Rajul Ramchandani & Conan Zhang

Professor Thomas Henderson

CS 4300 - 001

17 November 2016

Assignment A6: Kalman Filter State Estimation

1. INTRODUCTION: CONAN

A Kalman Filter is an estimation algorithm that takes in sensor data over time and combines it with statistical noise to generate estimates based on variables. The algorithm is most commonly used to model movement and navigation for artificial intelligence. We model a linear and projectile trajectory and track it with the Kalman Filter to test its robustness. The questions we seek to answer are:

1. What are the confidence intervals on the means and hypothesis test for modeling a linear trajectory with a Kalman Filter?

2. What are the confidence intervals on the means and hypothesis test for modeling a projectile trajectory with a Kalman Filter?

3. How do the estimated velocities for the projectile model compare to the actual velocities over time?

4. What are the confidence intervals on the means and hypothesis test for modeling a projectile velocity with a Kalman Filter?

2. METHOD: RAJUL

For this experiment, the first step was to write the Kalman Filter function. This function comprises of the multiple equations shown below:

```
mu_bar_t = (A_t * mu_tm1) + (B_t * u_t);
```

```
Sigma_bar_t = (A_t * Sigma_tm1 * A_t') + R_t;
K_t = (Sigma_bar_t * C_t' * inv(((C_t * Sigma_bar_t * C_t') + Q_t)));
mu_t = mu_bar_t + (K_t * (z_t - (C_t * mu_bar_t)));
Sigma_t = (eye(4,4) - (K_t * C_t)) * Sigma_bar_t;
```

These equations apply the Kalman Filter explained above with the given parameters. Then the matrices A, B, and C were set up accordingly to reflect the equations given. This along with making ax = 0, ay = 0 to model a linear trajectory. After this was completed, the next task was a step by step application of the Kalman Filter using sensor data. Values we retained were the actual locations, the estimated locations, and sensor location. This was then used to plot the graphs shown in section 4.

Similarly, for the projectile trajectory slight changes were made. Instead of having acceleration equal to zero for a linear trajectory, gravity was added as an opposing force. Again, the data was collected and graphed accordingly as well as a graph for the velocities.

3. VERIFICATION OF PROGRAM: CONAN

We verify our Kalman Filter by comparing our algorithm’s output with a hand calculated result. In the case where:

$$\mu_{t-1} = [0.0179; -0.0539; 0; 0]$$

$$\Sigma_{t-1} = [0, 0, 0, 0; 0, 0, 0, 0; 0, 0, 0, 0; 0, 0, 0, 0]$$

$$u_t = [0; 0]$$

$$z_t = [0.1126; 0.0477]$$

$$A_t = [1, 0, 0.1, 0; 0, 1, 0, 0.1; 0, 0, 1, 0; 0, 0, 0, 1]$$

$$R_t = [1.0e-3, 0, 0, 0; 0, 1.0e-3, 0, 0; 0, 0, 1.0e-3, 0; 0, 0, 0, 1.0e-3]$$

$$B_t = [0.005, 0; 0, 0.005; 0.1, 0; 0, 0.1]$$

$$C_t = [1, 0, 0, 0; 0, 1, 0, 0]$$

$$Q_t = [1.0e-3, 0; 0, 1.0e-3]$$

$$\overline{\mu_t} = (A_t * \mu_{t-1}) + (B_t * u_t) = [0.0179; -0.0539; 0; 0]$$

$$\overline{\Sigma_t} = (A_t * \Sigma_{t-1} * A_t^T) + R_t = [1.0e-3, 0, 0, 0; 0, 1.0e-3, 0, 0; 0, 0, 1.0e-3, 0; 0, 0, 0, 1.0e-3]$$

$$K_t = (\overline{\Sigma_t} * C_t^T) * ((C_t * \overline{\Sigma_t} * C_t^T) + Q_t)^{-1} = [0.005, 0; 0, 0.005; 0, 0; 0, 0]$$

$$\mu_t = \overline{\mu_t} + (K_t * (z_t - (C_t * \overline{\mu_t}))) = [0; 0]$$

$$\Sigma_t = (I - (K_t * C_t)) * \overline{\Sigma_t} = [5.0e-4, 0, 0, 0; 0, 5.0e-4, 0, 0; 0, 0, 1.0e-3, 0; 0, 0, 0, 1.0e-3]$$

Our algorithm outputs the same result as the hand calculations and verifies it for the case.

4. DATA AND ANALYSIS: RAJUL

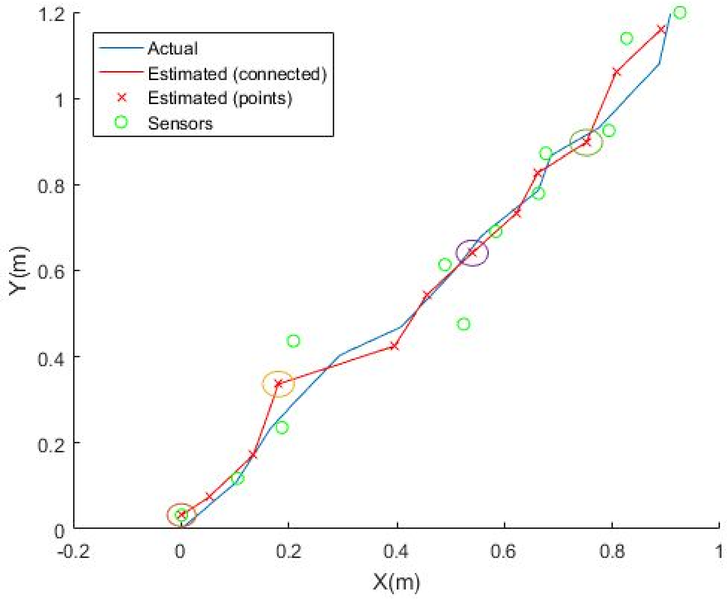

Figure 1: Results of Linear Trajectory KF.

In the graph above, the red line represents the Estimated Linear trajectory. We see that the Actual trajectory, although varied, is relatively close to the actual Linear Trajectory. We also used the error ellipse method provided to represent the error circles.

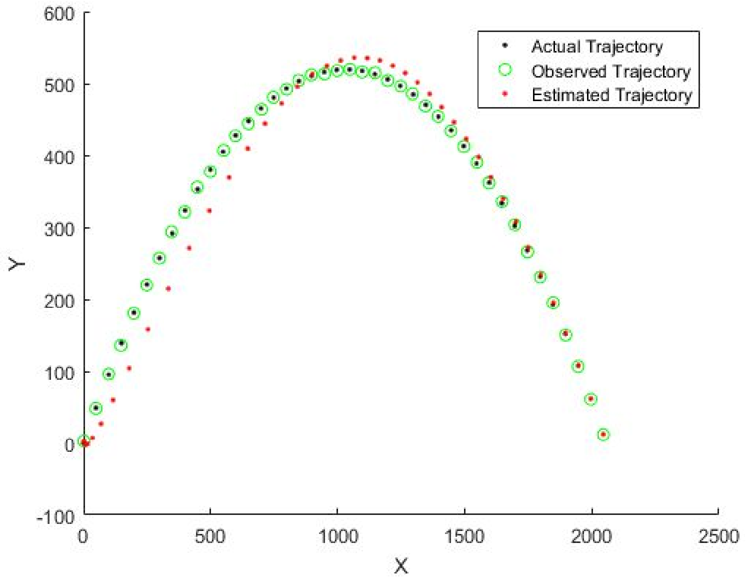

Figure 2: Results of Projectile Trajectory KF.

Figure 2 represents the projectile trajectory. Here, the algorithm was given gravity to map the projectile. Notice the estimated trajectory in red and the observed in green. As we move along the x-axis, we see that the two plots slowly start to align.

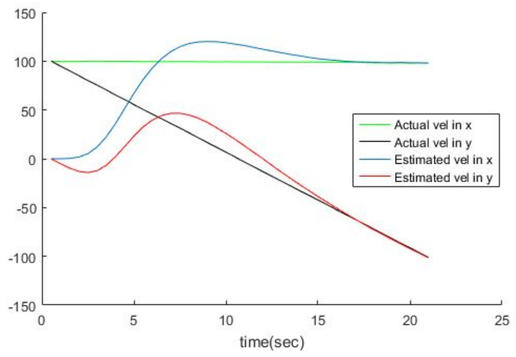

Figure 3: Results of Projectile KF Velocity Tracking.

Figure 3 represents the projectile trajectory model's velocity tracking over time.

Data Set	Mean	Variance	95% Confidence Interval Lower Bound	95% Confidence Interval Upper Bound
Linear Trajectory	0.0486	7.8501e-4	0.0328	0.0645
Projectile Trajectory	56.944	5.857e3	33.7981	80.0895
Projectile X Velocity	25.4287	1.1323e3	15.2521	35.6054
Projectile Y Velocity	25.7322	1.1325e3	15.5544	35.9101

Figure 4: Statistics for Data Sets

The statistical values in Figure 4 are based on the differences between the Kalman Filter estimates and the actual values of the model.

5. INTERPRETATION: CONAN

We can see in Figure 4 the Mean for the Linear Trajectory is a difference of 0.0486, the 95% confidence interval lower bound is 0.0328, and the 95% confidence interval upper bound is 0.0656. Also in Figure 4, the Mean for the Linear Trajectory is a difference of 56.944, the 95% confidence interval lower bound is 33.7981, and the 95% confidence interval upper bound is 80.0895. The values for the Projectile Trajectory are bigger simply due to the velocity we used to model and the maximum observation time we chose. Our linear trajectory works with a much smaller time interval. We can also see the Mean Projectile X velocity is a difference of 25.4287, the 95% confidence interval lower bound is 15.2521, and the 95% confidence interval upper bound is 35.6054. The Mean Projectile Y velocity is a difference of 25.7322, the 95% confidence interval lower bound is 15.5544, and the 95% confidence interval upper bound is 35.9101. These values are interesting in that they are very similar, indicating the Kalman Filter works the same for both velocity values.

We can see in Figure 3 that the estimated velocities for the projectile model eventually close in on actual velocities for the x and y directions. As the actual values are above the estimated velocities in the beginning, the estimated velocities increasing over time. After the actual y values decrease past the estimated y values, the Kalman Filter accounts for that and reduces the estimate in a rubber band like fashion. The same can be said for the estimated x velocity, which increases until it’s higher than the actual value, then decreases again.

6. CRITIQUE: RAJUL

For this experiment, important simulation concepts of graphing and distance formulas were used. A trial and error based method for finding the correct combination of parameters was used in order to obtain the given graphs. It might be better to consider a smarter way to choose more realistic world and sensor data to get realistic solutions.

Also, the addition of a real sensor will provide realistic noise and data as well which will help simulate the scenario more realistically.

7. LOG: RAJUL & CONAN

Rajul: I spent 5 hours on implementation and 6 hours on the Lab Report.

Conan: I spent 5 hours on implementation and 6 hours on the Lab Report.