CS 440 Machine Problem 5 Report

Deep Patel

December 12, 2018

1 2D Kalman Filter

1.1 Update Equations

Time update

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$$
$$P_k^- = AP_{k-1}A^T + Q$$

Measurement update

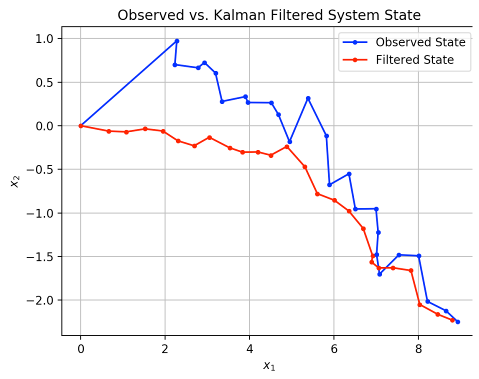
$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$
$$K_k = \frac{P_k^- H^T}{HP_k^- H^T + R}$$
$$P_k = (I - K_k H)P_k^-$$

Figure 1: Update Equations

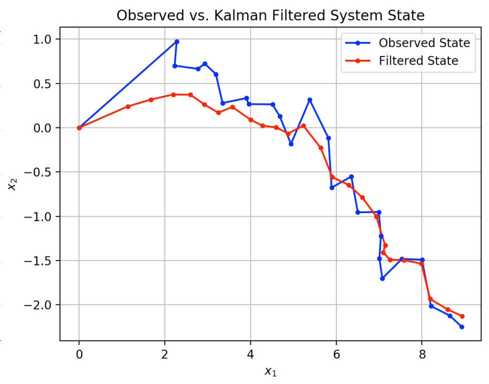
In the equations in Figure 1, you can see the general iterative equations that can be used for any system. In our specific system, the x variable is the 2-dimensional state mean, x_1 and x_2, and P is the system covariance matrix, which we initially set to a tuned guess. The matrices A and B are simply the identity matrices in this system. Q is the system noise given in the problem statement. In the measurement update step, we update out predicted mean that we got in the time update step by adding the difference between the observed state and the predicted state (times the observation model matrix, H) times the Kalman gain, K, which determines how much weight should be put on this observation difference. The Kalman gain is a function of the system covariance matrix, P, which we got during the time update, H which is the observation model matrix, and R which is the observation noise matrix, given in the problem statement. Using these equations we can implement a very basic Kalman filter and get the results shown in the next section.

1.2 Analyzing the Results

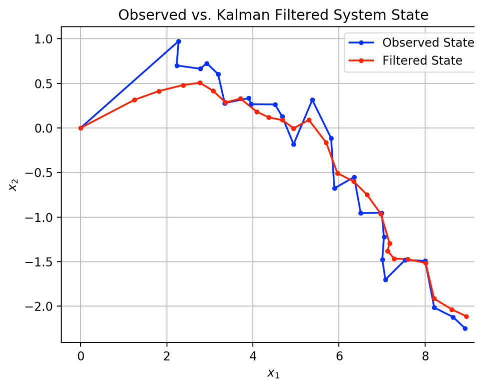
As you can see from Figure 2, the initial value of P makes a significant impact on the initial accuracy for the Kalman filter. It seems that setting it to values around 1 give good results. This is because if you set it too low in



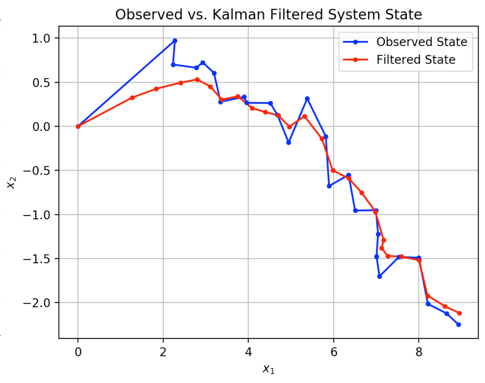
(a) P = I*0.01



(b) P = I*0.1



(c) P = I*1



(d) P = I*10

Figure 2: Observed vs. Filtered System State with different starting values of P, where I is the 2D identity matrix

the beginning, the Kalman filter initially is very confident in its state, although in reality, it should be more uncertain so that it can correct for itself through updating.

1.3 Kalman Filter for Shooting Game State Estimation

For the game, I implemented the same Kalman filter used in the first part, but using global variables to store the data from previous iterations. I took the expected x and y coordinates from the output of the Kalman filter, and decides to shoot based on two criteria. I recorded the x value from the P covariance matrix before and after the Kalman update and checked for a convergence in that, namely by taking the difference of the before and after and checked if it was smaller than some epsilon, as this shows that we might not be getting more confident in our prediction over time, so you might as well shoot now, than wait. The other criteria a threshold check for the norm of the covariance matrix, so if it small enough, then we are confident. I also tried to use the difference between the mean time updated state and the observation, and check that it was small enough for that particular step, but after implementing it, it did not seem to yield any benefits in terms of results, so I removed that.