Shamrock Sensor Fusion Modelling

Justin Kleiber

August 2019

State Representation

$$\hat{x} = \begin{bmatrix} X & Y & v & a & \theta \end{bmatrix} \tag{1}$$

X: X position, moving forward increases it, moving backwards decreases it
Y: Y position, left increases it, right decreases it
V: speed
a: acceleration
θ: heading in unit circle coordinate space

Motion Model

Overview

The motion model defines the behavior of the robot when it moves through space. The following equations and derivation leads us to the equation to calculate the robot's state, and the linearized system used to predict the state using the EKF.

General Form

$$\hat{x}_k = f(\hat{x}_{k-1}, \mathbf{u}, \Delta t) + w_k \tag{2}$$

x_k: the new state
x_{k-1}: the last state
$\mathbf{u}$: the control signal (ignored in this model)
Δt: the time step (time between each state estimate)
w_k: the process noise associated with the filter

Supporting Equations

The following equations are simple physics equations that will be used to approximate the robot's location. For the purposes of this filter, everything will be assumed to be in metric units.

Equation for X:

$$X_f = X_i + v_x \Delta t \tag{3}$$

Equation for Y:

$$Y_f = Y_i + v_y \Delta t \tag{4}$$

Equation for velocity:

$$v_f = v_i + a \Delta t \tag{5}$$

Velocity Decomposition:

$$v_x = v * cos(\theta) \tag{6}$$

$$v_y = v * sin(\theta) \tag{7}$$

Model Function

The following equation characterizes the robot's motion through space:

$$f(x, \mathbf{u}, \Delta t) = \begin{bmatrix} 1 & 0 & cos(\theta)\Delta t & 0 & 0 \\ 0 & 1 & sin(\theta)\Delta t & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} * x_{k-1} = \begin{bmatrix} X_{k-1} + vcos(\theta)\Delta t \\ Y_{k-1} + vsin(\theta)\Delta t \\ v + a\Delta t \\ a \\ \theta \end{bmatrix} \tag{8}$$

We need to linearize the system so the EKF will work. I will do this by finding the Jacobian of $f(x, \mathbf{u}, \Delta t)$. Taking the Jacobian of equation 8 (using the sympy python package to speed up the calculation process) yields:

$$F_k = \begin{bmatrix} 1 & 0 & cos(\theta)\Delta t & 0 & -vsin(\theta)\Delta t \\ 0 & 1 & sin(\theta)\Delta t & 0 & vcos(\theta)\Delta t \\ 0 & 0 & 1 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

Measurement Model

The measurement model seeks to correlate a given state with the available sensor values (measurements). A function is used to compute the sensor data, and then noise is added. The following equation demonstrates this.

General Form

$$z_k = h(x_k) + v_k \tag{10}$$

z_k: sensor values
x_k: current state
v_k: measurement noise of the sensors

Sensors

Shamrock has a few sensors to utilize when taking measurements:

- Wheel Encoders
- Accelerometer
- Gyroscope
- Magnetometer

Each of these sensors needs to be represented mathematically. To do so, I use the following vector:

$$z_k = \begin{bmatrix} v \\ a \\ \theta_g \\ \theta_m \end{bmatrix} \tag{11}$$

v: measured encoder velocity
a: measured acceleration
θ_g: gyroscope angle
θ_m: magnetometer angle (referenced from start angle)

Model Function

We need a way to convert a given state to the sensor values. To do this, the function $h(x_k)$ is constructed as follows:

$$h(x_k) = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} * x_k = \begin{bmatrix} v \\ a \\ \theta \\ \theta \end{bmatrix} \tag{12}$$

Like with the motion model, this system needs to be linearized in order for it to work. Luckily, this is taken care of by the matrix shown in equation 12.

$$H_k = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

Extended Kalman Filter Formulation

Overview

The modelling above will help run the EKF. To make an EKF, two phases are needed - the prediction phase and the update phase. First, the prediction phase estimates the state of the robot, and then the update phase uses sensor data to gain feedback on that prediction.

Reading Notes

To this point, we've been dealing with things we know to be true. For example, in the measurement model, a state was used to calculate the sensor values. In the EKF, this is no longer deterministic - predicted values will be used throughout the filter. In order to differentiate things that are predicted from things that are not predicted, a little "hat" will be used over the variable name:

$\hat{x}_k$: predicted state
x_k: actual, real-life state

This is generally consistent with the literature on EKFs out there right now. Since the modelling part is very concrete and the EKF uses similar equations (but with predictions), it is easy to get the two mixed up.

Prediction Phase

The prediction phase is described by two main steps - the state prediction step and the covariance update step. Put simply, the covariance of the state prediction is how uncertain the prediction is.

The Process

1. Predict the next state:
$$\hat{x}_k = f(\hat{x}_{k-1}, \mathbf{u}, \Delta t) \tag{14}$$
2. Recalculate the Jacobian, F_{k-1} using the values from $\hat{x}_{k-1}$
3. Then, update the covariance matrix P:
$$P_k = F_{k-1} P_{k-1} F_{k-1}^T + Q_{k-1} \tag{15}$$

It is important to note that Q is a diagonal square matrix representing the process noise of the motion model (see equation 2). Even a small value for Q is really helpful.

Update Phase

With a predicted state and its uncertainty calculated, the sensor measurements need to be considered. Then, once sensors are integrated, we will weight the uncertainties to find the tradeoff between the motion and measurement models. This tradeoff is called the Kalman Gain.

The Process

1. Calculate the innovation (the amount learned from this update). This is the difference in the actual sensor values and the predicted ones:
$$y_k = z_k - h(\hat{x}_k) \tag{16}$$
Note here that z_k is not predicted. It will be the vector defined in equation 11

2. Calculate the covariance of the innovation (the uncertainty due to measurement noise):

$$S_k = H_k P_{k-1} H_k^T + R_k \tag{17}$$

   Note here that R is a diagonal square matrix representing the measurement noise of the system (see equation 10). Data sheets or experimentation can be used to find the best values of R. In general, the square of each sensors standard standard deviation (called the variance and calculated as σ^2) is used on the corresponding entry.

3. Calculate the Kalman Gain

$$K_k = P_{k-1} H_k^T S_k^{-1} \tag{18}$$

4. Update the predicted state

$$\hat{x}_k = \hat{x}_k + K_k y_k \tag{19}$$

5. Update the estimated covariance

$$P_k = (I - K_k H_k) P_k \tag{20}$$

Running the EKF

When testing the robot, observe the P_k matrix value on a graph to make sure the EKF is converging. If it is not converging, or converging slowly, adjusting Q and R might be necessary.