HELP FOR STATE ESTIMATION ASSIGNMENT

DR. BHAN

The state estimation assignment is meant to teach material that is discussed briefly and cursorily in other courses. I hope that by completing it, you gain an edge over students from other universities. First, I discuss the sensors that are available on our UAV, and the equations that relate them to the UAV's motion. Then, I provide an example of using one of these models to construct an estimator. Lastly, I briefly describe sensor fusion.

1. Sensors

There are several sensors available that can be fused together to form an effective state-estimator. This guide discusses a few of the sensors–accelerometers, gyros, optical flow, and ultrasonic. A basic understanding of these sensors is critical to all state-estimation techniques on this UAV platform.

1.1. **IMU accelerations and rates.** An inertial measurement unit provides measurements of the UAV's inertial acceleration and angular velocity. The IMU combines a measurement of the vehicle's acceleration and the vehicle's angular velocity. Recall from the equations of motion that the sensor for applied specific force, a, measures the quantity

$$a = \frac{1}{m}\left(\vec{F}_a + \vec{F}_t\right),$$

where m is the vehicle's mass, $\vec{F}_a$ is the vehicle's aerodynamic forces, and F_t is the vehicle's propulsive forces. These forces relate to the equations of motion in body-axis as follows:

$$\dot{V} = -\omega \times V + \frac{\vec{F}_G\left(\phi, \theta, \psi\right)}{m} + a, \tag{1.1}$$

where V is the velocity, ω is the angular velocity, and $\vec{F}_G(\phi, \theta, \psi)$ is the gravity vector (it depends on ϕ, θ, and ψ; roll, pitch, and yaw, respectively). Recall that the force of gravity in components is

$$F_G = mg\left(R(\phi)R(\theta)R(\psi)\begin{bmatrix}0\\0\\1\end{bmatrix}\right) = mg\begin{bmatrix}-\sin(\theta)\\ \cos(\theta)\sin(\phi)\\ \cos(\theta)\cos(\phi)\end{bmatrix}.$$

At equilibrium, the following relation holds for specific force:

$$g\begin{bmatrix}-\sin(\theta)\\ \cos(\theta)\sin(\phi)\\ \cos(\theta)\cos(\phi)\end{bmatrix} = \hat{a}.$$

From this simplified relationship, we can back-calculate ϕ, θ, ψ by manipulating components of a. The estimate for θ and ϕ is:

$$\tan\phi = \frac{a_y}{a_z},$$

$$\sqrt{a_y^2 + a_z^2} = \sqrt{(\cos\theta\cos\phi)^2 + (\cos\theta\sin\phi)^2} = \cos\theta \quad \Rightarrow \quad \tan\theta = \frac{-a_x}{\sqrt{a_y^2 + a_z^2}}.$$

In this paradigm, roll and pitch are directly measured by trig relationships of the accelerometer measurements. These relationships for ϕ and θ fail on a digital computers because tangent or arctangent possess singularities over their support, and the possibility to divide by zero exists so we must approximate or limit these calculations. We can use a small-angle approximation to simplify them,

$$\cos\theta \approx 1, \quad \sin\theta \approx \theta.$$

These measurement equations are useful when the vehicle is nearly at equillibrium (i.e. in a hover). They are not accurate in accelerating flight.

The process model for these equations relate roll, pitch, and yaw relate to measured angular rates, p, q, r by

$$\begin{aligned} \dot{\phi} &= p + \tan\theta\left(q\sin\phi + r\cos\phi\right), \\ \dot{\theta} &= q\cos\phi - r\sin\phi, \\ \dot{\psi} &= \frac{q\sin\phi + r\cos\phi}{\cos\theta}, \\ J\dot{\omega} &= -\omega \times J\omega + \mathbb{M}. \end{aligned}$$

Note that the equations use $\mathbb{M}$, short-hand for the sum of all applied torques. They neglect conservation of angular momentum for the vehicle's rotors. These equations combine with the accelerator measurement equations to determine a model for the vehicle's attitudes. We further assumed that J was a diagonal moment of inertia tensor. Linearizing and decoupling these non-linear equations helps derive simple linear models that behave like double or triple integrators.

1.2. **Optical Flow.** The optical flow algorithm provides a measurement of the speeds u and v. This algorithm on the Parrot does not support estimates of position. Returning to the force state equations but focusing on the first two equations in this vector for u and v, acceleration appears as an input to

$$\begin{bmatrix} \dot{u} \\ v \\ w \end{bmatrix} = -\omega \times V + \frac{\vec{F}_G\left(\phi, \theta, \psi\right)}{m} + a. \tag{1.2}$$

Treating accelerometers as inputs to these equations allows us to correct estimates of w with information sensed from u and v from the optical flow algorithm.

Note that to hover, it is not strictly necessary to use optical flow, however, it does help with drift. You can close the loop on ϕ/p and θ/q, and then use optical flow to eliminate drift by adding feedback on u/v. Optical flow depends on height! Your optical flow sensor may not be super accurate as well, so precise tracking of speed is unrealistic.

Note the relationship for optical flow is approximately as follows:

$$\begin{bmatrix} u \\ v \end{bmatrix} \approx \kappa h \begin{bmatrix} u_{of} \\ v_{of} \end{bmatrix},$$

where h is altitude, u_{of} is and v_{of} is the raw optical flow data, and κ is a fudge factor (80% to 120%). Note that this formula depends on the focal length of the camera and is an approximation. You could actually determine the fudge factor by just measuring out a fixed distance, moving your UAV over the fixed distance with a timer, and measuring the value registered on the drone. Additionally, optical flow is actually affected by angular velocity so it can be corrupted when the drone is rapidly rotating.

1.3. **Ultrasonic Sensor.** To estimate the vertical channel we can combine the $\dot{w}$ equation with the equation for $\dot{Z}_{ned}$, which is the third equation in

$$R^T(\psi)R^T(\theta)R^T(\phi)\begin{bmatrix}u\\v\\w\end{bmatrix}=\begin{bmatrix}\dot{X}\\\dot{Y}\\\dot{Z}\end{bmatrix}_{NED}.$$

In reality, the range inferred from the ultrasonic sensor depends on θ and ϕ, and hence, one could in principle use it to infer information about these angles. Alternatively, knowledge of the tilt of the UAV would give more accurate measurements of the height above the ground. The ultrasonic sensor gives us a method of estimating w and z when combined with (1.2). Since the optical flow does not provide a position measurement for X and Y, without connecting to the optiTrack system, there is no way to my knowledge to infer X and Y position without image processing from the camera. I would consider that beyond the scope of this class, unless you wanted to attack this for your project.

1.4. **Pressure Sensor.** You can estimate the altitude from the barometric pressure sensor. A matlab live script in Canvas details how to use the formulas (hypsometric formula and the international barometric pressure formula). Note that the formulas will be referenced to sea level. To calculate the height above the ground, you will need to:

(1) Calculate the initial altitude using the initial pressure from the sensor calibration signals (when the UAV is on the floor);
(2) Subtract that value from the formula's application to the current pressure.

2. Example Architecture for the Kalman Filter

In this section we discuss how we can synthesize some models to provide an estimate of our UAV's states. Suppose we use a model for the vertical channel of the UAV. This model is the linear version of the nonlinear equations of motion. I selected a small, decoupled subset of the UAV's states. The continuous time equations for the UAV's vertical channel depends on the following terms:

Vertical distance (+ down): z;
Downward speed: w;
Normal acceleration: a_n;
Process bias: a_0;
Process noise: n_w, w_a;
Height above ground from ultrasonic Sensor: h_{US};
Measurement bias: h_0;
Measurement noise: v_h.

These terms populate the state-estimator's model,

$$\dot{x}=\begin{bmatrix}\dot{z}\\\dot{w}\end{bmatrix}=\begin{bmatrix}0&1\\0&0\end{bmatrix}\begin{bmatrix}z\\w\end{bmatrix}+\begin{bmatrix}0\\1\end{bmatrix}u=Ax+Bu,$$

$$z=-\left(h_{US}-h_0+v_h\right).$$

The accelerometer measurements pass into the Kalman filter's process model as if they were inputs [1]

$$u = a_n - a_0 + n_a.$$

It is easier to implement discrete equations on a digital computer. We can determine the exact digital form of these equations by using the state-transition matrix,

$$e^{At} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}.$$

Instead of digitalizing the noise with a formal process, we will take some measurements and calculate its covariance. We assume that u_k holds constant over times $\sigma \in [t, t+\Delta t]$, and then apply the variation of constants formula:

$$x(t+\Delta t) = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} x(t) + \int_0^{\Delta t} \begin{bmatrix} 1 & \sigma \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathrm{d}\sigma u_k$$

The new state space system has the following form

$$\begin{aligned} x_{k+1} &= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} (a_k - a_0) + \int_{t_t}^{t_{k+1}} e^{A(t_{k+1}-\sigma)} B n_a \mathrm{d}\sigma \\ y_k &= \begin{bmatrix} -1 & 0 \end{bmatrix} x_k + h_0 + v_h. \end{aligned}$$

Zero-mean, white noise n_a injected into the process creates a complicated integral of the white noise which we called a Wiener process. Let's call the Wiener process w_a. We used the notation w for vertical speed and also process noise in this course (another w, unfortunately).

2.1. Time-Varying Kalman Filter Model.

We now have the model in the standard form for a Kalman filter:

$$\begin{aligned} \begin{bmatrix} z_{k+1} \\ w_{k+1} \end{bmatrix} &= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} (a_k - a_0) + w_a \\ y_k &= \begin{bmatrix} -1 & 0 \end{bmatrix} \begin{bmatrix} z_k \\ w_k \end{bmatrix} + h_0 + v_h. \end{aligned}$$

We assume that the altitude measurement noise v_h and the accelerometer process noise w_a do not relate to each other or the process, i.e. statistical independence. We assume that w_a and v_h are zero-mean, discrete random sequences with constant covariances,

$$\mathbb{E} w_a w_a^T = Q_d, \quad \mathbb{E} v_h v_h^T = R_d.$$

For example, we could take some measurements of the accelerometer and ultrasonic sensor while holding the UAV still and calculate these measurements' standard deviations, σ_a for the accelerometer and σ_h for the ultrasonic sensor. This gives us a starting point to select Q and R. Recall that the Q matrix is the covariance of the process noise, and it is affected by the discretization process.

A good rule of thumb for small sampling times (as is the case in our class) is $Q_d = Q_c \Delta t$, where Q_d is the covariance matrix for the discrete-time model, Q is the original covariance matrix for the continuous-time model, and $\delta T$is the sampling interval. For example, for the altitude axis state estimator, we use the following covariances for the noise:

$$\begin{aligned} Q_d &= cov(a_z) * \begin{bmatrix} \frac{1}{3}\Delta t^3 & \frac{1}{2}\Delta t^2 \\ \frac{1}{2}\Delta t^2 & \Delta t \end{bmatrix} \approx \begin{bmatrix} 0 & 0 \\ 0 & cov(a_z)\Delta t \end{bmatrix}, \\ R_d &= cov(z). \end{aligned}$$

3. Sensor Fusion

Suppose I have two measurements for the same signal. How can one blend them? We already have learned all the theory for this! Use sensor fusion in your Kalman filter. If we have a redundant measurement, we need to modify the C matrix and the R matrix in the Kalman filter. For example, in our 2-state Kalman filter, suppose the first state is z and the 2nd state is w. If we have two measurements for z, z_1 and z_2 then the new C matrix is defined by

$$y = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = Cx = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} z \\ w \end{bmatrix}$$

Now the new R matrix is a 2x2 matrix (since there are now 2 measurements), and the diagonal elements will be

$$R = cov(z_1, z_2) = \begin{bmatrix} var(z_1) & 0 \\ 0 & var(z_2) \end{bmatrix}$$

Lastly, we can remove any biases in the two measurements, and now we have blended two measurements.

References

[1] P. Kabamba, A. Girard, *Fundamentals of Aerospace Navigation and Guidance*. Cambridge: Cambridge University Press, 2014. doi:10.1017/CBO9781107741751

[2] D. Simon, *Optimal State Estimation*, John Wiley & Sons, Inc., Hoboken, New Jersey 2006.