EKF ACCELEROMETER BASED WHEEL ODOMETRY

by

Jacob Morgan

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Electrical Engineering

Charlotte

2021

Approved by:

_____

Dr. James Conrad

_____

Dr. Robert Cox

_____

Dr. Andrew Willis

ABSTRACT

JACOB MORGAN. EKF Accelerometer Based Wheel Odometry. (Under the direction of DR. JAMES CONRAD)

Gathering information on robot motion is critical in determining how a mobile robot has interacted in its environment. The information about a robot's motion, called odometry is generally collected by tracking wheel behavior for ground based robots. Wheel encoders are a typical means for collecting this information. Wheel encoders generally split the circumference of the wheel into equally discrete distances to track the distance traveled along the wheel's circumference. The proposal of this research is to dismiss the use of the above discrete based wheel encoders and use an accelerometer as a wheel encoder for a more continuous reading of the wheel's position. Accelerometers are typically difficult to use for precise data collection because of noisy outputs causing inaccurate odometry information. This can be overcome by building a model to predict how the output of the accelerometer on the wheel should behave. With the accelerometer mounted on the wheel of a ground based robot, an expected output of the accelerometer should oscillate between positive and negative values of the gravity vector. This allows for a system that can be modelled as a sinusoid. Using the sinusoidal model, the raw data can then be filtered with an Extended Kalman Filter (EKF). The Extended Kalman Filter weights the measurement from the sensor along with the predicted value from the model to give a low noise, high precision output. The results from the newly filtered data proved to give acceptable accuracy that was desired for estimating position of the wheel. Based on the filtered data, the calculated distance proved to remain within %2.2 of the expected distance traveled. This error was seen to be the maximum error at start up and dropped well below %1 in each of the experiments.

ACKNOWLEDGEMENTS

The work completed in this research was supported by my academic advisor Dr. James Conrad and the other two board members for this thesis, Dr. Andrew Willis and Dr. Robert Cox. Dr. James Conrad provided the direction of what educational skill would be needed for a successful Masters focused on the robotics field. The basis of the Masters education was also taught by Dr. Conrad. I would like to also specifically thank Dr. Robert Cox for the insight in modeling provided in his lectures. It added the ability for me to analyze systems into mathematical models in order to provide proper control techniques as well as later be used for filtering. Dr. Andrew Willis must be noted for teaching in great detail the Kalman filtering techniques ultimately needed to make this thesis a success.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AMR An acronym for Autonomous Moving Robot.

EKF An acronym for Extended Kalman Filter.

I2C An acronym for Inter-Integrated Circuit

IMU An acronym for Inertial Measurement Unit

LED An acronym for Light Emitting Diode

M An acronym for Meters

MPS An acronym for Meters Per Second

MPSS An acronym for Meters Per Second Squared

POSE An acronym for Position Orientation System Estimate.

PPR An acronym for Pulses Per Rotation

RPM An acronym for Rotations Per Minute

CHAPTER 1: INTRODUCTION

1.1 Problem Statement

Information about the motion of a mobile robot is important for determining the location of the robot in its environment. The use of this information to determine the robot state is called odometry. For ground based mobile robotics, observing the actual movement of the robot is commonly achieved by monitoring the wheel movement. Knowing the diameter and wheel circumference, the distance traveled by the wheel is determined by recording the change in the wheel's position and recording how much of the wheel's circumference was traveled. This information is then used to determine the position and orientation of the mobile robot in its environment, known as the pose. Wheel encoders are typically used to collect the odometry information. This can include using magnetic, mechanical, or light based wheel encoders. These classic wheel encoders use a method of creating a set number of divisions of equal distance around the circumference of the wheel. The number of divisions represents the resolution of the encoder. Each change in division is tracked to determine the change in wheel position. The disadvantage to creating physical divisions is that regardless of the speed of the wheel, the resolution of the encoder is set to a given divisions per rotation. Therefore at slower wheel velocities, less changes in division can be recorded. This is overcome by placing an accelerometer directly on the wheels of a ground based mobile robot for a more continuous sensor reading. The sensing information from this method would be similar with that of the information coming from a typical wheel encoder except that the resolution would be variable with speed. This is because the limitations in the amount of information collected about the wheel is now due to how quickly the sensor can be sampled vs. physical limitation imposed on current wheel encoding methods.

1.2 Motivation

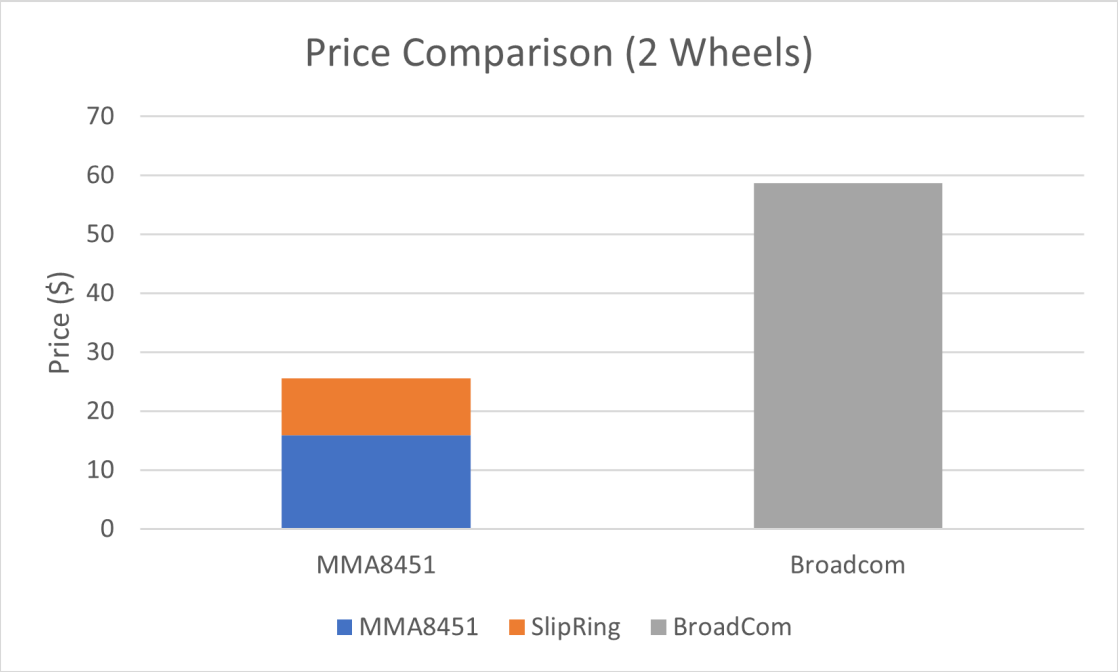
It may be noted that if the wheel encoder can already provide the information that the proposed solution with the accelerometer is offering, then there should be advantages to using the accelerometer instead of the traditional wheel encoder. These reasons can be analyzed by looking at cost, resolution, and the mounting of the accelerometer as an encoder verses other standard methods.

1.2.1 Cost Analysis

Wheel encoders generally fall in a price category based on the type. As mentioned previously, magnetic encoders are considered to be some of the most reliable encoders, as well as provide great resolution, but at a cost to price. In order to have an unbiased comparison, an encoder is picked based on similar performance as the accelerometer used in this research, regardless of type. Therefore a magnetic encoder will be used for comparison. The accelerometer being used in this research is the MMA8451 from Freescale Semiconductor[1]. It has a high end sampling rate of 800Hz. This means it has the potential to give feedback information 800 times a second. The traditional encoder used in comparison that can provide at minimum 800ppr is the Magnetic Encoder AEAT-6010-A06 from Broadcom[2].

The Broadcom encoder can give 1024 counts for every revolution of the wheel it is mounted on. This comes at heavy price with the new cost for getting comparable resolution at $29.32 per wheel. The MMA8451 already populated on a circuit board is priced on Adafruit at $7.95, with one being needed per wheel. Along with the MMA8451 sensor module, a slip ring must be used in order to allow the wheel to turn while keeping connection from the micro controller to the sensor and not twist the wires on themselves until tension causes a malfunction. For this research a Comidox, 6-wire slip ring is used and is priced at $4.83 at `https://www.ufontsa.com/index.php?main_page=product_info&products_id=216596`. This would bring the total

for monitoring 2 wheels with the suggested approach to $25.56 vs. $58.64 using the comparable Broadcom wheel encoder. Price comparison can be seen below in figure 1.1.



(a) Sensor Hardware Pricing

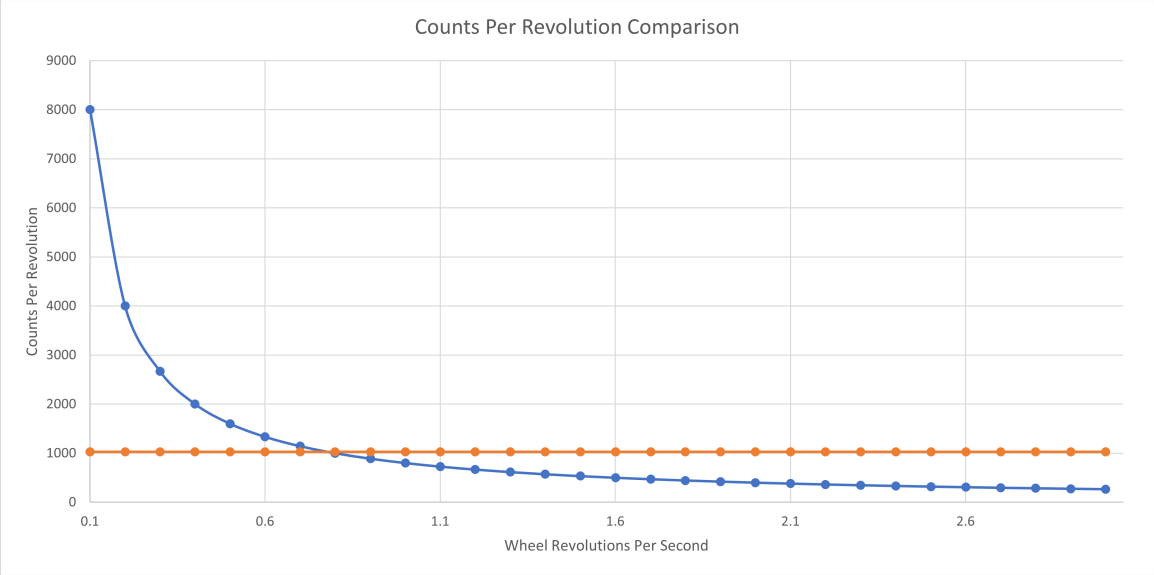
Figure 1.1: Above shows cost benefit of MMA8451 for wheel odometery vs. the similar performing Broadcom encoder.

It should also be noted that this does not include any other mounting cost for the magnetic encoder that may be incurred.

1.2.2 Sensor Resolution

Now that a more comparable wheel encoder has been selected, the resolution can be analyzed between the two sensors. The MMA8451 as stated previously can sample at a rate of 800 times per second. For our wheel period of 1.565s (38.339rpm), this means getting up to 1,252 samples within one revolution. It can be seen that the slower the wheel speed, the more samples that will be collected per revolution and means increased resolution of the wheel position. The wheel encoder is set to be 1024 pulses for every revolution no matter how slow the system. The resolution

of the wheel encoder only becomes better for wheel speeds faster than 46.875rpm. Therefore for applications with slow speeds, the accelerometer could be a preferred alternative. This can be seen in the following figure 1.2 that shows the number of revolutions per second vs. the number of counts from each sensor.



(a) Sensor Resolutions

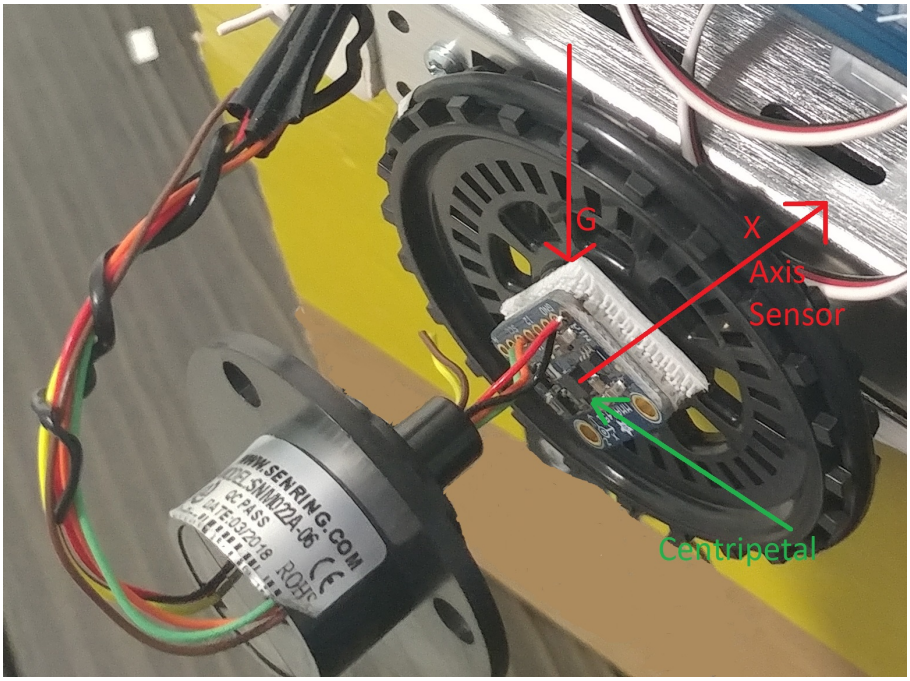
Figure 1.2: Resolutions of MMA8451 (blue) sensor and Broadcom (orange) over different wheel speeds in rotations per second.

The intersection of the lines shows how one would appropriately decide which sensor would be better for a project based on resolution. Given this research has a relative long wheel period, the MMA8451 is the best choice in the research scenario.

1.2.3 Sensor Mounting

Mounting is another consideration when choosing the correct sensor for the application. Without proper analysis of how to incorporate the sensor into the system, the mounting can become quite difficult and costly. For the magnetic wheel encoder, there are two parts. There is the polarized disk that mounts on the wheel, and the actual sensor for sensing the poles on the disk that is mounted on the wheel. The sensor generally has a sensitivity rating to tell how close it needs to be to the polarized

disk and also must be aligned properly. If this is not placed correctly, the sensor may give false information. For the accelerometer module, it just needs mounted to the center of the wheel using a preferred adhesive or mechanical insert method. Then the slip ring needs mounted parallel to the sensor so that as the wheel turns, the slip ring can move with it. This can mean having mounting hardware coming off the wheel itself to hold the slip ring in place. This removes any worry of the sensor information being lost. This setup is seen in figure 1.3 below.



(a) Mounting the Sensor

Figure 1.3: Sensor mounted on wheel with slip ring attached. Also shows x-axis, gravity vector, and centripetal acceleration

1.3 Previous Work

The Field of odometry based robot state estimation is a well explored field and the usage of accelerometers used to provide information on wheel movement has been previously studied as well. Accelerometers on wheels typically exploit the effect of the gravity vector on the accelerometer and orient the sensor so that 2 of the axis can measure the vector components of gravity. A simple usage is seen in the application

of an accelerometer to measure movement and direction of the wheel on a wheel chair [3]. This is done by measuring the accelerations on a radial and tangential (x and y) axis of the accelerometer and taking the inverse tangent of the two to get an approximate wheel position. Distances traveled were determined by the change in each sample taken of the wheel movement while the sign of the change would indicate the direction of movement. Speed of movement could then be calculated using the distances traveled over each time interval. Other experiments have been done to filter the information coming from the accelerometer on the wheel through a low pass filter for a more accurate position of the wheel [4]. The filter simply consists of a moving average for the sensor information read on the x-axis and y-axis. The direction of the wheel was determined by a magnometer also attached to the wheel. This allows the information from the accelerometer to be viewed as absolute values while the magnometer determines the sign for the change in position. Further work has improved the usage of accelerometer-based wheel odometry by using an inertial measurement unit (IMU) to get the angular velocity, from an on-board gyrosocpe combined with acceleration data from an accelerometer [5]. The data in this work is smoothed by using an Extended Kalman Filter. This means using the kinematic motion model of the wheel itself and comparing an expected output to the data received by the IMU.

1.4 Contribution

The work presented in this document aims to provide a method for using an accelerometer without the aid of other sensors to provide accurate position information for a wheel. It is desired to exploit the continuous sensor information of the accelerometer to always have feed back about the wheel. The model created for filtering will not have the benefit of gyroscopic angular velocity information or directional information from any other sensor sources. Therefore, the model must rely on command inputs sent to the motor to determine a state estimate. This command information will be

used to determine the expected influence of acceleration sources outside of gravity and how to predict the effects on the expected output of the sensor measurement. The previous works described in section 1.3 do not provide an effective way to calibrate the sensor distance from the center of the wheel that would be needed for determining centripetal acceleration. Some methods to overcome this were to place the sensor further from the center of the wheel in order for the effect of centripetal acceleration to be small. However, in this work, a robot with a small wheel is used, limiting the distance from the center of the wheel that is achievable. Also, to prove the accuracy of determining the centripetal acceleration, the accelerometer is placed as close to the center of the wheel as possible which will make the centripetal acceleration have more effect on the sensor. Therefore, it is desired to provide a procedure for determining this offset from the center of the wheel. By using command velocities to the wheel being analyzed and the found calibrated radius, accurate sensor expected outputs can be determined by the wheel motion model.

1.5 Topic Organization

In the following research, chapter 2 starts with the background information needed to understand how to implement the study. Chapter 3 covers the experimental set up and the hardware used to perform the study. Afterwards, chapter 4 explains in detail how to build the motion model needed to implement the Extended Kalman Filter and chapter 5 explains the implementation and build up of the filter itself. Lastly the results from the experiment are explained in chapter 6 along with the conclusion of the experiment in chapter 7.

CHAPTER 2: BACKGROUND

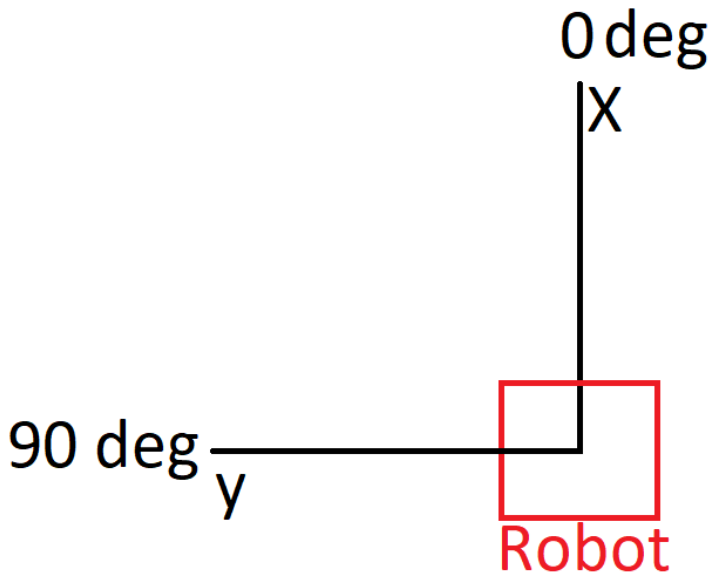
In order to implement this work, it is necessary to know the background information and will be described in this chapter. This includes knowledge of pose and coordinate systems in order to be able to understand how using odometry is used to determine robot state information. It will also be important to understand how traditional wheel encoders work to provide this odometry information. Then the output information of accelerometers will be explained. The final piece of information will be putting together a basic understanding of how the extended Kalman Filter works.

2.1 Pose and Coordinate Systems

To know the location of a mobile robot, it is important to know how the pose is treated and the coordinate frame of the world that the robot travels in. The pose of a ground based robot operating in only 2 dimensions includes a x and y Cartesian coordinates value, and the robot orientation in the environment [6]. This state can be illustrated as:

$$X = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{2.1}$$

The environment that the robot is in can be represented as a frame with a Cartesian coordinate system that has an origin that would give a pose of [0, 0 , 0]. The origin is typically at the center of the map, starts at one of the corners, or can be the starting position of the robot. The robot will also have its own coordinate frame and is shown below from a top view perspective of the robot:



(a) Robot Frame

Figure 2.1: Robot Cartesian coordinate frame with the origin at the robot center.

In figure 2.1, the positive x-axis is seen a pointing in front of and away from the robot. The positive x-axis also marks an orientation of zero degrees. The positive y-axis points to the left and away from the robot and marks an orientation of 90 degrees. The environment, denoted as the world frame will also use this same coordinate system layout.

2.2 Wheel Encoders

Wheel encoders are widely used for odometry information of a robot. Research has shown that position tracking of a robot based solely on odometry data can be reliable with little error in the actual physical position of the robot in an environment [7], [8]. Wheel encoding can be implemented in various methods. Encoders can be categorized as linear or rotary, and these two categories can also be subdivided into incremental or absolute[9]. The linear encoder is used to measure linear motion, while the rotary encoder is used to measure rotational motion. A rotary encoder would be used to monitor positional changes of a wheel. An incremental encoder receives a regular interrupt to its normal condition dependent on the type of encoder. Each

interrupt represents an even distribution along the wheel. These interrupts must be tracked to determine the wheel position. An absolute encoder uses a method that makes positions on the wheel unique. Therefore, the actual position of the wheel can be known.

Some rotary encoder categories would be mechanical, optical, and magnetic. These can be incremental or absolute as mentioned previously for rotary encoders. A commonly used incremental, mechanical encoder would be a potentiomter, commonly found in rotary switches, such as on thermostats [10]. A potentiomter translates rotational, mechanical motion into an electric signal that can then be used to determine how much the stator, or knob, has turned. These types of encoders have great resolution, but can become unreliable over time as parts wear out. This would mean having to replace the entire encoder on the wheel, which can be a difficult task.

A light based absolute encoder would use a disk of varying opaqueness. The differing opaqueness would be evenly spaced around the disk. A LED is would produce light onto the disk and on the other side, a photo-sensor sensitive to the level of light being received through the disk would be used to produce an electrical signal determining absolute position traveled [11]. These encoders are susceptible to dirt and dust. If the disk becomes contaminated with outside particulates, false readings of the photo-sensor can become inaccurate.

Magnetic encoders are reliable and they can offer high resolution [12]. These are typically deployed using a disk with alternating magnetic poles along each quadrant. These therefore sometimes referred to as quadrature encoders. A Hall effect sensor is then used to read the varying magnetic field changes as the disk is rotated. This get converted into an electrical signal for determining differentiation in wheel position. More poles can be added to the disk evenly spaced to give greater resolution. The downside to these types of encoders are that they are typically the most expensive and increase in price with the more precision that is desired.

The proposed method of placing an accelerometer on the wheel as encoder would be classified as an absolute, rotary encoder. The accelerometer can measure linear changes, but its use for this application is to measure rotational changes. It is absolute because of the sinusoidal output of the sensor. Each point on the sinusoid will correspond to a precise position on the wheel.

2.3 Accelerometers

Accelerometers are used to convert motion into electrical signals for determining the rate at which an object moved over a period of time. They consist of small membranes sensitive to accelerating forces on the 3 axis of x, y, and z [13]. As the membrane is moved due to accelerations, the membrane distance to its node differs. This creates a capacitance that can be measured and then translated to how much acceleration was acting on the axis.

Accelerometers have been used in robotics as input for tracking robotic motion. For ground based robots, they are typically placed near the center of the robot base and studies have shown reliable methods for using accelerometers to get accurate distance measurements [14]. The downside to accelerometers are that they are noisy because of their sensitivity and the measurements tend to drift from the true measurement. The research in this paper will use an Extended Kalman filter on the output of the accelerometer to overcome the drift and the noise of the measurement.

2.4 Extended Kalman Filter

A Kalman Filter is a belief based prediction and filtering algorithm for continuous systems [6]. This filter bases the predictions from a model of the system that is being observed and the prior belief of the state of the system. The noise of the system is assumed to be Gaussian, also known as a normal distribution. This means it will have a mean value representing the most likely state of our system with a variance that represents the noise in the system. The Kalman filter assumes that with a given

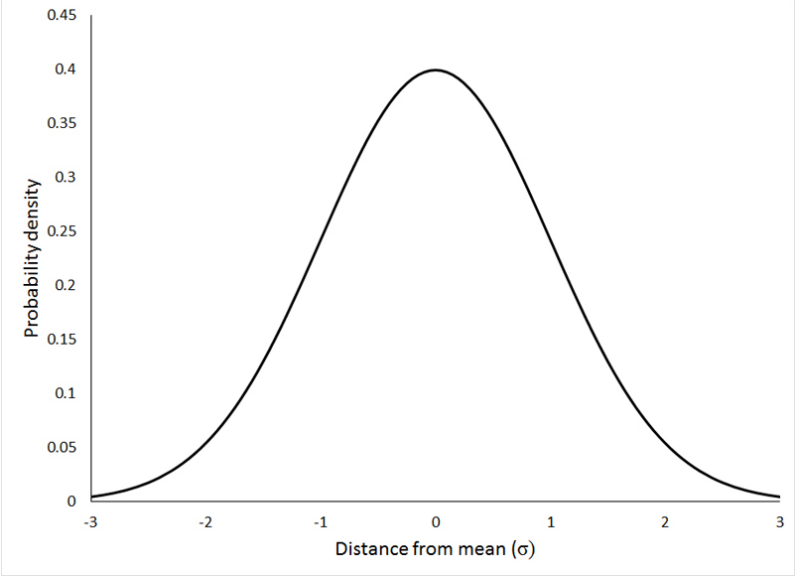
samples set of date observed from measuring a metric of a system, the output could be represented by the following equation:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-u}{\sigma})^2} \tag{2.2}$$

The above equation states that the system is able to be represented as having a mean u as the most likely value, and variance represented as the symbol sigma. The shorthand equation for representing this type of system is:

$$N(u, \sigma^2) \tag{2.3}$$

The shape of this system can be seen below in figure 2.2:



(a) Gaussian Curve [15]

Figure 2.2: This figure shows the general shape of a system that has noise that can be represented as having a normal distribution

Inputs from sensors that monitor the system are used as feedback to keep the prediction model from over confidently giving incorrect. The incorrect information in the system model is typically due to unforeseeable disturbances that a sensor can correct for. The algorithm for the Kalman filter can be written as [16]:

Algorithm 1 Kalman Filter algorithm for linear system

1: Prediction Step:

2: $x_k = a * x_{k-1}$

3: $p_k = a * p_{k-1} * a$ + q

4: Update Step:

5: $g_k = \frac{p_k}{(p_k+r)}$

6: $x_k = x_k + g_k * (z_k - x_k)$

7: $p_k = (1 - g_k) * p_k$

The Kalman Filter can broken down into two steps. These are the predict and update steps. During the prediction step, the current state is calculated from the previous state using a linear mathematical model. This is shown in line 2 of algorithm 1. Line 3 shows the prediction update for systematic error that exists from our prediction. The variable q, represents the noise that is part of the system being observed. The prediction state is usually calculated at regular time intervals and before each update step.

The update step occurs once a sensor observing the system has a value available. The time step can be chosen so that a sensor reading is available so that both a prediction and update are done on every interval. During the update step, a scalar called the Kalman gain is calculated and is shown in line 5 of algorithm 1. This gain is calculated from the systematic error of our prediction variable p_k and the sensor noise r. The next line is the state update that uses the new sensor value, current state estimate, and the newly calculated Kalman gain. It can be noted that for systems with sensor noise much greater than the systematic noise, the closer to zero the Kalman gain will be. This would mean in the state update step, the state would remain approximately what was calculated for the prediction. For scenario of the systematic error much greater than the sensor noise r, the Kalman gain is near one and the state update step will be approximately the sensor value that was observed.

The last line of algorithm 1, updates the systematic noise.

The Kalman Filter is reliable for linear systems. However, most systems cannot be observed in a linear way and limits the use of the regular Kalman filter. For systems with non-linear behavior, such as systems with sinusoidal motions, it has been shown that the Extended Kalman Filter (EKF) is a sufficient technique to estimate the state [17], [18]. In this version of the Kalman filter, the state variable, x_k can be represented as a matrix such as with the pose matrix from equation 2.1. The state transition equation from algorithm 1 is now represented as:

$$X_k = f(x_{k-1}, u_k)$$

The above equation states that like in the original linear Kalman filter, the next state is based on the previous and any inputs that can represented with u. The variable q, is now represented as Q to indicate that it is a matrix. This matrix is called the Co-variance matrix of the system. This matrix places the uncertainty of our non-linear system on the diagonal of the matrix. The below matrix shows the general layout of the co-variance matrix.

$$\begin{bmatrix} \sigma_{0,0} & 0 & ... & 0 \\ 0 & \sigma_{1,1} & ... & 0 \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ 0 & 0 & ... & \sigma_{m,n} \end{bmatrix} \tag{2.4}$$

The matrix is always a square matrix, meaning that the variable m and n above will be equal. The The sensor noise variable r, is also now represented as R since it can be providing multiple observations about our system, that will have their own uncertainties that need included. The matrix R will also follow the same outline as

the matrix from equation 2.4. This covers the multiple inputs for the EKF algorithm, but one of the most important parts of to the EKF is how non-linearity is dealt with.

Any non-linear function can be estimated linearly by differentiating over small time intervals. The smaller the time interval, the better the differentiated approximation is going to be. The EKF accomplishes this by taking the partial derivative of the state transition matrix with respect to each of its inputs. Inputs can include motion control, noise, and time. The partial derivative with respect to the inputs is stored in a matrix and represented as F. The Jacobian of the state transition matrix is then used to update the noise from our estimate P_k. With these improvements to the linear Kalman filter, the prediction step now looks as follows:

$$x_k = f(x_{k-1}, u_k) \tag{2.5}$$

$$P_k = F_{k-1} * P_k * F_{k-1}^2 + Q_{k-1} \tag{2.6}$$

Next, the Jacobian of the observation equation representing the sensor reading is also taken. All the partial derivative of the observation equation with respect to each of its inputs are all stored in a matrix H. The Kalman gain in line 5 of algorithm 1 was updated using the state estimation uncertainty variable p_k and the sensor noise variable r. These are now both matrices and therefore the Kalman gain will also be represented as a matrix labeled G. The Jacobian of the state observation equation will also be used to update the Kalman gain. The Kalman gain now looks as follows:

$$G_k = P_k H_k^T (H_k P_k H_k^T + R) \tag{2.7}$$

With the newly updated Kalman Gain, the state can be updated using the sensor observation. The Equation with, Z_k being the sensor inputs, is written as:

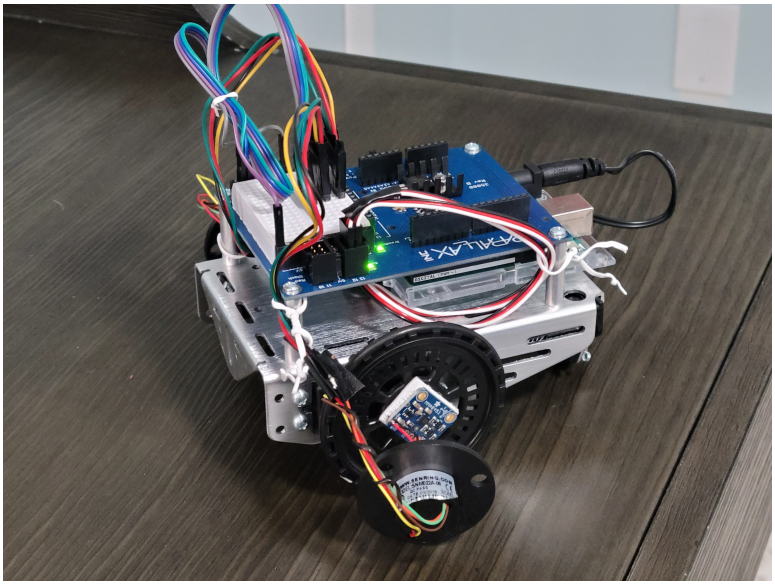
$$X_k = X_k + G_k * (Z_k - X_k) \tag{2.8}$$

As before with the uncertainties in the linear Kalman filter, the more trust on either the model or sensor reading, meaning the uncertainty is smaller, the more the values of the state variable is going to be towards either the predicted value or sensor readings respectfully. The last step of the update is the update for the uncertainty of our EKF. With I as the identity matrix, this is:

$$P_k = (I - G_k H_k) P_k \tag{2.9}$$

CHAPTER 3: CONFIGURATION

For this research, a Parallax BOEBot Robotics kit is used as the platform for testing and is intelligently powered by an Arduino Uno 3. A MMA8451 accelerometer Adafruit module is placed on one of the wheels of the kit. The wheel is driven by constant pulse width modulation (PWM) to keep as close to a relatively constant speed as possible. The connection from the accelerometer module to the Arduino is achieved by using a 6 wire slip ring. Only 4 need to be used with the I2C connection between the accelerometer and Arduino. The wheel is driven to have a full rotation about every 1.565s. The figure below shows the completed setup for the Parallax robot.



(a) Hardware Setup

Figure 3.1: The Parallax robot that was used for this research. Extra modifications can be seen for attaching the accelerometer to the wheel and using a slip ring for transferring data from the sensor to the Arduino Uno.

CHAPTER 4: MODELING THE EXPECTED SINUSOIDAL OUTPUT

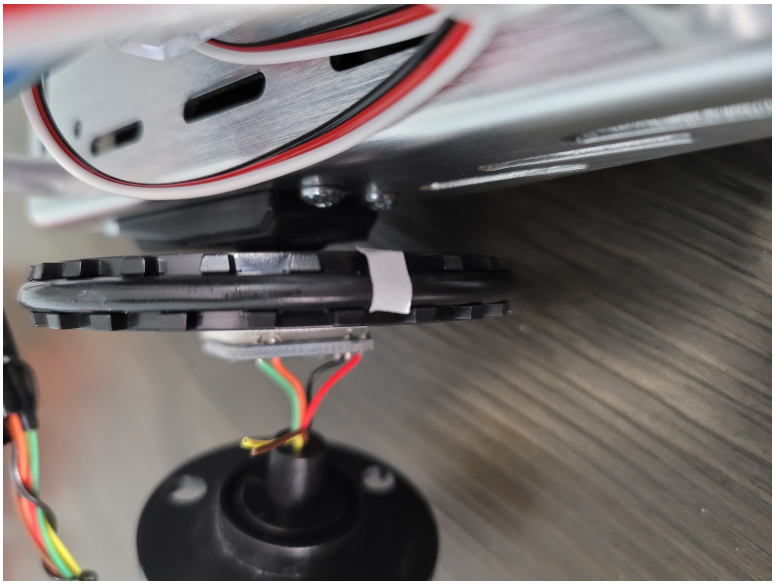
In order to use the accelerometer for reliable data output that is filtered using an EKF, a model must be created for estimating an expected output. For a perfectly placed sensor on the center of the wheel, an oscillating wave between a positive and negative average nominal value of the gravity vector would be expected. This would be between -9.80665 and +9.80665. This would mean that the model would look like:

$$9.80665 sin(2\pi f_w t) \tag{4.1}$$

, where t is the system time and f_w is the wheel frequency. This is the most simple model and will need to be expanded upon in order to capture all acting accelerations in the system as well as for any inaccuracies of user placement of the sensor itself.

4.1 Determining the Wheel Period

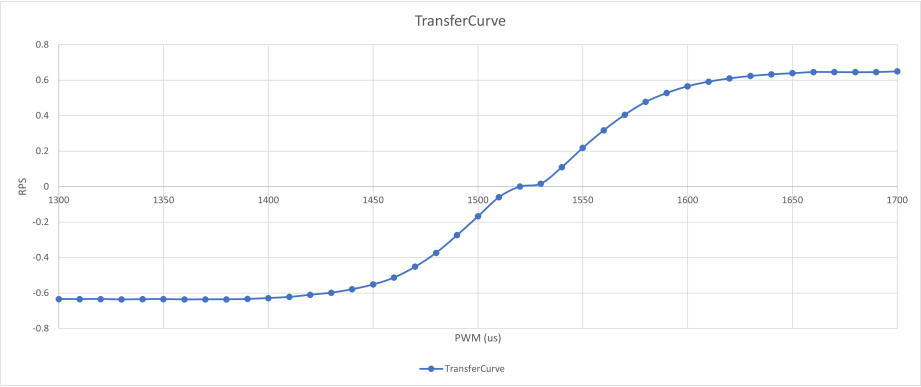
In order for the model to predict the expected position of the wheel over time, the wheel period needs to be known. Typically, this can be made obvious by knowing the expected velocity of the wheel. This was done by creating a look up table for wheel speeds given different motor driving PWM inputs. For the BOEBot robotics kit, the motor is driven using PWM inputs that range from 1300us to 1700us. According to the Parallax guide document [19], 1300us input is the fastest speed for clockwise rotation and 1700us input is the fastest speed for counter clockwise rotation. The closer to to the input of 1500us, reduces the speed of the wheel in either direction. In order to verify wheel speeds, the speed of the wheels at different input was measured and graphed. The measurements were done by placing reflective tape on the wheel being inspected as seen in figure 4.1 below.



(a) Reflector Tape Position

Figure 4.1: Light sensor is used to determine loop periods by differentiating between the black wheel color and reflective tape.

A light sensitive sensor is placed near the wheel. As the wheel moves, the sensor will read consistent values due to the uniformly black wheel color until the reflective tape intersects the sensing path. Many samples of the timing between each pass of the reflective tape were taken and averaged over 10us intervals over the range of 1300us to 1700us. The transfer curve looks as follows:



(a) Transfer Curve for PWM input to wheel speeds in rotations per second

Figure 4.2: Graph maps the PWM input to the wheel speed in rotations per second

The graph from figure 4.2 shows that 0 rotations per second is closer to an input of

1520us PWM input. also it can be seen that wheel speeds begin to not respond with much change once in the last 100us of each end of the curve. The velocity is just the multiplication of the frequency of the wheel by the circumference of the wheel.

$$V_{wheel} = f_w * C$$

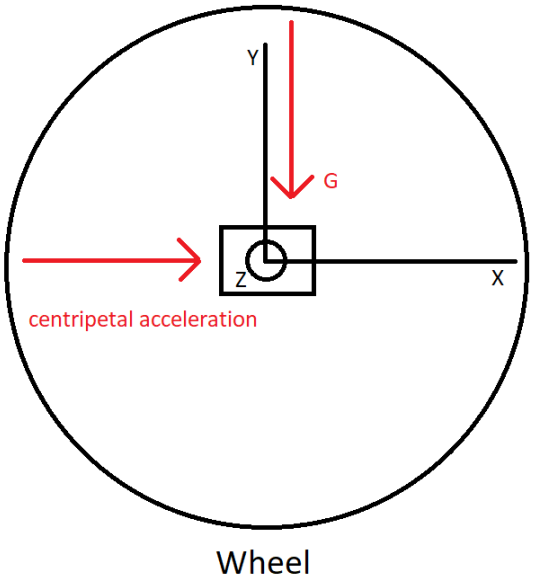
The above variable C is the circumference of the wheel. A PWM input of 1650us was used for the testing to give an approximate output of 0.639 rotations per second. With

$$T_w = \frac{1}{f_w}$$

where f _w is the frequency of the wheel just found in rotations per second, the wheel period is approximately 1.565s. For other velocity inputs, the period will be calculated by dividing the velocity by the wheel circumference, and then using the reciprocal of the result.

4.2 Adding Outside Acceleration Forces

The only issue with equation 4.1 is that it represents a perfect system. Any system will likely have offset error in the placement of the accelerometer module. Since the sensor is likely not perfectly centered, this will introduce a centrifugal acceleration that will need captured by the amplitude of the equation 4.1 and will increase the accuracy of the model as seen in other experiments [20]. The below image shows an outline of the sensor on the wheel and the acceleration vectors that would be seen during turning.



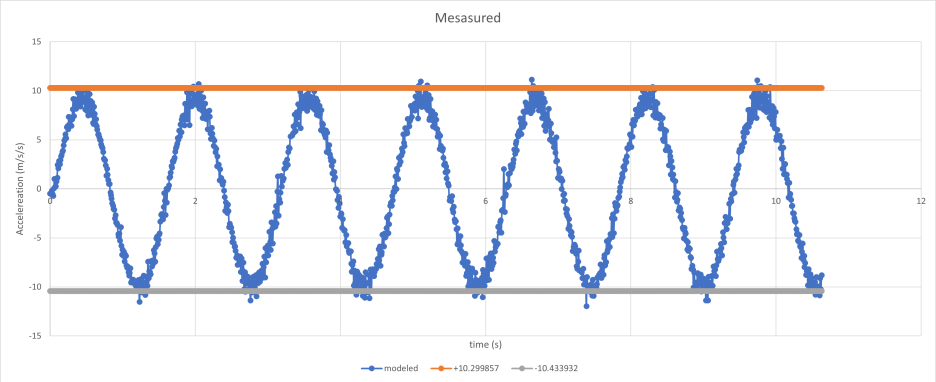
(a) Acceleration Vectors acting On Sensor

Figure 4.3: Gravity acceleration is constant from above. Centripetal acceleration is measured from any point on the edge of the wheel to the sensor in the center.

Figure 4.3 shows the two forces that must be accounted for when modeling our system. Gravity is taken as constant from above towards the ground. The centripetal force will be seen at any edge point of the wheel to the center where the sensor is located. This will be accounted for after looking at the effect of the offset in the sinusoidal model due to sensor offset from the center of the wheel.

4.3 Calibrating Zero Offset

There will likely be a offset in the oscillating amplitude, meaning it will not be centered around zero for a non-shifted sinusoid. This can be captured and adjusted for with some calibration. First the output of the sensor in place must be captured and graphed without any filtering.



(a) Accelerometer output over time

Figure 4.4: Accelerometer output from driving wheel with period 1.565s with measured amplitudes +10.299857 and -10.433932

From figure 4.4, it can be seen that the sinusoid is oscillation is measured between +10.299857mpss and -10.433932mpss. Using this information, the offset needed to center around zero can be found by

$$offset = \frac{abs(-Amp) - (+Amp)}{2} \tag{4.2}$$

With the given values from figure 4.4, this offset is -0.06708375mpss. This offsets now accounts for the sinusoid not centered around zero and will become part of our model equation. The centered Amplitude would be between positive and minus 10.3668945mpss and represented as CA. Using this new amplitude, the centrifugal acceleration can be computed by

$$A_c = CA - G = 0.5602445mpss$$

Given that the wheel diameter of the wheel is 66mm, the circumference is computed as

$$C = d * \pi = 0.207345115m$$

With the circumference and known wheel period, velocity is calculated as

$$V = C * f_w = 0.132489mps$$

The Sensor offset from center, r, can now be calculated from the following equation for the centrifugal acceleration.

$$A_c = \frac{V^2}{r} \tag{4.3}$$

$$r = \frac{V^2}{A_c} = 0.031332m$$

It is recommended multiple iterations of gathering sinusoidal amplitude data, determining the centripetal acceleration once gravity is removed, and then calculating the radius r be done. Then with a preferred averaging method, an accurate measurement for r will be acquired. Equation 4.3 shows that the placement of the sensor will affect how much centrifugal acceleration will need to be accounted for in the amplitude part of the model. The closer to the center of the wheel, {0, 0}, the sensor is, but not actually quite at zero, the more centrifugal acceleration that will be experienced. With a method to determine the centripetal acceleration available and the offset from zero available, equation 4.1 now becomes

$$(9.80665 + A_c)\sin(2\pi f_w t) + offset \tag{4.4}$$

Now that an estimate for the sensor distance from the center of the wheel is available, A_c can be recalculated using equation 4.3. This will account for the Velocity command sent to the wheel.

4.4 Determining Where on the Sinusoid to Start

There is still one more part of the model to account for. It cannot be guaranteed that the sensor will always begin at zero mpss as is expected by sine at system time zero. Therefore a method for determining a time offset must be accomplished. A model time can be calculated by

$$TM_i = \arcsin((X - offset)/(9.80665 + A_c)) * \frac{1}{2\pi f_w}$$

The inverse of sine will always return a value that tells which half of the sinsuoid to start in, as in the positive or negative part of the sinusoid. To overcome this, an average value for both the x and y axis are taken on start up of the system. Along with the calculated model start time, the x and y values helps to determine what quadrant of the sinusoid we are starting in and how to adjust our model time to match the desired quadrant. To analyze the x axis, it would look as such

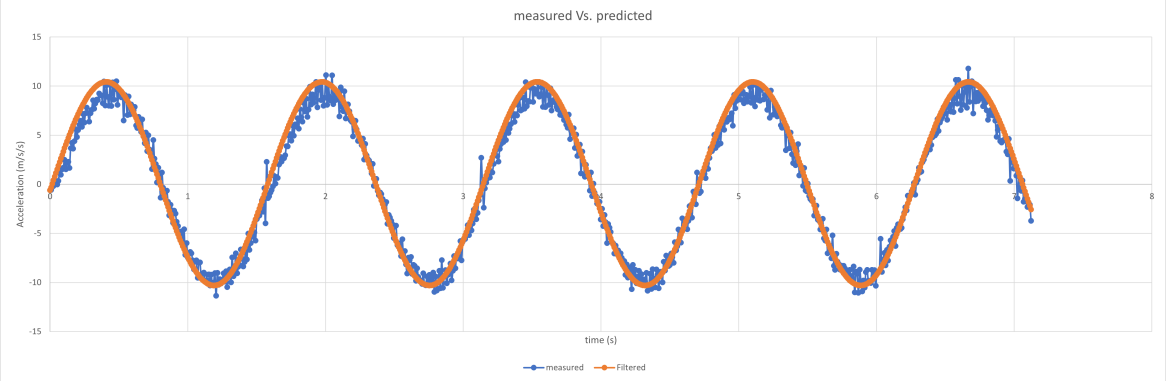
Algorithm 2 Algorithm to get start time for model

```
1: if X > 0 then
2:     if Y > 0 then
3:         q = 1;
4:     else
5:         q = 2
6:         TM_i = 1/(2f_w) − TM_i
7: else
8:     if Y > 0 then
9:         q = 4;
10:        TM_i += (1/f_w);
11:    else
12:        q = 3;
13:        TM_i = fabs(TM_i) + 1/(2f_w);
```

This analysis will initialize the model start to appropriately reflect the starting position of the sinusoid by initializing our first value to the averaged x value. Equation 4.4 now becomes

$$(9.80665 + A_c)\sin(2\pi f_w(t + TM_i)) + offset \tag{4.5}$$

Using the new model, the comparison to the sampled data from figure 4.4 can be compared to expected outputs of the model from equation 4.5 at each sampled time period of the raw sensor data by overlaying the two sinusoidal outputs.



(a) Modeled Output Data

Figure 4.5: Sample data (blue) graphed with model data (orange) for comparison

Figure 4.5 shows that the pure mathematical model follows very closely to the sinusoidal output of the sensor. This means that the sensor can have a predictable output. This is advantageous when building the Kalman filter in the next section for the sensor system.

CHAPTER 5: BUILDING THE EXTENDED KALMAN FILTER

The Extended Kalman Filter is used to filter a signal with Gaussian noise using a weighted value from the model and the sensor. The weights on these values determine how much trust to place on each value when fusing them together for the best final estimate. They represent the variance in the expected error of the measurement or prediction. As an example, an even weight would be to place weights of 0.5 and 0.5 for both the sensor measured value and the predicted value from the model. This would just give an average of the two for the filtered value. These weight can be moved up and down to determine how to get the best output empirically or an attempt to measure the expected error can be used.

5.1 Noise

As stated in the Arduino manual at pp. 260, table 28-1, the Arduino Uno can have a system clock error of %2 [21]. Therefore the system noise variance was first set as

$$\sigma_n = 0.02$$

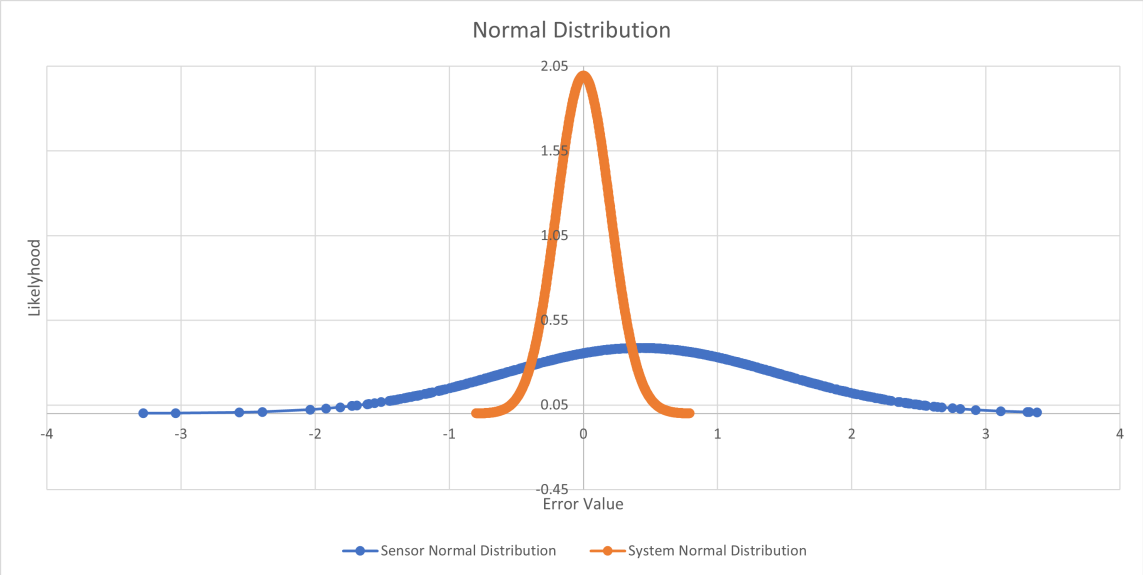
and empirically changed to narrow down desired filtered outputs. When analyzing figure 4.5, for each sample at time, t, the measured data, was compared to the model data. The model data is the ideal waveform data and therefore error was calculated at each point using

$$Error = ModelPoint - MeasuredPoint$$

From the list of errors, a standard deviation can be calculated in the error. This was determined to be

$$\sigma_g = 1.004$$

and was further empirically changed to get a desired output of the EKF. The normal distribution of both the sensor and the system can be seen in the following figure:



(a) Normal Distribution Curves

Figure 5.1: Sensor Distribution (blue) and System Distribution data (orange) for comparison

This shows that the error distribution of our system is much tighter than our sensor and will therefore be more reliable. Being more reliable means having a lower noise value as determined above. Likewise, changes to the measurement noise are higher to place more trust on the model and ultimately smooth the filtered output more. When analyzing figure 4.4, the output from the sensor is a noisy signal that seems to represent a sinusoidal figure. The data is not a constant decreasing or increasing output that would allow for easy determination of where on the sinusoidal output the model really should lie.

5.2 Transition Model

Building the EKF for the accelerometer measurements is largely similar to using a light sensor to detect the thread's position as it spins off a spool [22]. The current equation for our model from equation 4.5 is a time based estimate and does not use

any information about the system previous. An EKF should use prior estimations as part of the determination of the next estimate. This is difficult to do with a time based system. Therefore the state transition equation should then be represented similarly as

$$X = \begin{bmatrix} x_{t+1} = x_t + \frac{dx}{dt}_t * T \\ \frac{dx}{dt}_{t+1} = \frac{dx}{dt}_t + \sigma_n \\ h_{t+1} = sin(x_t) \end{bmatrix} \tag{5.1}$$

The first row of the transition matrix will represent the change in radians in each transition over time period, "T". The second row will be the change in radians for each transition. The last row will represent the new predicted estimated position of the accelerometer measurement. Equation 5.1 now becomes

$$X = \begin{bmatrix} x_t = x_t + \frac{dx}{dt}_t * T \\ \frac{dx}{dt}_t = 2\pi\ f_w + \sigma_n \\ h_t = (9.80665 + A_c)\sin(x_t) + offset \end{bmatrix} \tag{5.2}$$

It should be noted that the velocity is constant in the transition model. The update part of the EKF will correct the velocity to the value it currently is running.

5.3 Observation Model

The observation model represents the expected sensor output given the the variance of the sensor, also known as the noise. The model can be represented as

$$g_t = \begin{bmatrix} x_t + \sigma_a \\ \frac{dx}{dt} + \sigma_v \\ h_t + \sigma_g \end{bmatrix} \tag{5.3}$$

Each row of the observation has its own uncertainty associated with it. They are the uncertainty of the observed angle, velocity, and sinusoidal output. Since all observations from the measurement will be derived from the output of the sinusoid,

the uncertainties above will be

$$\sigma_a = \sigma_v = \sigma_g$$

The actual sensor input will be represented by

$$Y_t = \begin{bmatrix} arcsin(\frac{M-offset}{9.80665+A_c}) \\ (arcsin(\frac{M-offset}{9.80665+A_c} - x_{t-1}) * \frac{1}{T} \\ M \end{bmatrix} \tag{5.4}$$

Row 1 correlates to the angular position of the sinusoid. The correct quadrant can be determined by observing the sign of the other axis of the acceleromter. Row 2 is the change in velocity from the last state of the system and row 3 is the observed measurement, M, from the accelerometer.

5.4 Jacobians

Now that the transition model is set, the Jacobian Matrices must be found. These are used to linearize our non-linear system, taking many tangents across small time steps on our sinusoid. The first Jacobian is calculated taking the partial derivative of the state transition matrix, equation 5.2, with respect to the state variables x, dxdt, and h.

$$\frac{\partial X}{\partial x} = \begin{bmatrix} 1 & T & 0 \\ 0 & 1 & 0 \\ (9.80665 + A_c)\cos(x_t) & 0 & 0 \end{bmatrix} \tag{5.5}$$

Next, the partial derivative of the state transition equation with respect to the dynamic noises is taken as

$$\frac{\partial X}{\partial \sigma_n} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.6}$$

The partial derivative of the observation equation with respect to the state transition variables x_t, dx/dt, h is

$$\frac{\partial g}{\partial x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.7}$$

Then lastly the partial derivative of the observation equation with respect to sensor noises are in the following Jacobian matrix

$$\frac{\partial g}{\partial \sigma} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.8}$$

5.5 Co-Variance

The co-variance of they system noise will be represented by

$$Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \sigma_n & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.9}$$

The co-variance of the sensor noise will be represented by

$$R = \begin{bmatrix} \sigma_a & 0 & 0 \\ 0 & \sigma_v & 0 \\ 0 & 0 & \sigma_g \end{bmatrix} \tag{5.10}$$

5.6 Determining Distance Traveled

The distance traveled can be determined by measuring the change in wheel position over the circumference of the wheel for each time interval, T, and summing up all the values as the experiment is running. The method for determining the distance is to find the position in radians of the wheel based on the filtered output. This is similar to the first row of equation 5.4 and outlined below

$$\theta_{rad} = \arcsin(\frac{X[2] - offset}{9.80665 + A_c}) \tag{5.11}$$

As with finding the starting model time, sine will only return the positive or negative half of the sinusoid. Therefore the output must be adjusted similarly. The starting quadrant was already found using the code in algorithm 1. Now with each update of the model, the quadrant needs tracked and will therefore allow for the distance to be tracked as well. The new determined quadrant of each iteration will be compared to the last quadrant to help determine distance traveled. Also, before every prediction/update step of the EKF, the current state needs saved into a variable to help determine when to change quadrants. The distance over the travel interval, T, is calculated as:

$$\delta d_m = \theta_{rad} * \frac{C}{2\pi} \tag{5.12}$$

Determining the quadrant can be based on checking the other axis of the accelerometer or by using the following algorithm for tracking code for quadrants and distance, based on equation above.

Algorithm 3 Algorithm to get change in angle in radians

```
1: lastQ = q;
2: if q == 1 then
3:     if X[2] < lastPrediction then
4:         q = 2;
5: else if q == 2 then
6:     if X[2] > 0 then
7:         q = 3;
8: else if q == 3 then
9:     if X[2] > lastPrediction then
10:        q = 4;
11: else
12:    if X[2] > 0 then
13:        q = 1;
14: // update calculated angle based on new quadrant
15: if quadrant == 1 then
16:    θ_rad = fabs(θ_rad);
17: else if quadrant == 2 then
18:    θ_rad = PI − θ_rad;
19: else if quadrant == 3 then
20:    θ_rad = fabs(θ_rad) + PI;
21: else
22:    θ_rad += 2 ∗ PI;
23: // Sum change in distance
24: if lastQ == 4 AND q == 1 then
25:    d_M += θ_rad ∗ C/(2 ∗ PI);
26: else
27:    d_m += (θ_rad − θPrev_rad)(C/(2 ∗ PI);
```

5.7 Putting it all together

As stated before, the EKF is broken up into two steps. This would be the prediction and update steps.

5.7.1 Prediction

For the prediction part of the EKF, the state transition matrix will be updated every sample interval of time, T, using equation 5.2. Then the co-variance matrix, S, of the system must be updated. For the experiment, the co-variance is a 3x3 matrix that is initialized with ones on the diagonal to start. The prediction step of the co-variance will be as such

$$S = (\frac{\partial X}{\partial x})S(\frac{\partial X}{\partial x})^T + (\frac{\partial X}{\partial \sigma_n})Q(\frac{\partial X}{\partial \sigma_n})^T \tag{5.13}$$

Since the partial derivative of the transition equation 5.2 with respect to the system noise is all zero except for the 1 on the diagonal, the transpose of this matrix will be the same matrix.

$$\frac{\partial X}{\partial \sigma_n}^T = \frac{\partial X}{\partial \sigma_n}$$

Also since Q is all zero except for the systemNoise value on the diagonal, we have

$$Q * \frac{\partial X}{\partial \sigma_n} = Q$$

Therefore equation 5.13 becomes

$$S = (\frac{\partial X}{\partial x})S(\frac{\partial X}{\partial x})^T + Q \tag{5.14}$$

5.7.2 Update

The first step to determine the update to the estimated output is to determine the Kalman gain. The Kalman gain uses the system noise and sensor noise to weight the

tendency of the output more towards the sensor value or the predicted value from the model dependent on how much noise for the sensor measurement and noise for the model prediction that were set previously. The Kalman gain is found by

$$K = S(\frac{\partial g}{\partial x})^T * [(\frac{\partial g}{\partial x})S(\frac{\partial g}{\partial x})^T + (\frac{\partial g}{\partial \sigma})R(\frac{\partial g}{\partial \sigma})^T] \tag{5.15}$$

The Jacobian of the observation equation dg/dx is the identity matrix. The transpose of the identity matrix is the identity. Therefore

$$(\frac{\partial g}{\partial x})S(\frac{\partial g}{\partial x})^T = S$$

Also since dg/dsigma is the identity matrix, the transpose is also the identity matrix. This means that

$$(\frac{\partial g}{\partial \sigma})R(\frac{\partial g}{\partial \sigma})^T = R$$

Therefore equation 5.15 can be reduced to

$$K = S * (S + R) \tag{5.16}$$

Next, the state prediction is updated using the newly found Kalman gain. This is done by

$$X = X + K[Y - g] \tag{5.17}$$

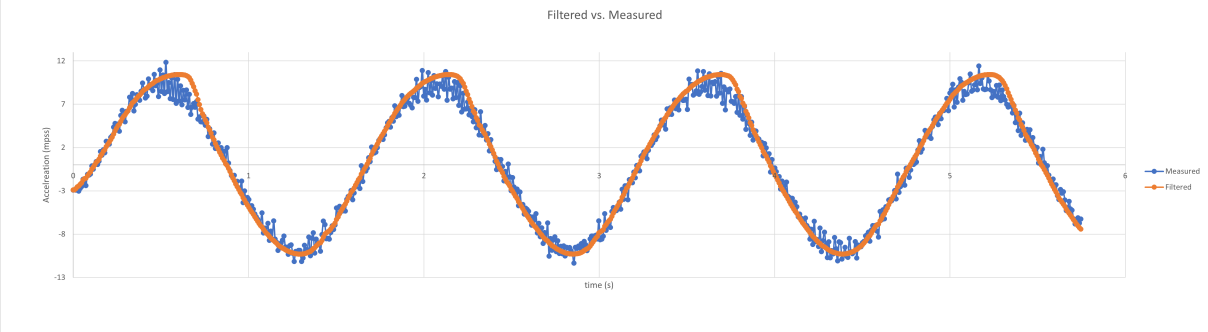
Equation 5.16 shows that the smaller the variance in the system noise compared to the variance in the sensor noise, the smaller K will be. This shows that more trust is placed into the system model prediction rather than the sensor measurement. K will be larger if more trust is placed into the sensor measurement. It can then be shown that in equation 5.17, the larger K is, the more the new state estimate is going to lean towards the measured sensor value. For smaller value of K, the new state estimate

will lean more towards the model prediction. The last piece of the EKF is to update the system co-variance. This is done with

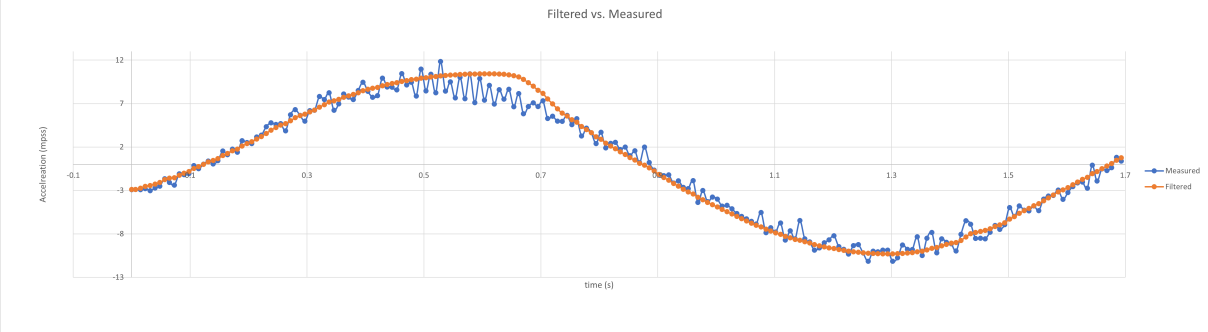
$$S = [I - K * (\frac{\partial g}{\partial x})]S \tag{5.18}$$

CHAPTER 6: RESULTS

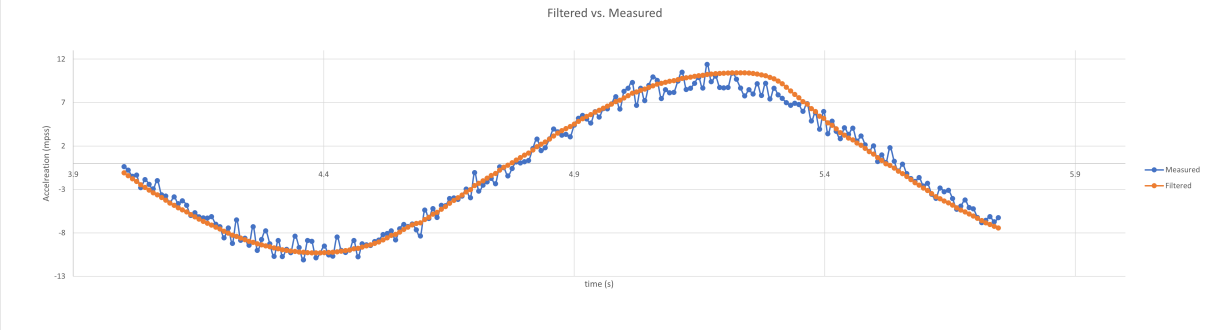
Using the EKF model decribed in the previous chapter, the following results were obtained:



(a) EKF Results over 5.75s



(b) EKF Results over first 1.69s



(c) EKF Results over last 1.74s

Figure 6.1: Comparison of measured output (blue) and filtered output (orange)

Figure 6.1 shows a full filtered output over 5.75s. The following two sub-graphs give a closer look of the filter at the start of the experiment as well as at the end.

It can be seen that the filtered output is a continuous wave with the noise vastly reduced. The filter deviates slightly at the top of the sinusoid, but the EKF corrects the estimate as designed. This deviation means that the error will oscillate. The difference of the filtered output to the sensor measurement can be affected by any number of disturbances in the system. These disturbances can cause the assumed constant velocity of the wheel to deviate. This could be overcome by placing more trust into the sensor measurement, but could add more of the inherent sensor noise back into the estimate. The new value does produce a wave with a smooth, continuous oscillation that is lacking the original noise. This allows for a good estimate of the wheel position in relation to the starting position. A noisy wave can cause false assumptions at discrete measurements about the position on the sine wave giving the wrong position of the wheel in turn.

The experimental results of the amount of travel for the test wheel was 0.759m. The expected amount of travel over 5.75s given the velocity from chapter 3 as 0.13133151mps, was 0.755m. The error is ultimately

$$\%error = 100 * abs[(exp - meas)/exp] = \%0.53 \tag{6.1}$$

The expected distance is compared to the filtered output and graphed.

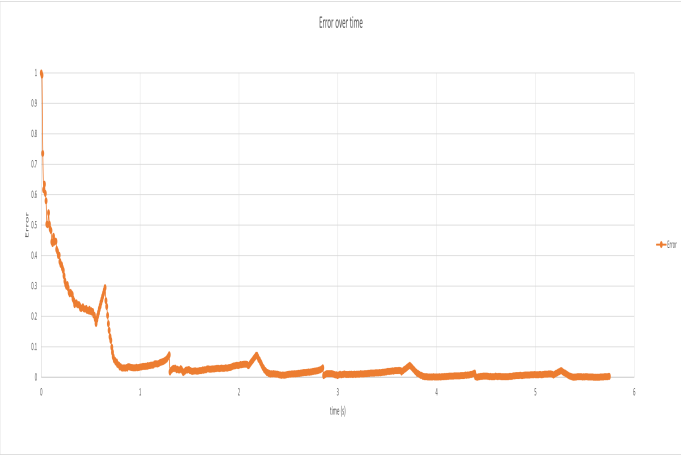
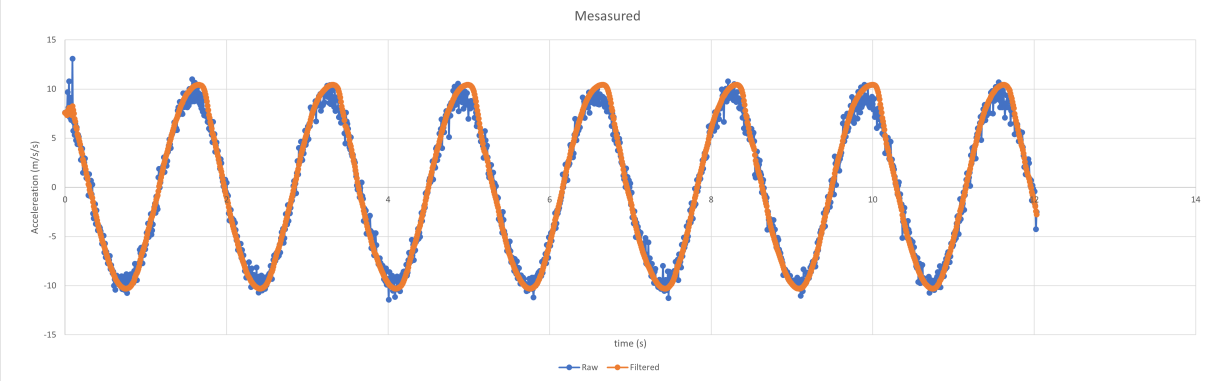


Figure 6.2: Error of filtered Output compared to Expected value

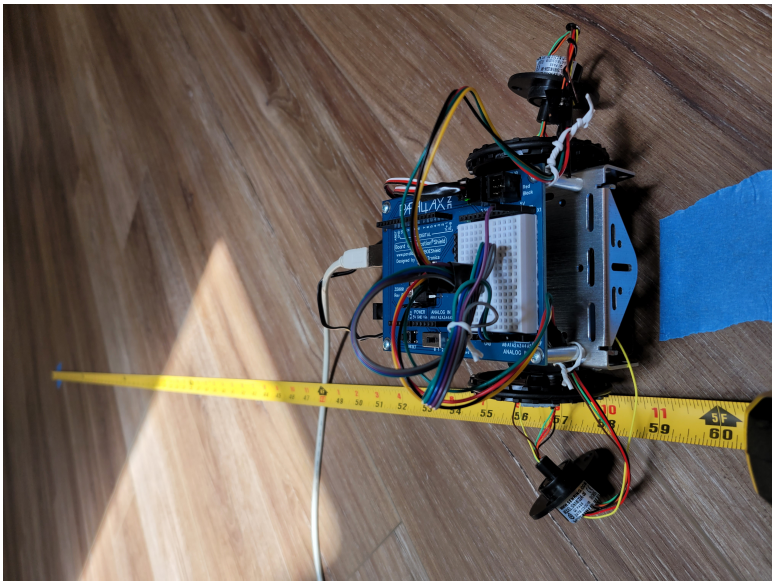
The error starts out high for the expected vs. the filtered output. This potentially means that the original starting estimate was a off, but the more the filter runs, the smaller the error becomes. The final second of error oscillates between %.007269 and %2.20. The small peaks are due to the slight overshoot of the filter from the measured sinusoid. Figure 6.2 also shows that the error appears to decrease the longer the filter is run. This shows that error is not inclined to rise. So for any error, E, at a time, t, error at time, 2t, will not correlate to an error of 2*E. The expected error will continue to decrease while oscillating between a set range. The recovery in error shows that the measurement over full revolutions are highly accurate and little information is lost.

Further testing was done to test the robustness of the EKF for further determining the viability of using the sensing method in practice. The first test done was to drive the robot straight on a flat surface to verify the distance measured by the previously described algorithm vs. a hand measured distance. The results are shown in the following figure as well as an image of the environment that the robot was operated in.



(a) EKF Results on hard surface

Figure 6.3: Very little disturbances other than some floor friction

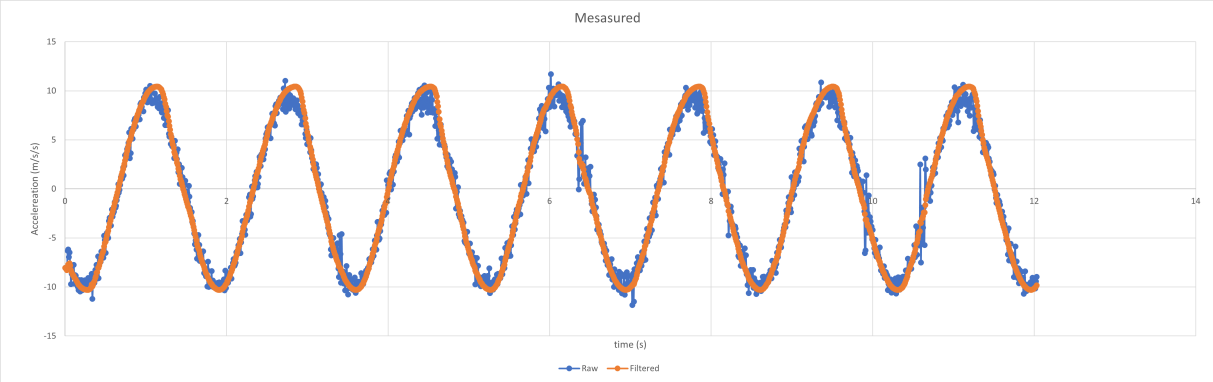


(a) Floor type = vinyl

Figure 6.4: Hard floor vinyl

The results of this experiment output a measured distance of 1.485m after running for nearly 12 seconds. With a constant velocity of 0.132m/s, the expected distance traveled by the left wheel would have been 1.584m. However, the measured distance came out to 1.4986m. The distance from the algorithm ended up being within 0.91% of the hand measured value. This shows that there is likely a slight mismatch in previous assumption or measurements. This could include mismatches in actual velocity given it was measured without any load, but using the EKF to fuse expected values and sensor values still produce accurate results.

The next experiment was done on some tile floor. The tile floor was mostly a smooth surface, but did have some disturbance events that would be introduced at the grout line. The results and picture environment can be seen in the next two figures.



(a) EKF Results on tile surface

Figure 6.5: Disturbances introduced by grout lines

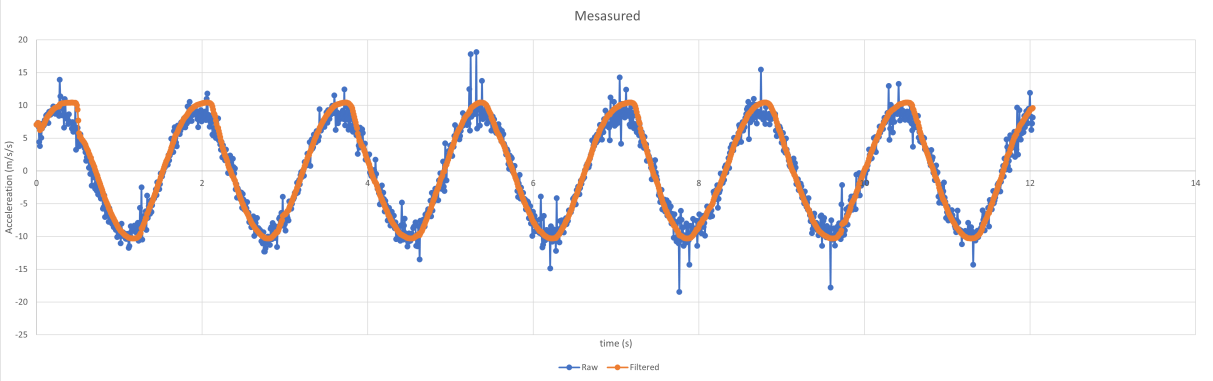
(a) Floor type = tile

Figure 6.6: Hard floor tile

When looking at the results with some disturbances, the raw data (blue) shows where the sensor data is completely off. An example can be seen around 10.59s where the data should be approaching zero on the rising sinusoid, but spikes can be seen on the rise. The EKF (orange) appears to continue rising smoothly to help keep giving accurate results. The measured distance from the algorithm was 1.74m and the hand measured value was 1.759m. This is an error of 0.99%. The total time of the run was

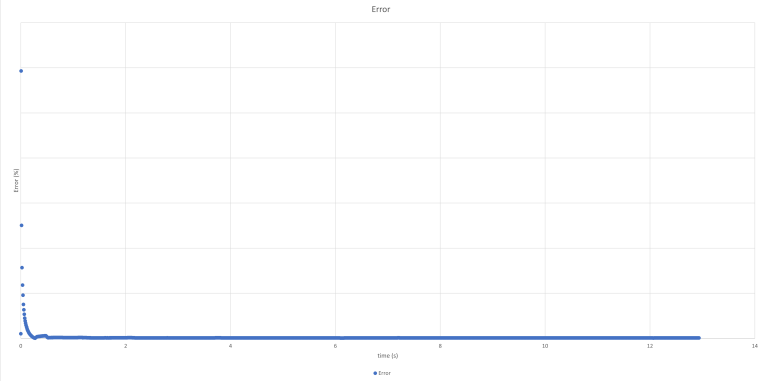
14.11s. This would mean an expected travel distance of 1.870m. This again shows the importance of the EKF and the importance for sensor feedback.

The last experiment was done on a system intended to introduce disturbances at a much higher rate than the tile floor. The experiment was done on a brick pattern entryway with multiple grout lines at about every 0.0889m vs. the tile introducing grout lines every 0.4445m. The results, measured error, and environment images are seen below.



(a) EKF Results on brick patio surface

Figure 6.7: Disturbances introduced by grout lines



(a) EKF Error on brick patio surface

Figure 6.8: Error over time

(a) Floor type = brick

Figure 6.9: Brick Patio

The initial estimate for the EKF appeared to be very wrong and was even moving in the wrong quadrant. This is seen in the beginning as the filters starts out well above the initial sensor reading and tries to move towards zero as if in the second quadrant of the sinusoid where it should have started in the first quadrant and move towards the peak of the sinusoid. It is not until a very under confident estimate is brought back in line of the sensor measurements around 0.54s that the filter begins to accurately measure the wheel position. Spikes in sensor measurements are seen periodically in the raw measurements due to the grout in the brick surface and is most notable around 5.31s. Even in this environment, the filter still produces usable results. The measured distance from the algorithm after 12.9s was 1.57413m where the hand measured results were 1.5748m. The is an error of 0.04%. The expected distance based on perfect assumptions was 1.71m. These results help to verify the usability of the accelerometer as a valid wheel odometer when used with an appropriate EKF and modeled well.

CHAPTER 7: CONCLUSION

The desire of this study was to use a standalone accelerometer and provide high accuracy position information about the wheel while exploiting the continuous sensor input of the accelerometer. The work done also was attempting to satisfy finding a method for determining an accurate description of the sensor's offset from the wheel center in order to accurately predict centripetal acceleration. This information was placed in a system model that was based on the expected sinusoidal reading of the gravity vector on one axis with the centripetal acceleration added as well. This was successfully implemented and the result of the model using the calibrated found radius could be seen in figure 4.5. It could be seen from figure 4.4 that the original accelerometer data for tracking the wheel position based on gravity, was too noisy for comprehensive analysis if accurate position was desired. The sinusoidal model was placed into an EKF and used to estimate wheel position. The model is not expected to be accurate in systems with disturbances. The Extended Kalman Filter is used so that if the estimate begins to drift from the actual sensor data due to these disturbances, the estimate will then be brought back to the proper tracking position. This is seen in figure 6.1. As the motion model drifts away from the sensor data, the sensor data forces the model back into a proper range of the sensor data. With the new accuracy provided by the estimate, this allows for the accelerometer to be a suitable choice for tracking wheel odometry. The method of using the accelerometer along with a well designed EKF was tested in multiple environments and was proven successful while the final result never went above 1% in error. This proposed method is shown to provide more continuous measurements at lower frequencies in contrast to the the typical wheel encoding methods. Using this continuous and accurate wheel position information and knowing the circumference of the wheel, distance information traveled by the wheel can used to determine a robot's pose. Also, by using the change in the position of the wheel, the velocity of the wheel

can be fed back into the robot system. Using this wheel velocity information and the kinematic model of the robot, a pose can be determined containing the robot coordinates in the world as well as heading.

7.1 Future Work

It is noted that improvements can be made by modeling the motors driving the wheels to have a more accurate assumption about the actual wheel speed and include any rise times for the start up of the motor, potentially giving a lower error on start up. The modeled motor can then be run through a controller to help account for disturbances in the system and provide more consistent expected wheel speeds. This will lead to more accurate assumption about the period of the wheel and provide better estimates of the wheel position. It should be noted that this method of sensing is restricted to ground based robotics on a level surface. Therefore, to remove this restriction, one can add an accelerometer to the center of the robot in order to be able to account for non-level surface environments. The connection of the accelerometer to the micro-controller using the slip ring potentially added torque on the motor on start up. This would cause the initial belief of the sensor value to be off as well until reaching a steady state. Furthermore, the wires are not well centered on the wheel. This would cause an elliptical circle pattern that the wires connecting to the sensor would follow. This may cause loosening and tightening of the wires placing temporary restriction on the wheel movement. This can be fixed with better alignment of the slip ring towards the wheel center. Another solution can be to use blue tooth and battery hardware instead of directly powering the sensor. This would add cost, but should still be cheaper than the magnetic encoder mentioned in the paper. Further work to validate this would be to implement this solution on different robot types and use the kinematic models of those robots to verify accurate pose information is able to be calculated. Long term testing should be done to determine the longevity of the accelerometer and slip ring combination as well.

REFERENCES

[1] Freescale Semiconductor. *Xtrinsic MMA8451Q 3-Axis, 14-bit/8-bit Digital Accelerometer* (2013). Accessed Oct. 20, 2020. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/MMA8451Q-1.pdf.

[2] Avago Technologies. *AEAT-6010/6012 Magnetic Encoder, 10 or 12 bit Angular Detection Device* (2011). Accessed Oct. 20, 2020. [Online]. Available: https://docs.broadcom.com/doc/AV02-0188EN.

[3] E. Coulter, P. Dall, L. Rochester, J. Hasler, and M. Granat, "Development and validation of a physical activity monitor for use on a wheelchair," *Spinal cord*, vol. 49, pp. 445–50, 03 2011.

[4] J.-D. Huang and T.-W. Wang, "Accelerometer based wireless wheel rotating sensor for navigation usage," in *2011 Fifth International Conference on Sensing Technology*, pp. 565–568, 2011.

[5] B. Gersdorf and U. Frese, "A kalman filter for odometry using a wheel mounted inertial sensor," *ICINCO 2013 - Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics*, vol. 1, pp. 388–395, 01 2013.

[6] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. Cambridge, Mass.: MIT Press, 2005.

[7] A. Jha and M. Kumar, "Two wheels differential type odometry for mobile robots," in *Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization*, pp. 1–5, 2014.

[8] M. A. Mahmud, M. S. Aman, H. Jiang, A. Abdelgawad, and K. Yelamarthi, "Kalman filter based indoor mobile robot navigation," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pp. 1949–1953, 2016.

[9] Anaheim Automation, Anaheim, CA, USA. "Encoders: Optical and Magnetic, Incremental and Rotary" (2020). Accessed Oct. 20, 2020 [Online]. Available: https://www.anaheimautomation.com/manuals/forms/encoder-guide.php unpublished.

[10] S. Paul G, A. Jason L, J. Gray, and R. Tim M, "Thermostat with mechanical user interface," Jan. 9 2007. Patent Number US 7,159,789 B2.

[11] U. Hideki, "Optical absolute rotary encoder," May 11 2010. Patent Number US 7,714,272 B2.

[12] K. Miyashita, T. Takahashi, and M. Yamanaka, "Features of a magnetic rotary encoder," *IEEE Transactions on Magnetics*, vol. 23, no. 5, pp. 2182–2184, 1987.

[13] W. Norman F, "Accelerometer system," July 22 1986. Patent Number US 4,601,206.

[14] C. Garcia-Saura, "Self-calibration of a differential wheeled robot using only a gyroscope and a distance sensor," 2015.

[15] D. Zelmer, "The normal distribution." (2010). Accessed Oct. 20, 2020. [Online]. Available: "http://sciences.usca.edu/biology/zelmer/305/norm/" unpublished.

[16] L. Simon D, "The extended kalman filter: An interactive tutorial for non-experts." (2020). Accessed Oct. 20, 2020. [Online]. Available: https://simondlevy.academic.wlu.edu/kalman-tutorial/ unpublished.

[17] B. Kaewkham-ai and K. Uthaichana, "Comparative study on friction compensation using coulomb and dahl models with extended and unscented kalman filters," in *2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 191–195, 2012.

[18] M. Sun and Z. Sahinoglu, "Extended kalman filter based grid synchronization in the presence of voltage unbalance for smart grid," in *ISGT 2011*, pp. 1–4, 2011.

[19] L. Andy. Parallax Inc. *Robotics with the BOE Sheild-Bot for Arduino* (2020). Accessed Oct. 20, 2020. [Online]. Available: https://learn.parallax.com/tutorials/robot/boe-bot/robotics-boe-bot.

[20] L. Armesto, S. Chroust, M. Vincze, and J. Tornero, "Multi-rate fusion with vision and inertial sensors," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 1, pp. 193–199 Vol.1, 2004.

[21] Atmel Corporation. *ATmega328P, 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash Datasheet* (2015). Accessed Oct. 20, 2020. [Online]. Available: $http : //ww1.microchip.com/downloads/en/DeviceDoc/Atmel - 7810 - Automotive - Microcontrollers - ATmega328P_Datasheet.pdf$.

[22] A. Hoover (2020) Lecture Notes: EKF sinusoid example [Online]. Available: http://cecas.clemson.edu/ ahoover/ece854/lecture-notes/lecture-ekf-sine.pdf.