INDOOR AND OUTDOOR LOCALIZATION OF A MOBILE ROBOT FUSING SENSOR DATA

A Thesis Presented

By

Md Maruf Ibne Hasan

to

The Department of Electrical and Computer Engineering

in partial fulfillment of the requirements

for the degree of

Master of Science

in the field of

Electrical & Computer Engineering

Northeastern University

Boston, Massachusetts

August, 2017

ACKNOWLEDGEMENTS

Firstly, I would like to thank my family members for their constant support and encouragement throughout my graduate studies. They have been a source of comfort during my difficult times and supported me mentally and financially throughout my degree.

Furthermore, I am grateful to my thesis advisor, Prof. Taskin Padir, whose teachings and guidance enabled me to better understand my thesis project which lead to a successful outcome. I sincerely thank him for his patience and support he has shown towards me throughout my thesis project.

In addition, I would like to thank the other members of this project, Rui Luo, Erik Silva, Mitch White and Jingwei Luo who have been a great support in solving the difficult challenges of this project.

I would also like to show gratitude to the thesis committee members, Prof. Hanumant Singh and Prof. Aatmesh Shrivastava for their questions and suggestions which helped me gather valuable insight.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

One of the major challenges of mobile robot navigation is to know the robot’s position and orientation with respect to an environment. This is known as robot localization. In order to get an accurate pose estimation, we cannot rely only on measurements from a single sensor source. Such sensor sources are susceptible to noise and introduces errors in the measurement. The error accumulates with time leading to a wrong estimate of the pose of the robot. To overcome this problem, fusion of sensor data coming from multiple sensors can be used to locate the robot in a known or unknown environment.

To calculate the pose of our mobile robot we use sensor information from incremental wheel encoders, Inertial Measurement Unit (IMU) and a GPS. We get information about how much the wheel encoders are rotating and extract odometry information. The IMU provides orientation, linear and angular velocity and linear acceleration and the GPS provides absolute position information in longitude and latitude. These sensor measurements along with their corresponding errors are fused and fed into an Extended Kalman Filter. The Extended Kalman filter uses these information and weighs between incoming measurements and its prediction to get the most accurate estimate of the robot’s current position compared to a single sensor used in isolation.

Chapter 1

Introduction

A critical challenge for mobile robot navigation is the ability to answer the question, “Where am I?”, “Where am I going?” and “How should I get there?” [1]. The solution to the first question is known as localization. Localization can be done using different sensors either in isolation or in combination which can provide both proprioceptive and exteroceptive data [2]. Several localization methods have been developed which can be classified into two broad categories: absolute and relative localization [3]. Absolute localization methods use positioning methods such as Global Navigation Satellite System (GNSS). On the other hand, relative localization uses methods such as inertial navigation and odometry from dead reckoning to determine a robot’s position and heading. However, dead reckoning methods are susceptible to unbounded growth of time integration errors with the increase of distance travelled by the robot and is unavoidable [4].

The goal of this project is to use a wheelchair which would autonomously drive itself and navigate from point A to point B with minimum or no control inputs from its user. The wheelchair would act as an aid for disabled people who are unable to move from one place to another without any support and who have little to no motor skills in their hands. We plan to make this wheelchair drive autonomously from Boston Home (2049 Dorchester Ave, Dorchester Center, MA 02124) to Symphony Hall using public transport such as the subway. There are four major parts of this project: mapping the environment, path planning, localization and obstacle avoidance. The experiments and results presented here are on the performance of wheelchair localization.

1.1 Problem Statement

A mobile robot often requires the fusion of measurements from multiple sensor sources for accurate state estimation. However, most of the sensors are imperfect and susceptible to noisy measurements. Fusing data measurements from multiple sensors can be a solution to this problem [2]. This will lead to an overall position estimate whose error would be less compared to the estimation error of a single sensor used in isolation. Often it has been observed that a greater amount of sensor input data will produce more accurate position estimates [2]. But it is critical to have the right selection of sensors.

Among the pool of absolute and relative positioning methods, Global Positioning System (GPS), Inertial Navigation System (INS) and dead reckoning are some of the most commonly used systems for mobile robot navigation [2]. The GPS provides absolute positions in latitude and longitude. However, the GPS accuracy in the horizontal plane of the earth is not very accurate. According to the United States Government, the error can reach up to 7.8 meters with a 95% confidence interval [5].

For such reasons, we introduce wheel encoders and Inertial Measurement Unit (IMU) to compensate for each other's positon estimation errors. Dead reckoning is a widely used method to determine momentary position of a mobile robot using wheel encoders [6]. Using the forward kinematic model of the mobile robot, it is possible to derive odometry information from the measurements if the wheel encoders. However, dead reckoning error accumulates over time as the robot travels due to systematic and non-systematic errors of the robot model [6].

We will be using a 9 degree of freedom inertial measurement unit comprising of a triple-axis magnetometer, triple-axis accelerometer and a triple-axis gyroscope. It provides orientation, velocity and acceleration measurement. These measurements are also susceptible to noise similar to the sensors mentioned above and leads to inaccuracies during position and heading estimation of the robot.

To overcome the inaccuracies of the individual sensors, we use a popular state estimation filter known as the Extended Kalman Filter (EKF). It is a mathematical tool that estimates the states of a non-linear system [8]. This type of filter works very well in practice and is used for mobile robot navigation and system integration [7]. One of the applications of EKF is the position and heading state estimation of mobile robots. Provided with the sensor measurements and their corresponding error estimations, the EKF fuses these data and provides a more accurate position and orientation estimation of the robot compared to the individual sensors used in isolation. Using the sensor measurements from the GPS, IMU and wheel encoder, we employ the EKF in Robotic Operating System (ROS) and improve the accuracy of the robot's true position estimation compared to an individual sensor.

The following contents of the book are organized as follows: In chapter 2, we introduce our robot model, Permobil C300 wheelchair, the sensors we are going to use and their specifications. Chapter 3 describes the calibration and error estimation of the different sensors and the University of Michigan Benchmark (UMBmark) test to calculate dead reckoning errors [6]. Chapter 4 presents the Extended Kalman filter model and algorithm, our observation vector from the sensor measurements and the state estimation of the wheelchair's true position. Chapter 5 shows the parameter configuration of our EKF localization software package in ROS. Chapter 6 shows our experimental results of localization done in indoor and outdoor conditions and finally chapter 7 is the conclusion.

Chapter 2

Hardware Specification

This chapter presents the wheelchair model and its hardware specifications. We introduce the wheel encoder, IMU and GPS and sensors used for robot localization and their specifications Finally we are going to describe how we use the encoders to build a wheel on wheel encoders.

2.1 Wheelchair Model

For localization, we are using a Permobil C300 wheelchair, NoRMA (Northeastern Robotic Mobility Assistant) as our mobile robot model. It is an electric wheelchair for outdoor and indoor driving and is intended for people with physical disabilities. The wheelchair consists of a chassis and a seat. The chassis contains the wheelchair's electronics, power supply and drive functions. The seat consists of a seat frame, seat plate/back rest, arm rest/leg rest and any accessories/options such as a head rest, calf rest, chest support, etc.

We equipped the wheelchair with a computer on the back for running ROS and interfacing different sensors. We also designed a frame and attached it to the back of the wheelchair for mounting sensors such as GPS, IMU, camera, LIDAR, etc. We attached two WoW (Wheel on Wheel) encoders in the left and right wheel.

The WoW encoders are spring loaded instead of a rigid contact with the wheels. This is done to ensure there is minimum physical stress on the wheels of the WoW encoders. Otherwise, the physical stress would wear away the surface of the WoW encoder wheels and would make it lose contact with the wheelchair wheels. So even if the wheelchair wheels are turning, we won't be able to get any readings from the encoders. We also mount an IMU and a GPS. Figure 2.1 – 2.3 shows the front view, side view and rear view of the wheelchair with the sensors mounted.

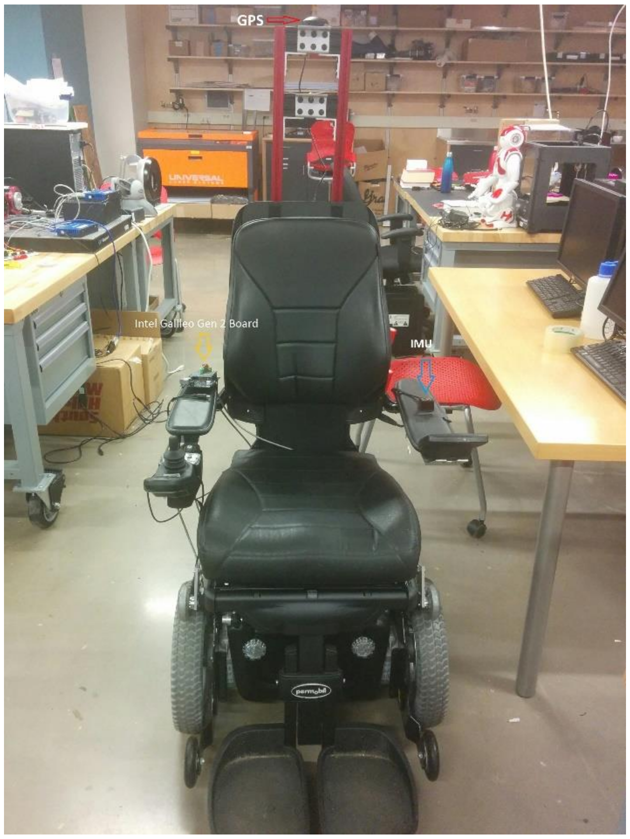


Figure 2.1: Front view of C300 wheelchair

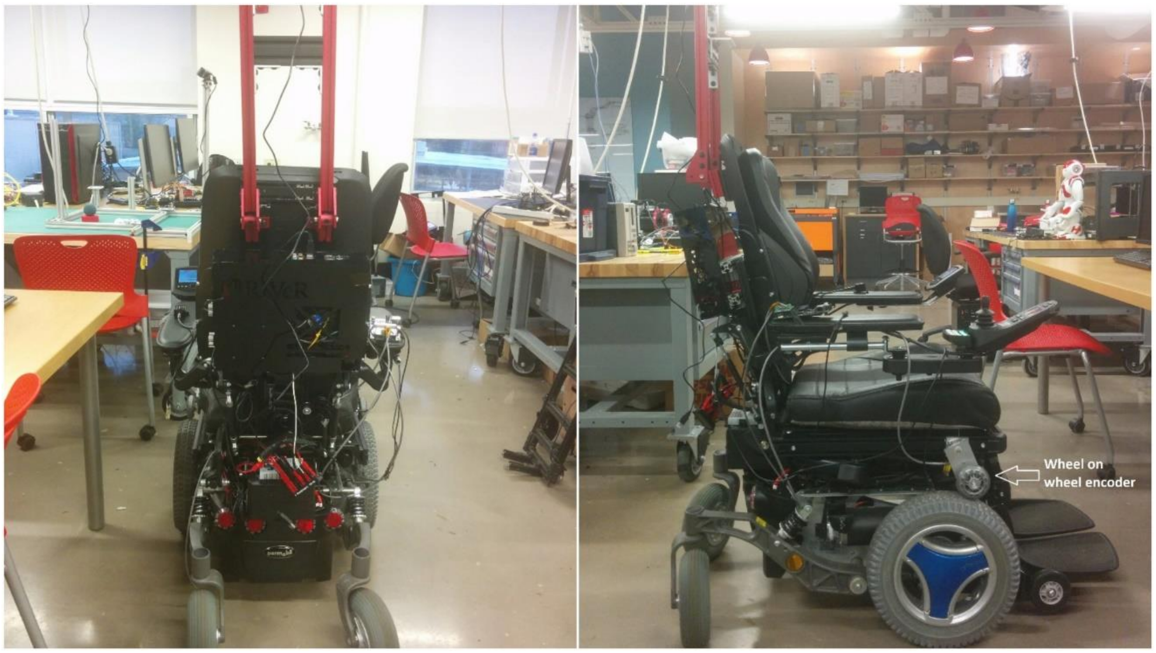


Figure 2.2: Rear and side view of C300 wheelchair

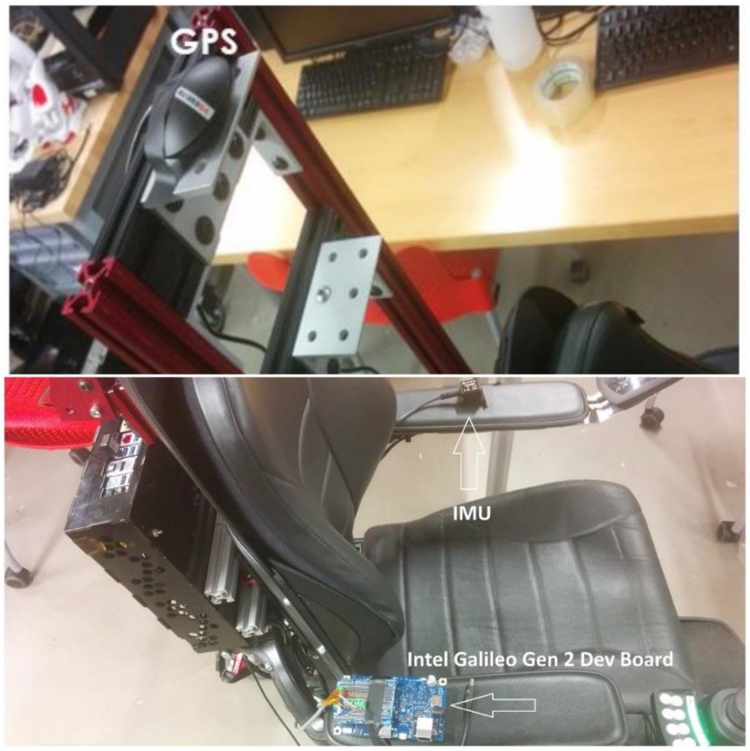


Figure 2.3: Sensor mounted on C300 wheelchair

Table 2.1 shows the technical specifications of the C300 wheelchair.

Table 2.1: Technical Specifications of C300 Wheelchair

Parameters	Value	Unit
Dimension (L x W x H)	1115 x 620 x 990	mm
Min. Ground Clearance	70	mm
Weight	153	Kg(s)
Battery Cell	2 x 60	Ah
Max Speed	7	Km/h
Max User Weight	120	Kg(s)
Kinematic model	Differential Drive	N/A

2.2 Wheel on Wheel (WoW) Encoder

For getting velocity information from wheels of the wheelchair, we interface two H1 optical shaft encoders from U.S. Digital (figure 2.4). The encoders provide 2-channel quadrature, TTL square wave outputs which is used to determine which way the encoder is rotating. It provides a 32 clock pulses per revolution providing enough resolution to estimate the displacement to the wheelchair.

Figure 2.4: H1 Optical Shaft Encoder

The encoders are equipped with skate wheels from Sparkfun to provide ample contact with the tires of the wheelchair. The skate wheels have and inner and outer diameter of 0.87 in and 4.9 in respectively with a width of 0.94 in. The skate wheels are interfaced with the encoder shaft and whole arrangement is attached to the wheels with a spring load contact (figure 2.5). This has been done to ensure that the encoder do not experience any physical stress when the wheelchair is moving in a rough terrain or over bumps and cracks.

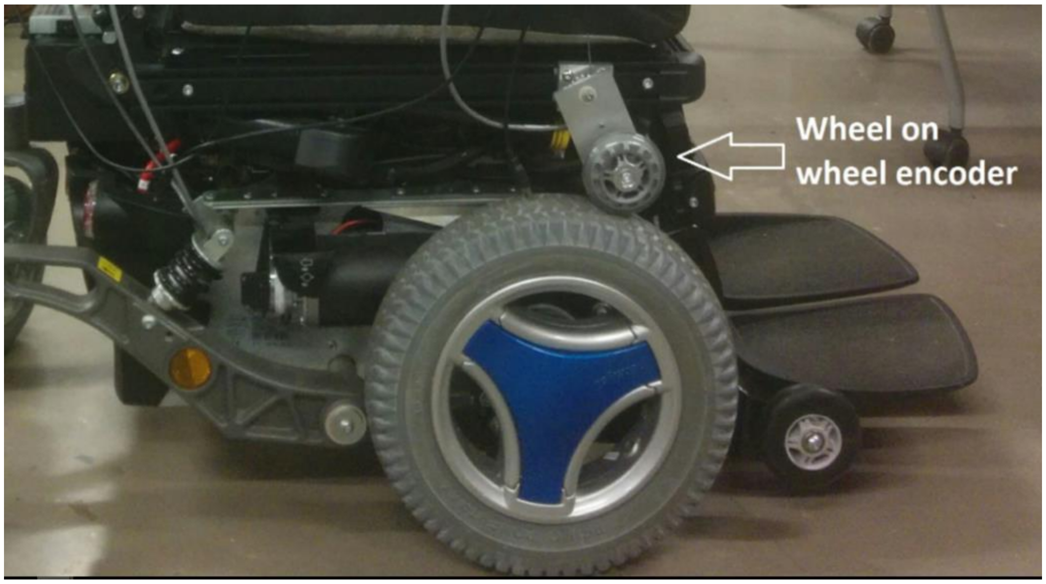


Figure 2.5: Wheel on wheel encoder arrangement

2.3 Inertial Measurement Unit

For getting orientation and velocity information, we are using the Orientus IMU from Advanced Navigation. Orientus is a ruggedized miniature orientation sensor and attitude and heading reference system (AHRS). An AHRS uses accelerometers, gyroscopes and magnetometers combined in a mathematical algorithm to provide orientation. Orientation consists of the three body angles roll, pitch and heading. It combines temperature calibrated accelerometers, gyroscopes and magnetometers in a sophisticated fusion algorithm to deliver reliable orientation measurements. The technical specifications are provided below.

Table 2.2: Technical Specifications of Orientus IMU

Parameters	Value
Roll and Pitch Accuracy (Static)	0.2 °
Heading Accuracy (Static)	0.5 °
Roll and Pitch Accuracy (Dynamic)	0.6 °
Heading Accuracy (Dynamic)	1.0 °
Output Data rate	Up to 1 KHz

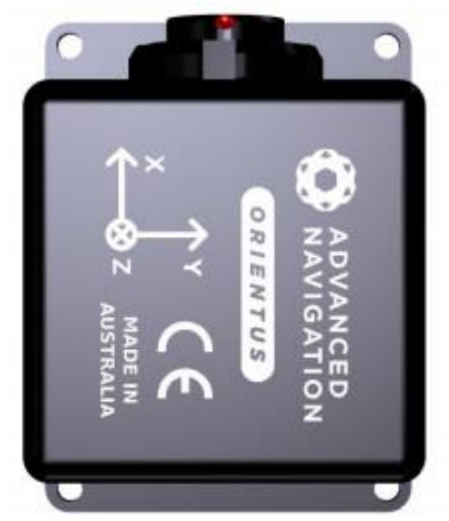


Figure 2.6: Bird's eye view of Orientus IMU showing axes marked on top

2.4 GPS Unit

We are using a GPS puck from GlobalSat (Model BU-353). It is equipped with a patch antenna and SiRF Star IV GPS Chipset. It publishes NMEA standard messages [9]. We investigate the messages that contain latitude, longitude and altitude data ($GPGGA messages) using a ROS package and publish that data over a ROS topic.

To summarize, the chapter describes our wheelchair model NoRMA and the types of sensors we used for localization. We describe the IMU, GPS and encoder's hardware specification and some of the design constraints we have due to WoW encoders.

Chapter 3

Sensor Calibration and Measurement Preprocessing

This chapter presents how we are going to calibrate the sensors in order to get more accurate measurements. The GPS, Inertial Measurement Unit (IMU) and wheel encoders doesn't directly provide pose and velocity information which conforms to the robot localization software package. So, each sensor data had to go through a transformation according to their type of data and were then provided to the ROS package and EKF. Furthermore, we try to model the error of each sensors which would be used in the EKF for a more accurate state estimation.

3.1 Differential Drive Model of the Wheelchair

The driving mechanism of the C300 wheelchair can be described as a differential drive model. It consists of two wheels that can be driven and is mounted on a common axis. The wheels are independent and can rotate forward or backward. However, the wheels do not have the ability to steer. Different types of motions are generated by controlling the velocity of the individual wheels and thus it can perform 360° rotation, steering left and right and controlling the sharpness of the turn. This is depicted in figure 3.1.

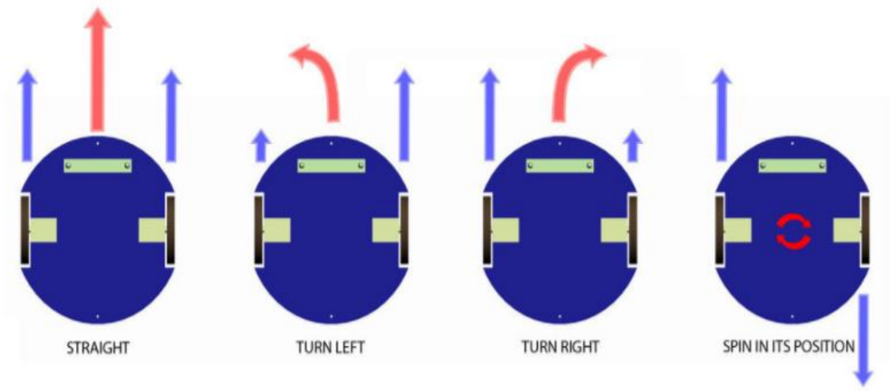


Figure 3.1: Various motions of a differential wheel drive

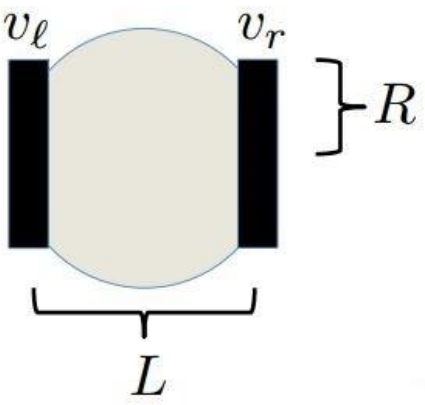


Figure 3.2: Simple model of a differential drive robot [10]

It is difficult to imagine the motion of the robot in terms of each individual wheel velocity. Instead we would like to derive terms from the wheel velocities that expresses the whole robot’s position (x, y) and heading (φ) (figure 3.3). In order to do that we need to know two parameters R and L. R is simply the wheel radius and L is called the wheel base of the robot. It is the distance between the two points where the wheels are in contact with the floor surface.

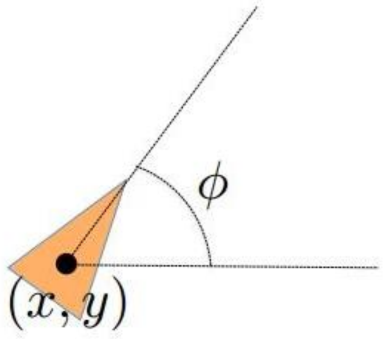


Figure 3.3: Desired parameters (x, y, φ) for a differential wheel drive robot [10]

We can know left and right wheel linear velocity, V_l and V_r respectively, from our wheel encoders by counting the ticks we receive from each encoder. We know the encoder resolution and we from there we know how much linear distance the individual wheels has travelled. We have the knowledge of time from a microcontroller and thus we get speed V_l and V_r. The following equations express the robots linear and angular velocity with respect to the wheel velocities and robot's parameters. R represents wheel radius, L represents wheelbase length and φ represents the wheelchair's heading.

$$\dot{x} = \frac{R}{2}(V_r + V_l)cos\varphi \qquad (3.1)$$

$$\dot{y} = \frac{R}{2}(V_r + V_l)sin\varphi \qquad (3.2)$$

$$\dot{\varphi} = \frac{R}{L}(V_r - V_l) \qquad (3.3)$$

We can further integrate these parameters to get position (x, y) and heading (φ) of the robot. But the linear and angular velocities are sufficient as an input for our EKF state estimation algorithm as we will be using absolute positions from the GPS as an input for the Extended Kalman Filter to estimate our robot's location.

3.2 Dead Reckoning and Wheel Encoder Error Estimation

In navigation, dead reckoning is the process of calculating one's current position by using a previously determined position, or fix, and advancing that position based upon known or estimated speeds over elapsed time and course. Dead-reckoning is can be accomplished by using incremental wheel encoders. But the biggest disadvantage of dead-reckoning is accumulation of error as time progresses. But it is possible to reduce the dead-reckoning error of a differential wheel drive by properly estimating the error that will accumulate over time.

An assumption that is made on dead reckoning is that wheel revolutions can be translated into linear displacement relative to the surface. But this account doesn't count wheel slippage. For example, if one wheel was to slip on an oil spill or muddy roads, then the associated encoder would register wheel revolutions even though these revolutions would not correspond to any linear displacement of the wheel. There are several other inaccuracies in the translation of the wheel encoder motion not only due to slippage but due to the inaccuracies of the physical robot and the physical robot. They can be categorized into two terms:

Systematic errors:

- Unequal wheel diameters
- Average of both wheel diameters differs from nominal diameter
- Misalignment of wheels
- Uncertainty about the effective wheelbase (due to non-point wheel contact with the floor)
- Limited encoder resolution
- Limited encoder sampling rate

Non-systematic errors:

- Travel over uneven floors
- Travel over unexpected objects on the floor
- Wheel-slippage due to:
  - Slippery floors
  - Over-acceleration
  - Skidding
  - External forces (interaction with external bodies)
  - Internal forces (e.g., castor wheels)
  - Non-point wheel contact with the floor

The systematic errors are caused by imperfections in the design and mechanical implementation of a mobile robot. Rubber tires are used in robots to improve traction. But it is very difficult to manufacture the tires with precision accuracy in diameter. Along with that, these kinds of tires are susceptible to unequal compression under asymmetric load distribution. Such imperfections cause dead-reckoning errors.

The contact points of the two wheels of the robot and the floor is known as the wheel base (L). While measuring the length of the wheelbase, there can be an uncertainty in the effective wheelbase length because the rubber tires contact the floor not on a point, but rather in a contact area. The resulting uncertainty about the effective wheelbase can be on the order of 1% in some commercially available robots.

Non-systematic dead-reckoning errors are caused by interaction of the robot with unpredictable features of the environment. This can be irregularities of the surface, such as bumps, potholes, rough surface, uneven terrain, debris, etc. Such imperfections of the environment will make the wheel rotate more as now we are also moving up and down in the vertical direction This will cause error in the estimation of the horizontal distance travelled and will accumulate as the robot moves forward. These errors are very hard to predict and thus we are not including into the error estimation of the dead reckoning. Fortunately, the Extended Kalman filter is able to handle such errors using the data coming from other sensors and tries to estimate such errors resulting in a better state estimation.

In order to estimate the error in position of the robot from dead reckoning happening due to systematic errors, we use a concept from the UMBmark benchmark test [6]. The UMBmark benchmark test tells the robot to start at a position x_0, y_0, θ_0, (Start) (figure 3.4). The starting area should be located near the corner of two perpendicular walls so that the walls can be used as a fixed reference before and after the run. We move forward and make three 90° clockwise turns and come at the start point. This would create a trajectory close to a square shape. We calculate from the encoder data how much we are off from the origin and record this data. The difference in the starting position and the calculated encoder data gives us the error due dead reckoning errors due to system imperfections. We make 5 runs like this and record the error for further processing. Now we perform the same task but in counter-clockwise direction and complete a square path. We do it for five more runs and record the data.

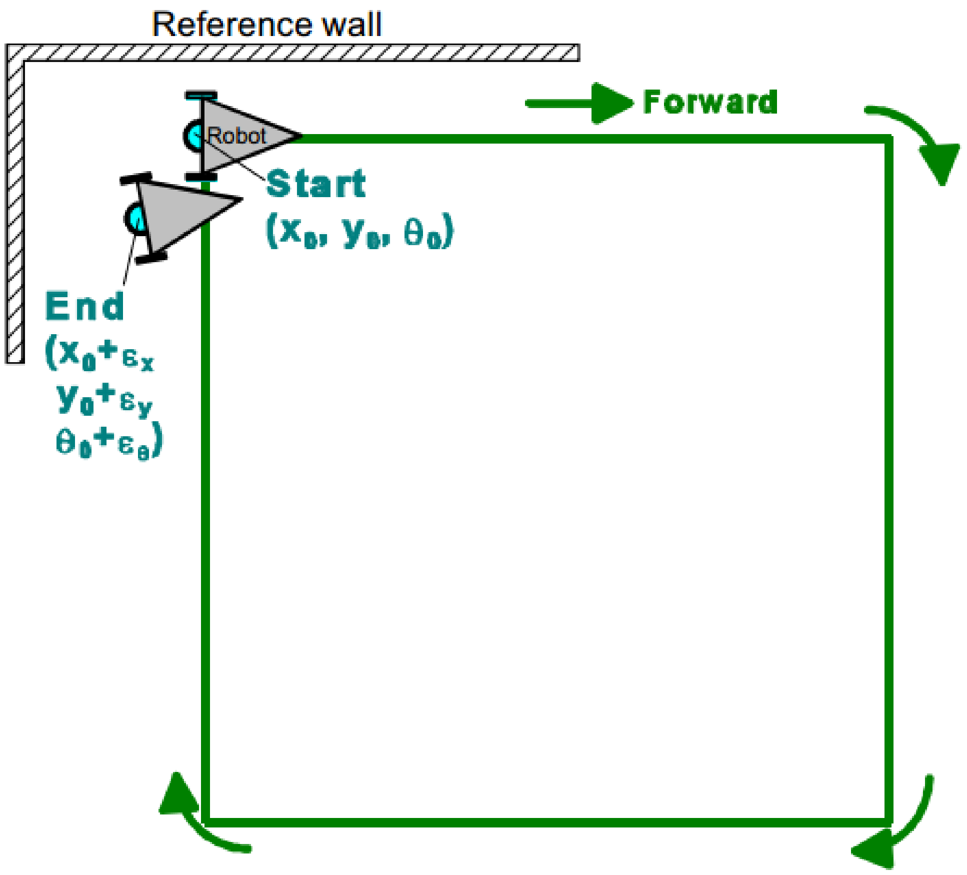


Figure 3.4: UMBmark benchmark square path experiment [6]

The reason for doing a clockwise and counter clockwise square trajectory is due to take account for the unequal wheel diameters. For example, if we perform a 90° rotation clockwise, the robot may rotate 93°. But as since there is one wheel will be slightly larger than the other, the same rotation in the counter clockwise direction would result in a different rotational angle, for example 86.5°.

So, we perform a square trajectory both in the clockwise and counter clock wise direction to account for the uncertainty in the wheel diameters. The clockwise trajectory will generate errors that would be off by some order of magnitudes compared to the errors produced by the counter clockwise trajectory.

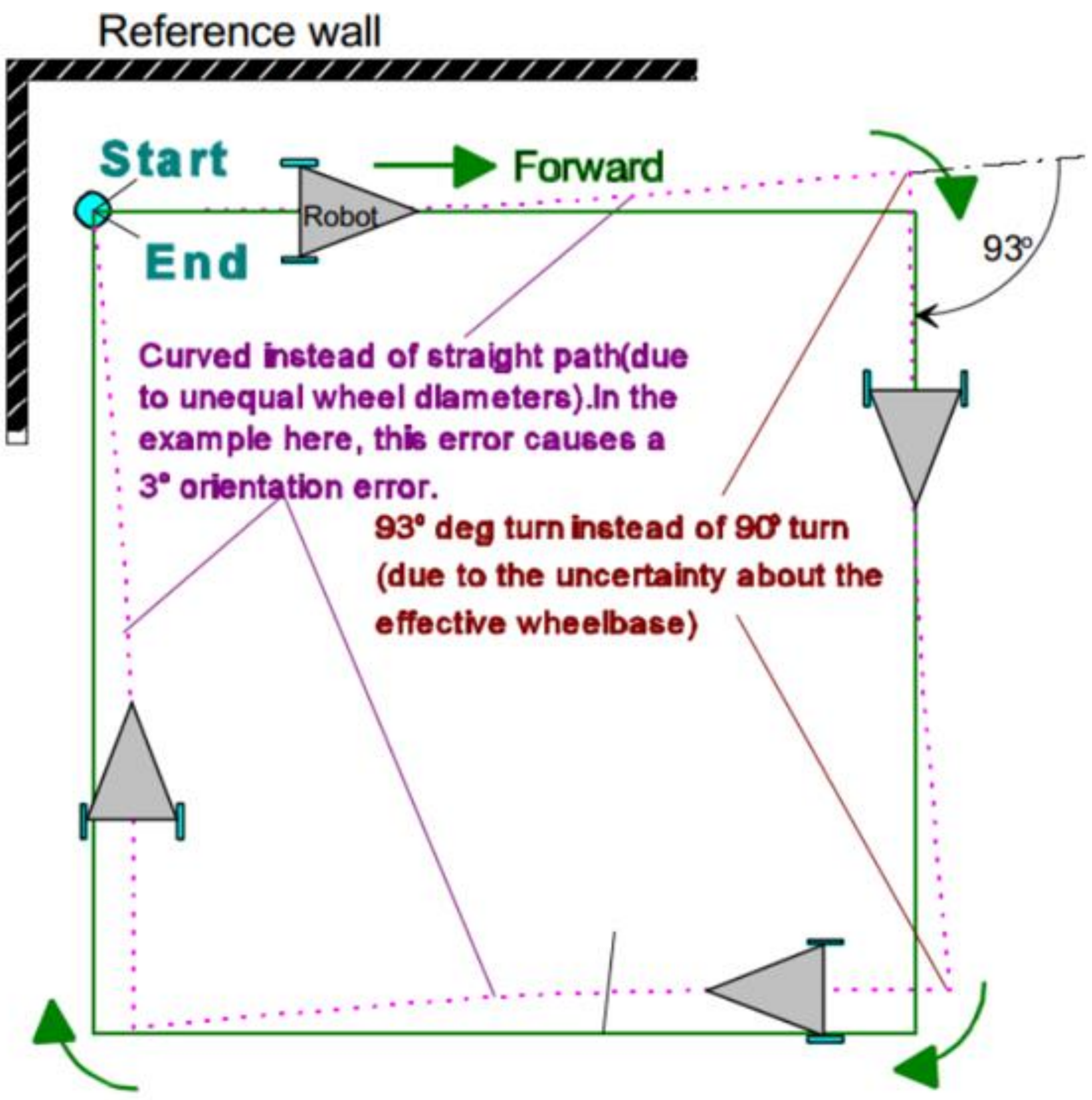

Reference wall
Start
End
Robot
Forward
93°
Curved instead of straight path(due to unequal wheel diameters).In the example here, this error causes a 3° orientation error.
93° deg turn instead of 90° turn (due to the uncertainty about the effective wheelbase)

Figure 3.5: Dead reckoning error due to unequal wheel diameters [6]

The results of our 5 runs on the clockwise and 5 runs on the counter clockwise square path trajectory can be illustrated with a graph in figure 3.6. The start position is at the origin (0,0) and the end positions of the whole run are clustered in two sides, one cluster is the result of the 5 runs in clockwise direction and the other 5 points represents the ending positions for the runs made in counter clockwise direction.

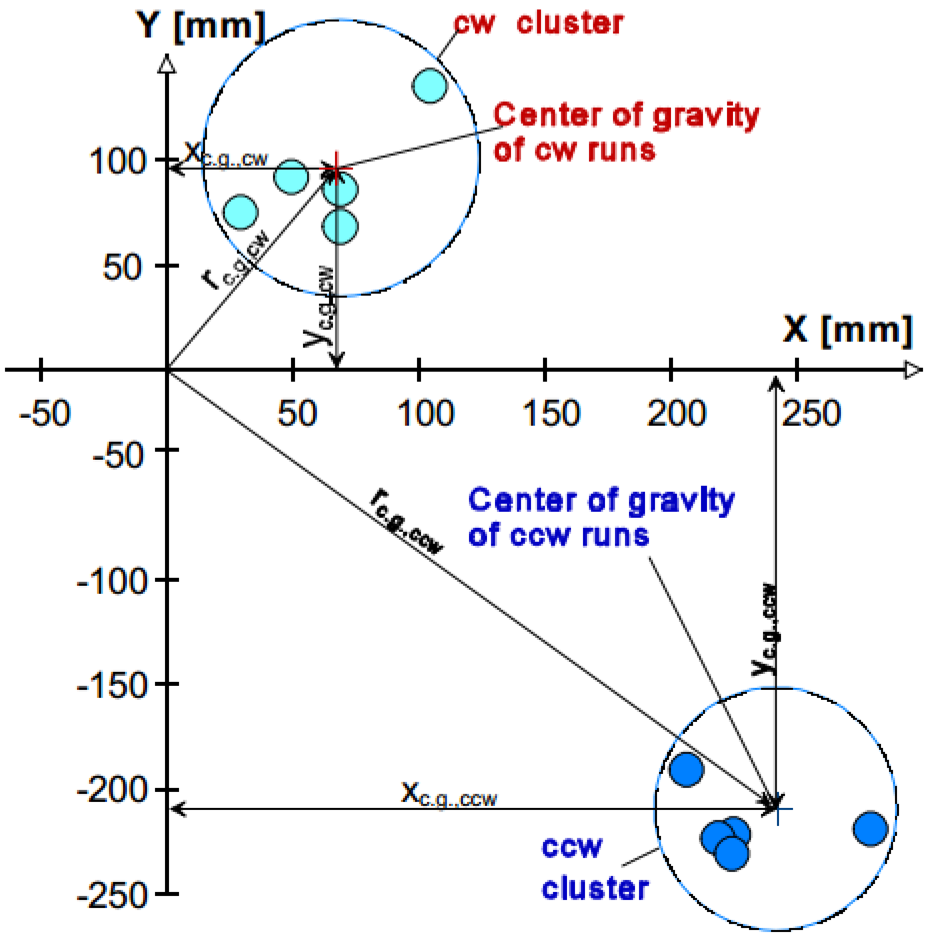


Figure 3.6: Results from running a differential wheel drive with uncalibrated encoder [6]

The UMBmark test calculates mean of the cluster which they call center of gravity for both clusters and calculates the distance from the origin to the means. They use the larger value to define it as the maximum dead reckoning error due to system imperfections and adjusts this for calculation of velocity. However, we are more interested in the distribution of the clusters. Because the Extended Kalman Filter expects the error model to be in standard deviation or covariance matrices. By calculating the standard deviation and variance of the clockwise and counter clockwise clusters, we can prepare this error model which is convenient as an error measurement input for the EKF.

3.3 IMU Calibration

Our robot localization software requires the IMU provides absolute heading. This means if the X axis of the IMU is pointing towards true north, the IMU yaw angle should provide a value of 0°. If its pointing towards south, it should provide a heading of 180°. But the IMU we are using wasn't providing a 0° value when it was facing true north. In order for the IMU to provide absolute heading angle, we needed to perform a hard iron and soft iron calibration of the magnetometer of the IMU. The IMU uses the earth's magnetic field to determine the magnetic north of the earth and thus can provide absolute heading value using it.

The earth's magnetic field is not uniform and is distorted due to external magnetic influences. This distortion is called hard iron and soft iron effect. If there wasn't any distortion in the earth's magnetic field, rotating a magnetometer (embedded in the IMU) through a minimum of 360° in the X-Y plane would result in a circle centered around the origin.

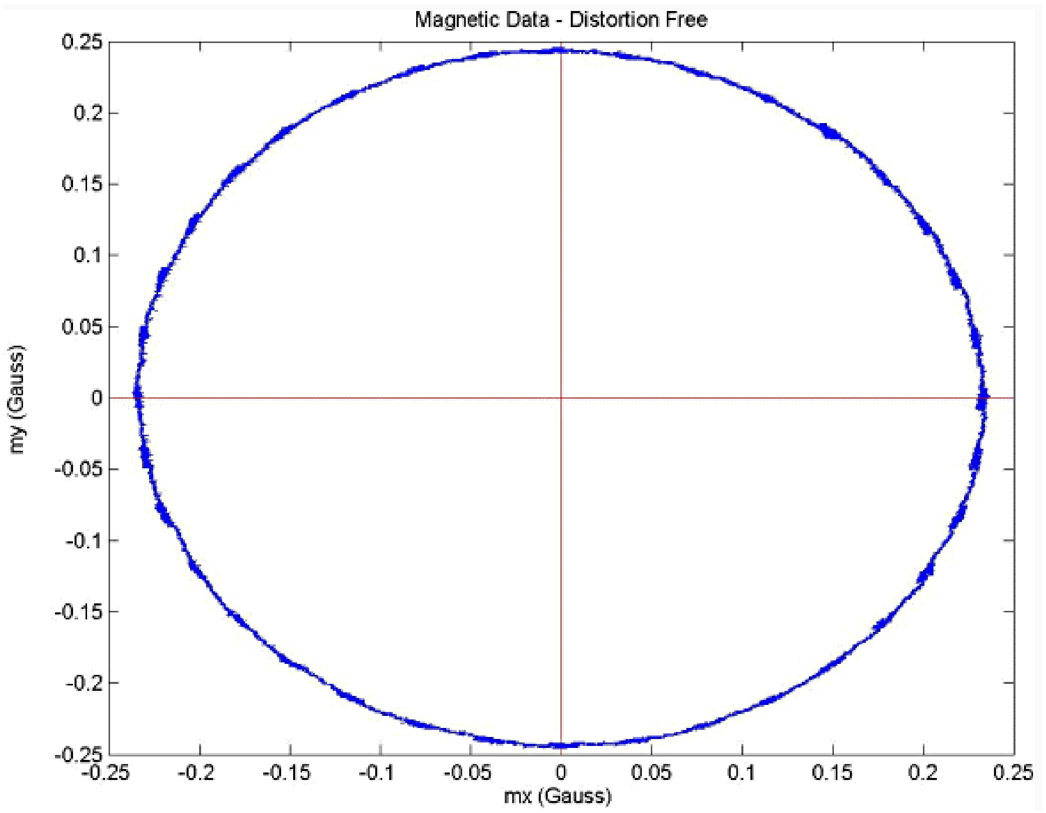


Figure 3.7: Magnetometer data without any distortion in the X-Y plane [11]

However, the presence of these magnetic distortions would have shifted the circle center from the origin due to a hard-iron effect, or change the shape of the circle to produce an ellipse in the case of a soft-iron effect. It is a very likely possibility that both of the hard and soft iron error would effect the circle.

Hard-iron distortion is produced by materials that exhibit a static, additive field to the earth's magnetic field, thereby generating a constant additive value to the output of each of the magnetometer axes. For example, a permanent magnet would produce a hard-iron distortion given that its position and orientation is constant relative to the magnetometers position and orientation. A hard iron error is shown in figure 3.8.

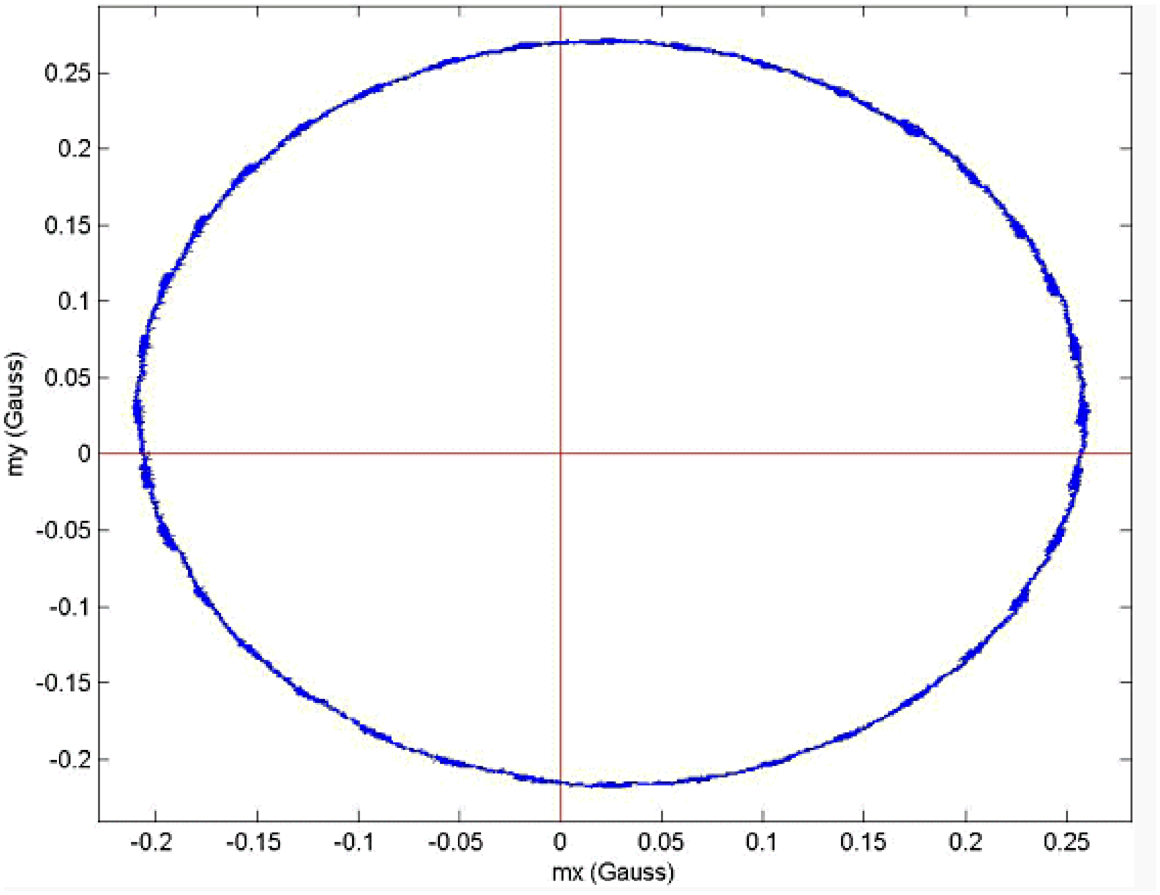


Figure 3.8: Plot of magnetometer data in the X-Y plane due to hard iron distortion [11]

To nullify the error due to hard iron distortion, the center of the circle is determined and the offset from the origin is calculated. This constant value is then added or subtracted with each measurement from the magnetometer.

Soft-iron distortion is the result of material that distorts or influences a magnetic field but the material necessarily doesn't produce a magnetic field itself. Thus, unlike the hard iron effect it is not additive to the magnetometer readings. For example, Nickel and iron will generate a soft-iron distortion. While hard-iron distortion is constant regardless of orientation, the distortion produced by soft-iron materials is dependent upon the orientation of the material relative to the sensor and the magnetic field. Thus, soft-iron distortion cannot be compensated with a simple constant. Figure 3.9 shows a distortion due to a soft iron which deforms the circle into an ellipse. To compensate for the soft iron effect requires the ellipse to align its major and minor axis with x and y axis and changing its form to a circle. Thus, it cannot be just compensated with a single constant like the hard iron error.

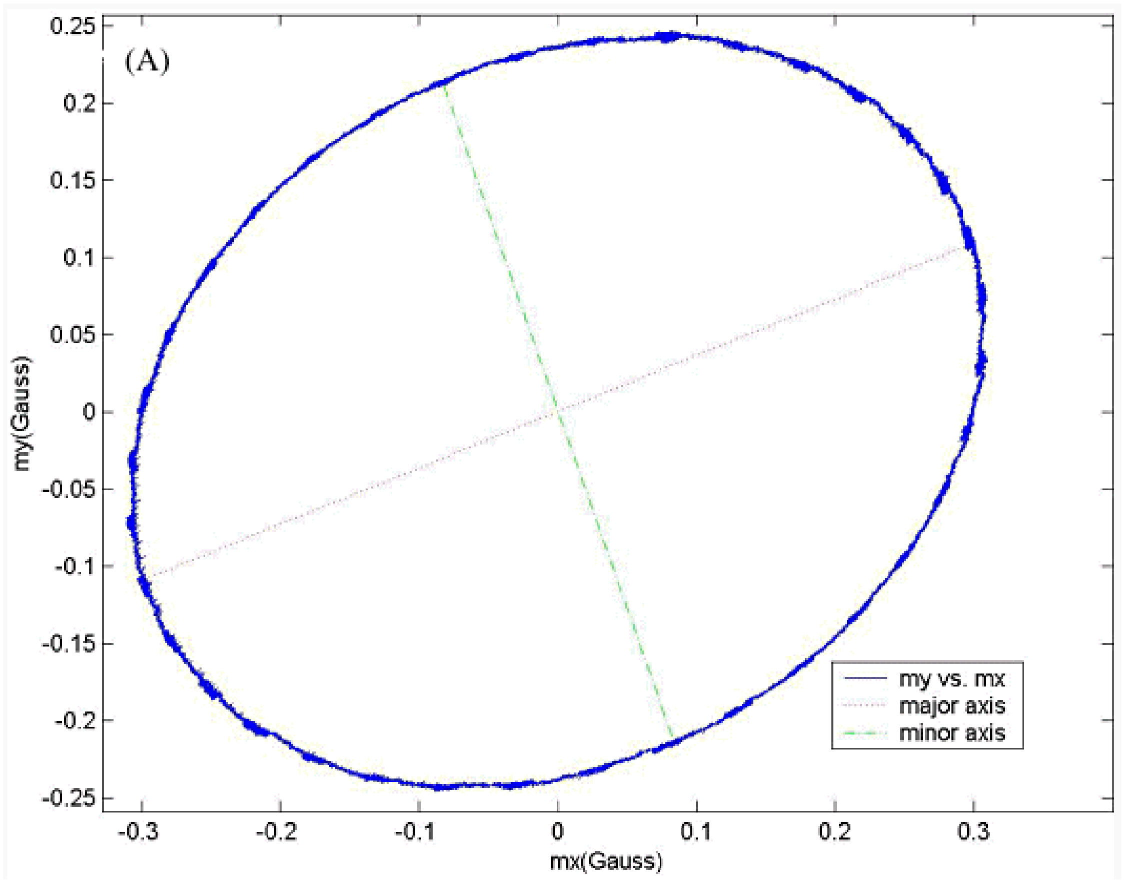
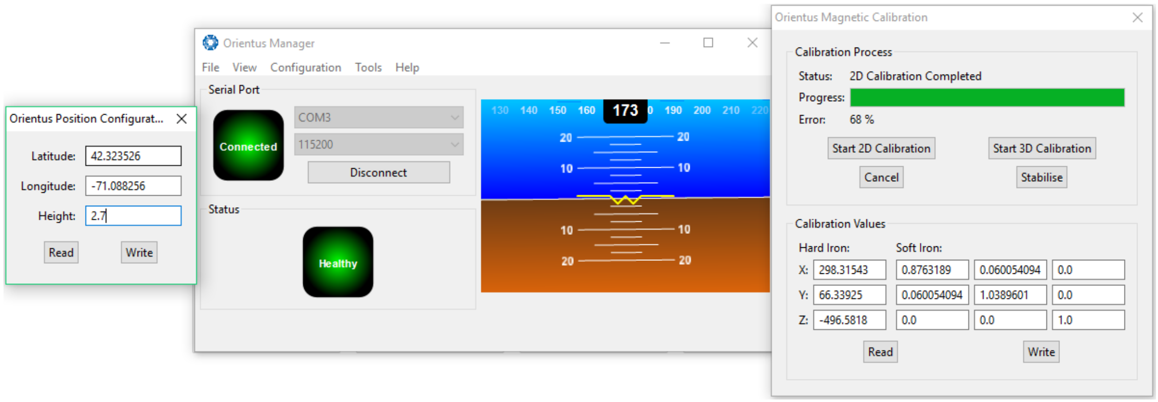


Figure 3.9: Plot of magnetometer data in the X-Y plane due to soft iron distortion [11]

The Advanced Orientus IMU that we are using provides a magnetic calibration tool to find out the hard and soft iron parameters. In order to use the tool, we provide the location in latitude, longitude and elevation of the IMU. We start the calibration process by taking the IMU and keeping it away from magnetic interference as much as possible and rotate it in the X-Y plane of the IMU three times either in clockwise or anti-clockwise direction. This provides six parameters, three for hard iron and three for soft iron effects along the IMU's x, y and z axis (figure 3.10). We use these parameters to write to the appropriate register of the IMU from the tool and the IMU uses these parameters to compensate for the hard and soft iron effects when providing orientation output. Before calibration, the IMU was providing a 0° heading in the SW direction. After the calibration, we saw the IMU to be providing almost 0° heading at the true north based on the compass from a smartphone.



Latitude: 42.323526
Longitude: -71.088256
Height: 2.7

COM3
115200

Status: 2D Calibration Completed
Progress:
Error: 68 %

	Hard Iron:	Soft Iron:		
X:	298.31543	0.8763189	0.060054094	0.0
Y:	66.33925	0.060054094	1.0389601	0.0
Z:	-496.5818	0.0	0.0	1.0

Figure 3.10: Hard and soft iron calibration of the Orientus IMU

3.4 GPS Sensor

For the GPS sensor, there is no calibration required for the GPS puck we are using. As soon as we connect the GPS to the wheelchair, it takes about 10 seconds time to establish connection with the satellites and starts proving latitude, longitude and altitude information in standard NMEA messages.

To summarize, we describe in this chapter how we estimate the error from our encoders using UMBmark test, how we calibrated the IMUs by reducing hard and soft iron effects and the delay of GPS for establishing communication with the satellite.

Chapter 4

Sensor Measurements Fusion with Extended Kalman Filter

This chapter describes the extended Kalman filter algorithm implemented in the robot localization software package. It describes how the filter weighs between the incoming sensor measurements and its predicted state based on the Kalman gain to get the best possible state estimation. We are also going to discuss about the state vector and the observation vector, error estimations and covariance matrices. Finally, we are going to present how different measurements and their corresponding error estimations that are being fused from the GPS, IMU and odometry from the wheel encoder into the Extended Kalman Filter and its corresponding the output.

4.1 Extended Kalman Filter Algorithm

The extended Kalman filter is used for nonlinear version of the Kalman filter which linearizes about an estimate of the current mean and covariance. The state estimation model and the measurement process is given by the following two equations:

$$x_k = f(x_{k-1}, u_{k-1} + w_k) \qquad (4.1)$$

$$z_k = h(x_k) + v_k \qquad (4.2)$$

The meaning of the variables in the equation are as follows:

x_k: state vector

u_{k-1}: control input

w_{k-1}: the process noise

z_k : measurement/observation vector

v_k : measurement noise

To predict state from the previous estimate, the function f(.) is used. Similarly, to compute the predicted measurement from the predicted state function h is used. However, f and h functions cannot be applied to the covariance directly. Instead a matrix of partial derivatives is computed called the Jacobian.

The Extended Kalman Filter has a predict and update step. In the predict step, the filter tries to predict the current state estimate based on the previous state estimate and the process covariance matrix (P_k). The process covariance matrix represents the error in the filter estimate and is calculated based on the previous P_{k-1} and process noise covariance Q_{k-1}. Estimation of the the Kalman gain (K_k) determines how much emphasis to put on the current predicted state estimate and current measurements and is derived from P_k and sensor noise covariance matrix R_k. Using the Kalman gain, we update our state estimate x_k and P_k. F_k and H_k are the state transition and observation matrices respectively and S_k is the residual covariance. The prediction and update model is provided below.

Predict Cycle:

$$\hat{x}_{k|k-1} = f\left(\hat{x}_{k|k-1}, u_{k-1}\right) \quad (4.3)$$

$$P_{k|k-1} = F_{k-1} P_{k-1|k-1} F_{k-1}^T + Q_{k-1} \quad (4.4)$$

Update Cycle:

$$\tilde{y}_k = z_k - h\left(\hat{x}_{k|k-1}\right) \quad (4.5)$$

$$S_k = H_k P_{k-1|k-1} H_k^T + R_k \quad (4.6)$$

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \quad (4.7)$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k \quad (4.8)$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \quad (4.9)$$

Since we are interested in the robot's position and orientation in the environment, our state estimation vector will contain ($x, y, z, \alpha\ \beta$ φ) where x, y, z represents the coordinates in a 3D Cartesian frame and $\alpha\ \beta$ φ represents roll, pitch and yaw respectively. Assuming we are moving in a near planar environment, we can exclude the effects of the z position and rolling and pitching. So, our state estimate vector becomes:

$$x_k : [x, y, \varphi]$$

Our observation vector z_k are the different measurements that are arriving from different sensors. The sensors and their corresponding measurements are provided below.

IMU: $[\alpha\ \beta\ \varphi\ \dot{x}\ \dot{y}\ \dot{z}\ \dot{\alpha}\ \dot{\beta}\ \dot{\varphi}\ \ddot{x}\ \ddot{y}\ \ddot{z}]$

Wheel Encoder: $[x, y\ \varphi\ \ \dot{x}\ \ \dot{y}\ \ \dot{\varphi}]$

GPS: $[x, y]$

Here $\dot{x}\ \dot{y}\ \dot{z}$ are linear velocities, $\ddot{x}\ \ddot{y}\ \ddot{z}$ are linear acceleration and $\dot{\alpha}\ \dot{\beta}\ \dot{\varphi}$ are angular velocities respectively.

Using these different measurements and fusing them together, we feed the observation vector to the EKF so that the EKF provides us the best state estimate (x, y, φ) and use that to monitor the trajectory of our wheelchair.

Thus, our observation vector and state estimation vector becomes:

$$z_k : [\mathrm{x}\ \mathrm{y}\ \varphi\ \dot{x}\ \dot{y}\ \dot{\varphi}\ \ddot{x}\ \ddot{y}]$$
$$x_k : [x, y, \varphi]$$

In this chapter, we presented the Extended Kalman Filter model and its algorithm. We described what are the sensor parameters we are going to use to populate the observation matrix and what are the parameters we are going to use for our state estimation. In our case, it is the x, y and φ parameters.

Chapter 5

ROS Configuration

The chapter describes the ROS nodes we are going to use for robot localization. There are We describe each nodes what they are publishing and subscribing to and the finally we discuss the tf tree being published by our localization node.

5.1 ROS Nodes

There are total three drivers we are using to get measurements from the sensors and publish standard ROS messages. The differential drive node reads encoder data coming from an Intel Galileo Gen 2 development board. The board reads the encoder ticks and sends it to the computer via USB where the differential drive node subscribes to these readings. The node calculates the linear and angular velocities relative displacement and publishes it as a ROS standard odometry message containing pose and twist information of the wheelchair.

The orientus driver reads the packets coming from the Orientus IMU. It then publishes standard ROS sensor messages containing linear velocity, angular velocity, linear acceleration and orientation data.

The nmea navsat driver reads standard NMEA messages provided by our GPS and converts those messages into standard ROS GPS fix messages which contains latitude, longitude and altitude information.

The robot localization package subscribes to these three type of sensor messages. The package is running a Extended Kalman filter node which takes these sensor readings, processes them and publishes the wheelchair’s position and orientation estimation in a standard odometry message format. We use these odometry data to visualize our robot’s true position and trajectory in the map.

5.2 Tf tree

The robot localization package is publishing a tf tree containing the map frame, odom frame, baselink frame, imulink and gpslink. The map frame is a world fixed frame and is set by the current locations GPS readings. Once the node gets the first GPS reading from the GPS fix messages, it sets that location as the origin of the map frame.

The odom frame is a local frame and is the same as the map frame. The reason for having two frames is because the localization package publishes translation from the odom frame to the baselink using odometry and sensor information. The map frame wouldn't be able to publish this translation and hence it uses another frame.

The baselink is frame that is rigidly attached to the wheelchair. The origin and the orientation of this frame provides the wheelchairs pose in the map. From the baselink, there are two other links attached which are the gpsink and imulink and the position of these two links provides the actual distance where our sensors are in the real world with respect to the wheelchair baselink.

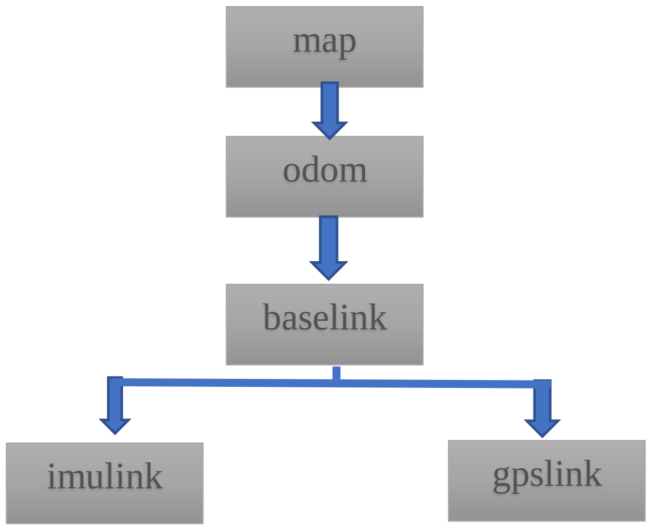


Figure 5.1: Tf tree published by robot localization package

The chapter describes the ROS configuration of our localization experiment. We describe the different ROS node drivers for our sensors, the robot localization package containing an EKF filter and the Tf tree published by this package.

Chapter 6

Experimental Results

The chapter presents our findings on the accuracy of our state estimation using the Extended Kalman Filter. We describe the outputs being published by the robot localization ROS package and display the trajectory of our wheelchair in both indoor and outdoor conditions. We perform loop closure accuracy to estimate performance with our sensor configurations. We also display the results of wheel encoder calibration and how much error we estimated using the UMBmark algorithm [6].

6.1 Dead Reckoning Errors from Wheel encoders

In chapter 2 we discussed the calibration of our wheelchair by dead reckoning using the UMBmark algorithm and performed a run in a square trajectory both in clockwise and counter clockwise direction. We did a total of ten runs, five each in clockwise and counter clockwise respectively. We observed how far we are from our starting point and calculated the standard deviation for the datasets. We chose the higher standard deviation and calculated the variance. The following table shows our result.

Table 6.1: UMBmark benchmark test dead reckoning error results

Direction	**Observed error from starting point (5 data sets in meters)**	**Standard Deviation**	**Variance**
Clockwise	4.24, 3.22, 3.59, 2.18, 3.17	0.798	0.637
Counter Clockwise	3.34, 5.16, 3.93, 3.71, 3.58	0.722	0.522

The benchmark test results show that we had a larger distribution of the dear reckoning errors when we were making clockwise turns to follow a square trajectory. The standard deviation and variance is larger for this run and hence we used the variance 0.637 as an error estimate for our wheel encoders which would be used as an error measurement covariance in the Extended Kalman Filter.

6.2 Loop Closure Accuracy

For our wheelchair model, loop closure is defined as starting from a position, making a trajectory and returning to that exact position. We perform loop closure and check how accurate it is with our wheelchair setup. It was equipped with two wheel encoders, one in the right and the other in left wheel. The IMU was mounted on the hand rest area with a tape tightly so that it may experience vibration as less as possible. The GPS was attached to the external frame of the wheelchair at a high elevation so that it doesn’t loses communication with the satellite. The following section describes our results for indoor and outdoor conditions.

6.3 Indoor Conditions

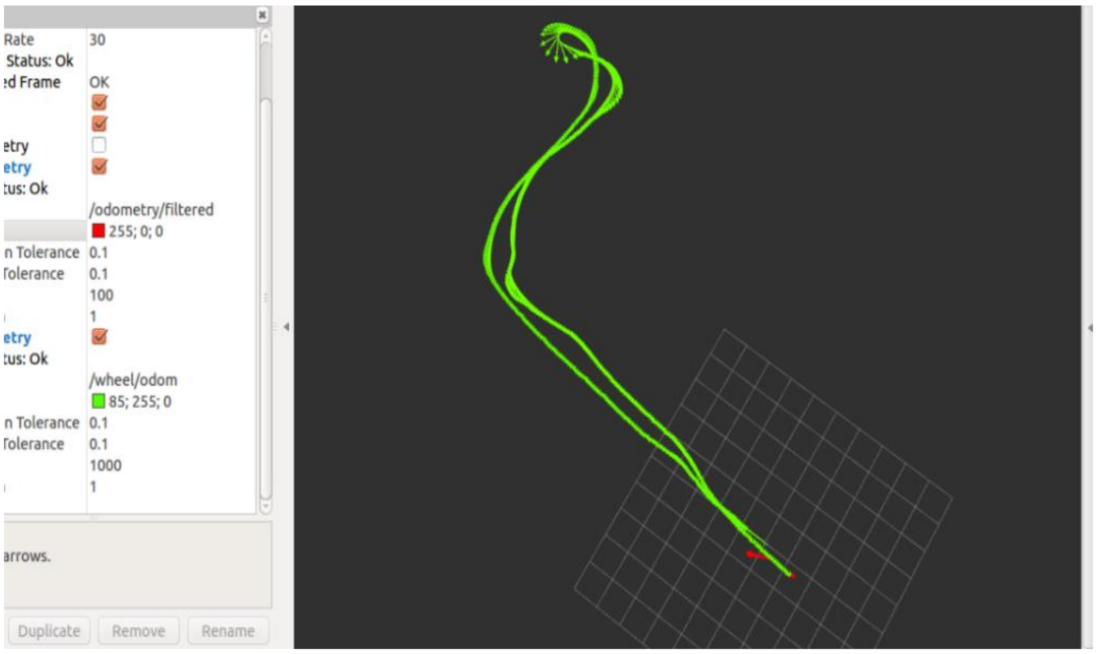
The indoor experiment was done in a corridor with sharp 90° turns at Richards Hall, Northeastern University. We are interested in the distance between the robot’s start and end positions, as reported by the EKF localization node in ROS. We would like the end position values to be as close to the starting point of our wheelchair. But ideally that is not possible due to the errors of our sensors and human caused errors for unable to finish exactly at the starting position due to the bulkiness of the wheelchair. We use only the IMU and wheel encoder sensor data and exclude GPS as it is not possible to have GPS reception inside the building. With these two sensors, we observe the loop closure accuracy in indoor conditions.

Table 6.2 describes which configuration for each sensor we fused to get the state estimation from the EKF localization node. Not all the parameters are used as this leads to undesired behavior and unstable position estimation from the EKF.

Table 6.2: Sensor configuration in indoor conditions

Sensors	**x**	**y**	**z**	$\boldsymbol{\alpha}$	$\boldsymbol{\beta}$	**φ**	$\dot{x}$	$\dot{y}$	$\dot{z}$	$\dot{\alpha}$	$\dot{\beta}$	$\dot{\varphi}$
Odometry (Encoders)	0	0	0	0	0	0	1	0	1	0	0	1
IMU	0	0	0	1	1	1	1	1	1	1	1	1

We start at a position, in the corridor and move forward, make one 90° turn clockwise, move forward, make one 90° turn counter clockwise, move forward and make a 180° rotation and follow the same trajectory to return to our starting locations. Figure 6.1 - 6.4 show the trajectory of our wheelchair in RViz published by the EKF localization node and the starting and end position of the wheelchair after making the whole run.



```
Rate            30
 Status: Ok
ed Frame        OK
                ✓
                ✓
etry            ☐
etry            ✓
tus: Ok
                /odometry/filtered
                🟥 255; 0; 0
n Tolerance     0.1
Tolerance       0.1
                100
                1
etry            ✓
tus: Ok
                /wheel/odom
                🟩 85; 255; 0
n Tolerance     0.1
Tolerance       0.1
                1000
                1

arrows.
```

Figure 6.1: Wheelchair trajectory in indoor condition for test run 1

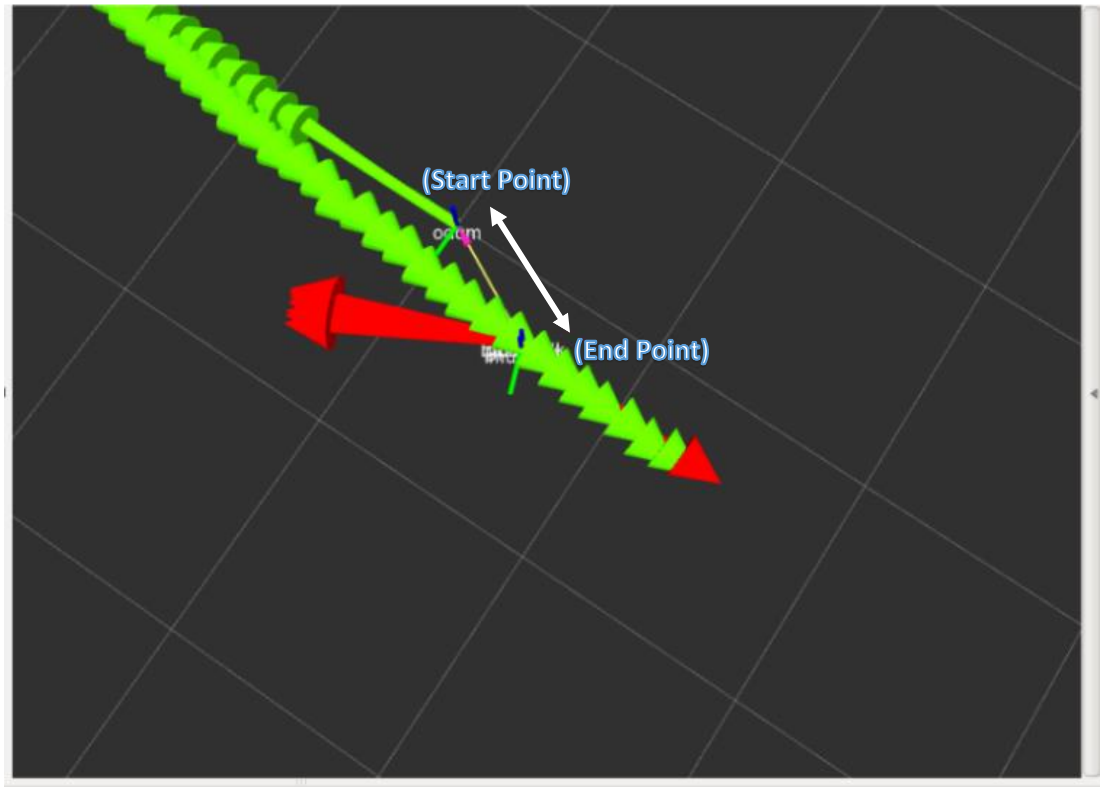


Figure 6.2: Start and end points for indoor test run 1

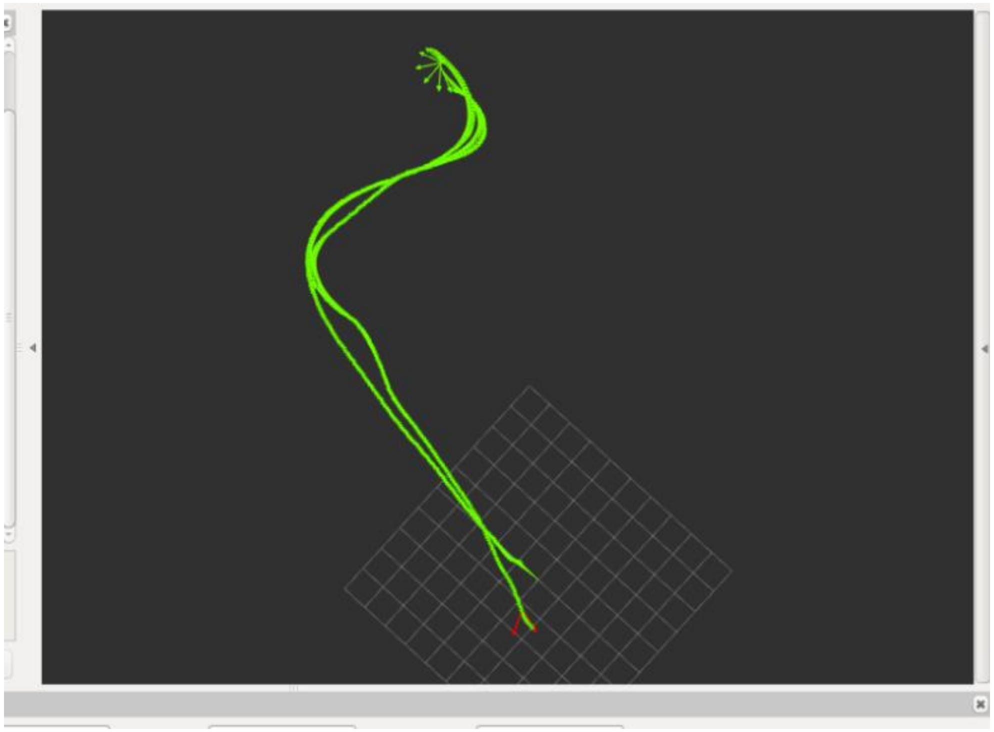


Figure 6.3: Wheelchair trajectory in indoor condition for test run 2

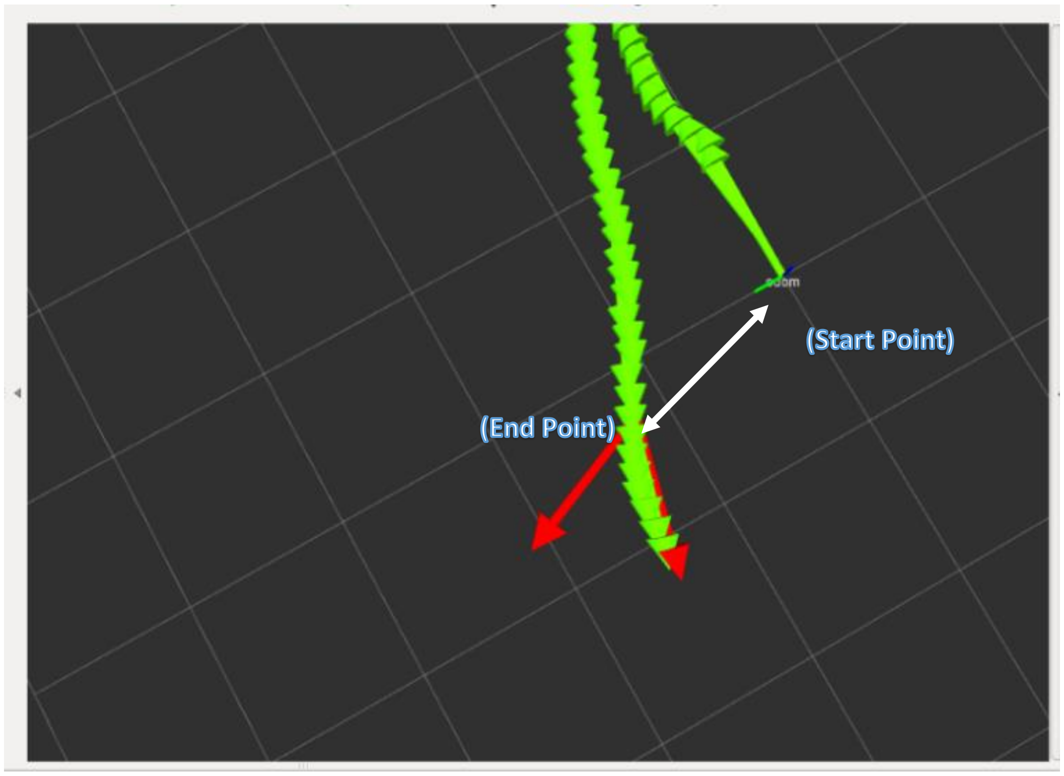


Figure 6.4: Start and end points for indoor test run 2

The grid size in the figures 6.1 – 6.4 are 1 meter in dimension. By observing the above figures, we can see the error in the start position and the end position for the different runs is approximately to be around 1 meter. The green arrow specifies in which direction the wheelchair is heading as published by the odometry source from the wheel encoders. The red arrow specifies the heading published by the Extended Kalman filter node fusing IMU and encoder data.

We made a total of three runs indoors. The corresponding error estimations of the EKF localization and from dead reckoning is shown for comparison:

Table 6.3: Loop accuracy error comparison for indoors

Sensor Set	Loop Closure Error with respect to starting position (in meters)	Standard Deviation
Odometry (Dead Reckoning, CW)	2.1m ~ 4.2m	0.798
IMU + Odometry	0.5m ~ 1.7m	0.602

From the above error estimations, we can deduce that the EKF localization is providing a better result. The distance from the start and end positions of the robot is much lower compared to the dead reckoning errors. The standard deviation is for the IMU and encoder configuration is also lower meaning the end points are much more clustered together compared to dead reckoning.

6.4 Outdoor Conditions

The outdoor experiment was performed in front of Snell Library, Northeastern University (figure 6.5). The green line specifies the path we followed with our wheelchair. Here it should be mentioned that it was a rough terrain made of bricks and there were a lot of bumps and manhole covers which made the encoders and IMU experience a lot of vibration. We use the IMU, GPS and the wheel encoder sensor data since we are outdoors and would like to see the effects on the state estimation of the EKF fusing these three sensor measurements. Like indoors, we now observe the loop closure accuracy in outdoor conditions.

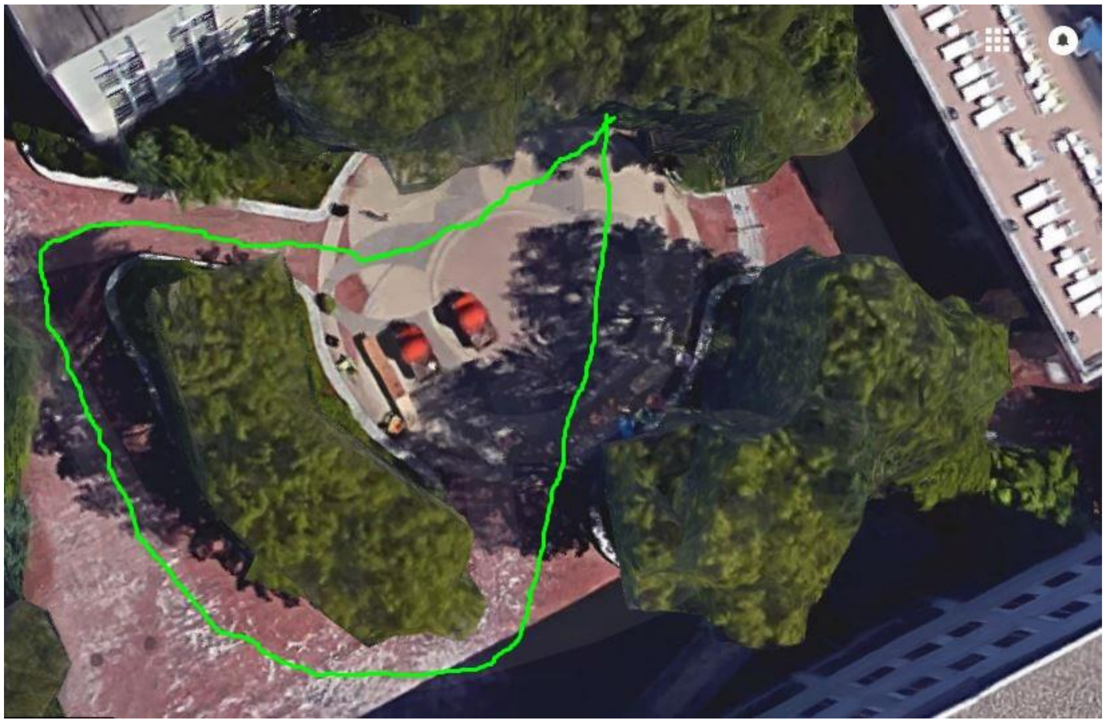


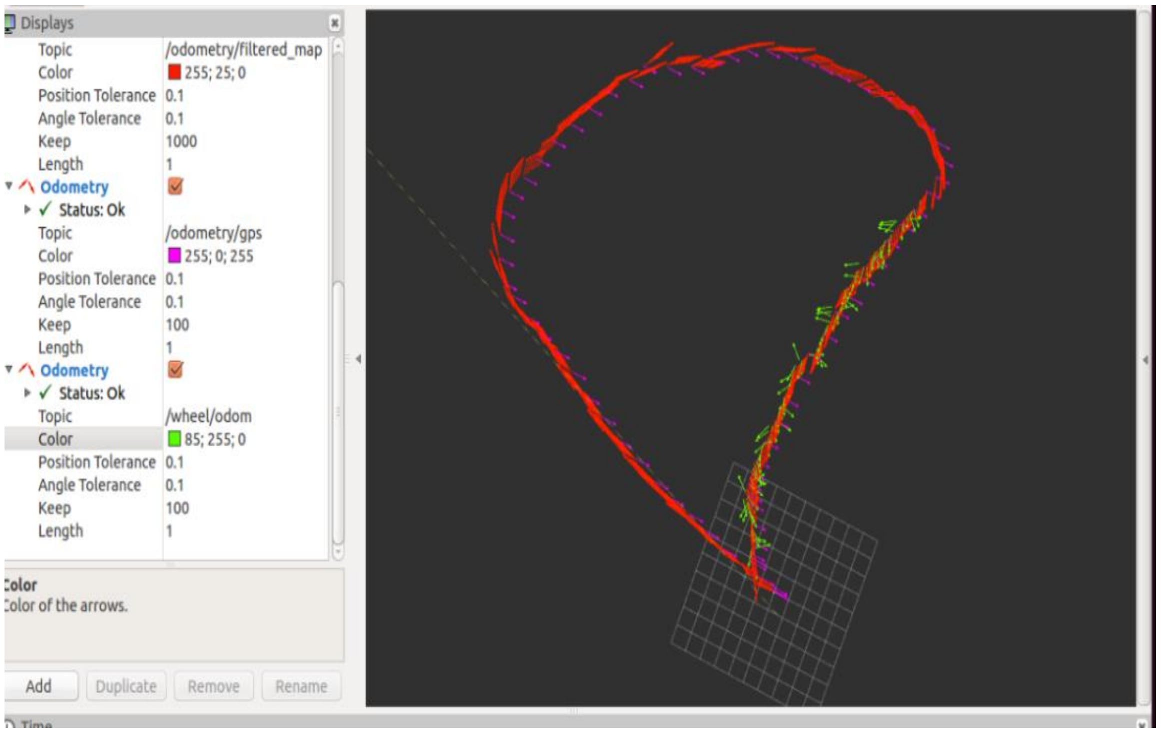
Figure 6.5: Followed trajectory by the wheelchair in outdoor conditions

Table 6.4 describes which configuration for each sensor we fused to get the state estimation from the EKF localization node.

Table 6.4: Sensor Configuration in outdoor conditions

Sensors	**x**	**y**	**z**	**x**	**y**	**z**	$\boldsymbol{\alpha}$	$\boldsymbol{\beta}$	**φ**	$\dot{\boldsymbol{x}}$	$\dot{\boldsymbol{y}}$	$\dot{\boldsymbol{z}}$
Odometry (Encoders)	0	0	0	0	0	0	1	0	1	0	0	1
IMU	0	0	0	1	1	1	1	1	1	1	1	1
GPS	1	1	0	0	0	0	0	0	0	0	0	0

Like indoors, we start at a point and follow a specific trajectory (figure 6.5) and return to that point and measure error in distance between the start and end points. Figures 6.6 – 6.9 show the trajectory of our wheelchair in RViz published by the EKF localization node and the starting and end position of the wheelchair after making the whole run.



Displays

Topic	/odometry/filtered_map
Color	🟥 255; 25; 0
Position Tolerance	0.1
Angle Tolerance	0.1
Keep	1000
Length	1
Odometry	✓
✓ Status: Ok	
Topic	/odometry/gps
Color	🟪 255; 0; 255
Position Tolerance	0.1
Angle Tolerance	0.1
Keep	100
Length	1
Odometry	✓
✓ Status: Ok	
Topic	/wheel/odom
Color	🟩 85; 255; 0
Position Tolerance	0.1
Angle Tolerance	0.1
Keep	100
Length	1

Color
Color of the arrows.

Figure 6.6: Wheelchair trajectory in outdoor condition for test run 1

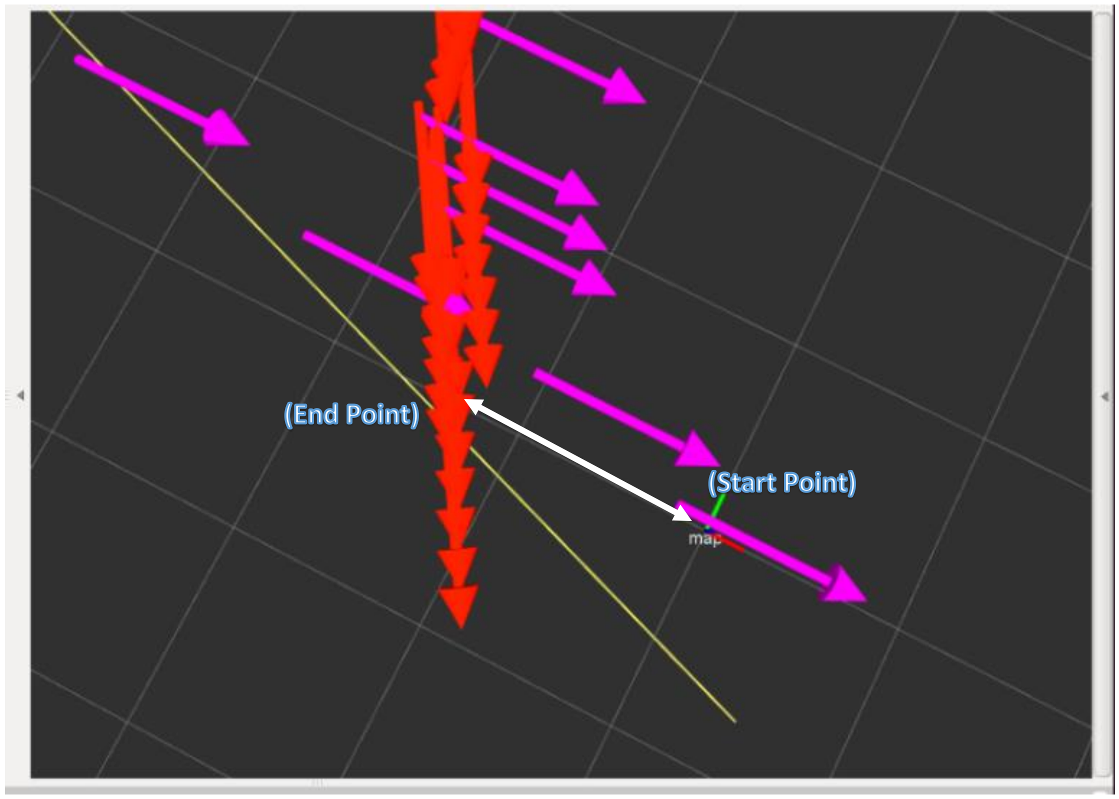


Figure 6.7: Start and end points for outdoor test run 1

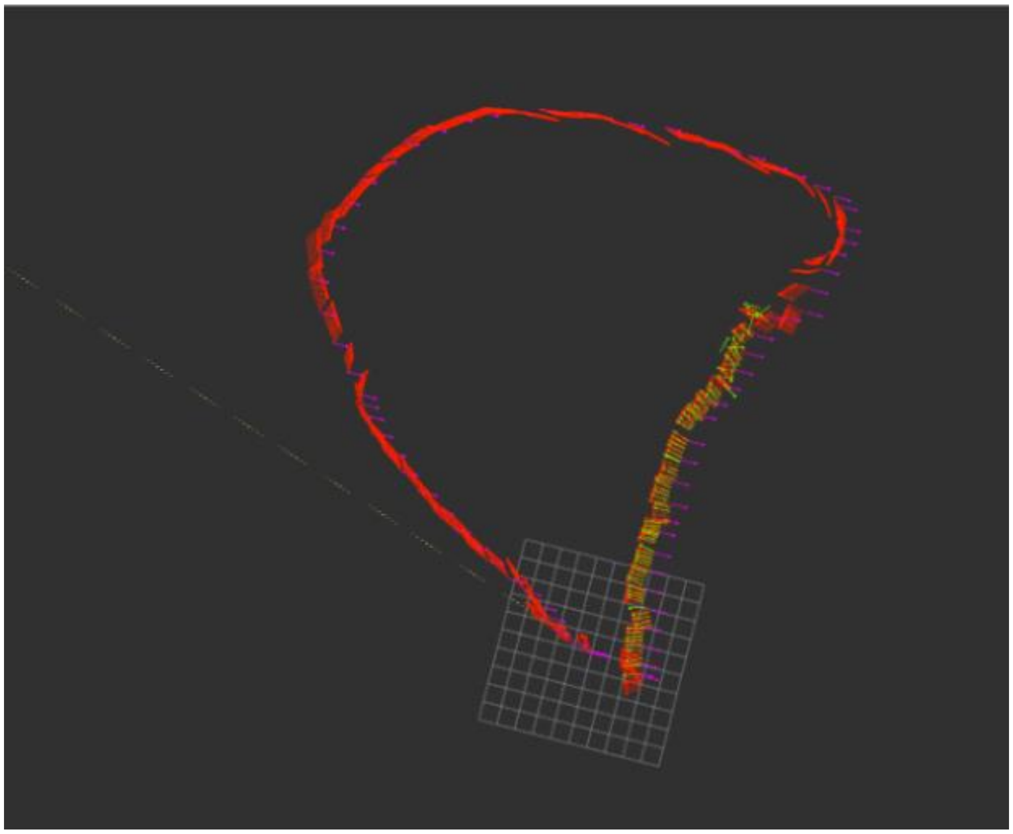


Figure 6.8: Wheelchair trajectory in outdoor condition for test run 2

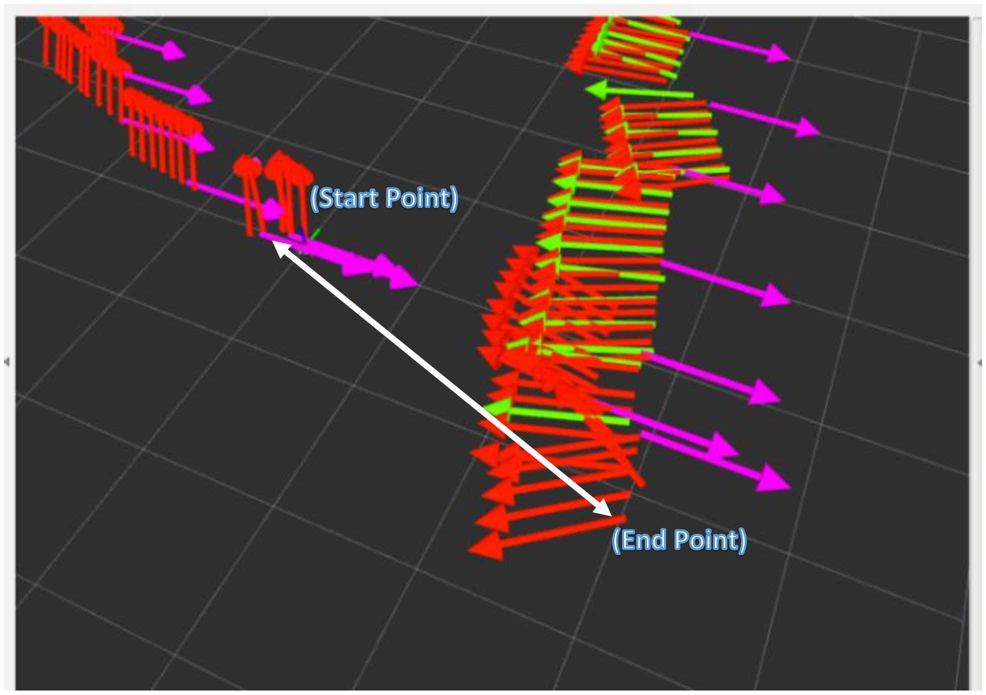


Figure 6.9: Start and end points for outdoor test run 2

The green arrow specifies in which direction the wheelchair is heading as published by the odometry source from the wheel encoders. The red arrow specifies the heading published by the Extended Kalman filter node fusing IMU, GPS and encoder data. The purple arrow is pointing towards North to give us an idea what is our orientation compared to a world fixed frame.

We made a total of four runs outdoors. The loop closure accuracy using the EKF localization node, dead reckoning and IMU along with dead reckoning is provided below.

Table 6.5: Loop accuracy error comparison for outdoors and indoors

Sensor Set	**Loop Closure Error with respect to starting position (in meters)**	**Standard Deviation**
Odometry (Dead Reckoning, CW)	2.1m ~ 4.2m	0.798
IMU + Odometry	0.5m ~ 1.7m	0.602
IMU + Odometry + GPS	1.2m ~ 3.2m	0.941

Among the four runs in outdoor conditions, one of the run experienced large vibrations due to bumps and manholes. We have noticed that the IMU fails due to such vibrations giving a sensor over range error. Yet the EKF localization node performed well in such a case of sensor failure and could estimate our end position with an accuracy of 3.2 meters approximately.

This chapter describes in details about our experiments with the wheelchair in indoor and outdoor condition and compare results with different sensor configuration. We see how our loop closure accuracy is using the EKF and visualize it in RViz. Finally, we compute a standard deviation of our measurements and have an estimate of how spread our measurement distribution is.

Chapter 7

Conclusion

To increase the state estimation accuracy of our robot, we proposed fusing sensor data from GPS, IMU and encoders and implement in an Extended Kalman Filter. We tested our wheelchair both indoors and outdoors. Before running the state estimation algorithm for our wheelchair localization in the environment, we compensated for the effects of hard iron and soft iron effects in the magnetometer of the IMU to get absolute heading which is fused in the EKF algorithm implemented in the robot localization ROS package. The encoders errors were estimated using UMBmark test which gives us dead reckoning errors due to physical imperfections of the wheelchair [6]. We model our wheelchair as a differential wheel drive robot and using its forward kinematics model we calculate linear and angular velocity information which is fused in the EKF along with its corresponding error estimates. Finally, we integrate the GPS in our setup to get absolute position and fuse it to the EKF to get best state estimation possible.

From the loop closure accuracy tests, we find that combining the three sensor measurements with the EKF gives better and stable loop closure accuracy compared to the sensors used in isolation. Indoor environment conditions provide better accuracy compared to outdoor conditions. This is understandable as we are moving in a flat and smooth planar surface indoors where the sensors face less vibration. Whereas, the sensors experience a lot of vibrations due to uneven rough terrains, bumps and potholes, which effects the loop closure accuracy. Vibration dampers can be used to minimize the effect of vibrations on the sensors to get better measurements.

For future applications, we can use visual odometry that uses optical flow and odometry from pointcloud provided by the LIDAR. Integrating these methods would further increase state estimation accuracy using the Extended Kalman Filter.

REFERENCES

[1] Leonard, J. J., & Durrant-Whyte, H. F. (1991). Mobile robot localization by tracking geometric beacons. IEEE Transactions on robotics and Automation, 7(3), 376-382.

[2] Moore, T., & Stouch, D. (2016). A generalized extended kalman filter implementation for the robot operating system. In Intelligent Autonomous Systems 13 (pp. 335-348). Springer, Cham.

[3] Borenstein, J., Everett, H. R., Feng, L., & Wehe, D. K. (1997). Mobile robot positioning: Sensors and techniques.

[4] Chen, L., Hu, H., & McDonald-Maier, K. (2012, September). Ekf based mobile robot localization. In Emerging Security Technologies (EST), 2012 Third International Conference on (pp. 149-154). IEEE.

[5] Global Positioning System Standard Positioning Service Performance Standard, 3rd Edition

[6] Borenstein, J., & Feng, L. (1994). UMBmark: A method for measuring, comparing, and correcting dead-reckoning errors in mobile robots.

[7] Suliman, C., Cruceru, C., & Moldoveanu, F. (2009). Mobile robot position estimation using the Kalman filter. Scientific Bulletin of the" Petru Maior" University of Targu Mures, 6, 75.

[8] Freeston, L. (2002). Applications of the kalman filter algorithm to robot localisation and world modelling. Electrical Engineering Final Year Project.

[9] Betke, K. (2001). The NMEA 0183 protocol. Standard for Interfacing Marine Electronics Devices, National Marine Electronics Association, Severna Park, Maryland, USA

[10] Papadopoulos, E., & Misailidis, M. (2007, July). On differential drive robot odometry with application to path planning. In Control Conference (ECC), 2007 European (pp. 5492-5499). IEEE.

[11] Caruso, M. J. (2000). Applications of magnetic sensors for low cost compass systems. In Position Location and Navigation Symposium, IEEE 2000 (pp. 177-184). IEEE.

[12] Ganganath, N., & Leung, H. (2012, January). Mobile robot localization using odometry and kinect sensor. In Emerging Signal Processing Applications (ESPA), 2012 IEEE International Conference on (pp. 91-94). IEEE.

[13] Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2002, July). FastSLAM: A factored solution to the simultaneous localization and mapping problem. In Aaai/iaai (pp. 593-598).

[14] Jetto, L., Longhi, S., & Venturini, G. (1999). Development and experimental validation of an adaptive extended Kalman filter for the localization of mobile robots. IEEE Transactions on Robotics and Automation, 15(2), 219-229.