Kalman Filter: Simple Object Tracking

Benjamin Brown

ECSE 456 - Notes

February 18, 2015

1 Background

A Kalman filter is used to combine continuous predictions from a theoretical system model with continuous measurements from a real implementation of the same system. This is done to help reduce the effects of noise on system measurements, and can provide a prediction of the system's next state if for some reason the measurements fail or contain lots of noise.

Implementation of a Kalman filter is highly intuitive for a sensor that has been characterized with testing. But for object tracking, it is much more difficult to understand as all of the Kalman filter jargon is focused around "sensors" and "measurements" that sound odd when referring to an object tracking system. The goal of these notes is to break down the Kalman filter theory and apply the theory to a single, 2D object tracking system.

The following two sources were used to develop these notes:

- http://en.wikipedia.org/wiki/Kalman_filter
- http://robotsforroboticists.com/kalman-filtering/

2 Kalman Equations

The Kalman filter equations can be divided into two key sets: the predication equations and the update equations.

The Prediction Equations:

$$\vec{x_{k+1}} = F \cdot \vec{x_k} + B \cdot \vec{u_k} \tag{1}$$

$$P_{k+1} = F \cdot P_k \cdot F^T + Q \tag{2}$$

Intermediate Calculations:

$$\vec{y_{k+1}} = \vec{z_k} - H \cdot \vec{x_{k+1}} \tag{3}$$

$$S_{k+1} = H \cdot P_{k+1} \cdot H^T + R \tag{4}$$

$$K_{k+1} = P_{k+1} \cdot H^T \cdot S_{k+1}^{-1} \tag{5}$$

The Update Equations:

$$\hat{\vec{x_{k+1}}} = \vec{x_{k+1}} + K_{k+1} \cdot \vec{y_{k+1}} \tag{6}$$

$$\hat{P_{k+1}} = (I - K_{k+1} \cdot H) \cdot P_{k+1} \tag{7}$$

Here I have chosen to only use subscripts on vectors and matrices that are going to change over time (i.e. dynamic). I have also chosen to break it up into three sets of equations, because the values produced in the intermediate calculations aren't very interesting to a beginner. The recursive calculation is essentially make a predication of the state of the system, take a measurement of the system, and adjust the predication based on the measurement.

3 Variables Explained

The equations present a lot of variables, which can be daunting, but most of them are very simple or arbitrary for a simple object tracking system.

The State & Input Vector:

- $\vec{x}$ (nx1): The state vector containing the state variables of the system. These are the values you want to control and use, like position, velocity, and acceleration.
- $\vec{u}$ (mx1): The controlled inputs to the system that effect the states, for instance force or voltage which effect the system's velocity or acceleration.

Static Variables: These are the values that aren't going to change as time goes on, and you obtain new measurements.

- F (nxn): This is the state transition matrix, sometimes denoted A, and describes how the system states change over a time-step (i.e. from k to $k+1$). The entries of this matrix are derived using physical equations like kinematic equations, Lagrange equations, or Newton's laws. If you have a state space model of your system, or have a transfer function of the system then this matrix is found with ease. Essentially, this matrix describes how your system should act in theory.
- B (nxm): This is the control input model. If you derived the state transition matrix using equations mentioned above, if there are any external inputs being added to the states, this matrix would have the coefficients on those inputs.
- H (nxn): This is the measurement model. If for some reason the space in which you measure data is different than the space which the predictions are in, you use this matrix to make them the same. For instance, if your measurement is in 3D coordinates ordered (y,x,z) and your system model is (x,y,z) this matrix would be a permutation matrix swapping rows. The dimensions might not be nxn, they might be mxm if your some reason you can't measure something (i.e. you just get position and use a derivative to get velocity). For the sake of these notes, I'm going to assume you are able to somehow measure each state.
- Q (nxn): This matrix is a measure of the process error, meaning the error in your theoretical system model. This could be something like friction or air resistance. The matrix is a diagonal (all zeros except along diagonal) with the variance of each state error in the entry. For instance, if your process error is characterized by some vector $\vec{w} = [w_{x_1}, w_{x_2}]$ where each entry is the error for each state variable, then this matrix has the variance of each entry of $\vec{w}$ along the diagonal.
- R (nxn): This is the exact same idea of Q except with your measurements. So think of Q as theoretical error and R as practical error.

4 The Object Tracking Scenario

4.1 Measurement Model

4.2 System Model