Lab 2: Extended Kalman Filter

Timothy Lee

Ruochen Wang

Christian Yu

**Contents**

[**I.** **Background** 3](#_Toc32281623)

[**II.** **Background – Extended Kalman Filter (EKF)** 3](#_Toc32281624)

[**III.** **Actuation Model, Sensor Models, and Error** 3](#_Toc32281625)

[**A.** **Actuation Model** 4](#_Toc32281626)

[**B.** **Sensor Model** 4](#_Toc32281627)

[**IV.** **System Dynamics and EKF Dynamics Propagation** 5](#_Toc32281628)

[**A.** **Position** 5](#_Toc32281629)

[**B.** **Compass Heading** 7](#_Toc32281630)

[**C.** **Rotational Velocity** 8](#_Toc32281631)

[**D.** **Summary** 8](#_Toc32281632)

[**V.** **Measurement Model and EKF Measurement Update** 9](#_Toc32281633)

[**A.** **Position** 9](#_Toc32281634)

[**B.** **Compass Heading** 9](#_Toc32281635)

[**C.** **Gyroscope Sensor** 9](#_Toc32281636)

[**D.** **Summary** 10](#_Toc32281637)

[**VI.** **Kalman Filter** 10](#_Toc32281638)

**Github**: <https://github.com/rrcwang/ECE183DA-Labs>

Please see calibration\_data.xlsx for raw calibration data. Details and other information are contained in the README.

### **Introduction**

In this lab, we explore the use of Kalman Filtering (KF) in creating state estimates of a two-wheeled robot using the mathematical, actuation, and sensor models developed in the previous lab. This state estimator is then validated against simulations and physical results. In this report, we will provide background on Extended Kalman Filters (EKF), our experimental setup and mathematical models before explaining our findings.

### **Background – Experiment Setup and**

To explore Kalman Filtering in creating a state estimate, we used a two-wheeled robot called “paperbot”. The rough dimensions of the robot without the modules required for computation are shown in the figures below. In addition to the controller, the paperbot includes two laser distance sensors mounted on the front and right sides.

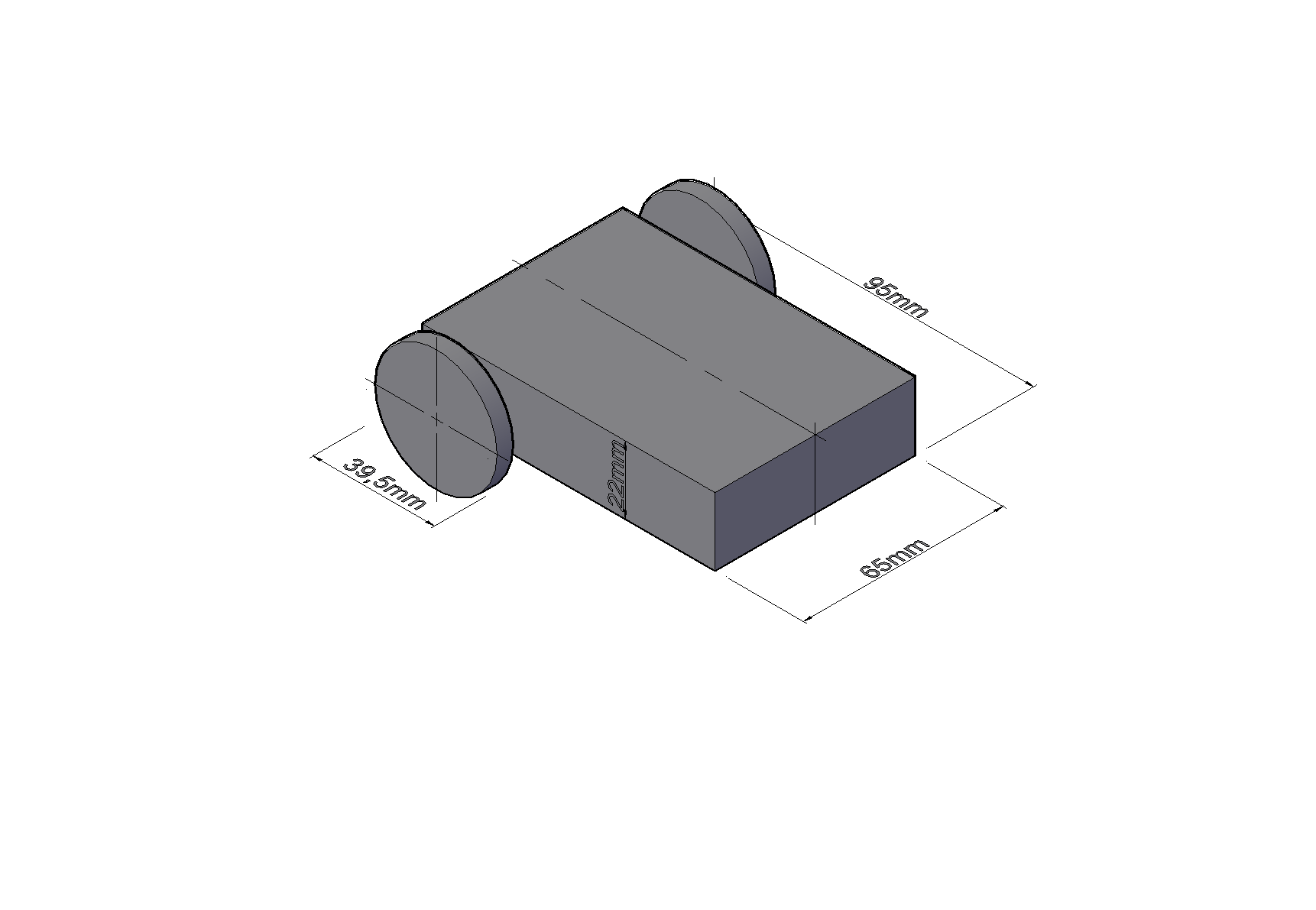
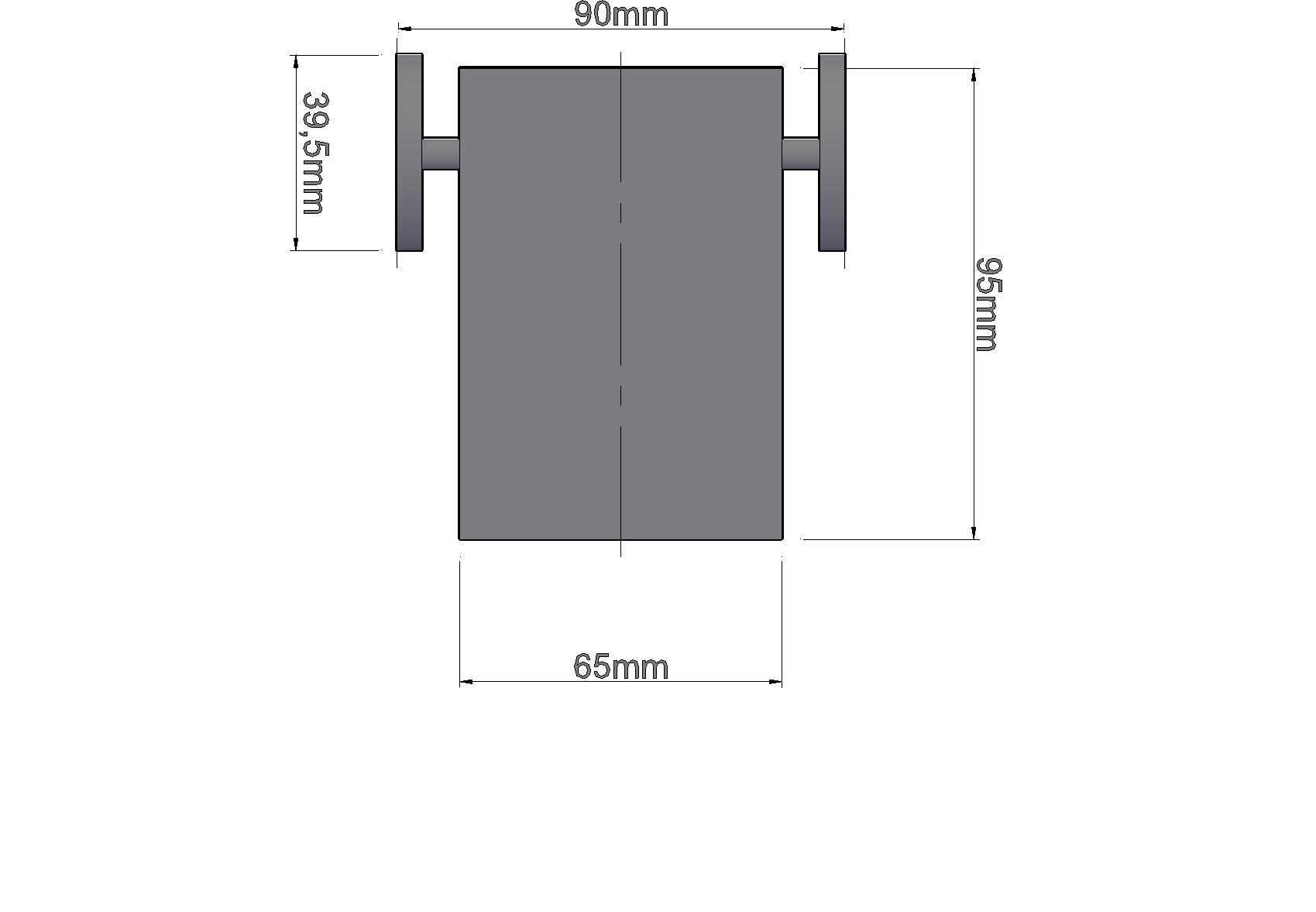


Fig. 1. Dimensions of our test platform.

We use a standard shipping cardboard box with the dimensions 45 cm x 63 cm as our testing environment for the paperbot. The floor space is bounded by the walls of the cardboard box, with one corner designated as the point of origin. Coordinate axes were marked in centimeter units along the environment floor. To enable greater range of forward motion during our testing with the paperbot, we designated the 45 cm dimension as x and the 63 cm dimension as y. As per the Cartesian coordinate system conventions, positive directions are designated as x+ right and y+ up, matching the paperbot sensor placements at the front- and right- facing sides. In this configuration, the paperbot is flush with the left and lower walls of the box when placed at the origin.

To test our state estimator, we must provide an initial state and time series of inputs and sensor data. In this experiment, the state is defined by positional data – the linear displacement values and and the rotational values , or angular displacement and , or rotational (angular) velocity. The initial state is represented simply by placing the paperbot at the origin aligned with the sides of the box. We then provide create a time series of inputs to provide to both the state estimator and the actual robot itself. Through this, we can observe the difference between the estimated state and the actual observed state using the axes measured in the box at corresponding times. The basic cases that we must test are:

1. Moving forward at the zero angle,
2. Moving forward at an angle orthogonal to zero,
3. Moving backward at the zero angle,
4. Moving backward at an angle orthogonal to zero,
5. Rotating right,
6. Rotating left,
7. Moving forward at an angle not associated with the cardinal points,
8. Moving backward at an angle not associated with the cardinal points, and
9. Remaining still.

An example of an initial and end state is depicted in the figure below. The figure only includes coordinate positional data, but actual experiments record rotation and rotational velocity as components of the robot state.

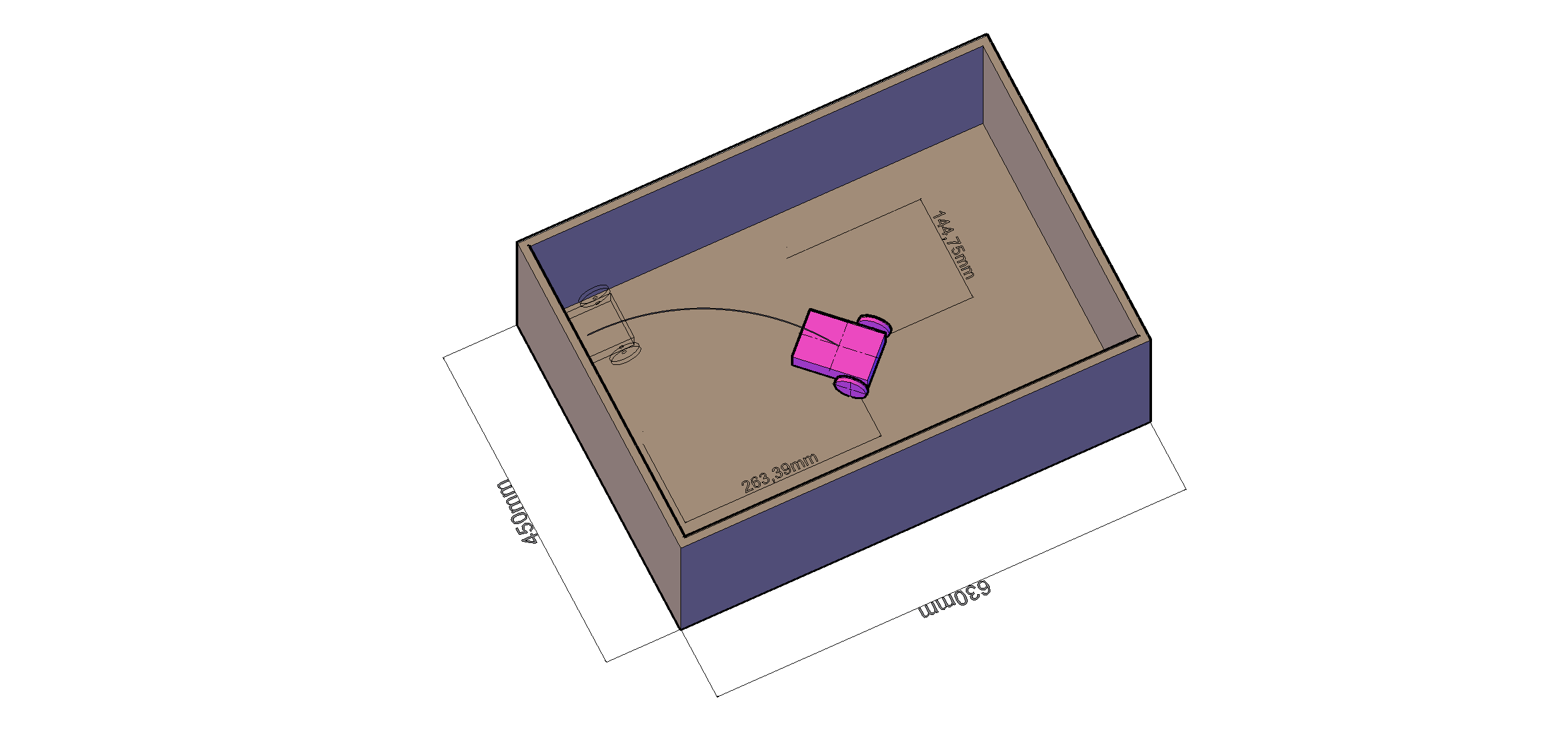


Fig. 2. Rendering of an experiment’s trial.

We used the inside of a 63cm x 45cm (inside dimensions) four-walled box as a testing environment. We performed re-calibrations and noise analysis of our sensors and created a new actuation model with data gathered from running tests in the box. The re-calibrations we performed in the testing environment increases the accuracy of our state estimator.

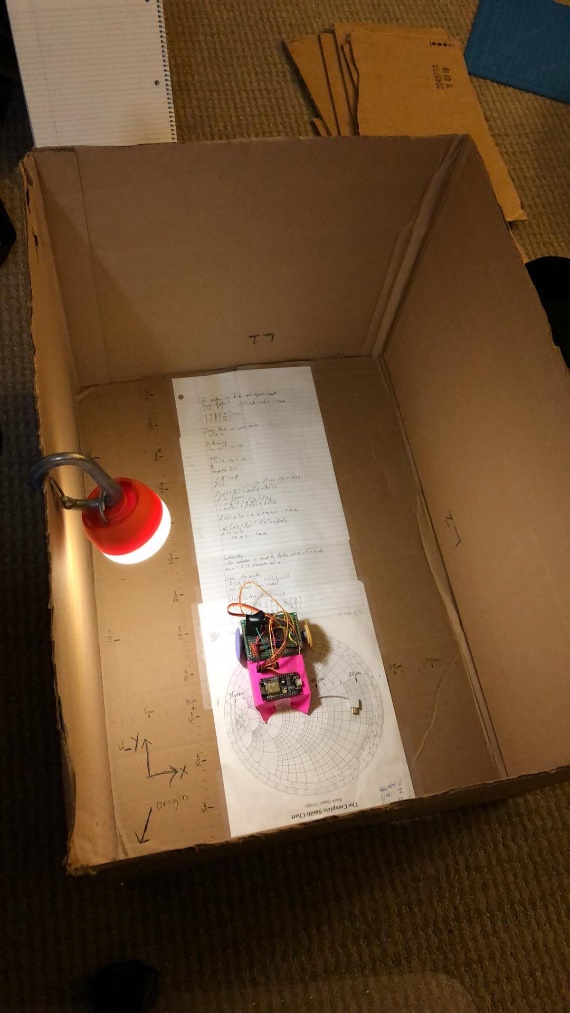


Fig. 3. The paperbot inside our illuminated, four-walled testing environment.

### **Background – Extended Kalman Filter (EKF)**

The EKF consists of two main steps, the **dynamics propagation** and the **measurement update**. First, we will need the discrete time system dynamics and measurement models of the robot, as

Here and are the noises in the dynamics and measurement noises, modeled as additive white gaussian noise (AWGN) with covariances and respectively.

With knowledge of some initial state from the physical system, the EKF is described as follows.

The state transition and observation matrices and are defined to be the Jacobians evaluated as

1. First, we **initialize** the state for time
2. Compute the initial state estimate
3. Compute the covariance estimate
4. For the time , the **dynamics propagation** step consists of:
   1. Predict the next state, using system dynamics
   2. Predict the covariance estimate
5. The **measurement update** consists of
   1. Compute measurement residuals
   2. Compute covarianceof residuals
   3. Compute the Kalman gain
   4. Update the state estimate
   5. Update the covariance estimate

Step 1 and Step 2 are iterated over.

### **Actuation Model, Sensor Models, and Error**

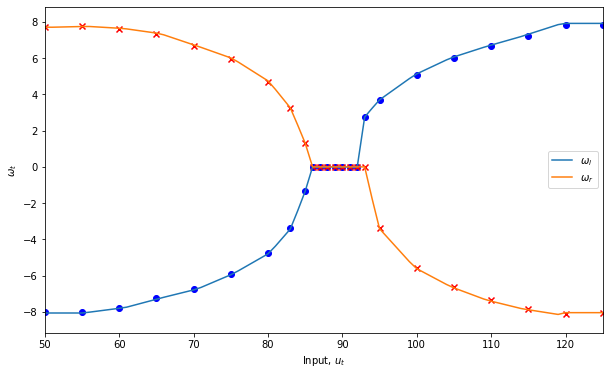
We define our state as  
where are cartesian position of the robot. is the compass heading of the robot. is the in-plane rotational velocity.

We define the translational and rotational center of the car at , with being the distance between the two wheels, the length of the robot, and the radius of the wheels respectively.

### **Actuation Model**

Let the **system input** be , where is the input value to the motors and at time .

The **actuation model** where the first and second elements represent the rotational velocity of the left and right wheels respectively. This was found through experimentation as described in Lab 1. The calibration data is reproduced below in Fig. OM.



**Fig. 4.** Ideal motor actuation curves

We will model the output as constant in the saturation regions outside of the graph for input values in the ranges .

We will assume that the actuation noise is constant for all time . Then the linearized form the actuation model is given as

Here, each is an affine function of with **input error** caused by physical factors such as slippage given by , a 2-D Gaussian distribution. To make the expression linear, we have added also defined , an augmented version of including . These are the constant DC offsets in the linearization of .

To characterize the error of the motor input, we considered separately the linear and rotational motions of the car. Using the fact that linear velocity at some radius away from the center of a rotating body is given as , we define the **linear** **velocity** of robot in the body basis to be  
where is the radius of the wheel.

To model the noise in our actuation model, we used the Wi-Fi interface to command paperbot to move forward for 2 seconds. We then used the linear sensor output data to determine the distance travelled as well as the average velocity. It should be noted that the noise in our sensors was significantly smaller (1 or 2 magnitudes) than the noise in our actuation model so we considered it to be negligible. After running 10 trials, we calculated the average and standard deviation of the distances travelled, about 10 mm worth of error.

We will also define the **input** **rotational velocity** of the robot as follows  
where is the distance between the two wheels.

We conducted similar experiments to find the average and standard deviation of the driven angular velocity. We observed the magnetometer output as we drove the paperbot to rotate for a 2 second time interval to calculate these values.

### **Sensor Model**

Let the **measurement** be . We will also assume that these measurement errors are constant for all time . The first two components are the distances measured from the right and forward distance sensors respectively.

**Fig. 5.** Sensor Calibration data for the distance sensors.

To characterize the noise in these sensors, we take distance measurements in our environment from a variety of positions and orientations for several seconds. The standard deviation and was found to be anywhere between 1.5 and 3.3 scaling with short to long measurement distances. Taking all the average case gives us a standard deviation . To avoid adding additional complexity to our mathematical model of the robot. We measurements at a typical position (heading angle = 0°, ~45cm from the wall) and used the standard deviation of 2.56 to characterize the added white Gaussian noise of the sensors.

The third measurement is the magnetometer sensor’s compass heading output in degrees. There is a **standard deviation** in the measurements of . Our magnetometer outputs measurements “mx” and “my” which can be plotted in a cartesian coordinate system. To obtain the heading angle from these measurements, we perform .

A calibrated magnetometer’s mx and my will form a circle centered at (0, 0). This ensures that our angle computation properly reflects the physical heading angle of the paperbot. To calibrate the magnetometer, we placed it in our environment and observed sensor output as the paperbot made several rotations. Starting from the raw, uncalibrated x and y magnetometer data, we first scaled our outputs to normalize the shape of the output. We then performed the test again to retrieve the constant offsets to center the output circle at (0,0). The process is illustrated by our sensor output plots.

**Fig. 6.** Raw magnetometer output before scaling and offset is approximately elliptical with center at approximately (-2, -20).

To normalize the shape of the output, we calculated a scaling factor with the following formulas:

**Fig. 7.** Output after scaling is now approximately circular with a clearer center at (-15.71, -8.09). We took these as our final linear offsets to mx and my respectively.

**Fig. 8.** Output after scaling and offsets is approximately circular and centered at the origin.

The final value is the rotational acceleration measured by the gyroscope sensor in . We find that the standard deviation in the rotational velocity measured is about . For lack of a better and more accurate method of measurement, we used our magnetometer data to scale our gyroscope output from units of magnitude to radians/s. We measured the average angular velocity by rotating the robot at an approximately constant angular velocity. We then calculated the average angular velocity using the magnetometer data.

**Fig. 9**. Gyroscope measurement of equal rotational speeds in the left (blue) and right (orange) directions respectively.

### **System Dynamics and EKF Dynamics Propagation**

Using our knowledge of the physical system and actuation model, we will define the dynamics update , state estimate, and covariance estimate. In the previous sections we have already defined:

### **Position**

To begin we will consider the cartesian position component of the state:

We can also define as estimate of as a function of the inputs as a linear function, linearized at time as done in :

We then use this to estimate the linear velocity,

We find that , since is found to be the sum of the two Gaussian distributed noises in with covariance of two uncorrelated Gaussian distributions.

The change in is then found as a function of the state variable

If the **time difference between state updates** is , then the **dynamics propagation** to is

Using the estimate given in to complete ,  
with additive white gaussian noise The **covariance matrix** of can then be found as

In matrix form, the linearization is found to be

### **Compass Heading**

Recall that the **input** **rotational velocity** of the car can be found by :  
where is the distance between the two wheels. Then the dynamics update to the rotational angle and velocity are given by

We can find a linearization for using once again:

Where:  
Similar to the previous section C, we find that as is the sum of the two uncorrelated Gaussian distributed noises in .

With this we can write the EKF linear estimate of the next state to be

The noise term for the variable is then . By definition of variance,

Rewriting in the linearized matrix form of the dynamics propagation,

### **Rotational Velocity**

We will directly update the rotational velocity estimate to be equal to the input rotational velocity

Reusing the linearization found in step will yield the desired dynamics propagation expression

### **Summary**

We have considered up to now each state component independently. To put them together, we will need to first consider the dynamics propagation **Step 1.1**. We have found the expressions

These can be put together into a neater form using matrices:

Observe that

Next, we will need to find the linearized Jacobian matrices that describe . This is given by our linear expressions

Similar to what we did for , the linearized dynamics propagation can be found as

Assuming the three variables are independent of each other, the **dynamics covariance matrix** is

Step 1.2 can then be easily computed with and .

### **Measurement Model and EKF Measurement Update**

The measurements of our system are given by . Like section III, we will look at each measurement component separately first, then summarize and walk through Step 2 of the EKF as applied to our system.

### **Position**

The devices will give us two distance measurements orthogonal to each other, and . The algorithm we implement to find the expected measurement is as follows:

1. In the direction of sensor , project a sufficiently long vector from body origin .
2. For each of the four walls, check which wall interests with and call that point .
3. If there are no intersections, throw an error.
4. Return .

Let and be two points on the walls as above. The Jacobian can be found as

### **Compass Heading**

When we predict our next state to be , we expect the measured heading to be directly that which we predicted. Hence

The Jacobian for this function is simply

### **Gyroscope Sensor**

Similar to the case for the compasses, we expect our calibrated measurement to match our calculated speed from the dynamics propagation step.

Similarly,

### **Summary**

Taking the previous sections into a complete equation, we find that

This is the matrix required to complete Step 2 of the Kalman filtering process.

### **Kalman Filter and Simulator**

By now we have defined each individual definition required. For our initial conditions, we will assume that the position is uniformly distributed in position. Rotation is uniformly distributed over but is cyclic, hence it does not matter what value we initialize to. is modelled as a Gaussian centered at 0. Then the default initialization of our state estimator will take values .

The constructor for our state estimator can be initialized to arbitrary values, but will likely immediately throw an out of bounds error if the position is beyond the zone of the box. The simulator operates with a known time interval between measurements , which is the rate at which the sensor values are sampled.

For our initial covariance estimate , we will assume that each state component is independent of each other and has very high uncertainty. With this in mind, we will initialize  
where .

### **Testing and Evaluating the Kalman Filter**

In each of the below plots, compare the trajectory and observe the rapid convergence within a few iterations of initializing the state estimator.

**Fig. 10**. Experiment 1. Car rolls forwards then back in a smooth manner.

### **References**

<http://blog.tkjelectronics.dk/2012/09/a-practical-approach-to-kalman-filter-and-how-to-implement-it/>

<https://www.cse.sc.edu/~terejanu/files/tutorialEKF.pdf>

<https://bryceboe.com/2006/10/23/line-segment-intersection-algorithm/>