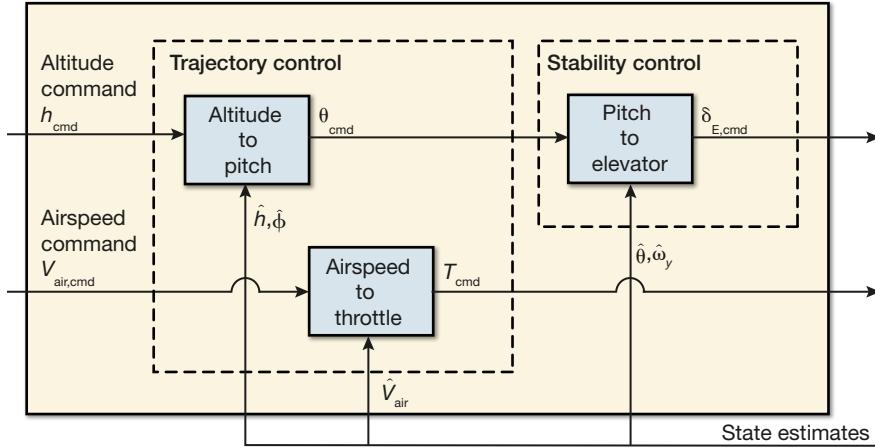


Figure 15. Longitudinal flight control method where pitch is used to control altitude and throttle controls airspeed. The inner-loop stability control portion achieves the desired pitch angle using the elevator command. Some autopilots will use this method during altitude hold modes.

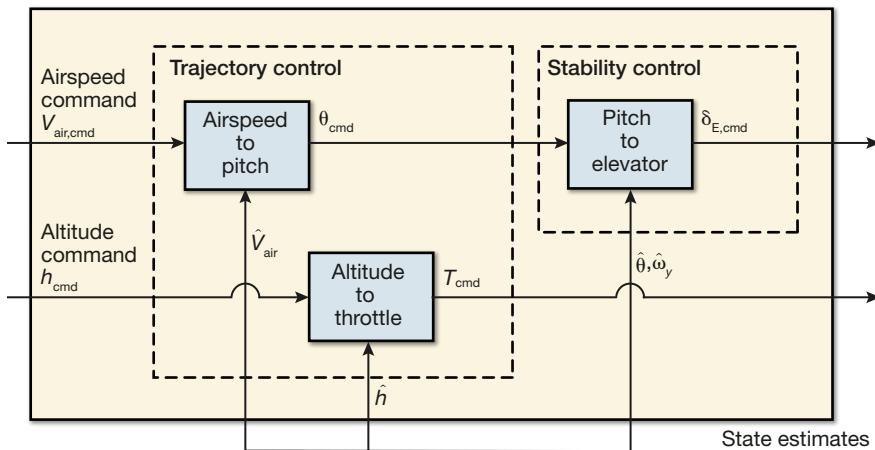


Figure 16. Longitudinal flight control method where pitch is used to control airspeed and throttle controls altitude. The inner-loop stability control portion achieves the desired pitch angle using the elevator command. Some autopilots will use this method during climbs and descents.

for a small UAS. In Fig. 15, altitude error is used to generate a pitch command (θ_{cmd}), which in turn generates an elevator command using appropriate state feedbacks. (The pitch command computation can be aided with the roll estimate to account for the increased pitch command needed during coordinated turns.) Airspeed error is regulated directly with the throttle command. In Fig. 16, it is airspeed error that generates the pitch command, which is in turn used to generate an elevator command via appropriate feedbacks. Altitude error is then regulated using the throttle command. As shown in Fig. 17, a small fixed-wing UAS might use either one of these longitudinal control architectures or it might switch between the two based on flight mode. For example, using throttle to control altitude error is generally more energy efficient over sustained climbs and descents, so an autopilot might use the longitudinal control architecture in Fig. 16 during

a climb to a higher altitude command. In contrast, the elevator channel is generally faster than throttle in regulating small altitude errors, so the longitudinal control architecture in Fig. 15 might be used during a sustained altitude hold. Both of the described longitudinal control architectures used pitch command (θ_{cmd}) for inner-loop longitudinal stability control. Some small UAS autopilots will use an altitude rate command (\dot{h}_{cmd}) and its respective state estimate ($\frac{d}{dt}\hat{h}$) for inner-loop longitudinal stability control in lieu of pitch. Although it is not shown in the figures, the UAS might also override pitch and throttle commands during certain automatic flight phases such as takeoff and landing.

Flight Control Tuning

Flight control algorithms for modern missiles and large aircraft are typically developed based on aerodynamic modeling from either a wind tunnel or a computational fluid dynamics analysis. Using those aerodynamic models, control engineers derive the flight control algorithms and parameters to meet desired analytical flight response and stability characteristics. Using a similar procedure to

derive flight control algorithms on a small UAS is generally not feasible because of the excessive cost of generating an aerodynamic model. In contrast, a small UAS developer will typically manually tune the control parameters to achieve the desired flight response. As mentioned above, small UAS flight control algorithms are generally based on classical control methods like PID, feed-forward terms, and trim offsets, where the response is dictated by a set of autopilot gains and parameters. A common gain-tuning procedure is to have an experienced R/C pilot fly the small UAS manually to achieve smooth, level flight and then successively enable and “tune” the gains for each lateral and longitudinal control loop.

Takeoff

Small fixed-wing UAS autonomous launch procedures obviously vary based on the airframe, vehicle

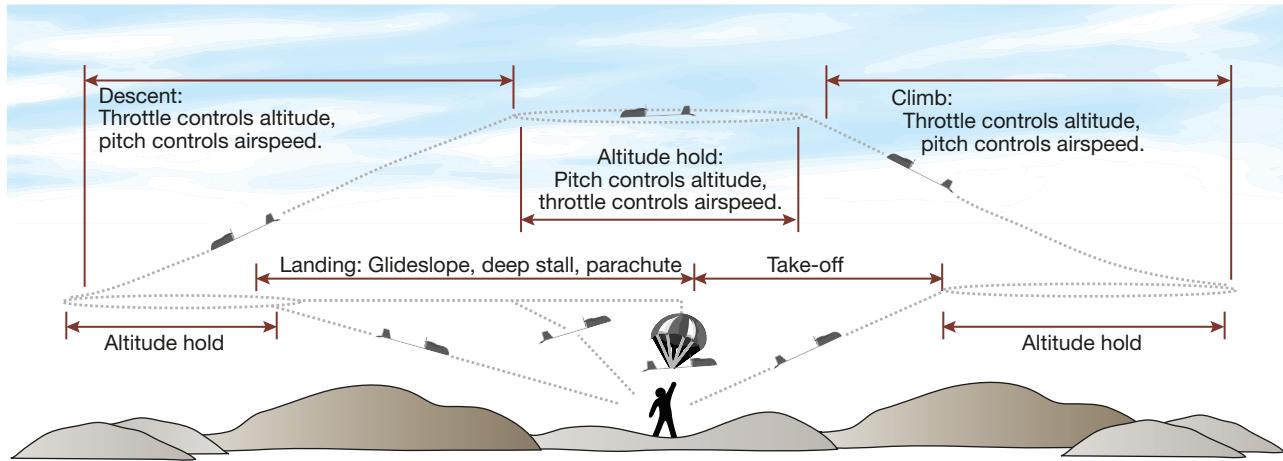


Figure 17. Longitudinal flight control encompasses achieving both a desired altitude and airspeed, and some autopilots switch longitudinal control methods during different phases of flight. For example, during climbs and descents an autopilot may use throttle to control altitude and pitch to control airspeed. Similarly, pitch might control altitude and throttle might control airspeed during altitude holds. The figure also represents takeoff and a variety of landing modes.

weight, and required flight speed. Many small airframes with a sufficiently slow stall speed can be hand-launched by a single operator. Other airframes might require a takeoff airstrip, a bungee launcher, or a pneumatic launch device in order to achieve the requisite airspeed. Once the UAS is airborne with an airspeed above the stall speed of the aircraft, the takeoff flight control algorithms are generally some variation of regulating roll, controlling pitch to a desired ascent pitch profile, and setting throttle to a fixed high value (e.g., maximum). Provided sufficient thrust and pitch, the UAS will then ascend until a threshold altitude is reached. Upon reaching the threshold altitude, the UAS transitions to the lateral and longitudinal flight logic described above. Taking off into the prevailing wind is always desirable and is sometimes quite necessary in order to exceed the stall speed of the aircraft.

Landing

As shown in Fig. 17, automatic landing for a small UAS can be achieved via a parachute, a controlled glide slope, or a deep stall. A parachute landing can generally provide safe and reliable recoveries, assuming proper parachute packing by the operator, but at the expense of horizontal landing accuracy in higher winds and from higher altitudes. A controlled glide slope can be achieved by regulating roll, controlling airspeed via pitch, and using throttle to achieve a desired altitude profile. Using an altitude estimate, operator input, or a height-above-ground sensor to flare (pitch up) immediately before ground impact is desirable. Many small UASs do not have a landing gear (wheels) and thus might glide to a belly landing, preferably into the wind and on a soft surface such as grass. Other UASs perform a controlled glide into a capture mechanism such as a

net. In a deep-stall landing,²⁰ the throttle is cut and the elevator is quickly deflected upward and maintained at a high angle (e.g., 45°), thus causing a suitable aircraft to achieve a steady angle-of-attack much higher than the stall angle-of-attack. (Angle-of-attack is the angle between the airframe and the wind-relative velocity vector, and it largely dictates the resulting aerodynamic lift and drag forces.) Although the flow about the UAS wing becomes unstable near the stall angle, it does recover stable airflow at angles-of-attack well above the stall angle. Moreover, the aircraft has notably increased drag at this stable, high angle-of-attack condition, thus allowing horizontal speed to be arrested quickly. The result is a trimmed descent at a very high glide-slope angle, allowing the UAS to be landed in a confined area with vertical obstructions that would have prevented a controlled glide-slope landing. Of note, the Raven UAS from Fig. 1 uses a deep-stall landing and is designed to intentionally break apart into its separate components upon impact to reduce the landing shock.

Fail-Safes

As a final component of flight control, UAS autopilots often include a set of user-configurable fail-safes that are automatically enacted in certain situations. For example, the UAS might automatically enter a “Return To Launch Point” mode if it exceeds a specified distance from the launch point, communications are lost, or the battery voltage falls below a certain threshold. If GPS signals are lost or degraded and the UAS can no longer determine its location, it will usually enter an altitude hold and constant roll mode until GPS is (hopefully) restored. Finally, if the battery voltage falls below some critical threshold, the UAS might immediately enter an automatic landing mode.

MISSION CONTROL

Using flight control algorithms similar to those described in limited detail above, a UAS is able to take off, achieve a desired altitude and speed, follow a prescribed trajectory (e.g., waypoints or loiter commands), and automatically land when commanded. Although not described in this article, a UAS with a gimbaled camera payload can even automatically control the gimbal to continuously point the camera at (or relatively near) a desired ground location. In order to successfully accomplish a desired mission, however, a mission controller must translate mission objectives into flight commands (see Fig. 3).

For most UASs, the mission controller is a human operator who uses ground control software (see Fig. 18) to specify a set of trajectory commands and, if applicable, payload pointing commands to meet a desired objective and manually adjusts those commands as new information becomes available. Relying on the operator to adjust UAS trajectory and payload commands is a highly effective and robust means of acquiring video-based ISR, for example, but at the expense of operator attention, operator workload, reliance on consistent communications, and in some cases a delayed response time. APL and other research organizations are currently advanc-

ing the art of full mission autonomy in unmanned vehicles by developing and demonstrating mission control mechanisms that convert tasks and objectives into evolving trajectory and payload commands for one or more vehicles. Two autonomy efforts at APL involving multivehicle cooperative control are Closed-Loop Collaborative ISR and Mission-Level Autonomy. Both of these efforts close the outer mission control loop shown in Fig. 3 by automatically adjusting trajectory and payload commands based on UAS observations of an evolving scenario.

Closed-Loop Collaborative ISR²¹ is an automated ISR management effort wherein multiple UASs are dynamically tasked from a centralized location to search for and track ground vehicles over a large region of interest. The centralized mission controller ingests either raw sensor measurements or processed target tracks from all UASs, uses a multiple hypothesis tracker to generate a fused picture of tracked ground vehicles, and automatically redirects the UASs using a receding horizon controller such that the existing tracks can be maintained and other areas can be explored for new ground vehicles to track. Closed-Loop Collaborative ISR uses sensor fusion and path planning expertise at APL and leverages simulation capabilities to demonstrate mission control effectiveness against evolving scenarios.

In contrast, Mission-Level Autonomy²² is a decentralized mission control mechanism wherein unmanned vehicles, such as UASs, share information and collaborate to achieve mission objectives. In Mission-Level Autonomy, each vehicle maintains a belief map (i.e., an operational picture) based on the combination of its own experiences and those of any neighbors in communications range. By using a behavior-based control called Dynamic Co-Fields to convert beliefs and objectives into time-varying attractors and repulsors, each vehicle can determine which tasks to pursue, which direction to travel, and what actions to take to complete the objectives. Using Mission-Level Autonomy, APL has flight demonstrated heterogeneous teams of fully autonomous small UASs cooperating to perform mission-level tasks such as area and road network search, convoy protection, airspace deconfliction, communications relay, and chemical plume tracking.

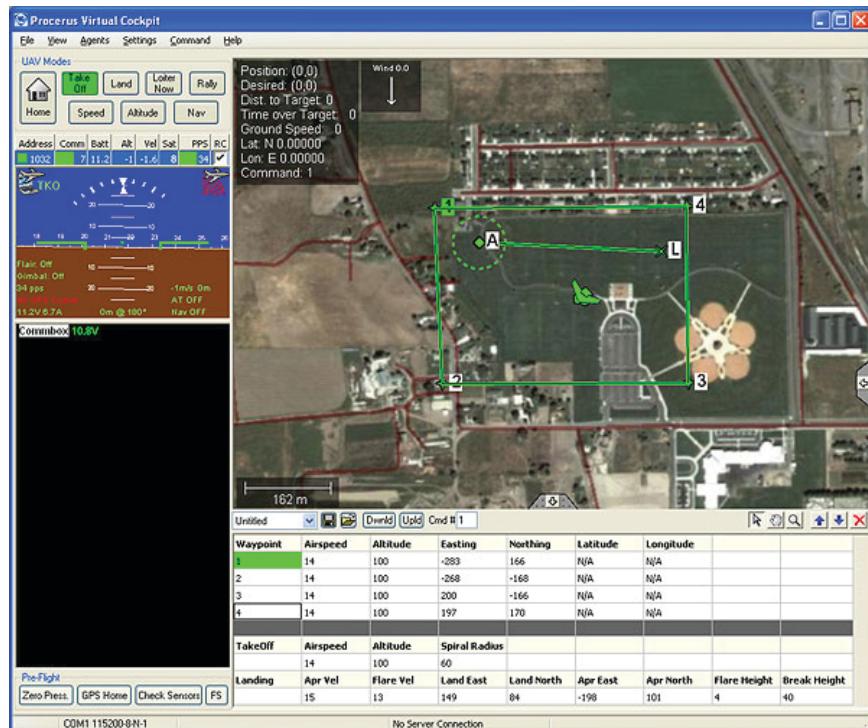


Figure 18. Example ground station user interface for controlling a UAS. The interface (Procerus Technologies Virtual Cockpit for their Kestrel autopilot) uses intuitive graphical features to provide live telemetry information as well as a means of updating the UAS in-flight with new trajectory commands. (Reprinted with permission from Procerus Technologies © 2004–2011. All rights reserved.)

SUMMARY

The key objective of this article was to describe the fundamentals of state estimation and control for small fixed-wing UASs. Proper state estimation is of critical importance on a UAS because it provides the feedback used in all UAS flight and payload control routines. By comparing trajectory commands with the current state estimates, the UAS autopilot uses flight control algorithms to convert trajectory commands such as altitude, speed, and the desired route into aerodynamic surface and throttle commands. Most UASs in use today require a human operator to interpret mission objectives into trajectory commands and modify those commands as new information becomes available. In contrast, current research at APL and other organizations is enabling full mission autonomy by closing the mission control loop with algorithms that automatically generate new flight commands for one or more vehicles on the basis of observations of the evolving scenario. Using automatic mission control algorithms reduces operator workload, improves vehicle reaction time to evolving scenarios, and in the decentralized case, increases robustness to communications dropouts.

REFERENCES

- ¹Cai, G., Lum, K.-Y., Chen, B. M., and Lee, T. H., "A Brief Overview on Miniature Fixed-Wing Unmanned Aerial Vehicles," in *Proc. 8th IEEE International Conf. on Control and Automation*, Xiamen, China, pp. 285–290 (2010).
- ²Reuder, J., Brisset, P., Jonassen, M., Müller, M., and Mayer, S., "SUMO: A Small Unmanned Meteorological Observer for Atmospheric Boundary Layer Research," *Proc. IOP Conf. Ser.: Earth Environ. Sci.* 1, paper 012014 (2008).
- ³Chao, H., Cao, Y., and Chen, Y., "Autopilots for Small Unmanned Aerial Vehicles: A Survey," *Int. J. Control Autom. Syst.* 8(1), 36–44 (2010).
- ⁴DIY Drones website, www.diydrones.com (accessed 24 Sept 2012).
- ⁵Paparazzi website, paparazzi.enac.fr (accessed 24 Sept 2012).
- ⁶Gluonpilot website, www.gluonpilot.com (accessed 24 Sept 2012).
- ⁷Stombaugh, T., McLaren, D., and Koostra, B., "The Global Positioning System," *University of Kentucky Cooperative Extension Circular AEN-88* (2005).
- ⁸Barber, B., "Accurate Target Geolocation and Vision-Based Landing with Application to Search and Engage Missions for Miniature Air Vehicles," master's thesis, Brigham Young University, Provo, UT (2007).
- ⁹Bryson, M., and Sukkarieh, S., "Vehicle Model-Aided Inertial Navigation for a UAV Using Low-Cost Sensors," in *Proc. Australasian Conf. on Robotics and Automation*, Canberra, Australia, <http://www.araa.asn.au/acra/acra2004/papers/bryson.pdf> (2004).
- ¹⁰Premerlani, W., and Bizard, P., "Direction Cosine Matrix IMU: Theory," <http://gentlenav.googlecode.com/files/DCMDraft2.pdf> (accessed 24 Sept 2012).
- ¹¹Euston, M., Coote, P., Mahony, R., Kim, J., and Hamel, T., "A Complementary Filter for Attitude Estimation of a Fixed-Wing UAV," in *Proc. International Conf. on Intelligent Robots and Systems*, Nice, France, pp. 340–345 (2008).
- ¹²Kingston, D., Beard, R., McLain, T., Larsen, M., and Ren, W., "Autonomous Vehicle Technologies for Small Fixed Wing UAVs," in *Proc. 2nd AIAA "Unmanned Unlimited" Conf. and Workshop and Exhibit*, San Diego, CA, paper AIAA-2003-6559 (2003).
- ¹³National Oceanic and Atmospheric Administration, NOAA's Geophysical Data Center—Geomagnetism, <http://www.ngdc.noaa.gov/geomag-web/#declination> (accessed 24 Sept 2012).
- ¹⁴Beard, R. W., and McLain, T. W., "Navigation and Control Technologies for Autonomous Micro Vehicles," Brigham Young University Technical Report, <http://hdl.lib.byu.edu/1877/65> (25 Aug 2005).
- ¹⁵Taylor, B., Bil, C., and Watkins, S., "Horizon Sensing Attitude Stabilization: A VMC Autopilot," in *Proc. 18th International UAV Systems Conf.*, Bristol, UK, www.ctie.monash.edu.au/hargrave/horizon_sensing_autopilot.pdf (2003).
- ¹⁶Brisset, P., Drouin, A., Gorraz, M., Huard, P.-S., and Tyler, J., "The Paparazzi Solution," in *Proc. 2nd US-European Competition and Workshop on Micro Air Vehicles*, Sandestin, FL, pp. 1–15, www.recherche.enac.fr/paparazzi/papers_2006/mav06_paparazzi.pdf (2007).
- ¹⁷Brown, R. H., and Hwang, P. Y. C., *Introduction to Random Signals and Applied Kalman Filtering*, 2nd Ed., John Wiley and Sons, New York (2001).
- ¹⁸Eldredge, A. M., "Improved State Estimation for Miniature Air Vehicles," master's thesis, Brigham Young University, Provo, UT (2006).
- ¹⁹Christophersen, H. B., Pickell, R. W., Neidhoefer, J. C., Koller, A. A., Kannan, S. K., and Johnson, E. N., "A Compact Guidance, Navigation, and Control System for Unmanned Aerial Vehicles," *J. Aerosp. Comput. Inf. Commun.* 3(5), 187–213 (2006).
- ²⁰Taniguchi, H., "Analysis of Deepstall Landing for UAV," in *Proc. 26th International Congress of the Aeronautical Sciences*, Anchorage, AK, pp. 1–6 (2008).
- ²¹Newman, A. J., Martin, S. R., DeSena, J. T., Clarke, J. C., McDermott, J. W., Preissler, W. O., and Peterson, C. K., "Receding Horizon Controller Using Particle Swarm Optimization for Closed Loop Ground Target Surveillance and Tracking," in *Signal Processing, Sensor Fusion, and Target Recognition XVIII*, Proc. of SPIE, Vol. 7336, I. Kadar (ed.), SPIE, Bellingham, WA, pp. 73360M-1–73360M-12 (2009).
- ²²Chalmers, R. W., Scheidt, D. H., Neighoff, T. M., Witwicki, S. J., and Bamberger, R. J., "Cooperating Unmanned Vehicles," in *Proc. AIAA 1st Intelligent Systems Technical Conf.*, Chicago, IL, paper AIAA 2004-6252 (2004).

The Author

Jeffrey D. Barton is a member of APL's Senior Professional Staff and has extensive professional experience with guidance and control of air vehicles, including missiles, aerial targets, and UAVs. During the previous 7 years, Mr. Barton has successfully crashed small UAVs in eight states and in environments as diverse as bear-infested mountains and tarantula-infested deserts. Thus far, he has managed to walk away from each incident unharmed. His e-mail address is jeffrey.barton@jhuapl.edu.

The Johns Hopkins APL Technical Digest can be accessed electronically at www.jhuapl.edu/techdigest.