State Estimation for Micro Air Vehicles

Randal W. Beard

Department of Electrical and Computer Engineering
Brigham Young University, Provo, Utah
beard@ee.byu.edu

Summary. Autopilots for small UAVs are generally equipped with low fidelity sensors that make state estimation challenging. In addition, the sensor suite does not include units that measure angle-of-attack and side-slip angles. The achievable flight performance is directly related to the quality of the state estimates. Unfortunately, the computational resources on-board a small UAV are generally limited and preclude large state Kalman filters that estimate all of the states and sensor biases. In this chapter we describe simple models for the sensors typically found on-board small UAVs. We also describes a simple cascaded approach to state estimation that has been extensively flight tested using the Kestrel autopilot produced by Procerus Technologies. Our intention is to provide a tutorial of continuous-discrete Kalman filtering with application to state estimation for small UAVs.

High fidelity estimates of the position, velocity, attitude, and angular rates are critical for successful guidance and control of intelligent UAVs. The achievable fidelity of the state estimates depends upon the quality of the sensors on-board the UAV. Unfortunately, high quality sensors are usually heavy and expensive. This is particularly true for sensors that directly measure the attitude of the UAV. In this chapter we focus on the problem of state estimation using light weight, inexpensive, low quality sensors. In doing so, our target platforms are small and micro air vehicles with limited payload capacity.

In recent years, several autopilots for small UAVs have appeared on the commercial market. These include the Procerus Kestrel [4], the Cloudcap Piccolo [2], and the Micropilot MP2028 [3]. Each of these autopilots use the following sensors:

- rate gyros,
- accelerometers,
- pressure sensors, and
- GPS.

We will assume throughout this chapter that these are the only sensors that are available for state estimation.

The limited payload capacity of small UAVs not only restricts the type and quality of the sensors, it also limits the computational resources that can be placed on-board the UAV. For example, the Procerus Kestrel autopilot has an 8-bit Rabbit microcontroller with 512K of memory. Therefore, Kalman filters that estimate all of the states as well as the sensor biases are not feasible. The objective of this chapter is to describe simple attitude estimation techniques for small UAVs that require limited computational resources.

The chapter will be organized as follows. In Section 1 we define and briefly describe the states that need to be estimated. In Section 2 we describe the sensors that are generally available on small UAVs and develop mathematical models of their behavior. Section 3 briefly describes the simulation environment that is used to demonstrate the algorithms described in this chapter. Section 4 describes simple state estimation techniques that use digital low pass filters and sensor model inversion. In Section 5 we provide a brief review of the continuous-discrete Kalman filter. Finally, Section 6 describes the application of the continuous-discrete extended Kalman filter to roll, pitch, position, and heading estimation.

1 UAV State Variables

Aircraft have three degrees of translational motion and three degrees of rotational motion. Therefore, there are twelve state variables as listed below:

$$
\begin{aligned}
p_n &= \text{the inertial north (latitude) position of the UAV,} \\
p_e &= \text{the inertial east (longitude) position of the UAV,} \\
h &= \text{the altitude of the UAV,} \\
u &= \text{the body frame velocity measured out the nose,} \\
v &= \text{the body frame velocity measured out the right wing,} \\
w &= \text{the body frame velocity measured through the belly,} \\
\phi &= \text{the roll angle,} \\
\theta &= \text{the pitch angle,} \\
\psi &= \text{the yaw angle,} \\
p &= \text{the roll rate,} \\
q &= \text{the pitch rate,} \\
r &= \text{the yaw rate.}
\end{aligned}
$$

The state variables are shown schematically in Figure 1. As an alternative to expressing the velocity vector as $(u, v, w)^T$, it can be expressed in terms of the airspeed V_a, the angle-of-attack α, and the side-slip angle β. The transformation between the two representations is given by [22]

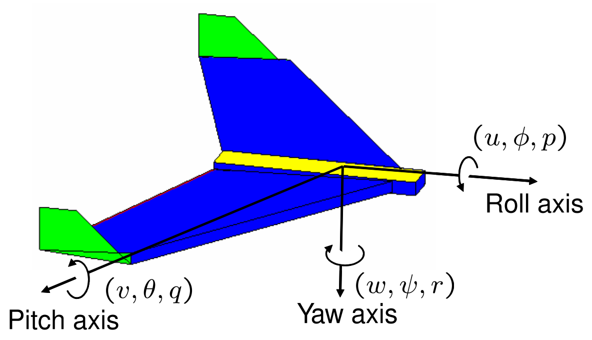


Fig. 1. This figures depicts some of the UAV state variables. The forward velocity u and the roll rate p are defined along the roll axis which points out the nose of the UAV. The side slip velocity v and the pitch rate q are defined along the pitch axis which points out the right wing of the UAV. The downward velocity w and the yaw rate r are defined with respect to the yaw axis which points out the belly of the UAV. The Euler angles are defined by first yawing ψ about the yaw axis, pitching θ about the transformed pitch axis, and finally rolling ϕ about the transformed roll axis.

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = V_a \begin{pmatrix} \cos\alpha\cos\beta \\ \sin\beta \\ \sin\alpha\cos\beta \end{pmatrix}. \tag{1}$$

$$\begin{aligned} V_a &= \sqrt{u^2 + v^2 + w^2} \\ \alpha &= \tan^{-1}\left(\frac{w}{u}\right) \\ \beta &= \tan^{-1}\left(\frac{v}{\sqrt{u^2 + w^2}}\right). \end{aligned} \tag{2}$$

There are several other quantities that are also of interest for guidance and control of UAVs including the flight path angle γ, the course angle χ, and the ground velocity V_g. The flight path angle defines the inertial climb angle of the UAV and is given by

$$\gamma = \theta - \alpha\cos\phi - \beta\sin\phi.$$

Note that in wings level flight, this formula reduces to the standard equation $\gamma = \theta - \alpha$. The course angle defines the inertial heading of the UAV which may be different than the yaw angle ψ due to wind. If $(w_n, w_e)^T$ is the wind vector in the inertial frame, then we have the following relationships

$$V_g \begin{pmatrix} \cos\chi \\ \sin\chi \end{pmatrix} = V_a \begin{pmatrix} \cos\psi\cos\gamma \\ \sin\psi\cos\gamma \end{pmatrix} + \begin{pmatrix} w_n \\ w_e \end{pmatrix}$$

$$V_g = \sqrt{V_a^2\cos^2\gamma + 2V_a\cos\gamma\sqrt{w_n^2+w_e^2}\cos(\psi - \tan^{-1}\left(\frac{w_e}{w_n}\right)) + w_n^2 + w_e^2}$$

$$\chi = \tan^{-1}\left(\frac{V_a\sin\psi\cos\gamma + w_e}{V_a\cos\psi\cos\gamma + w_n}\right).$$

The kinematic evolution of the Euler angles are given by [22]

$$\begin{pmatrix} \dot\phi \\ \dot\theta \\ \dot\psi \end{pmatrix} = \begin{pmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)\sec(\theta) & \cos(\phi)\sec(\theta) \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix}, \tag{3}$$

and the navigational equations of motion are given by

$$\dot r_n = V_a\cos\psi\cos\gamma + w_n = V_g\cos\chi \tag{4}$$

$$\dot r_e = V_a\sin\psi\cos\gamma + w_e = V_g\sin\chi \tag{5}$$

$$\dot h = V_a\sin\gamma. \tag{6}$$

2 Sensor Models

This section derives mathematical models for sensors typically found on small and micro UAVs. In particular, we discuss rate gyros, accelerometers, pressure sensors, and GPS sensors.

2.1 Rate Gyros

A MEMS rate gyro contains a small vibrating lever. When the lever undergoes an angular rotation, Coriolis effects change the frequency of the vibration, thus detecting the rotation. A brief description of the physics of rate gyros can be found in Ref [9, 15, 23].

The output of the rate gyro is given by

$$y_{\text{gyro}} = k_{\text{gyro}}\omega + \beta_{\text{gyro}}(T) + \eta_{\text{gyro}},$$

where y_{gyro} is in Volts, k_{gyro} is a gain, ω is the angular rate in radians per second, β_{gyro} is a temperature dependent bias term, and η_{gyro} is a zero mean Gaussian process with known variance. The bias term $\beta_{\text{gyro}}(T)$ is a function of the temperature T and can be effectively determined by use of a temperature chamber before flight.

If three rate gyros are aligned along the x, y, and z axes of the UAV, then the rate gyros measure the angular body rates p, q, and r as follows:

$$\begin{aligned} y_{\text{gyro},x} &= k_{\text{gyro},x} p + \beta_{\text{gyro},x}(T) + \eta_{\text{gyro},x} \\ y_{\text{gyro},y} &= k_{\text{gyro},y} q + \beta_{\text{gyro},y}(T) + \eta_{\text{gyro},y} \\ y_{\text{gyro},z} &= k_{\text{gyro},z} r + \beta_{\text{gyro},z}(T) + \eta_{\text{gyro},z}. \end{aligned}$$

We will assume that $k_{\text{gyro},*}$, $\beta_{\text{gyro},*}(T)$, and the covariance of $\eta_{\text{gyro},*}$ have been determined *a priori* and are known in-flight. MEMS gyros are analog devices that are sampled by the on-board processer. We will assume that the sample rate is given by T_s. As an example, the Procerus Kestrel autopilot samples its rate gyros at approximately 120 Hz.

2.2 Accelerometers

A MEMS accelerometer contains a small plate attached to torsion levers. The plate rotates under acceleration which changes the capacitance between the plate and the surrounding walls. The change in capacitance is proportional to the linear acceleration [1, 23].

The output of the accelerometers is given by

$$y_{\text{acc}} = k_{\text{acc}} a + \beta_{\text{acc}}(T) + \eta_{\text{acc}},$$

where y_{acc} is in Volts, k_{acc} is a gain, a is the acceleration in meters per second squared, β_{acc} is a temperature dependent bias term, and η_{acc} is zero mean Gaussian noise with known variance.

Accelerometers measure the specific force in the body frame of the vehicle. A physically intuitive explanation is given in [22, p. 13-15]. An additional explanation is given in [19, p. 27]. Mathematically we have

$$\begin{aligned} \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} &= \frac{1}{m}\left(\mathbf{F} - \mathbf{F}_{\text{gravity}}\right) \\ &= \dot{\mathbf{v}} + \boldsymbol{\omega} \times \mathbf{v} - \frac{1}{m}\mathbf{F}_{\text{gravity}}. \end{aligned}$$

In component form we have

$$\begin{aligned} a_x &= \dot{u} + qw - rv + g\sin\theta \\ a_y &= \dot{v} + ru - pw - g\cos\theta\sin\phi \\ a_z &= \dot{w} + pv - qu - g\cos\theta\cos\phi. \end{aligned}$$

The output of an accelerometer is usually in units of [g], therefore $k_{\text{acc}} = 1/g$. The output of the accelerometers are therefore given by

$$\begin{aligned} y_{\text{acc},x} &= \frac{\dot{u} + qw - rv + g\sin\theta}{g} + \beta_{\text{acc},x}(T) + \eta_{\text{acc},x} \\ y_{\text{acc},y} &= \frac{\dot{v} + ru - pw - g\cos\theta\sin\phi}{g} + \beta_{\text{acc},y}(T) + \eta_{\text{acc},y} \qquad (7) \\ y_{\text{acc},z} &= \frac{\dot{w} + pv - qu - g\cos\theta\cos\phi}{g} + \beta_{\text{acc},z}(T) + \eta_{\text{acc},z}. \qquad (8) \end{aligned}$$

As with the rate gyros, we will assume that the biases and noise statistics are known and available in-flight. MEMS accelerometers are analog devices that are sampled by the on-board processer. We will assume that the sample rate is given by T_s.

2.3 Pressure Sensors

Small autopilots typically have two pressure sensors: a static pressure sensor which is used to measure altitude, and a dynamic pressure sensor which is used to measure airspeed. These sensors will be discussed in the following two sections.

Altitude Sensor

Pressure is a measure of force per unit area or

$$P = \frac{F}{A},$$

where P is the pressure, F is the force, and A is the area. The static pressure at a particular altitude is determined by the force exerted by a column of air at that altitude:

$$P = \frac{m_{\text{column}} g}{A},$$

where m_{column} is the mass of the column of air, g is the gravitational constant, and A is the area upon which the column is exerting pressure. The density of air is the mass per unit volume. Since the volume is given by the area times the height we get

$$P = \rho h g,$$

where ρ is the density of air and h is the altitude [8, 11].

Therefore, the output of the static pressure sensor is given by

$$y_{\text{static pres}} = \rho g h + \beta_{\text{static pres}} + \eta_{\text{static pres}},$$

where $\beta_{\text{static pres}}$ is a slowly varying bias and $\eta_{\text{static pres}}$ is a zero mean Gaussian process. To remove the bias, we collect multiple measurements of the pressure on the ground and average to remove the Gaussian noise to obtain

$$\bar{y}_{\text{static pres}}(h_{\text{ground}}) = \rho g h_{\text{ground}} + \beta_{\text{static pres}}.$$

If we know the altitude of the ground station above sea level and the density of the surrounding air, then the bias $\beta_{\text{static pres}}$ can be determined. If, on the other hand, we are interested in the height above the ground station then we can subtract the calibrated ground measurement to obtain

$$\begin{aligned} y_{\text{static pres}}(\Delta h) &\overset{\triangle}{=} y_{\text{static pres}}(h) - \bar{y}_{\text{static pres}}(h_{\text{ground}}) \\ &= \rho g (h - h_{\text{ground}}) + \eta_{\text{static pres}}(t), \\ &= \rho g \Delta h + \eta_{\text{static pres}}(t), \end{aligned}$$

where Δh is the height above the ground station.

Air Speed Sensor

When the UAV is in motion, the atmosphere exerts dynamic pressure on the UAV parallel to the direction of flow. The dynamic pressure is given by [8]

$$P_I = \frac{1}{2}\rho V_a^2,$$

where V_a is the airspeed of the UAV. Bernoulli's theorem states that [8]

$$P_s = P_I + P_O,$$

where P_s is the total pressure, and P_O is the static pressure.

Therefore, the output of the differential pressure sensor is

$$\begin{aligned} y_{\text{diff pres}} &= P_s - P_O + \eta_{\text{diff pres}} \\ &= \frac{1}{2}\rho V_a^2 + \eta_{\text{diff pres}}(t), \end{aligned}$$

where $\eta_{\text{diff pres}}$ is a zero mean Gaussian process with known variance.

The static and differential pressure sensors are analog devices that are sampled by the on-board processer. We will assume that the sample rate is given by T_s.

2.4 GPS

There are several sources of GPS error. Table 1 lists the sources of error and the respective error budget. The data was obtained from
http://www.montana.edu/places/gps/lres357/slides/GPSaccuracy.ppt.

The current weather affects the speed of light in the atmosphere. However, this inaccuracy should be relatively constant for a given day. We will model the effect of the atmosphere by a random variable drawn from a Gaussian distribution with a standard deviation equal to 5 meters.

Effect	Ave. Horizontal Error	Ave. Vertical Error
Atmosphere	5.5 meters	5.5 meters
Satellite Geometry (Ephemeris) data	2.5 meters	15 meters
Satellite clock drift	1.5 meters	1.5 meters
Multipath	0.6 meters	0.6 meters
Measurement noise	0.3 meters	0.3 meters

Table 1. This table lists average error estimates for commercial grade GPS units. Atmosphere, satellite geometry, clock drift, and multipath produce a near constant bias term. The measurement noise is modeled as an additive Gaussian process.

The geometry of the Satellites viewed by the receiver is used to triangulate the location of the GPS receiver. Triangulation is much more effective in the horizontal plane than in the vertical direction. The satellite geometry is slowly changing in time. Therefore we will measure the effect of satellite geometry as a sinusoid with amplitude equal to $2.5\sqrt{2}$ (RMS=2.5), with a constant but unknown frequency ω_{geometry} and a phase that is a random variable drawn from a uniform distribution over $[-\pi, \pi]$.

We will assume that the clock drift is relatively constant over time. Therefore, we will model the clock drift by a constant random variable drawn from a Gaussian distribution with standard deviation of 1.5 meters.

Multipath is a function of the position of the UAV. Therefore we will assume that the error is a sinusoidal signal with a magnitude of $0.6\sqrt{2}$, a frequency equal to $\omega_{\text{multipath}}$ and a random phase drawn from a uniform distribution over $[-\pi, \pi]$.

We will model the measurement noise as a zero mean Gaussian process with a variance equal to 0.3 meters. The model for the GPS signal is therefore given by

$$\begin{aligned}
y_{\text{GPS},n}(t) &= p_n + \nu_{n,\text{atmosphere}} + \nu_{\text{clock}} + \eta_{n,\text{measurement}}(t) \\
&\quad + 2.5\sqrt{2}\sin(\omega_{\text{geometry}}t + \nu_{n,\text{geometry}}) + 0.6\sqrt{2}\sin(\omega_{\text{multipath}}t + \nu_{n,\text{multipath}}) \\
y_{\text{GPS},e}(t) &= p_e + \nu_{e,\text{atmosphere}} + \nu_{e,\text{clock}} + \eta_{e,\text{measurement}}(t) \\
&\quad + 2.5\sqrt{2}\sin(\omega_{\text{geometry}}t + \nu_{e,\text{geometry}}) + 0.6\sqrt{2}\sin(\omega_{\text{multipath}}t + \nu_{e,\text{multipath}}) \\
y_{\text{GPS},h}(t) &= h + \nu_{h,\text{atmosphere}} + \nu_{h,\text{clock}} + \eta_{h,\text{measurement}}(t), \\
&\quad + 15\sqrt{2}\sin(\omega_{\text{geometry}}t + \nu_{h,\text{geometry}}) + 0.6\sqrt{2}\sin(\omega_{\text{multipath}}t + \nu_{h,\text{multipath}}),
\end{aligned}$$

where p_n, p_e, and h are the actual earth coordinates and altitude above sea level respectively. The GPS receiver also computes estimated ground speed and heading from the measurements listed above. Accordingly, we have

$$\begin{aligned}
y_{\text{GPS},V_g} &= \sqrt{\left(\frac{y_{\text{GPS},n}(t+T_s) - y_{\text{GPS},n}(t)}{T_s}\right)^2 + \left(\frac{y_{\text{GPS},e}(t+T_s) - y_{\text{GPS},e}(t)}{T_s}\right)^2} \\
y_{\text{GPS},\text{course}} &= \tan^{-1}\left(\frac{y_{\text{GPS},e}(t+T_s) - y_{\text{GPS},e}(t)}{y_{\text{GPS},n}(t+T_s) - y_{\text{GPS},n}(t)}\right).
\end{aligned}$$

The update rate of a GPS receiver is typically on the order of $T_{\text{GPS}} = 1$ second. However, the update rate can vary between $0.1 - 2$ seconds, depending on the GPS receiver.

3 Simulation Environment

We will illustrate the quality of the state estimation techniques proposed in this chapter via simulation. This section briefly describes the simulation environment which is a six degree-of-freedom nonlinear flight simulator called Aviones, developed at Brigham Young University using C/C++, and which runs on the Microsoft Windows operating system. The sensor models described in the previous section were implemented in Aviones using the parameters shown in Table 2. We have assumed that sensor biases are estimated before flight and are therefore not included in the simulator, with the exception of GPS, where it is not possible to estimate the biases.

Parameter	Value	Units
$\sigma_{\text{gyro},x}$	0.005	rad/sec
$\sigma_{\text{gyro},y}$	0.005	rad/sec
$\sigma_{\text{gyro},z}$	0.005	rad/sec
$\sigma_{\text{acc},x}$	0.005	m/sec^2
$\sigma_{\text{acc},y}$	0.005	m/sec^2
$\sigma_{\text{acc},z}$	0.005	m/sec^2
$\sigma_{\text{static pres}}$	0.4	meters
$\sigma_{\text{diff pres}}$	0.4	meters/sec
$\sigma_{\text{mag},x}$	500	nanotesla
$\sigma_{\text{mag},y}$	500	nanotesla
$\sigma_{\text{mag},z}$	500	nanotesla
$\sigma_{\text{GPS},n}$	0.5	meters
$\sigma_{\text{GPS},e}$	0.5	meters
$\sigma_{\text{GPS},h}$	0.5	meters
$\bar{\nu}_{\text{atmosphere}}$	5.5	meters
$\bar{\nu}_{\text{clock}}$	1.5	meters
$\bar{\nu}_{\text{geometry}}$	2.5	meters
$\bar{\nu}_{\text{multipath}}$	0.6	meters

Table 2. Sensor parameters used in the Aviones flight simulator. σ_* denote the variance of a zero mean Gaussian process. ν_* denotes a random variable drawn uniformly from the set $[0, \bar{\nu}_*]$.

The state estimate plots shown in this chapter are all associated with a similar flight trajectory which was dictated by the following autopilot commands (using full state feedback).

- **Throughout maneuver:**
  Hold airspeed at 10.0 m/s.

- $0 \leq t \leq 2.5$ **seconds:**
  Hold a pitch angle of 20 degrees.
  Hold a roll angle of 30 degrees.
- $2.5 \leq t \leq 5.0$ **seconds:**
  Hold a pitch angle of -20 degrees.
  Hold a roll angle of 0 degrees.
- $5.0 \leq t \leq 8.0$ **seconds:**
  Hold a pitch angle of 20 degrees.
  Hold a roll angle of -30 degrees.
- $8.0 \leq t \leq 10.0$ **seconds:**
  Hold a pitch angle of -20 degrees.
  Hold a roll angle of 0 degrees.
- $8.0 \leq t \leq 10.0$ **seconds:**
  Hold a pitch angle of -20 degrees.
  Hold a roll angle of 0 degrees.
- $10.0 \leq t \leq 13.0$ **seconds:**
  Hold a pitch angle of 20 degrees.
  Hold a roll angle of 30 degrees.
- $13.0 \leq t \leq 30.0$ **seconds:**
  Hold a pitch angle of 0 degrees.
  Hold a roll angle of 0 degrees.

A plot of the state variables during this maneuver is shown in Figure 2.

4 State Estimation via Model Inversion

The objective of this section is to demonstrate that computationally simple state estimation models can be derived by inverting the sensor models. As we shall demonstrate, the quality of the estimates produced by this method is, unfortunately, relatively poor for some of the states.

4.1 Low Pass Filters

All of the state estimation schemes require low-pass filtering of the sensor signals. For completeness, we will briefly discuss digital implementation of a first order low-pass filter.

The Laplace transform representation of a simple unity DC gain low-pass filter is given by

$$Y(s) = C(s)U(s) \overset{\triangle}{=} \frac{a}{s+a}U(s),$$

were $u(t)$ is the input of the filter and $y(t)$ is the output. Taking the inverse Laplace transform we obtain

$$\dot{y} = -ay + au. \tag{9}$$

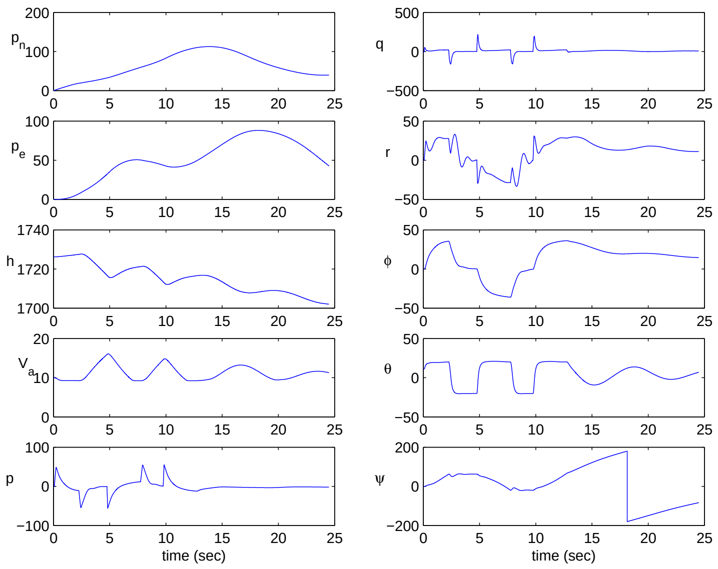


Fig. 2. Actual states during the simulated test maneuver used throughout the article. The positions p_n and p_e are in units of meters from home base, h is in units of meters above sea level, V_a is in meters/sec, p, q, and r are in units of degrees/sec, and ϕ, θ, and ψ are in units of degrees.

By introducing an integrating factor, it is straightforward to show that the solution to this differential equation is given by

$$y(t+T) = e^{-aT}y(t) + a\int_0^T e^{-a(T-\tau)}u(\tau)\,d\tau.$$

Assuming that $u(t)$ is constant between sample periods results in the expression

$$\begin{aligned} y(t+T) &= e^{-aT}y(t) + a\int_0^T e^{-a(T-\tau)}\,d\tau u(t) \\ &= e^{-aT}y(t) + (1-e^{-aT})u(t). \end{aligned} \tag{10}$$

Note that this equation has a nice physical interpretation: the new value of y (filtered value) is a weighted average of the old value of y and u (unfiltered value). We will use the notation $C(s)\{\cdot\}$ to represent the low-pass filter operator. Therefore $\hat{x} = C(s)\{x\}$ is the low-pass filtered version of x.

4.2 State Estimation by Inverting the Sensor Model

In this section we will derive the simplest possible state estimation scheme based on inverting the sensor models. While this method is effective for angular rates, altitude, and airspeed, it is not effective for estimating the position and Euler angles.

Position and Heading

The position variables p_n, p_e and the course heading χ can be estimated by low-pass filtering the GPS signals:

$$\hat{p}_n = C(s)\{y_{\text{GPS,n}}\} \tag{11}$$
$$\hat{p}_e = C(s)\{y_{\text{GPS,e}}\} \tag{12}$$
$$\hat{\chi} = C(s)\{y_{\text{GPS,course}}\}. \tag{13}$$

Figure 3 shows the actual and estimated states using this scheme. Note that since the measurements are only received every second, the estimates have a sampled data characteristic that includes significant delay.

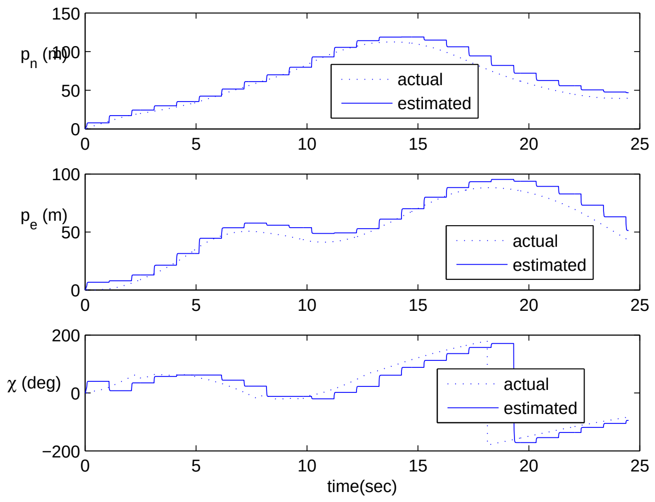


Fig. 3. Actual and estimated values of p_n, p_e, and h after low pass filtering the GPS sensor. The actual and estimated values of χ are wrapped so that they lie between ± 180 degrees.

Angular Rates

Similarly, the angular rates p, q, and r can be estimated by low-pass filtering the rate gyro signals:

$$\hat{p} = C(s)\{y_{\text{gyro,x}}\}/k_{\text{gyro,x}} \tag{14}$$
$$\hat{q} = C(s)\{y_{\text{gyro,y}}\}/k_{\text{gyro,y}} \tag{15}$$
$$\hat{r} = C(s)\{y_{\text{gyro,z}}\}/k_{\text{gyro,z}}. \tag{16}$$

Figure 4 shows the actual and estimated states using this scheme. Note that low pass filtering the rate gyros results in acceptable estimates of p, q, and r.

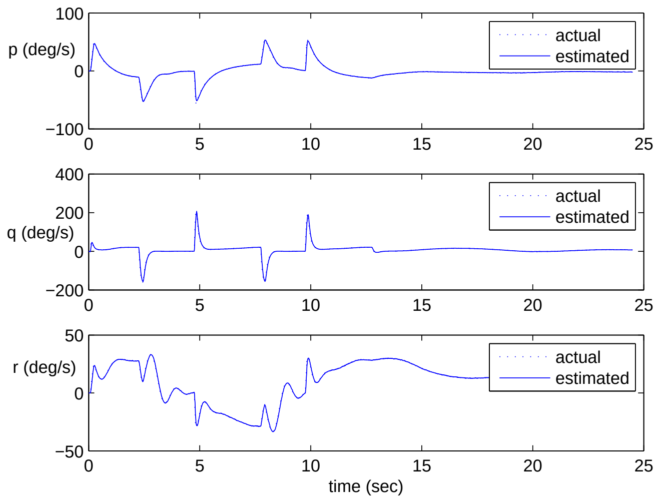


Fig. 4. Actual and estimated values of the angular rates p, q, and r after low pass filtering the rate gyros.

Altitude

GPS is not accurate enough to estimate the altitude. Therefore, we will use the absolute pressure sensor. Recall that

$$y_{\text{static pressure}} = \rho g (h - h_{\text{ground}}) + \eta_{\text{static pressure}}.$$

Therefore, a simple estimation scheme is

$$\hat{h} = h_{\text{ground}} + \frac{C(s)\{y_{\text{static pressure}}\}}{\rho g}. \tag{17}$$

Airspeed

Recall that

$$y_{\text{diff pres}} = \frac{1}{2}\rho V_a^2 + \eta_{\text{diff pres}}.$$

Therefore, a simple estimation scheme is

$$\hat{V}_a = \sqrt{\frac{2}{\rho} C(s)\{y_{\text{diff pres}}\}}. \tag{18}$$

Figure 5 shows the actual and estimated altitude and airspeed using this scheme. Again note that inverting the sensor models results in acceptable estimates of altitude and airspeed.

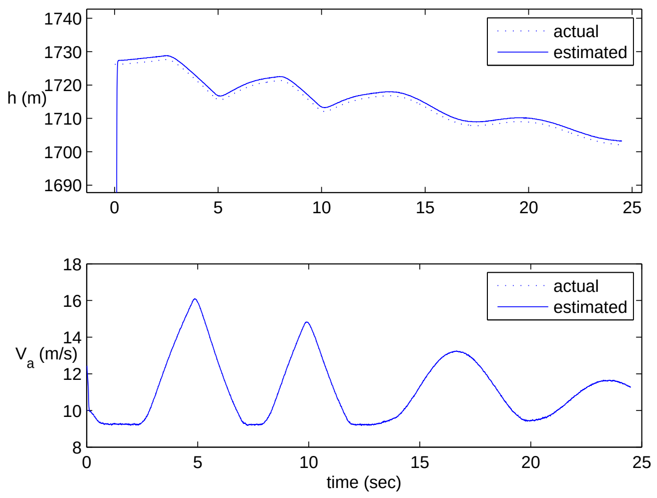


Fig. 5. Actual and estimated values of h and V_a after low pass filtering the pressure sensors and inverting their models.

Roll and Pitch Angles

Roll and pitch angles are the most difficult variables to estimate well on small UAVs. A simple scheme, that works in unaccelerated flight, can be derived as follows. Recalling that

$$\begin{aligned}
y_{\text{accel,x}} &= \frac{\dot{u} + qw - rv + g\sin\theta}{g} + \eta_{\text{accel,x}} \\
y_{\text{accel,y}} &= \frac{\dot{v} + ru - pw - g\cos\theta\sin\phi}{g} + \eta_{\text{accel,y}} \\
y_{\text{accel,z}} &= \frac{\dot{w} + pv - qu - g\cos\theta\cos\phi}{g} + \eta_{\text{accel,z}}.
\end{aligned}$$

and that in unaccelerated flight $\dot{u} = \dot{v} = \dot{w} = p = q = r = 0$, gives

$$\begin{aligned} C(s)\{y_{\text{accel,x}}\} &= \sin\theta \\ C(s)\{y_{\text{accel,y}}\} &= -\cos\theta\sin\phi \\ C(s)\{y_{\text{accel,z}}\} &= -\cos\theta\cos\phi. \end{aligned}$$

Solving for ϕ and θ we get

$$\hat{\phi}_{\text{accel}} = \tan^{-1}\left(\frac{C(s)\{y_{\text{accel,y}}\}}{C(s)\{y_{\text{accel,z}}\}}\right) \tag{19}$$

$$\hat{\theta}_{\text{accel}} = \tan^{-1}\left(\frac{C(s)\{y_{\text{accel,x}}\}}{\sqrt{C(s)\{y_{\text{accel,y}}\}^2 + C(s)\{y_{\text{accel,z}}\}^2}}\right). \tag{20}$$

Figure 6 shows the actual and estimated roll and pitch angles during the sample trajectory using this scheme. Note that the sample trajectory severely violates the unaccelerated flight assumptions. Clearly, model inversion does

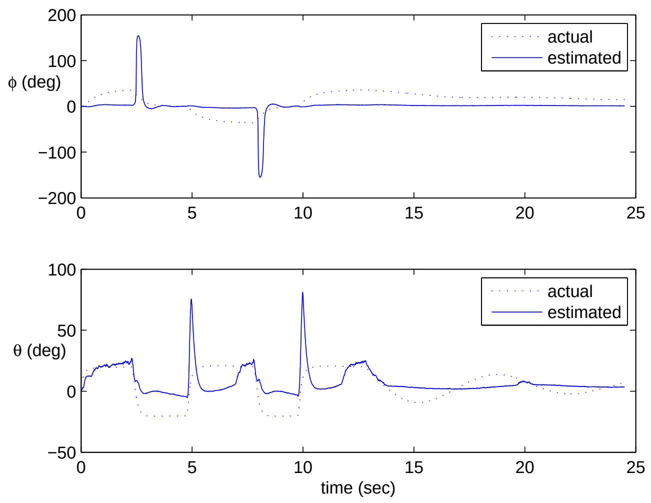


Fig. 6. Actual and estimated values of roll angle ϕ and pitch angle θ using simple model inversion.

not work well for attitude estimation during accelerated flight. Another idea is to combine model inversion with the integral of roll and pitch as estimated by the rate gyros.

Recalling that

$$\begin{aligned} \dot{\phi} &= p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \\ \dot{\theta} &= q\cos\phi - r\sin\phi \end{aligned}$$

and assuming that $\phi \approx 0$ and $\theta \approx 0$ we get

$$\dot{\phi} = p$$
$$\dot{\theta} = q.$$

Therefore we can integrate these equations to obtain an additional estimate of ϕ and θ:

$$\hat{\phi}_{\text{int}} = \int_{-\infty}^{t} p(\tau)\, d\tau$$
$$\hat{\theta}_{\text{int}} = \int_{-\infty}^{t} q(\tau)\, d\tau.$$

Combining the estimate from the integrator and the accelerometers we obtain

$$\hat{\phi} = \kappa\hat{\phi}_{\text{int}} + (1-\kappa)\hat{\phi}_{\text{accel}}$$
$$\hat{\theta} = \kappa\hat{\theta}_{\text{int}} + (1-\kappa)\hat{\theta}_{\text{accel}},$$

where $\kappa \in (0,1)$.

Figure 7 shows the actual and estimated roll and pitch angles using this scheme. It can be observed that the integration of the rate gyros causes a drift in the estimate of ϕ and θ.

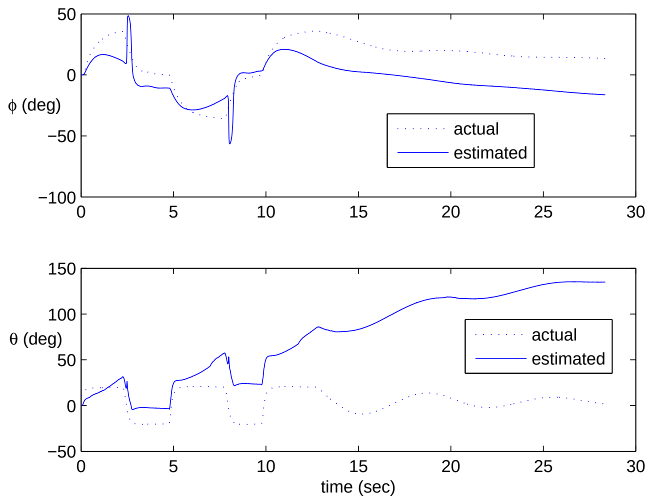


Fig. 7. Actual and estimated values of roll angle ϕ and pitch angle θ combining model inversion with the integral of the rate gyros.

While low pass filtering and model inversion work well for estimates of p, q, r, V_a and h, we need more sophisticated techniques to adequately estimate

p_n, p_e, χ, ϕ, and θ. In Section 5 we will review the basics of Kalman filter theory. In Section 6 we use two extended Kalman filters to obtain estimates for p_n, p_e, χ, ϕ, and θ.

5 The Continuous-Discrete Kalman Filter

The objective of this section is to give a brief review of Kalman filter theory. There are many excellent references on Kalman filtering including [12, 13, 14, 16, 5]. We will provide a brief derivation and then focus on the application of the Kalman filter to UAV state estimation.

5.1 Dynamic Observer Theory

As a first step in deriving the Kalman filter, we briefly review dynamic observer theory. Consider the linear time-invariant system modeled by the equations

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx. \end{aligned}$$

A continuous-time observer for this system is given by the equation

$$\dot{\hat{x}} = \underbrace{A\hat{x} + Bu}_{\text{copy of the model}} + \underbrace{L\,(y - C\hat{x})}_{\text{correction due to sensor reading}}, \tag{21}$$

where $\hat{x}$ is the estimated value of x. Letting $\tilde{x} = x - \hat{x}$ we get that

$$\dot{\tilde{x}} = (A - LC)\tilde{x}$$

which implies that the observation error decays exponentially to zero if L is chosen such that the matrix $A - LC$ is Hurwitz [20].

In practice, the sensors are usually sampled and processed in digital hardware at a sample rate T_s. How should the observer equation shown in Eq. (21) be modified to account for sampled sensor readings? The typical approach is to propagate the system model between samples using the equation

$$\dot{\hat{x}} = A\hat{x} + Bu \tag{22}$$

and then to update the estimate when a measurement is received using the equation

$$\hat{x}^+ = \hat{x}^- + L(y(t_k) - C\hat{x}^-), \tag{23}$$

where t_k is the instant in time that the measurement is received and $\hat{x}^-$ is the state estimate produced by Eq. (22) at time t_k. Equation (22) is then re-instantiated with initial conditions given by $\hat{x}^+$. The continuous-discrete observer is summarized in Table 3 [16]. The observation process is shown graphically in Figure 8. Note that a fixed sample rate is not required. The continuous-discrete observer can be implemented using Algorithm 1 which is listed below.

System model:
$\dot{x} = Ax + Bu$
$y(t_k) = Cx(t_k)$
Initial Condition $x(0)$.
Assumptions:
Knowledge of A, B, C, $u(t)$.
No measurement noise.
In between measurements ($t \in [t_{k-1}, t_k)$)**:**
Propagate $\dot{\hat{x}} = A\hat{x} + Bu$.
Initial condition is $\hat{x}^+(t_{k-1})$.
Label the estimate at time t_k as $\hat{x}^-(t_k)$.
At sensor measurement ($t = t_k$)**:**
$\hat{x}^+(t_k) = \hat{x}^-(t_k) + L\left(y(t_k) - C\hat{x}^-(t_k)\right)$.

Table 3. Continuous-discrete observer for linear time-invariant systems.

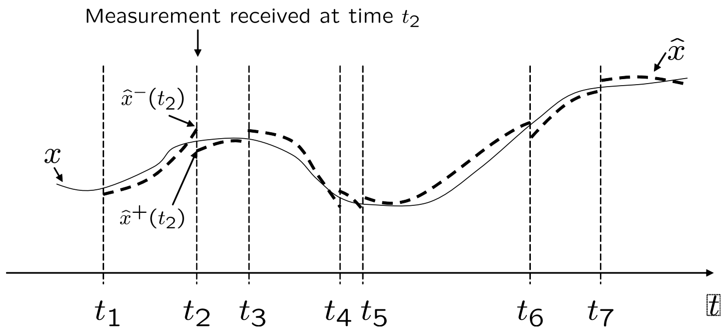


Fig. 8. This figure shows qualitatively the evolution of the state estimate. The solid line represents the actual state variable and the dashed line represents the state estimate. Measurements are received at discrete times denoted by t_i. Between measurements, the state estimate is computed by propagating the state model. At the measurements, the estimate is updated via a weighted average of the current estimate and the measurement.

5.2 Essentials from Probability Theory

Let $X = (x_1, \ldots, x_n)^T$ be a vector whose elements are random variables. The mean, or expected value of X is denoted by

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_n \end{pmatrix} = \begin{pmatrix} E\{x_1\} \\ \vdots \\ E\{x_n\} \end{pmatrix} = E\{X\},$$

where

$$E\{x_i\} = \int \xi f_i(\xi)\, d\xi,$$

Algorithm 1 Continuous-Discrete Observer

```
1: Initialize: $\hat{x} = 0$.
2: Pick an output sample rate $T_{out}$ which is much less than the sample rates of the
   sensors.
3: At each sample time $T_{out}$:
4: for $i = 1$ to $N$ do {Propagate the state equation.}
5:     $\hat{x} = \hat{x} + \left(\frac{T_{out}}{N}\right)(A\hat{x} + Bu)$
6: end for
7: if A measurement has been received from sensor $i$ then {Measurement Update}
8:     $\hat{x} = \hat{x} + L_i\,(y_i - C_i\hat{x})$
9: end if
```

and $f(\cdot)$ is the probability density function for x_i. Given any pair of components x_i and x_j of X, we denote their covariance as

$$cov(x_i, x_j) = \Sigma_{ij} = E\{(x_i - \mu_i)(x_j - \mu_j)\}.$$

The covariance of any component with itself is the variance, i.e.,

$$var(x_i) = cov(x_i, x_i) = \Sigma_{ii} = E\{(x_i - \mu_i)(x_i - \mu_i)\}.$$

The standard deviation of x_i is the square root of the variance:

$$stdev(x_i) = \sigma_i = \sqrt{\Sigma_{ii}}.$$

The covariances associated with a random vector X can be grouped into a matrix known as the covariance matrix:

$$\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} & \cdots & \Sigma_{1n} \\ \Sigma_{21} & \Sigma_{22} & \cdots & \Sigma_{2n} \\ \vdots & & \ddots & \vdots \\ \Sigma_{n1} & \Sigma_{n2} & \cdots & \Sigma_{nn} \end{pmatrix} = E\{(X - \boldsymbol{\mu})(X - \boldsymbol{\mu})^T\} = E\{XX^T\} - \boldsymbol{\mu}\boldsymbol{\mu}^T.$$

Note that $\Sigma = \Sigma^T$ so that Σ is both symmetric and positive semi-definite, which implies that its eigenvalues are real and nonnegative.

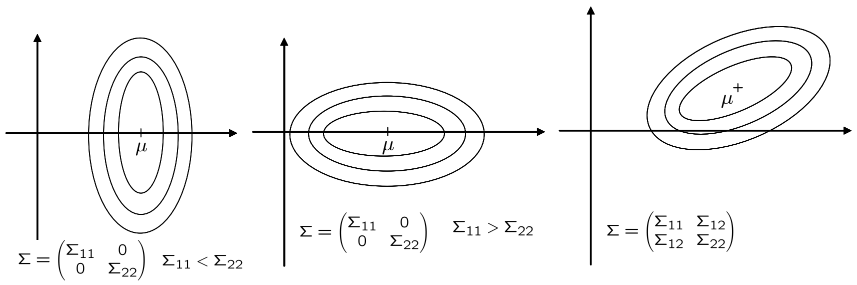
The probability density function for a Gaussian random vector is given by

$$f_X(X) = \frac{1}{\sqrt{2\pi \det \Sigma}} \exp\left[-\frac{1}{2}(X - \boldsymbol{\mu})^T \Sigma^{-1} (X - \boldsymbol{\mu})\right],$$

in which case we write

$$X \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma),$$

and say that X is normally distributed with mean $\boldsymbol{\mu}$ and covariance Σ. Figure 9 shows the level curves for a 2D Gaussian random variable with different covariance matrices.



μ
$\Sigma = \begin{pmatrix}\Sigma_{11} & 0 \\ 0 & \Sigma_{22}\end{pmatrix}$ $\Sigma_{11} < \Sigma_{22}$
μ
$\Sigma = \begin{pmatrix}\Sigma_{11} & 0 \\ 0 & \Sigma_{22}\end{pmatrix}$ $\Sigma_{11} > \Sigma_{22}$
μ
$\Sigma = \begin{pmatrix}\Sigma_{11} & \Sigma_{12} \\ \Sigma_{12} & \Sigma_{22}\end{pmatrix}$

Fig. 9. Level curves for the pdf of a 2D Gaussian random variable. On the left is the pdf when the covariance matrix is diagonal with $\Sigma_{11} < \Sigma_{22}$. In the middle is a pdf when $\Sigma_{22} < \Sigma_{11}$. On the right is a pdf for general $\Sigma = \Sigma^T > 0$. The eigenvalues and eigenvectors of Σ define the major and minor axes of the level curves of the pdf.

5.3 Continuous-Discrete Kalman Filter

In this section we assume the following state model:

$$\begin{aligned} \dot{x} &= Ax + Bu + G\xi \\ y_k &= Cx_k + \eta_k, \end{aligned} \tag{24}$$

where $y_k = y(t_k)$ is the k^{th} sample of y, $x_k = x(t_k)$ is the k^{th} sample of x, η_k is the measurement noise at time t_k, ξ is a zero-mean Gaussian process with covariance Q, and η_k is a zero-mean Gaussian random variable with covariance R. Note that the sample rate does not need to be be fixed. The covariance R can usually be estimated from sensor calibration, but the covariance Q is generally unknown and therefore becomes a system gain that can be tuned to improve the performance of the observer.

We will use the observer given by Eqs. (22) and (23). Define the estimation error as $\tilde{x} = x - \hat{x}$. The covariance of the estimation error is given by

$$P(t) = E\{\tilde{x}(t)\tilde{x}(t)^T\}.$$

Note that $P(t)$ is symmetric and positive semi-definite, therefore its eigenvalues are real and non-negative. Also small eigenvalues of $P(t)$ imply small variance, which implies low average estimation error. Therefore, we would like to choose L to minimize the eigenvalues of $P(t)$. Recall that

$$tr(P) = \sum_{i=1}^{n} \lambda_i,$$

where $tr(P)$ is the trace of P and λ_i are the eigenvalues. Therefore, minimizing $tr(P)$ minimizes the estimation error covariance. Our objective is to pick the estimation gain L in Table 3 to minimize $\text{tr}(P(t))$.

Between Measurements.

Differentiating $\tilde{x}$ we get

$$\begin{aligned}\dot{\tilde{x}} &= \dot{x} - \dot{\hat{x}} \\ &= Ax + Bu + G\xi - A\hat{x} - Bu \\ &= A\tilde{x} + G\xi,\end{aligned}$$

which implies that

$$\tilde{x}(t) = e^{At}\tilde{x}_0 + \int_0^t e^{A(t-\tau)}G\xi(\tau)\,d\tau.$$

We can compute the evolution for P as

$$\begin{aligned}\dot{P} &= \frac{d}{dt}E\{\tilde{x}\tilde{x}^T\} \\ &= E\{\dot{\tilde{x}}\tilde{x}^T + \tilde{x}\dot{\tilde{x}}^T\} \\ &= E\left\{A\tilde{x}\tilde{x}^T + G\xi\tilde{x}^T + \tilde{x}\tilde{x}^T A^T + \tilde{x}\xi^T G^T\right\} \\ &= AP + PA^T + GE\{\xi\tilde{x}^T\}^T + E\{\tilde{x}\xi^T\}G^T,\end{aligned}$$

where

$$\begin{aligned}E\{\xi\tilde{x}^T\} &= E\left\{\xi(t)\tilde{x}_0 e^{A^T t} + \int_0^t \xi(t)\xi^T(\tau)G^T e^{A^T(t-\tau)}\,d\tau\right\} \\ &= \frac{1}{2}QG^T,\end{aligned}$$

which implies that
$$\dot{P} = AP + PA^T + GQG^T.$$

At Measurements.
At a measurement we have that

$$\begin{aligned}\tilde{x}^+ &= x - \hat{x}^+ \\ &= x - \hat{x}^- - L\left(Cx + \eta - C\hat{x}^-\right) \\ &= \tilde{x}^- - LC\tilde{x}^- - L\eta.\end{aligned}$$

Therefore

$$\begin{aligned}P^+ &= E\{\tilde{x}^+\tilde{x}^{+T}\} \\ &= E\left\{\left(\tilde{x}^- - LC\tilde{x}^- - L\eta\right)\left(\tilde{x}^- - LC\tilde{x}^- - L\eta\right)^T\right\} \\ &= E\left\{\tilde{x}^-\tilde{x}^{-T} - \tilde{x}^-\tilde{x}^{-T}C^T L^T - \tilde{x}^-\eta^T L^T\right. \\ &\qquad - LC\tilde{x}^-\tilde{x}^{-T} + LC\tilde{x}^-\tilde{x}^{-T}C^T L^T + LC\tilde{x}^-\eta^T L^T \\ &= \qquad \left. -L\eta\tilde{x}^{-T} + L\eta\tilde{x}^{-T}C^T L^T + L\eta\eta^T L^T\right\} \\ &= P^- - P^- C^T L^T - LCP^- + LCP^- C^T L^T + LRL^T. \qquad (25)\end{aligned}$$

Our objective is to pick L to minimize $\mathrm{tr}(P^+)$. A necessary condition is

$$\begin{aligned}
\frac{\partial}{\partial L}\mathrm{tr}(P^+) &= -P^-C^T - P^-C^T + 2LCP^-C^T + 2LR = 0 \\
&\Longrightarrow 2L(R + CP^-C^T) = 2P^-C^T \\
&\Longrightarrow L = P^-C^T(R + CP^-C^T)^{-1}.
\end{aligned}$$

Plugging back into Eq. (25) give

$$\begin{aligned}
P^+ &= P^- + P^-C^T(R + CP^-C^T)^{-1}CP^- - P^-C^T(R + CP^-C^T)^{-1}CP^- \\
&\quad + P^-C^T(R + CP^-C^T)^{-1}(CP^-C^T + R)(R + CP^-C^T)^{-1}CP^- \\
&= P^- - P^-C^T(R + CP^-C^T)^{-1}CP^- \\
&= (I - P^-C^T(R + CP^-C^T)^{-1}C)P^- \\
&= (I - LC)P^-.
\end{aligned}$$

Extended Kalman Filter.

If instead of the linear state model given in (24), the system is nonlinear, i.e.,

$$\begin{aligned}
\dot{x} &= f(x,u) + G\xi \\
y_k &= h(x_k) + \eta_k,
\end{aligned} \tag{26}$$

then the system matrices A and C required in the update of the error covariance P are computed as

$$\begin{aligned}
A(x) &= \frac{\partial f}{\partial x}(x) \\
C(x) &= \frac{\partial h}{\partial x}(x).
\end{aligned}$$

The extended Kalman filter (EKF) for continuous-discrete systems is given by Algorithm 2.

6 Application of the EKF to UAV State Estimation

In this section we will use the continuous-discrete extended Kalman filter to improve estimates of roll and pitch (Section 6.1) and position and course (Section 6.2).

6.1 Roll and Pitch Estimation

From Eq. 3, the equations of motion for ϕ and θ are given by

Algorithm 2 Continuous-Discrete Extended Kalman Filter

```
1: Initialize: $\hat{x} = 0$.
2: Pick an output sample rate $T_{out}$ which is much less than the sample rates of the
   sensors.
3: At each sample time $T_{out}$:
4: for $i = 1$ to $N$ do {Propagate the equations.}
5:     $\hat{x} = \hat{x} + \left(\frac{T_{out}}{N}\right) f(\hat{x}, u)$
6:     $A = \frac{\partial f}{\partial x}(\hat{x})$
7:     $P = P + \left(\frac{T_{out}}{N}\right)\left(AP + PA^T + GQG^T\right)$
8: end for
9: if A measurement has been received from sensor $i$ then {Measurement Update}
10:    $C_i = \frac{\partial h_i}{\partial x}(\hat{x})$
11:    $L_i = PC_i^T(R_i + C_iPC_i^T)^{-1}$
12:    $P = (I - L_iC_i)P$
13:    $\hat{x} = \hat{x} + L_i\,(y_i - C_i\hat{x})$.
14: end if
```

$$\begin{aligned}\dot{\phi} &= p + q\sin\phi\tan\theta + r\cos\phi\tan\theta + \xi_\phi \\ \dot{\theta} &= q\cos\phi - r\sin\phi + \xi_\theta,\end{aligned}$$

where we have added the noise terms $\xi_\phi \sim \mathcal{N}(0, Q_\phi)$ and $\xi_\theta \sim \mathcal{N}(0, Q_\theta)$ to model the sensor noise on p, q, and r. We will use the accelerometers as the output equations. From Eq. (7), the output of the accelerometers is given by

$$y_{\text{accel}} = \begin{pmatrix} \frac{\dot{u}+gw-rv}{g} + \sin\theta \\ \frac{\dot{v}+ru-pw}{g} - \cos\theta\sin\phi \\ \frac{\dot{w}+pv-qu}{g} - \cos\theta\cos\phi \end{pmatrix} + \eta_{\text{accel}}. \tag{27}$$

However since we do not have a method for directly measuring $\dot{u}$, $\dot{v}$, $\dot{w}$, u, v, and w, we will assume that $\dot{u} = \dot{v} = \dot{w} \approx 0$ and we will use Eq. (1) and assume that $\alpha \approx \theta$ and $\beta \approx 0$ to obtain

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} \approx V_a \begin{pmatrix} \cos\theta \\ 0 \\ \sin\theta \end{pmatrix}.$$

Substituting into Eq. (27) gives

$$y_{\text{accel}} = \begin{pmatrix} \frac{qV_a\sin\theta}{g} + \sin\theta \\ \frac{rV_a\cos\theta - pV_a\sin\theta}{g} - \cos\theta\sin\phi \\ \frac{-qV_a\cos\theta}{g} - \cos\theta\cos\phi \end{pmatrix} + \eta_{\text{accel}}.$$

Letting $x = (\phi, \theta)^T$, $u = (p, q, r, V_a)^T$, $\xi = (\xi_\phi, \xi_\theta)^T$, and $\eta = (\eta_\phi, \eta_\theta)^T$, we get the nonlinear state equation

$$\begin{aligned}\dot{x} &= f(x, u) + \xi \\ y &= h(x, u) + \eta,\end{aligned}$$

where

$$f(x,u) = \begin{pmatrix} p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \\ q\cos\phi - r\sin\phi \end{pmatrix}$$

$$h(x,u) = \begin{pmatrix} \frac{qV_a\sin\theta}{g} + \sin\theta \\ \frac{rV_a\cos\theta - pV_a\sin\theta}{g} - \cos\theta\sin\phi \\ \frac{-qV_a\cos\theta}{g} - \cos\theta\cos\phi \end{pmatrix}.$$

Implementation of the extended Kalman filter requires the Jacobians

$$\frac{\partial f}{\partial x} = \begin{pmatrix} q\cos\phi\tan\theta - r\sin\phi\tan\theta & \frac{q\sin\phi - r\cos\phi}{\cos^2\theta} \\ -q\sin\phi - r\cos\phi & 0 \end{pmatrix}$$

$$\frac{\partial h}{\partial x} = \begin{pmatrix} 0 & \frac{qV_a}{g}\cos\theta + \cos\theta \\ -\cos\phi\cos\theta & -\frac{rV_a}{g}\sin\theta - \frac{pV_a}{g}\cos\theta + \sin\phi\sin\theta \\ \sin\phi\cos\theta & \left(\frac{qV_a}{g} + \cos\phi\right)\sin\theta \end{pmatrix}.$$

The state estimation algorithm is given by Algorithm 2.

Figure 10 shows the actual and estimated roll and pitch attitudes obtained by using this scheme, where we note significant improvement over the results shown in Figures 6 and 7. The estimates are still not precise due to the approximation that $\dot{u} = \dot{v} = \dot{w} = \beta = \theta - \alpha = 0$. However, the results are adequate enough to enable non-aggressive MAV maneuvers.

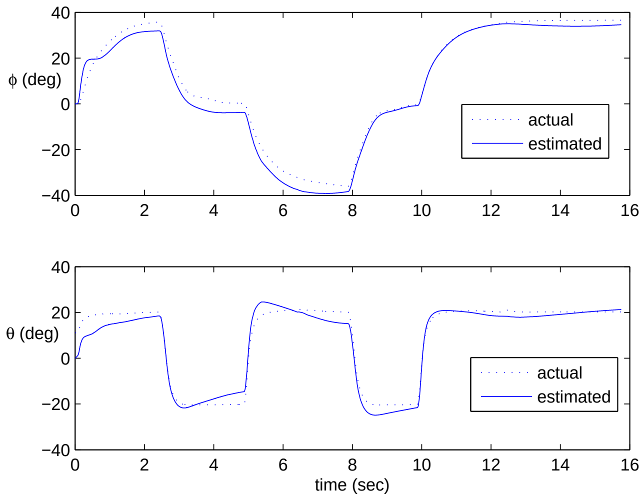


Fig. 10. Actual and estimated values of ϕ and θ using the continuous-discrete extended Kalman filter.

6.2 Position and Course Estimation

The objective in this section is to estimate p_n, p_e, and χ using the GPS sensor. From Eq. (3), the model for χ is given by

$$\dot{\chi} = \dot{\psi} = q\frac{\sin\phi}{\cos\theta} + r\frac{\cos\phi}{\cos\theta}.$$

Using Eqs. (4) and (5) for the evolution of p_n and p_e results in the system model

$$\begin{aligned}\begin{pmatrix}\dot{p}_N \\ \dot{p}_E \\ \dot{\chi}\end{pmatrix} &= \begin{pmatrix} V_g \cos\chi \\ V_g \sin\chi \\ q\frac{\sin\phi}{\cos\theta} + r\frac{\cos\phi}{\cos\theta}\end{pmatrix} + \xi_p \\ &\overset{\triangle}{=} f(x,u) + \xi_p,\end{aligned}$$

where $x = (p_n, p_e, \chi)^T$, $u = (V_g, q, r, \phi, \theta)^T$ and $\xi_p \sim \mathcal{N}(0, Q)$.

GPS returns measurements of p_n, p_e, and χ directly. Therefore we will assume the output model

$$y_{\text{GPS}} = \begin{pmatrix} p_n \\ p_e \\ \chi \end{pmatrix} + \eta_p,$$

where $\eta_p \sim \mathcal{N}(0, R)$ and $C = I$, and where we have ignored the GPS bias terms. To implement the extended Kalman filter in Algorithm 2 we need the Jacobian of f which can be calculated as

$$\frac{\partial f}{\partial x} = \begin{pmatrix} 0 & 0 & -V_g \sin\chi \\ 0 & 0 & V_g \cos\chi \\ 0 & 0 & 0 \end{pmatrix}.$$

Figure 10 shows the actual and estimated values for p_n, p_e, and χ obtained by using this scheme. The inaccuracy in the estimates of p_n and p_e is due to the GPS bias terms that have been neglected in the system model. Again, these results are sufficient to enable non-aggressive maneuvers.

7 Summary

Micro air vehicles are increasingly important in both military and civil applications. The design of intelligent vehicle control software pre-supposes accurate state estimation techniques. However, the limited computational resources on board the MAV require computationally simple, yet effective, state estimation algorithms. In this chapter we have derived mathematical models for the sensors commonly deployed on MAVs. We have also proposed simple state estimation techniques that have been successfully used in thousands of hours of actual flight tests using the Procerus Kestrel autopilot (see for example [7, 6, 21, 10, 17, 18]).

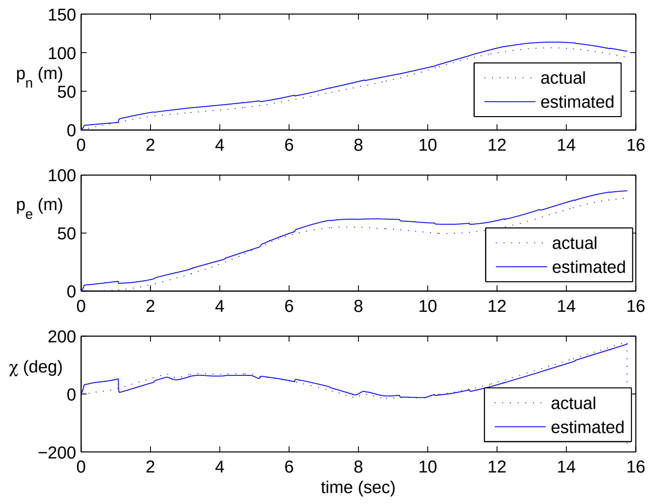


Fig. 11. Actual and estimated values of p_n, p_e, and χ using the continuous-discrete extended Kalman filter.

Acknowledgments

This work was partially supported under grants AFOSR grants FA9550-04-1-0209 and FA9550-04-C-0032 and by NSF award no. CCF-0428004.

References

1. `http://www.silicondesigns.com/tech.html`.
2. Cloudcap technology. `http://www.cloudcaptech.com`.
3. Micropilot. `http://www.micropilot.com/`.
4. Procerus technologies. `http://procerusuav.com/`.
5. Brian D. O. Anderson and John B. Moore. *Optimal Control: Linear Quadratic Methods*. Prentice Hall, Englewood Cliffs, New Jersey, 1990.
6. D. Blake Barber, Stephen R. Griffiths, Timothy W. McLain, and Randal W. Beard. Autonomous landing of miniature aerial vehicles. In *AIAA Infotech@Aerospace*, Arlington, Virginia, September 2005. American Institute of Aeronautics and Astronautics. AIAA-2005-6949.
7. Randal Beard, Derek Kingston, Morgan Quigley, Deryl Snyder, Reed Christiansen, Walt Johnson, Timothy McLain, and Mike Goodrich. Autonomous vehicle technologies for small fixed wing UAVs. *AIAA Journal of Aerospace, Computing, Information, and Communication*, 2(1):92–108, January 2005.
8. Robert E. Bicking. Fundamentals of pressure sensor technology. `http://www.sensorsmag.com/articles/1198/fun1198/main.shtml`.
9. Crossbow. Theory of operation of angular rate sensors. `http://www.xbow.com/Support/Support_pdf_files/RateSensorAppNote.pdf`.

10. Stephen Griffiths, Jeff Saunders, Andrew Curtis, Tim McLain, and Randy Beard. Obstacle and terrain avoidance for miniature aerial vehicles. *IEEE Robotics and Automation Magazine*, 13(3):34–43, 2006.
11. David Halliday and Robert Resnick. *Fundamentals of Physics*. John Wiley & Sons, 3rd edition, 1988.
12. Andrew H. Jazwinski. *Stochastic Processes and Filtering Theory*, volume 64 of *Mathematics in Science and Engineering*. Academic Press, Inc., New York, New York, 1970.
13. R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions ASME Journal of Basic Engineering*, 82:34–35, 1960.
14. R. E. Kalman and R. S. Bucy. New results in linear filtering and prediction theory. *Transaction of the ASME, Journal of Basic Engineering*, 83:95–108, 1961.
15. Robert P. Leland. Lyapunov based adaptive control of a MEMS gyroscope. In *Proceedings of the American Control Conference*, pages 3765–3770, Anchorage, Alaska, May 2002.
16. Frank L. Lewis. *Optimal Estimation: With an Introduction to Stochastic Control Theory*. John Wiley & Sons, New York, New York, 1986.
17. Timothy W. McLain and Randal W. Beard. Unmanned air vehicle testbed for cooperative control experiments. In *American Control Conference*, pages 5327–5331, Boston, MA, June 2004.
18. Derek R. Nelson, D. Blake Barber, Timothy W. McLain, and Randal W. Beard. Vector field path following for miniature air vehicles. *IEEE Transactions on Robotics*, in press.
19. Marc Rauw. *FDC 1.2 - A SIMULINK Toolbox for Flight Dynamics and Control Analysis*, February 1998. Available at `http://www.mathworks.com/`.
20. Wilson J. Rugh. *Linear System Theory*. Prentice Hall, Englewood Cliffs, New Jersey, 2nd edition, 1996.
21. Jeffery B. Saunders, Brandon Call, Andrew Curtis, Randal W. Beard, and Timothy W. McLain. Static and dynamic obstacle avoidance in miniature air vehicles. In *AIAA Infotech@Aerospace*, number AIAA-2005-6950, Arlington, Virginia, September 2005. American Institute of Aeronautics and Astronautics.
22. Brian L. Stevens and Frank L. Lewis. *Aircraft Control and Simulation*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2nd edition, 2003.
23. Navid Yazdi, Farrokh Ayazi, and Khalil Najafi. Micromachined inertial sensors. *Proceedings of the IEEE*, 86(8):1640–1659, August 1998.