William Argus
004610455

ECE 183DA - Design of Robotic Systems I

Lab Report - Lab 2

William Argus

Lab Group: Buffalo

February 4th, 2019

William Argus
004610455

Contents:

:

William Argus
004610455

1. Introduction

The goal of Lab 2 is to understand and implement a Kalman Filter to improve the estimation of the state of the Paperbot. In order to do this, a mathematical model of the complete dynamics of the robot system was developed. The mathematical model was used to develop a state estimator that has feedback through a Kalman filter to more accurately determine the state of the Paperbot. The Kalman Filter was then evaluated by comparing the output of the state estimator using a Kalman Filter to the actual state of the car, and to the output of the state estimator without a Kalman Filter.

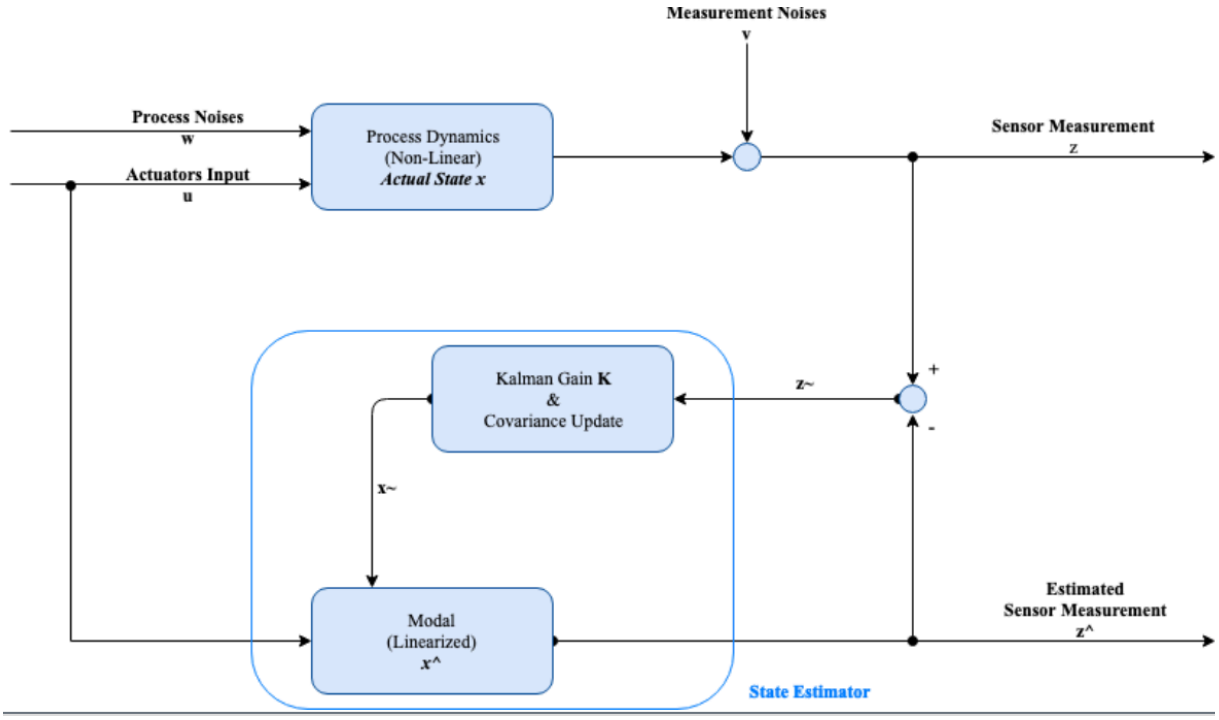

Figure 1, Block Diagram of System with Kalman Filter implemented.

2. THEORETICAL, GEOMETRICAL MODEL

Note: This model was developed for lab 1 and is included here for reference only.

Variables:

P: position of robot in 2D space

v: linear velocity of robot

N: noise associated

θ: Orietnation of the car in 2D space

ω: Angular velocity of the robot

D: distance from rotational axis to the car's reference point (the top right of the car)

Robot state update dynamics

The objective of this section is to mathematically find a theoretical, geometric model of the robot state update dynamics which will be of the form:

$$x'(t) = x_{t+1} = f(x_t, u_t, t) = Ax_t + Bu_t$$

In order to be in complete control of the car, the movements of the car are split into two cases:

Case 1: Traveling along a straight line (forward, backward)

$$p_{x,t+1} = p_{x,t} + (v_t + N_v)t * cos\theta_t$$

$$p_{y,t+1} = p_{y,t} + (v_t + N_v)t * sin\theta_t$$

$$\theta_{t+1} = \theta_t$$

Expressing in matrix form,

$$x_{t+1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{x,t} \\ p_{y,t} \\ \theta_t \end{bmatrix} + \begin{bmatrix} (v_t + N_v)t * cos\theta_t \\ (v_t + N_v)t * sin\theta_t \\ 0 \end{bmatrix}$$

Which agrees with the form:

$$x_{t+1} = I_3 x_t + Bu_t = Ax_t + Bu_t$$

Case 2: Turning about its rotational axis (clockwise, counter-clockwise)

Note that $p_{x,t+1} \neq p_x$ and $p_{y,t+1} \neq p_y$ because rotational axis is different from the car reference point.

$$p_{x,t+1} = p_{x,t} - D[cos\theta_t - cos\theta_{t+1}]$$

$$p_{y,t+1} = p_{y,t} + D[sin\theta_t - sin\theta_{t+1}]$$

$$\theta_{t+1} = \theta_t + (\omega_t + N_\omega)t$$

Expressing in matrix form,

$$x_{t+1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{x,t} \\ p_{y,t} \\ \theta_t \end{bmatrix} + \begin{bmatrix} -D[cos\theta_t - \cos(\theta_t + (\omega_t + N_\omega)t)] \\ D[sin\theta_t - \sin(\theta_t + (\omega_t + N_\omega)t)] \\ (\omega_t + N_\omega)t \end{bmatrix}$$

Which agrees with the form:

$$x_{t+1} = I_3 x_t + Bu_t = Ax_t + Bu_t$$

Note that the equations work for all θ_t.

3. Mathematical Model

Methodology Description

In order to develop the mathematical model, the robot's current state is defined as (P_x, P_y, θ). The x-coordinate is P_x, the y-coordinate is P_y, and the angle relative to true north, measured clockwise, is θ.

To find the distance from the Paperbot's reference point, its forward right corner if the wheel side is defined as forward, a line is drawn perpendicular to the respective edge of the car that each of the lidar sensors sit on.

For example, if the car is facing due North, the front lidar sensors would be also facing due north, so the line drawn would be in the direction of due north, perpendicular to the front edge of the car.

The line drawn will intersect with two out of the four edges of the box, assuming that these lines continue infinitely in both directions outside the box. The equations of these lines are

Left edge: x = 0

Right edge: x = L

Bottom edge: y = 0

Top edge: y = W

Note that one of the intersections will occur at the edge of the area of the box, the other will occur outside the area of the box, as the lines extend infinitely in both directions. See Figure 2 on following page.

Knowing the two intersection points, and the current state of the car, the distance between the car and each of the points is calculated. After these distances are calculated, the smaller of the two distances corresponds to the distance between the sensor and the wall.

This distance between the sensor and the edge of the box will give the ideal geometric reading of the lidar sensor.

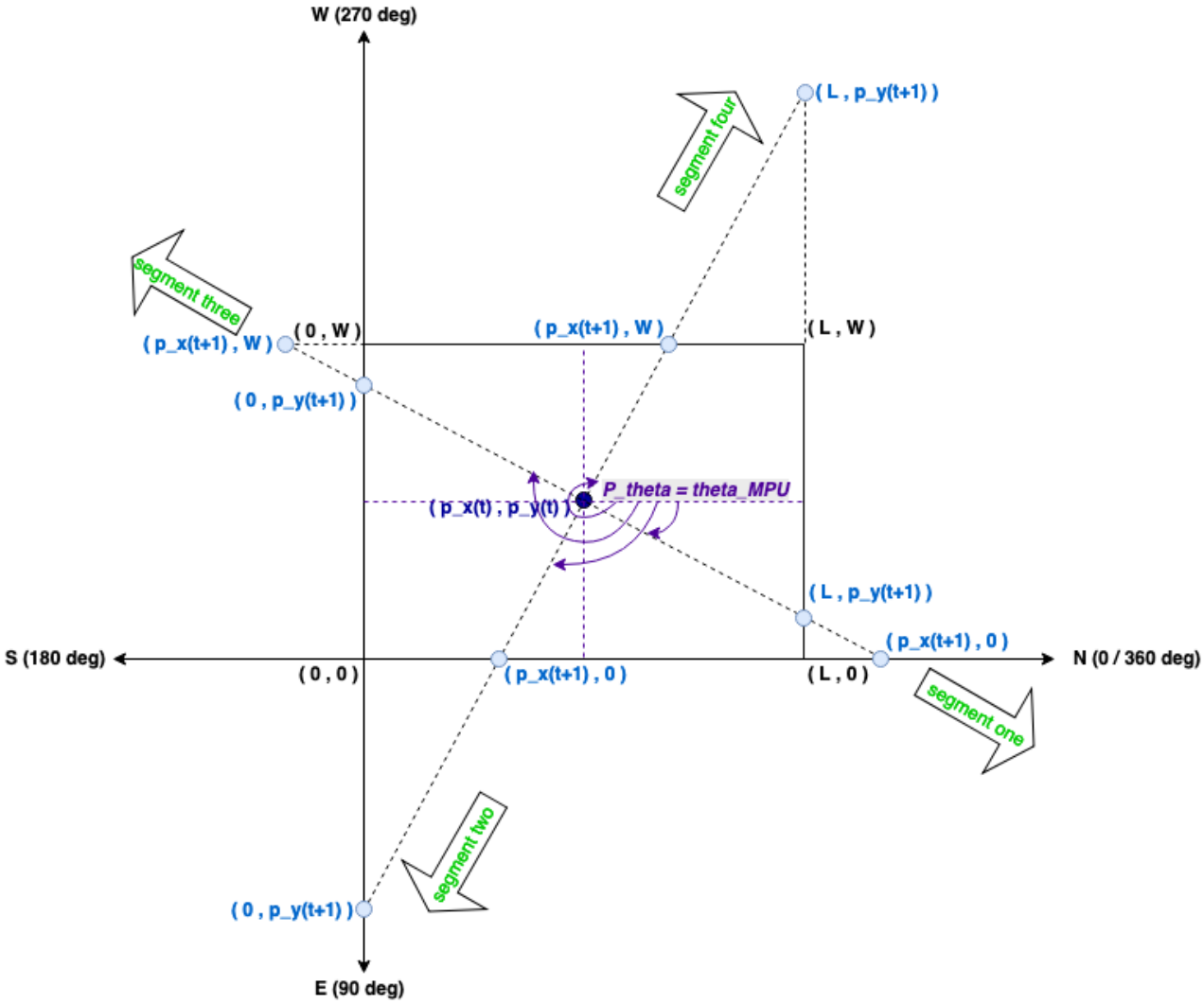

Figure 2, Paperbot Geometry. This shows the geometry used to develop the mathematical model discussed above.

Equations of Mathematical Model

The team separated the mathematical model into 4 different cases, depending on the angle of the Paperbot relative to true north.

For each lidar sensor, front and right, its intersection points with the lines making up the box are defined as follows:

$$Y_f = Front\ lidar\ sensor's\ Y\ intersection$$

$$Y_r = Right\ lidar\ sensor's\ Y\ intersection$$

$$X_f = Front\ lidar\ sensor's\ X\ intersection$$

$$X_r = Right\ lidar\ sensor's\ X\ intersection$$

$For\ Angles\ of\ 0° \leq \theta < 90°$

$$Y_f = 0$$

$$X_f = L$$

$$Y_r = 0$$

$$X_r = 0$$

$For\ Angles\ of\ 90° \leq \theta < 180°$

$$Y_f = 0$$

$$X_f = 0$$

$$Y_r = W$$

$$X_r = 0$$

$For\ Angles\ of\ 180° \leq \theta < 270°$

$$Y_f = W$$
$$X_f = 0$$
$$Y_r = W$$
$$X_r = L$$

$For\ Angles\ of\ 270° \leq \theta < 360°$

$$Y_f = W$$
$$X_f = L$$
$$Y_r = 0$$
$$X_r = L$$

The front lidar sensor's intersection points with the lines forming the box can be calculated as follows:

$$p'_x = p_x + {p_y - Y_f}/{tan(p_\theta)}$$

$$p'_y = p_y + (p_x - X_f)(tan(p_\theta))$$

The right lidar sensor's intersection points with the lines forming the box can be calculated as follows:

$$p'_x = p_x + +(Y_r - p_y)(tan(p_\theta))$$

$$p'_y = p_y + {X_r - p_x}/{tan(p_\theta)}$$

The distances of the intersections of each pair of the lines found above can be calculated as follows:

$$r_{x_f} = r_{x_r} = \sqrt{(p'_x - p_x)^2 + (Y - p_y)^2}$$

$$r_{y_f} = r_{y_r} = \sqrt{(X - p_x)^2 + (p'_y - p_y)^2}$$

Linearization

From the mathematical model shown in the previous section, the location of the reference point can be determined. The next step is to linearize the model so that it can be used with an extended Kalman Filter.

Take the intersection point to be (X_I, Y_I)

Take the lidar reading of distance between sensor and wall to be:

$L_I = \sqrt{(p_{x0} - X_i)^2 + (p_{y0} - Y_i)^2}$

Linearizing the mathematical equations previously discussed about $(p_{x_0},\ p_y, p_{\theta_0})$ will give $z_{3x1} = H_{3x3}x_{3x1} + D_{3x1}$, to be used in the extended Kalman Filter.

Putting the above equations in matrix form gives:

$$\begin{bmatrix} l_f \\ l_r \\ \theta_{MPU} \end{bmatrix} = z = \begin{bmatrix} \frac{p_{x_0} - X_I}{L_I} & \frac{p_{y_0} - Y_I}{L_I} & 0 \\ \frac{p_{x_0} - X_I}{L_I} & \frac{p_{y_0} - L_I}{L_I} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_\theta \end{bmatrix} + \begin{bmatrix} -p_{x0}\frac{p_{x0} - X_I}{L_I} \\ -p_{x0}\frac{p_{y0} - Y_I}{L_I} \\ L_I \end{bmatrix}$$

$$= Hx + D$$

$$H = \begin{bmatrix} \frac{p_{x_0} - X_I}{L_I} & \frac{p_{y_0} - Y_I}{L_I} & 0 \\ \frac{p_{x_0} - X_I}{L_I} & \frac{p_{y_0} - L_I}{L_I} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} -p_{x0}\frac{p_{x0} - X_I}{L_I} \\ -p_{x0}\frac{p_{y0} - Y_I}{L_I} \\ L_I \end{bmatrix}$$

The H and D matrices can be derived for all cases listed previously, in the mathematical model section by plugging in intersection points corresponding to those cases.

The above equations will not hold for some special cases, so we derive the H and D matrices specifically for those situations. This situations arise because of the following instances:

$$tan(0°) = tan(180°) = tan(360°) = 0$$

$$tan(90°) = tan(270°) = undefined$$

The special instances of H and D matrices are as follows:

$For\ angles\ 0°\ and\ 360°$

$$H = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} L \\ 0 \\ 0 \end{bmatrix}$$

$For\ angle\ 90°$

$$H = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$For\ angle\ 180°$

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \\ W \\ 0 \end{bmatrix}$$

$For\ angle\ 270°$

$$H = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} W \\ L \\ 0 \end{bmatrix}$$

4. Kalman Filter

Since the non-linear system has been linearized in the above section, an Extended Kalman Filter can be implemented on the system.

1. Prediction

$$\hat{x}_{t+1}^{-} = A\hat{x}_t^{+} + Bu_t$$

$$P_{t+1}^{-} = AP_t^{+}A^T + Q$$

2. Gain (K value)

$$K_{t+1} = P - C^T(CP - C^T + R)^{-1}$$

3. Measurement Update

$$\hat{x}^{+} = \hat{x}^{-} + K(z - H\hat{x}^{-})$$

$$P^{+} = (I - KH)P^{-}$$

Note that:

$e = x(t) - \hat{x}^{+}(t)$

$P = E[ee^T] = Error\ Covariance$

$Q = E[ww^T] = 0$

$R = E[vv^T]$

5. Results

Paperbot State

The state estimator of Paperbot was done using Matlab. The plot of the state estimation that Matlab outputted can be seen in Figure 3. The box used had a length of 750 mm and a width of 500mm. These dimensions are reflected in the plot of the Paperbot's state. The Paperbot is represented by a rectangle, with the red star point indicating the location of the Paperbot's reference point, calculated using the mathematical model with the Kalman filter implemented that was presented previously.

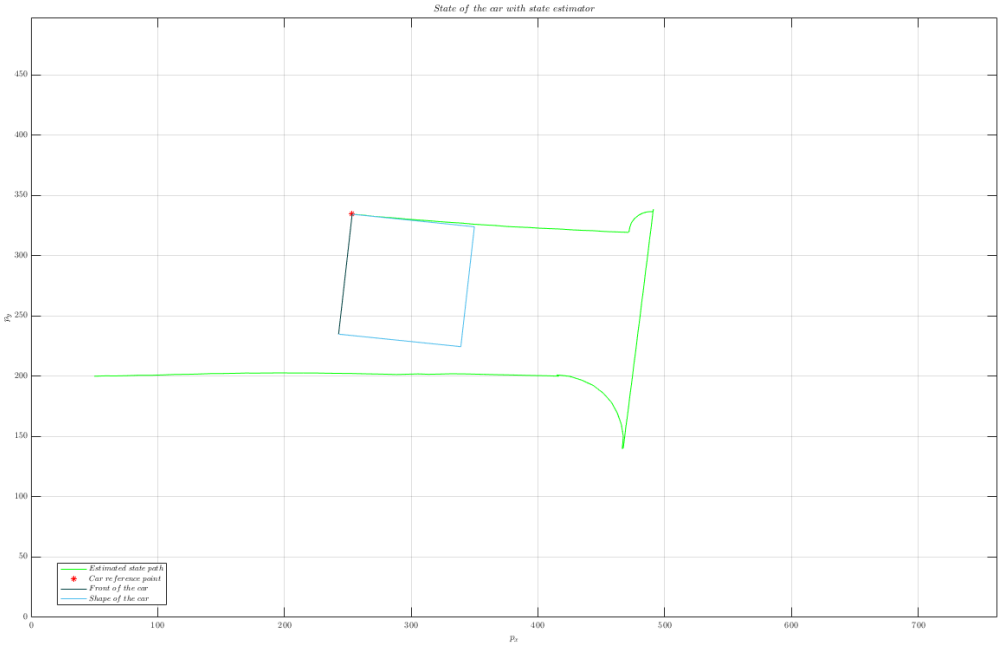

Figure 3, State Estimation of Paperbot.

In order to validate the state estimator using a Kalman Filter, the car was placed in a box, representing a simulated environment. The box was in the shape of a rectangle, measuring 750mm by 500 mm. The walls bounding the box were covered in white paper to ensure that the lidar sensors were able to give accurate readings. The floor was marked with squares with sides of length 5 cm, to allow the true state to be determined by a camera or observer. A camera was setup to record a bird's eye view of the car's movement, so that the actual state of the Paperbot could be compared to the state given by the state estimator both with, and without the Kalman Filter implemented.

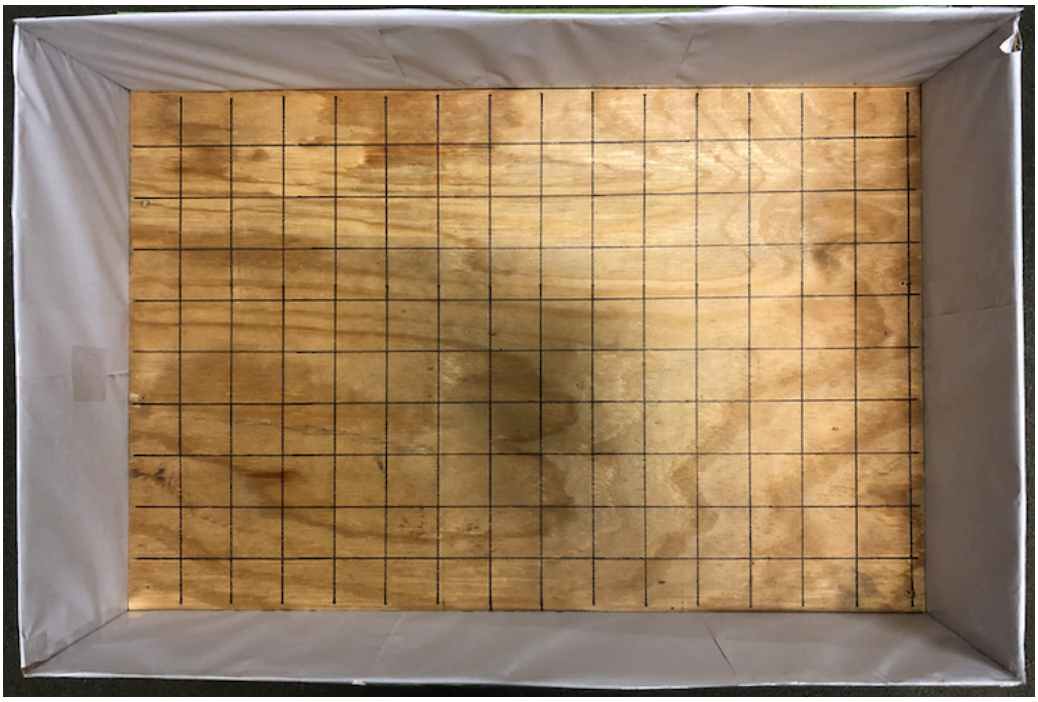

Figure 4, Simulated Environment for Paperbot.

Comparison

The video of the Paperbot moving can be found in the Git Repository. The data, PWM inputs and sensor outputs, was saved from that trial and run through the state estimator

both with and without the Kalman Filter implemented. The output of the closed loop state estimator when given that set of data can be seen in Figure 3. The true state of the Paperbot corresponding to that set of movements can be seen in the video previously mentioned.

It was observed that the state estimation tracks closely with the actual state of the Paperbot in that instance. When compared with the open loop state estimation, the closed loop state estimator was more accurate, due to the correcting its state by using input from the sensors. This was not always the case, unfortunately. Due to the initial state of the Paperbot being known in this case, the error covariance was small.

It was observed, however, that when the initial state was not known, and the error covariance grew large, the closed loop state estimator became very inaccurate. It was determined that the error was due to the Kalman filter implementation, and not with the mathematical model. This was because the open loop estimate was not affected by a large error covariance, while the closed loop estimate was; this makes sense as the Kalman Filter makes use of error covariance. Another way of saying this is that when there was an increase in the noise in the system, the open loop state estimator remained the same as it did with a small error covariance: reasonably accurate, but not perfect. The closed loop state estimation, however, became very inaccurate with a large error covariance. The inaccuracy was so bad that the closed loop state estimation because less accurate than the open loop state estimation. Clearly, this is a problem with the implementation of the Kalman Filter if it is decreasing accuracy under a lot of noise instead of increasing it.

Future Improvements

It will be necessary in the future to fix the implementation of Kalman Filter so that the closed loop state estimator works correctly and is able to accurately determine the state even when starting from an unknown initial state. Another improvement that can be made to the Paperbot is the programming of the Paperbot to be completely wireless, so that it is able to send its data to the state estimator in Matlab wirelessly, and in real time. Part of the motivation behind this is to allow for live state estimation updates, and part of the motivation is to allow the car to transmit its data while moving and untethered to the laptop. In the latter case, it is especially necessary that the car not have a wire trailing behind it, as the wire was suspected by the team to be a large source of noise in the system as it dragged behind that car, often pulling it to one side instead of moving in a straight line.

6. Appendix

All Matlab code, pictures, videos, and data can be found on Github.

Link to Git Reposity: https://github.com/IouSC/UCLA_EE183DA_TeamBuffalo