# DESIGN OF HIGH PRECISION INDOOR LOCALIZATION ALGORITHM WITH KALMAN FILTER AND MOTION CONTROL SCHEME FOR AN AUTONOMOUS MOBILE ROBOT

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

M.S.

by

Wenbo Lou

August 2021

## ABSTRACT

Autonomous mobile robot nowadays has more and more applications, ranging from everyday application such as food delivery to scientific applications. In many operation scenario, the robot needs to move to a target with high precision under disturbances, sensor noises and many other restrictions. The CCTA-p project presents such a complex setting for the mobile robot to operate, in which a differential drive robot traverses telescope's mirrors to place a measurement device to a set of predefined measurement points. This thesis addresses some of challenges aroused from the project's requirements, such as arriving at the target with very high precision and handling different emergency situations, through the designs of a localization algorithm and a motion control scheme. Meanwhile, the thesis also presents a novel approach for indoor localization by fusing inertial measurements with the laser based external measurement (Etalon) through Kalman Filter. The performance of the designs would be studied in a customizely developed simulation software.

## BIOGRAPHICAL SKETCH

Wenbo Lou is a M.S. student in the Space Imaging and Optical Systems Lab (SIOS) at Cornell University. He received his B.S. degree in Mechanical Engineering from Rice University. His research interests include robotics, motion control and localization.

This document is dedicated to my parents for their love and support all along the way.

## ACKNOWLEDGEMENTS

I would like to express my greatfulness to my advisor, Professor Dmitry Savransky, for his support and feedbacks during my graduate study. I would also like to thank Professor Silvia Ferrari for being my second member in the committee, and for her time and questions during the thesis defense. I would like to thank all the members in my team of the CCAT-p project, Xiaotian Liu, Ryan Gao, Seth MaCall, Phil Si, and Robert Whitney, for their works and suggestions along the way.

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

CHAPTER 1

# INTRODUCTION

## 1.1 Background

The Cerro Chajnantor Atacama Telescope prime (CCAT-p) is a 6-meter diameter telescope that starts to operate in 2021 and is sited at 5600 meters elevation on Cerro Chajnantor in the Atacama desert of northern Chile [5]. The telescope consists of two large mirror made of aluminum panels. Figure 1.1 shows the telescope structure. The primary mirror (M1) has 87 panels and is 30° inclined; the secondary mirror (M2) has 78 panels and is 20° overhang [7]. The orientation of each panel can be adjusted independently by the five adjusters underneath. The surfaces of both M1 and M2 mirrors are not planar, but have curvatures that can be described by a polynomial equation of the general form,

$$z(x,y) = \sum_{i=0}^{k} \sum_{j=0}^{k} a_{ij} (\frac{x}{R_N})^i (\frac{y}{R_N})^j \tag{1.1}$$

where $z$ is the sag at position $(x, y)$ in the mirror frame, $R_N$ is the normalization radius, $k$ is the maximum polynomial power, and $a_{ij}$ are the coefficients [8].

The observatory is interested to know if the orientations of the panels are in the expected alignment, which could be done by measuring whether the locations of the nine *measurement points* on the panel are in the expected locations in space. A retro-reflector (i.e. *puck*) is to be placed on the measurement point so that the laser based measurement system (i.e. Etalon) could measure the position of the puck. As such, an autonomous system is designed to carry the puck

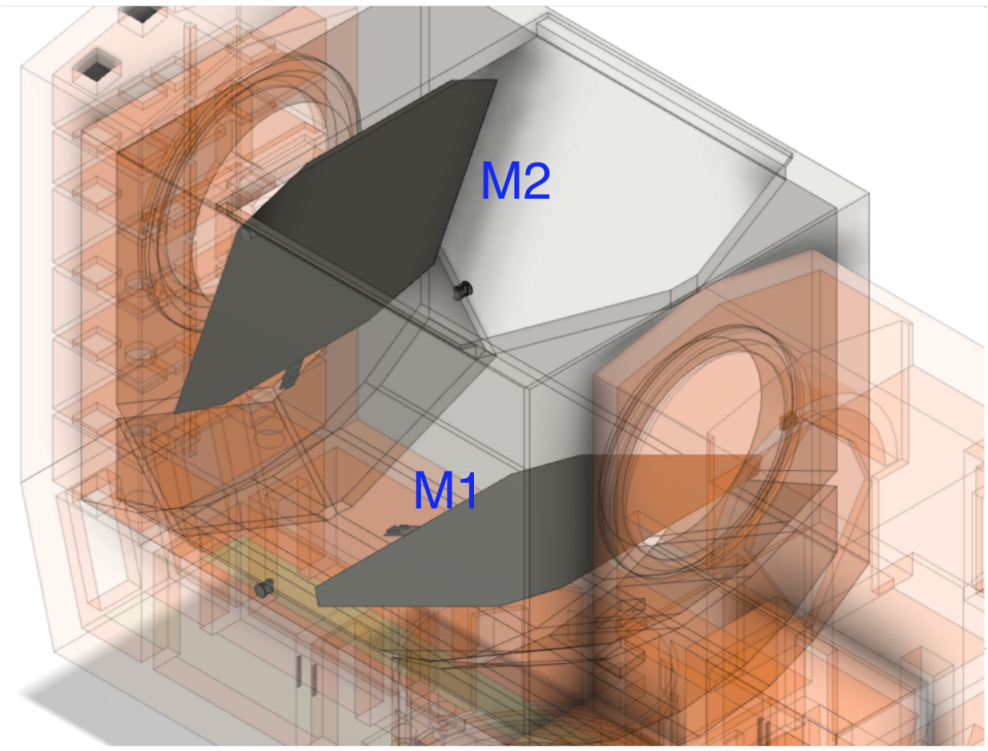


Figure 1.1: The telescope in the observatory. The telescope has two mirrors, M1 and M2 mirrors as labeled in the diagram.

and traverse to all the measurements points so that the Etalon measurement could be carried out. One interesting and challenging aspect of this project is that since the M2 mirror is 20° overhang, to traverse the M2 mirror the robot essentially needs to "climb the wall". To stay on the overhang mirror, the robot uses two powerful fans to push the itself onto the mirror, as shown in figure 1.2.

### 1.1.1 Requirements

The system needs to address several requirements.

- The Etalon measurement system requires that the puck should be placed within 1$cm$ of the measurement point [6].

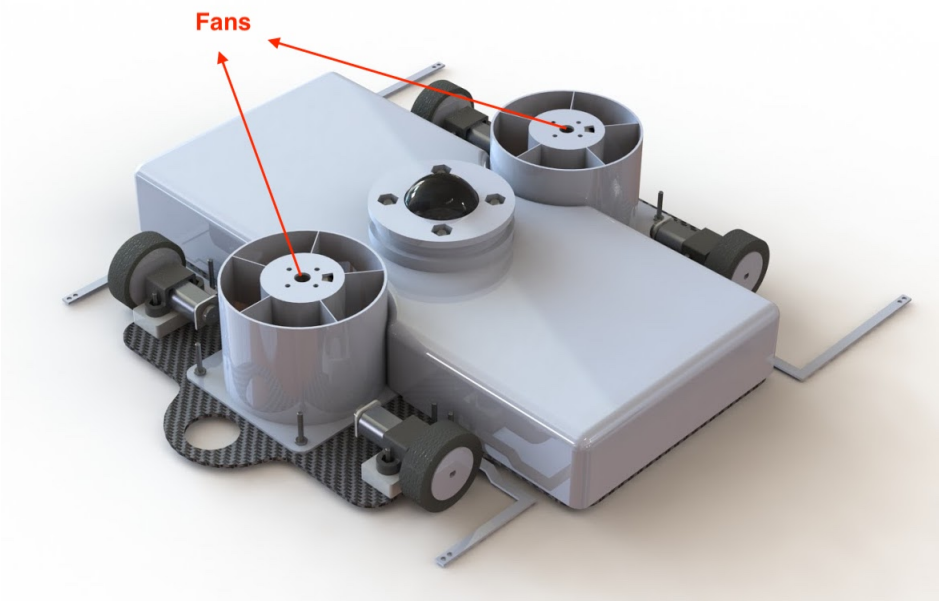


Figure 1.2: The rendering CAD model of the differential drive robot.

- The robot should prevent itself from falling off the edge of the mirror.
- At each measurement point, the robot should stop for measurement and send a start measurement request to the Etalon multi-line server, then resume traversing to the next measurement point.
- The total operation time to traverse the entire mirror should not exceed one hour.
- The robot's total mass **must** not exceed $1kg$.

## 1.2 Problem Statement

While the design of the system involves many aspects including structural design, electronics, motion control, localization and software development, this thesis focuses on the designs of the autonomous functionalities. More precisely,

the purpose of this thesis is to present the designs of the localization system and the motion control system that address the functional requirements stated previously. Meanwhile, the thesis would discuss some unique features of the implementations, and analyze the effectiveness of the designs through a simulation software. The thesis also presents a novel approach for indoor localization by fusing inertial measurement system (INS) with the laser based external measurement (Etalon) through Kalman Filter.

## 1.3 Design Overview

The system consists of a differential drive robot with on-board electronics, an off-board computer, and the Etalon system. The computationally heavy tasks such as localization, motion control and decision making are handled by the off-board computer, while the on-board computer focus on simple tasks such as sending driving commands and fan speed commands, and formating and passing sensor data from the data acquisition device to the off-board computer. Each remote component communicates with each other through TCP/IP protocol, while the on-board sub-components interface with each other through serial and I2C, as shown in figure 1.3.

For the system to perform the autonomous functionalities, the design consists of four components: localization, motion control with way-points tracking, path-planning, and a continuously running loop checking for dangerous conditions (i.e. *danger loop*). Figure 1.4 shows the dependencies between the components. The localization component estimates the robot's current pose by continuously filtering the sensor data and occasionally fusing with Etalon mea-

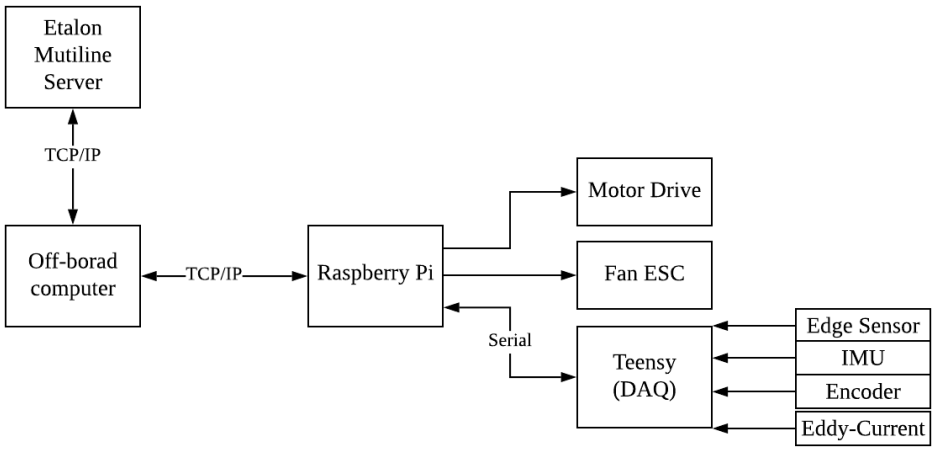


Figure 1.3: The electronic components.

surement and the orientation computed from edge sensors. The path-planning component defines a path consisted of all the measurement points (783 for M1 and 702 for M2). The motion control component computes the next driving command based on the distance from robot's current position to the next measurement point in the path, and the flags from the danger loop, which runs in a separate process and uses sensor data to check for various of conditions.

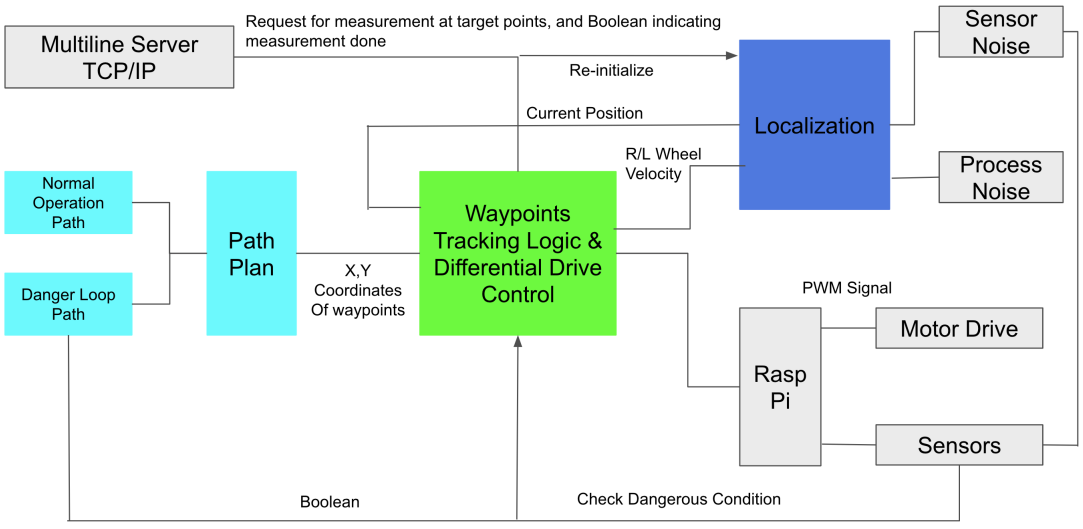


Figure 1.4: An overview of the system design.

## 1.4 Review of Related Work

One of the major challenges that the design needs to address is to provide highly precise localization estimates during the full one-hour operation, under the constraints and requirements of this project. Many approaches exist for localizing a robot in the indoor environment. This section reviews some of the common approaches and suggests why the approach proposed in this thesis is the most suitable solution for this project.

Inertial navigation system (INS) is the common approach for indoor localization. INS uses sensors such as the Inertial Measurement Unit (IMU) and the encoder to compute the position by dead-reckoning, without any knowledge of external references [1]. The INS approach has very high precision for short distance localization. However, INS would suffer from drift and lose its precision dramatically in long distance localization. Therefore, INS often fuses with external measurements such as GPS and visual based measurement to eliminate the accumulated errors, while still being able to provide high precision localization in between short distance.

There are several choices of external measurements. The global positioning system (GPS) is an common approach for fusing with the INS. However, the robot operates in an indoor environment, which makes it not suitable to use the GPS, since the satellite signal is attenuated by the construction materials and obstructions [12], such as the dome of the observatory. The precision of GPS in the indoor environment averages at $5 - 50$ meters, which is far below the required precision for this project [4]. Therefore, GPS is not applicable.

The other common solution for indoor localization is Wi-Fi positioning sys-

tem. Many techniques exist for the Wi-Fi based localization, but they all fail to meet the precision requirement for this project. For example, the first indoor Wi-Fi positioning system, RADAR, has precision of 2 to 3 meters [2]; other techniques such as Horus system and grid-based Bayesian system have precision of at most 1 meter for most of the time [13] [14].

Visual-based measurement that uses camera to measure the relative position of the camera against a set of known external features is another common approach. However, when applied to this project, the visual-based approach has a few drawbacks. First, since the robot has a strict mass limit of less than one kilogram, and a camera usually has mass more than 100 grams, installing a camera on-board would add significant weights to the mass budget. Second, Etalon requires that the puck needs to have at least 120° of clearance around the $z$-axis and no structures are to cross the clearance [6]. Therefore, the on-board camera could not extend above the puck clearance and the field-of-view (FOV) of the camera could be limited. If an off-board camera system is to be used to track the robot's position on the mirror, many cameras are needed to be installed in the observatory, which would drastically increase the overall cost of the system.

In comparison with all the choices above, the Etalon measurement is readily available at each measurement point, and thus requires no additional sensors either on-board or off-board, which would save both the mass-budget and the overall cost. The Etalon measurement also has micrometer level precision, which could effectively eliminate the accumulated error in the position estimation.

## 1.5 Thesis Layout

In chapter 2, the thesis would present the localization system in detail, including the Kalman Filter used to continuously filter the IMU and encoder data, the external measurements used to eliminate the accumulated error, and the implementation that addresses the problem of asynchronous measurement. The chapter would demonstrate and analyze the performance of the localization system through the simulation software. Chapter 3 would present the motion control scheme, including path planning, the design of closed-loop control using waypoint-tracking logic and PID controller, and the danger loop which determines when to respond to dangerous situations. The chapter would also demonstrate the effectiveness of the motion control scheme in the simulation and analyze the performances.

CHAPTER 2

# LOCALIZATION

## 2.1 Algorithm

The purpose of the localization algorithm is to estimate the position and the orientation of the robot on each panel. To be more specific, the output of the algorithm is the pose $(x, y, \theta)_p$, where the subscript $p$ stands for panel frame, which is defined in figure 2.2.

Meanwhile, the context of this project presents some unique challenges for performing localization on the robot. First, as explained in section 1.4, the choice of sensors available for localization is limited due to the project's requirements and practical design choices. Second, the robot is required to localize itself with one centimeter precision so that the measurement at the measurement point is meaningful. Third, the localization algorithm needs to maintain its precision during the full one-hour operation. A Kalman Filter with sensors such as an encoder and accelerometer that only inform the robot's relative position to the previous time step would accumulate large error over time. Therefore, a localization algorithm is designed based on Kalman Filter to fuse encoder data and Inertial Measurement Unit (IMU) data, and occasionally the high precision Etalon measurement data, which could be used to inform the robot's position on the panel, $(x, y)_p$ with proper frame transformations, to eliminate the accumulated error over time and provide high precision estimates of the robot's pose on the panel over the course of the operation.

### 2.1.1 Reference Frames

There are four reference frames that need to be specified to describe the localization algorithm.

First, the origin of the robot frame is attached to the robot's geometric center. The $x - y$ plane is parallel to the plane of the chassis, which is assumed to be a rigid two-dimensional plane. The positive $x$-axis points along the heading of the robot, while the $y$-axis is perpendicular to the $x$-axis, with it's direction following the right-hand rule. The robot frame is shown in figure 2.1. In all subsequent sections, the quantities in robot frame are denoted with the subscript $b$.

Second, the origin of the panel frame is attached to the center measurement point of the panel. The panel is assumed to be a rigid two-dimensional plane. The $x$-axis is pointing along horizontal direction, while the $y$-axis is pointing along vertical direction, with its direction following the right-hand rule. The panel frame is shown in figure 2.2. In all subsequent sections, the quantities in the panel frame are denoted with the subscript $p$. The panel frames in all panels are defined in the same method.

The mirror frame is attached to the center of the mirror. Figure 2.3 shows the two mirror frames of the M1 and M2 mirrors. The mirror frame is used to describe the position of a point on the mirror surface, which is defined in equation 1.1. The world frame is an inertial reference frame with its origin attached to the intersection of the primary mirror's boresight axis and secondary mirror's EL axis, as shown in figure 2.3. The Etalon measurement is represented in the world frame.

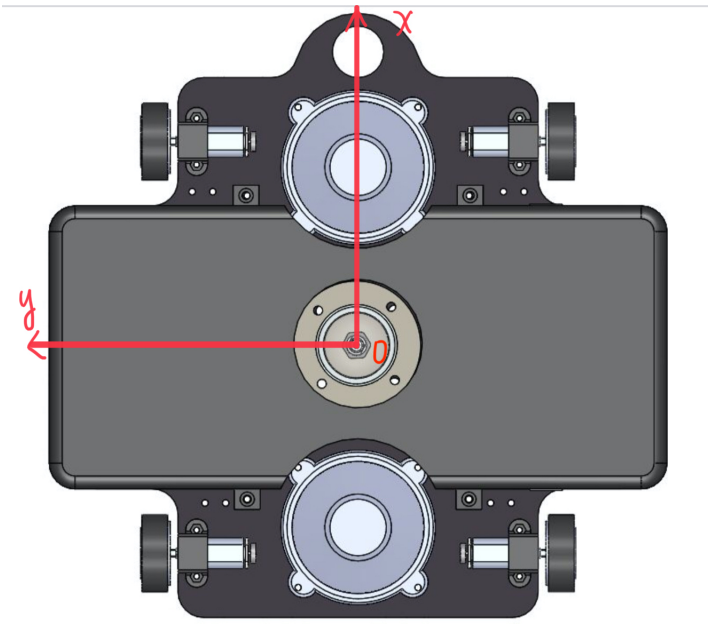


Figure 2.1: The top-down view of the robot and the attached robot frame. The origin of the frame, O, is attached to the robot's center of mass. The x-axis points along the heading of the robot, while the y-axis is perpendicular to the x-axis and follows the right-hand rule.

### 2.1.2 Kalman Filter

Kalman Filter is a Gaussian Filter that estimates the state recursively[10]. The state that Kalman Filter estimates is shown in equation 2.1,

$$\mathbf{q} = \begin{bmatrix} v_b \\ a_b \\ \theta_p \\ \omega_p \end{bmatrix} \tag{2.1}$$

where:

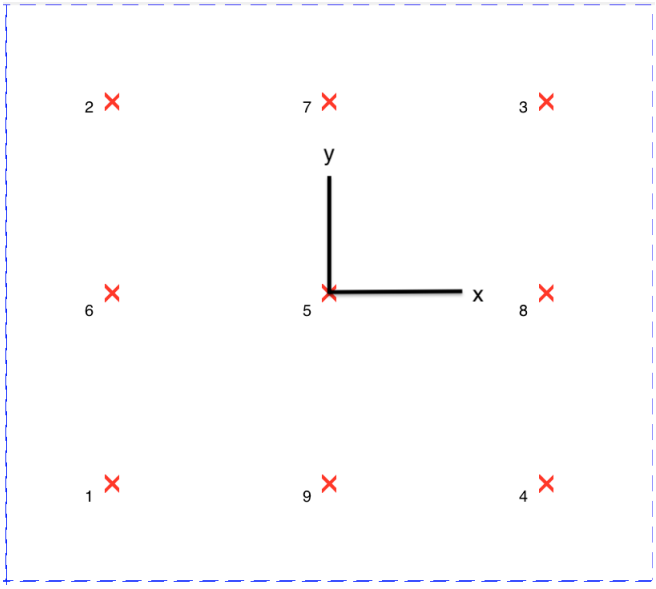


Figure 2.2: The top-down view of the panel and the attached panel frame. The origin of the frame is attached to the panel's central measurement point. The x-axis points along the horizontal direction, while the y-axis points along the vertical direction.

$v_b$ = the forward velocity in robot's frame $x$-axis, $[m/s]$

$a_b$ = the linear acceleration in robot's frame $x$-axis, $[m^2/s]$

$\theta_p$ = the orientation of the robot's frame $x$-axis with respect to panel frame's $x$-axis, $[rad]$

$\omega_p$ = the angular velocity of the robot's frame $x$-axis with respect to panel frame's $x$-axis, [rad/s]

and the state is propagated from time step $k$ to the time step $k+1$ following equation set 2.2.

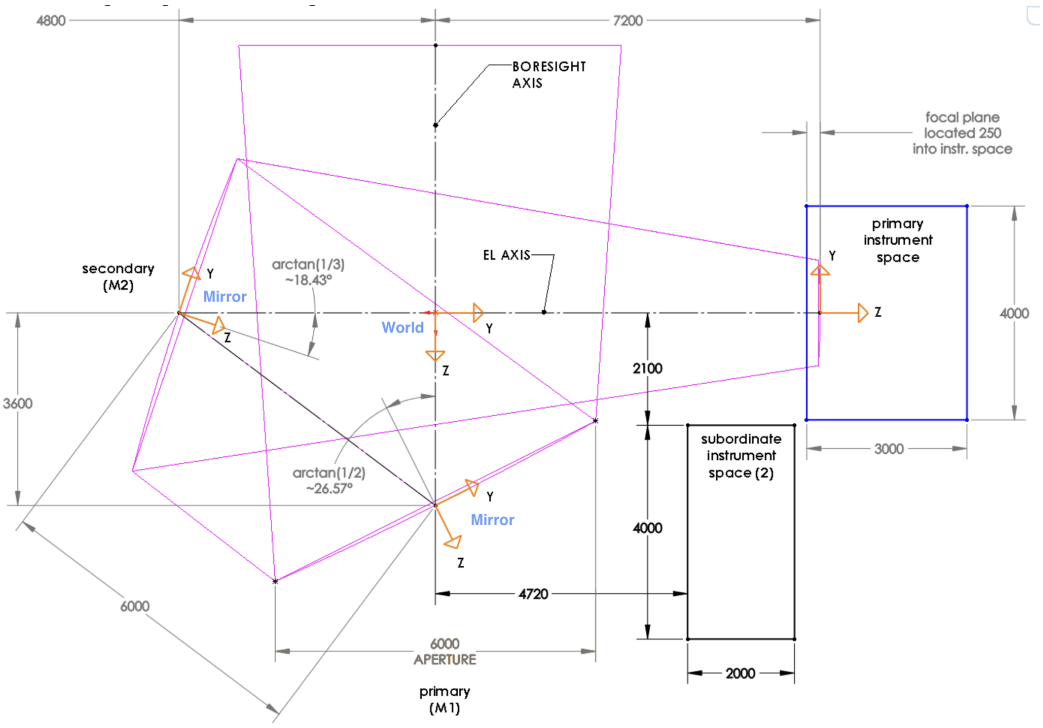




Figure 2.3: The world frame and the two mirror frames of the primary (M1) and secondary (M2) mirror.

$$
\begin{aligned}
v_{b,k+1} &= v_{b,k} + a_{b,k}\Delta t \\
a_{b,k+1} &= a_{b,k} \\
\theta_{p,k+1} &= \theta_{p,k} + \omega_{p,k}\Delta t \\
\omega_{p,k+1} &= \omega_{p,k}
\end{aligned}
\tag{2.2}
$$

The measurement vector is shown in equation 2.3,

$$\mathbf{z} = \begin{bmatrix} a_x \\ \omega_R \\ \omega_L \\ \theta_z \end{bmatrix} \tag{2.3}$$

where:

$a_x$ = the linear acceleration in robot's frame $x$-axis, $[m^2/s]$

$\theta_z$ = the orientation of the robot's frame $x$-axis with respect to panel frame's $x$-axis, $[rad]$

$\omega_R$ = the angular velocity of the robot's right wheel, $[rad/s]$

$\omega_L$ = the angular velocity of the robot's left wheel, $[rad/s]$

and the state and measurement are related by the measurement function in equation 2.4.

$$\begin{bmatrix} a_x \\ \omega_R \\ \omega_L \\ \theta_z \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1/r & 0 & 0 & L/2r \\ 1/r & 0 & 0 & -L/2r \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} v_b \\ a_b \\ \theta_p \\ \omega_p \end{bmatrix} \tag{2.4}$$

where:

r = the wheel's radius, $[m]$

L = the width of the wheel base, $[m]$

Since the both state propagation and the measurement function can be described by sets of linear equations, the linearity assumptions of Kalman Filter are obeyed [10].

One iteration of state estimation in Kalman Filter involves two steps: prediction and update [3]. The state prediction follows equation 2.5,

$$\bar{\mathbf{x}}_{k+1} = F_k \mathbf{x}_k + G_k \mathbf{u}_{k+1} \tag{2.5}$$

where the state is predicted using the linear state propagation model, as represented by the matrix $F_k$. $u_{k+1}$ is the input to the system at time step $k+1$. The state co-variance prediction follows equation 2.6,

$$\bar{P}_{k+1} = F_k P_k F_k^T + Q_k \tag{2.6}$$

where $P_k$ denotes the state co-variance matrix, and $Q_k$ denotes the co-variance matrix of the process noise. The measurement prediction follows equation 2.7,

$$\bar{\mathbf{z}}_{k+1} = H_{k+1} \bar{\mathbf{x}}_{k+1} \tag{2.7}$$

where $\bar{\mathbf{z}}_{k+1}$ is the predicted measurement given the predicted state, and $H$ is the measurement model that relates the state and the measurement. All the overhead bar denotes the prediction value.

Equation 2.8 computes the Kalman gain,

$$W_{k+1} = \bar{P}_{k+1} H_{k+1}^T (H_{k+1} \bar{P}_{k+1} H_{k+1}^T + R_{k+1})^{-1} \tag{2.8}$$

where $W_{k+1}$ denotes the Kalman gain, and $R_{k+1}$ denotes the co-variance ma-

trix of the measurement noise. The state update follows equation 2.9,

$$\mathbf{x}_{k+1} = \bar{\mathbf{x}}_{k+1} + W_{k+1}(\mathbf{z}_{k+1} - \bar{\mathbf{z}}_{k+1}) \tag{2.9}$$

where $z_{k+1}$ is the actual measurement vector. The state co-variance follows equation 2.10,

$$P_{k+1} = (I - W_{k+1}H_{k+1})\bar{P}_{k+1} \tag{2.10}$$

where $I$ is an identity matrix.

The dimensions of the matrices in the above equations is described as follow. If state is a $n - by - 1$ vector and measurement is a m-by-1 vector:

$F$ = n-by-n
$P$ = n-by-n
$Q$ = n-by-n
$H$ = m-by-n
$R$ = m-by-m
$W$ = n-by-m
$I$ = n-by-n

### 2.1.3 Algorithm outline

The localization algorithm is designed based on the Kalman Filter. The outline of the algorithm is shown in figure 2.4. The Kalman Filter is continuously running to update the state using encoders data and IMU data. Then,

the state is used to compute the robot's position estimate on the panel, $(x, y)_p$, using the robot's kinematic equation. Combining with the robot's orientation on the panel, $\theta_p$, the localization algorithm outputs the robot's pose estimate, $(x, y, \theta)_p$. At each measurement point, the robot receives the Etalon measurement that measures the position of the puck center in global frame. The Etalon measurement is transformed to the robot's position in panel frame to correct the position estimate. Since the robot stops at the measurement point to take measurement, the acceleration, linear velocity, and angular velocity are known to be zero. Therefore, the Kalman Filter can be re-initialized by setting $v_b, a_b, \omega_p$ of the state to zero, and resetting the corresponding diagonal entries in the state covariance. When the robot crosses the panel, the edge sensors compute the angle at which the robot enters the next panel. The entering angle is used to compute the robot's orientation in the panel frame, and is used both to correct the orientation in the pose estimate and re-initialize the orientation in the Kalman Filter state.

The Kalman Filter is implemented using the equations described in the previous section. It is assumed that system time-invariant. Therefore, the matrices are constant and the subscript that denotes the time step is dropped. Formulate the state propagation in 2.2 in the format of Kalman Filter defined in equation 2.5, the matrix $F$ is defined in equation 2.11, while input $\mathbf{u}$ is zero.

$$F = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

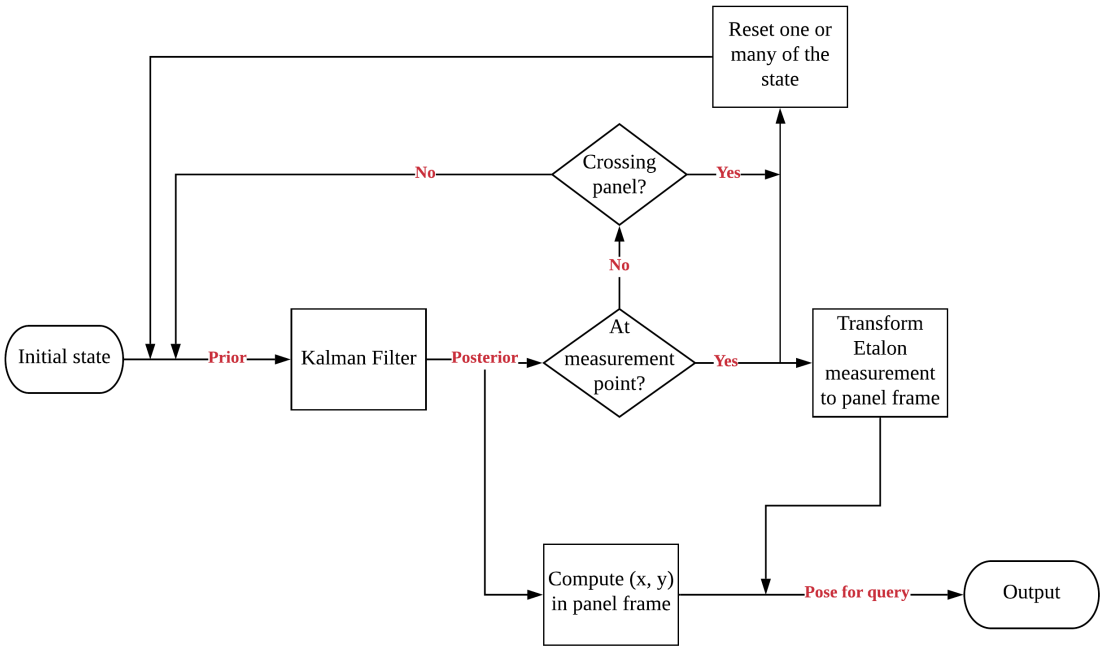


Figure 2.4: Localization Algorithm Outline

Formulate the measurement data and the measurement function that maps the relationship between the state and measurement data in the format of equation 2.7, the measurement vector $\mathbf{z}$ and the matrix $H$ are defined in equation 2.12.

$$\mathbf{z} = \begin{bmatrix} a_x \\ \omega_R \\ \omega_L \\ w_z \end{bmatrix} \qquad H = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1/r & 0 & 0 & L/2r \\ 1/r & 0 & 0 & -L/2r \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.12}$$

The implementation of Kalman Filter is summarized in the set of equations 2.13.

$$
\begin{aligned}
\bar{\mathbf{q}}_{k+1} &= F\mathbf{q}_k \\
\bar{P}_{k+1} &= FP_kF^T + Q \\
\bar{\mathbf{z}}_{k+1} &= H\bar{\mathbf{q}}_{k+1} \\
W_{k+1} &= \bar{P}_{k+1}H^T(H\bar{P}_{k+1}H^T + R)^{-1} \\
\mathbf{q}_{k+1} &= \bar{\mathbf{q}}_{k+1} + W_{k+1}(\mathbf{z}_{k+1} - \bar{\mathbf{z}}_{k+1}) \\
P_{k+1} &= (I - W_{k+1}H)\bar{P}_{k+1}
\end{aligned}
\tag{2.13}
$$

The forward velocity, $v$, and the robot's orientation with respect to x axis of the panel frame, $\theta$, are then used to compute the robot's $(x, y)$ position in the panel frame, following the robot's kinematic model in equation 2.14.

$$
\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} x_k + v_{k+1}cos(\theta_{k+1}\Delta t) \\ y_k + v_{k+1}sin(\theta_{k+1}\Delta t) \end{bmatrix} \tag{2.14}
$$

### 2.1.4 Asynchronous Measurement

The issue of asynchronous measurement could happen when the sensor samples data at different frequencies. For this project, the maximum sampling rate of the IMU, which includes the accelerometer and the gyroscope, is 100 Hz [9], while maximum the sampling rate of the encoder is 1000 Hz. Different sampling rates mean that when the four sensor data are packed together and sent to the main-computer, the four sensor date may have different time-stamps. When such issue arises, one solution is to only use the measurement data with the latest time-stamp at the moment when the measurement data is queried, and discard all measurement data with earlier time-stamps.

Implementing such solution in the Kalman Filter implies that the dimension of the measurement vector, $z_k$, might be different and needs to be adjusted in each iteration. Accordingly, the dimensions of the measurement function, $H$, and the measurement noise co-variance, $R$, also need to be adjusted. In specific, when the measurement vector has dimension $m \times 1$, the measurement function $H$ has dimension $m \times n$, and co-variance $R$ has dimension $m \times m$, where $n$ is the dimension of the state, which is four in this project. The dimension of the Kalman gain, $K$, also varies accordingly. However, since $K$ is the computed result of other matrices as shown in equation 2.13, it does not need to be adjusted manually. The dimensions of the rest matrices, namely, $F, Q, P, \mathbf{q}$, remain unchanged.

Therefore, one way to implement the modified Kalman Filter is to pass in the matrices with varying dimensions as parameters before each iteration. In specific, the measurement vector, $\mathbf{z}_k$, would be constructed based on the available measurement data, while the corresponding rows of the complete measurement function and measurement noise matrices are extracted to construct the $H$ and $R$ matrices for each iteration. The detail of this implementation is shown in the next section.

With the modified Kalman Filter, the complete outline of the localization algorithm can be summarized in figure 2.5. The output of the localization algorithm is the pose of the robot on the panel, $(x, y, \theta)_p$, which would be queried by other processes such as the motion control and path planning.

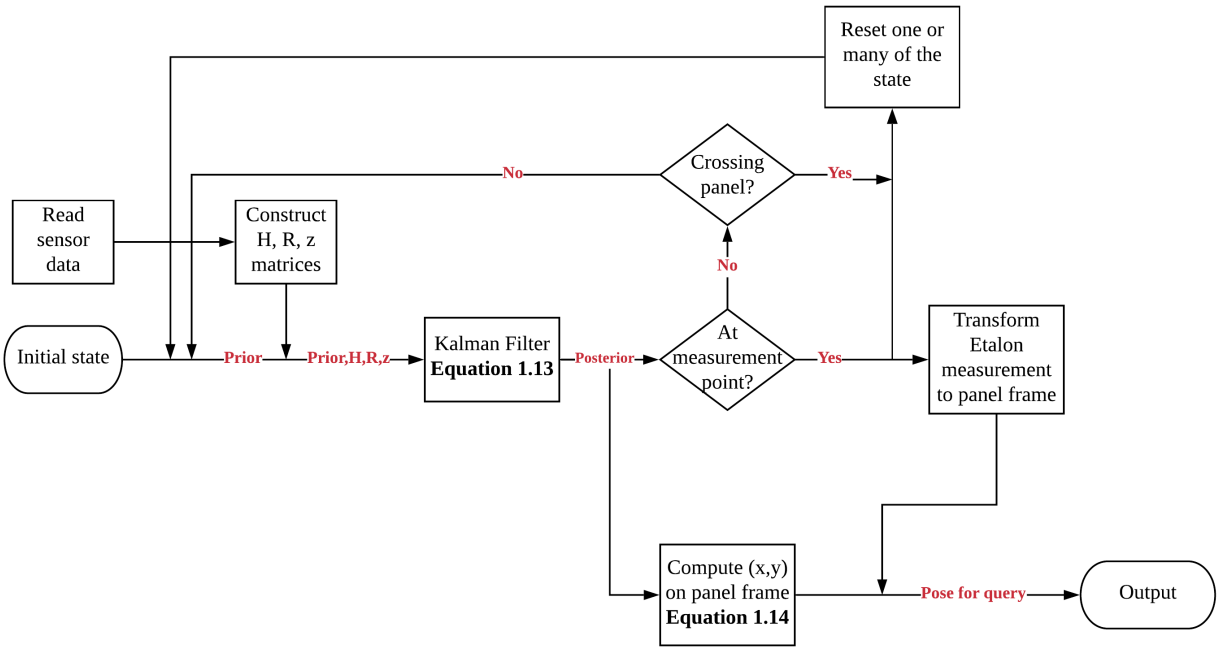


Figure 2.5: Complete outline of the localization algorithm.

### 2.1.5 Cross-Panel Adjustment

There is accumulated error in the orientation estimate, which could cause the localization algorithm to lose its accuracy. To eliminate the accumulated error, the localization algorithm should be provided with another source of orientation measurement. This orientation measurement could come from the angle at which the robot enters the panel when it crosses from one panel to another. The angle of entering the panel is measured by the four edge sensors placed at the four corners of the robot, as shown in figure 2.6.

Figure 2.7 shows the geometric relationship between robot's front two edge sensors and the panel's edge. According to the geometric relationship, the angle

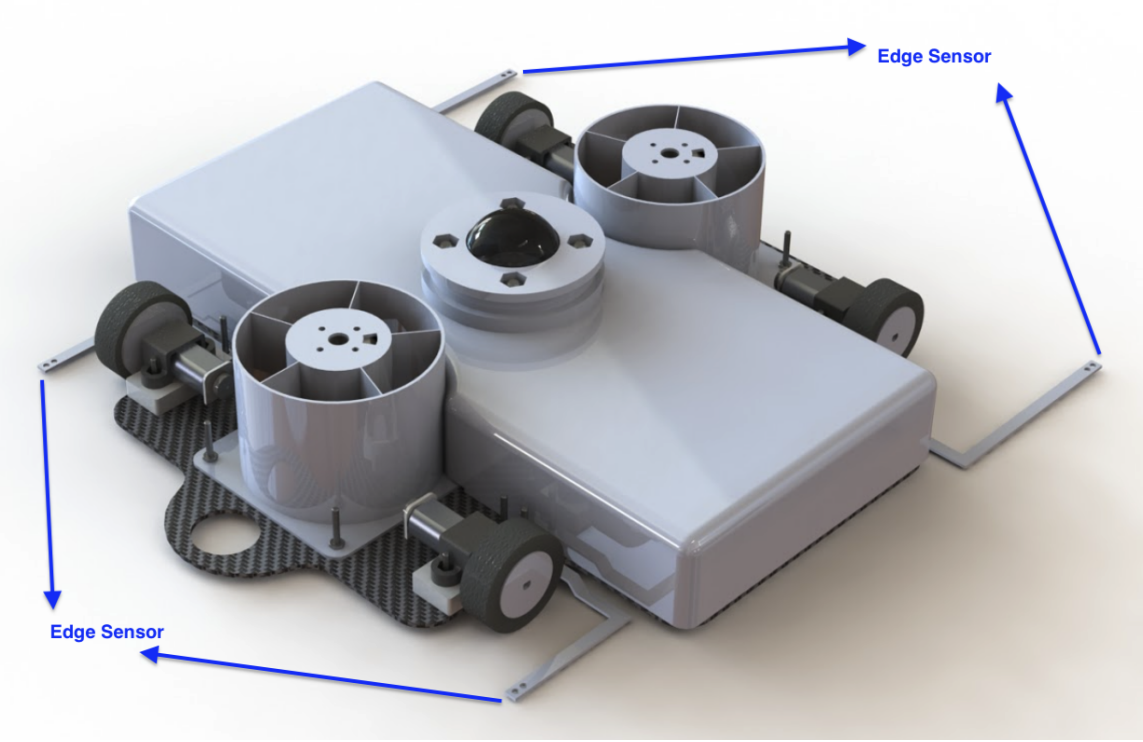


Figure 2.6: The four edge sensors are positioned at four corners of the robot. The edge sensors are attached to the L-shape supports, which extruded from the robot's chassis, so that the robot could detect the edge ahead of time.

of entering the panel, $\alpha$, can be computed using equation 2.15,

$$\begin{aligned} \alpha &= arctan(d/L) \\ d &= v_b t \end{aligned} \tag{2.15}$$

where $t$ is the time interval between the first and second edge sensor entering the panel, and $d$ is the distance the robot travels during this interval at forward velocity $v_b$. To convert the angle of entering the panel to the robot's orientation with respect to the panel frame, one could follow equation 2.16,

$$\theta = \pi/2 - \alpha \tag{2.16}$$

The angle of entering panel can be computed in the same way with the two back edge sensors. Since there are four edge sensors around the robot, the angle

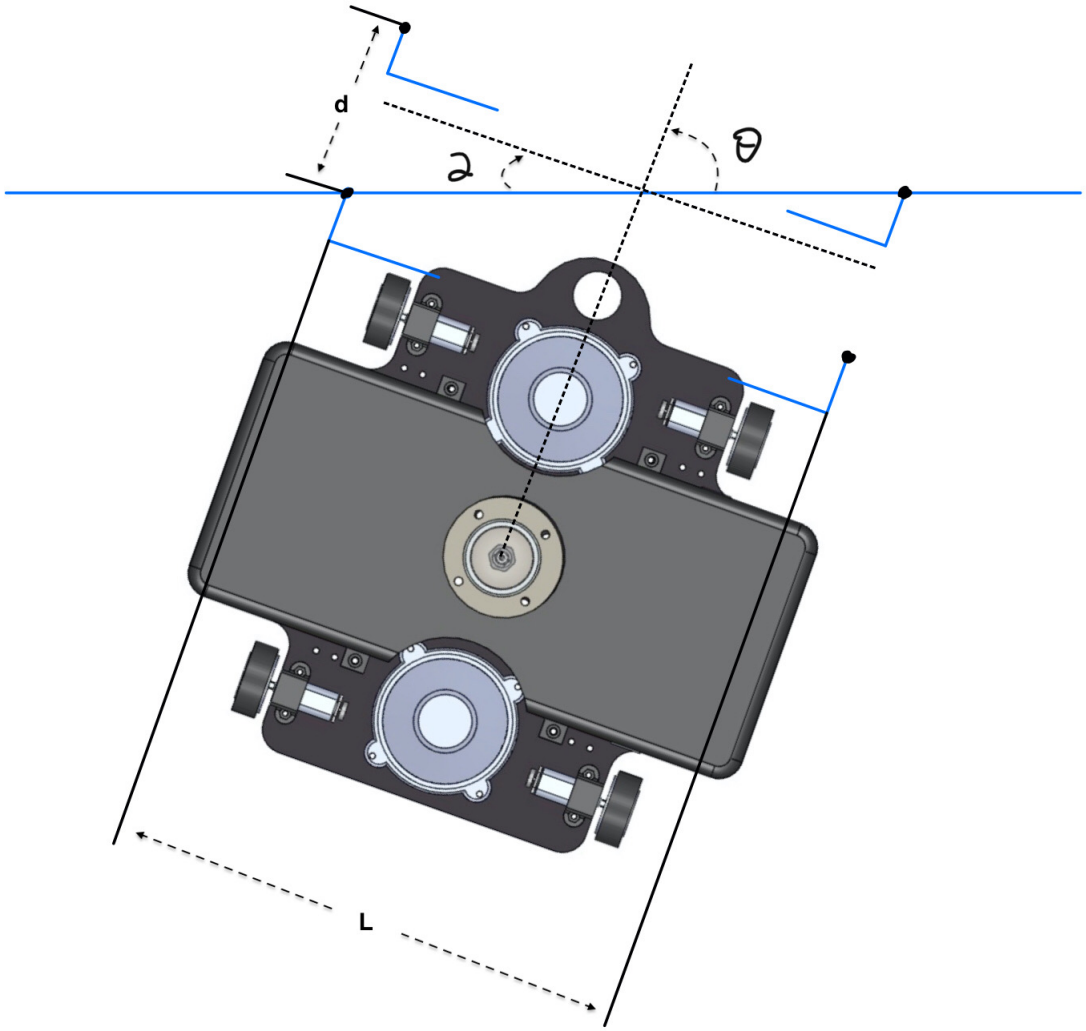


Figure 2.7: The geometric relationship between the robot's chassis and the edge. $\alpha$ is the angle of entering panel, $\theta$ is the robot's orientation in the panel frame, $d$ is the distance travels by the edge sensor, $L$ is the distance between the two edge sensors.

of entering the panel could be computed twice with the front two sensors and the back two sensors.

## 2.2 Implementation

As shown in figure 2.5, the localization algorithm consists of a continuous filtering process and an occasional re-initialization process. Therefore, it is best to implement the two processes in two objects. In specific, the Filter interface implements a filter such as the Kalman Filter to continuously estimate the state, while the Localizer class is composited with a Filter and is responsible for re-initializing the filter by resetting $\mu$ and $\Sigma$. In addition, the Localizer class can also handle the asynchronous measurement data issued by adjusting the dimensions of the matrices for the Kalman Filter based on the available sensor data before each iteration.

Figure 2.8 shows the class diagram of the localization algorithm. The Localizer class reads the measurement data and constructs the matrices of the Kalman Filter with the correct dimensions, using the method **contructKalmanMat()**. It then delegates the filtering process to the Filter object in **update()**. For this project, the Filter is implemented using Kalman Filter, with the equations described in 2.13 in specific. Using the estimated forward velocity and the orientation, the Localizer computes the position of the robot, $(x, y)_p$, on the panel, as described in equation 2.14, and thus completes the process of estimating the robot pose, $(x, y, \theta)_p$. The **resetFilter($\mu$, $\Sigma$)** method is used to re-initialize the filter.

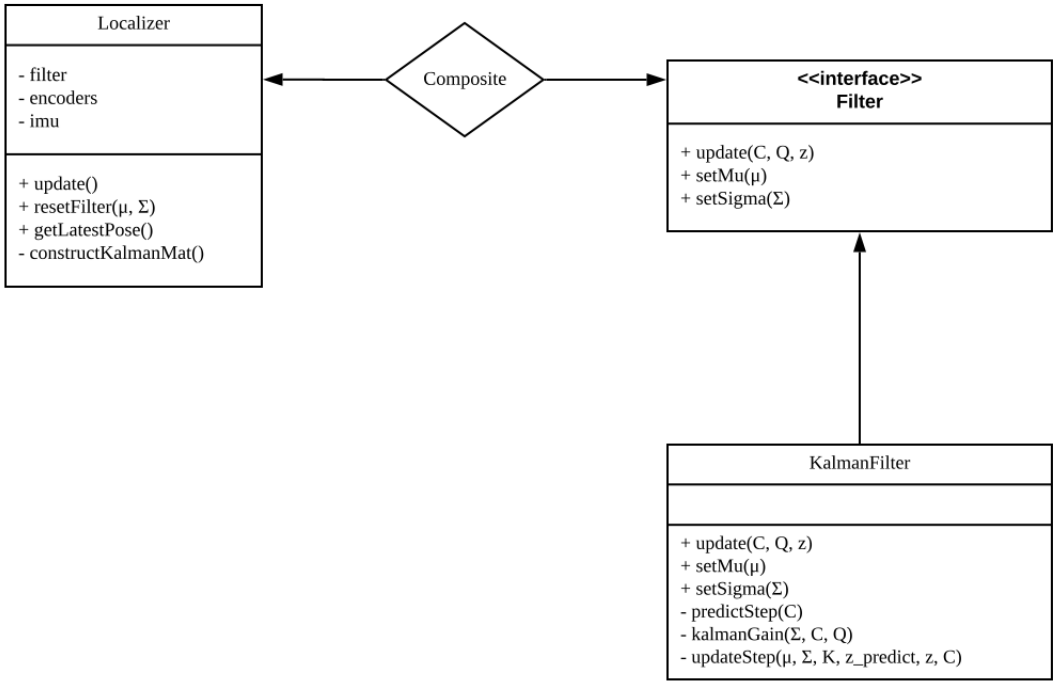


Figure 2.8: Unified Modelling Language (UML) of localization system's implementation

## 2.3 Simulation and Result

### 2.3.1 Simulation Setup

The purpose of the simulation software is to demonstrate the functional performance of all implementations, which are the actual implementations that would be used in the real operation; the entire localization algorithm described above is implemented in the simulation software, along with the motion control scheme, which would be introduced in the next chapter. However, due to the lack of the actual hardware, the simulation should also simulate the behaviour of the actual robot to supply "fake" sensor data. Therefore, in addition to the programs used in the real operation, the simulation software has another independent process,

the "DataLoop", which essentially acts as the virtual robot to receive speed command from the main loop process and respond with the "fake" sensor data. The "DataLoop" is also responsible for simulating the noise of each sensor. In specific, the noises of the accelerometer, gyroscope and encoder are modelled as additive white noises $N_w$ described by the zero-mean Gaussian distribution in equation,

$$N_w = N(0, \sigma) \tag{2.17}$$

where $\sigma$ is the variance representing the strength of the white noise. One benefit of the simulation software is that the noise levels of the sensors can be adjusted to study the performance of the localization algorithm and motion control scheme under different situations. To be specific, there are two noise levels: the noise level that matches specification of the actual sensor and the noise level twice as much as that of the actual sensor. The reason of testing for higher noise level is that the noise generally varies proportionally with bandwidth, which is currently set at 100 $Hz$ for the accelerometer and the gyroscope, but could be set to higher bandwidth to meet any potential changes in the project's requirements in the future. The specifications of the BNO055 Inertial Measurement Unit used for this project [9] described the noises of the accelerometer and the gyroscope, while the noise of the encoder is manually measured by collecting encoder data and computing the standard deviation. The values of the noise strength used in the simulation are shown in table 2.1 below.

The measurement noise and sensor noise in the Kalman Filter are adjusted accordingly. The performance of the localization algorithm is analyzed in the

| Noise level | Accelerometer ($m/s^2$) | Gyroscope ($rad/s$) | Encoder ($rad/s$) |
|---|---|---|---|
| normal | 0.0059 | 0.0087 | 0.001 |
| twice | 0.0118 | 0.0175 | 0.002 |

Table 2.1: The strengths of sensor noises used in the simulation.

section below, while the performance of the motion control scheme would be shown in the next chapter.

### 2.3.2 Result and Analysis

The simulation is run with normal noise level first. Figure 2.9 shows the robot's true positions and the estimated positions in one complete simulation run. The L2 error can be used to measure the localization error during the simulation, which is the distance between the true and estimated position and can be described by equation 2.18 [11],

$$error_{L2} = \sqrt{(x_{true} - x_{estimate})^2 + (y_{true} - y_{estimate})^2} \tag{2.18}$$

The L2 error is computed for every data point during the above simulation run, and the histogram in figure 2.10 visualizes the distribution of the L2 errors. The distribution plot shows that the localization is very accurate for most of the time, with 96.21% of the errors fall below 0.01 m, mean error equals to 0.0034 m and largest error equals to 0.0181 m.

The simulation is then run with higher noise level. Figure 2.11 shows the robot's true positions and the estimated positions in the second run, while figure 2.12 shows the distribution of the L2 errors. The localization is still accurate enough for the robot to track the waypoints, although the localization errors are

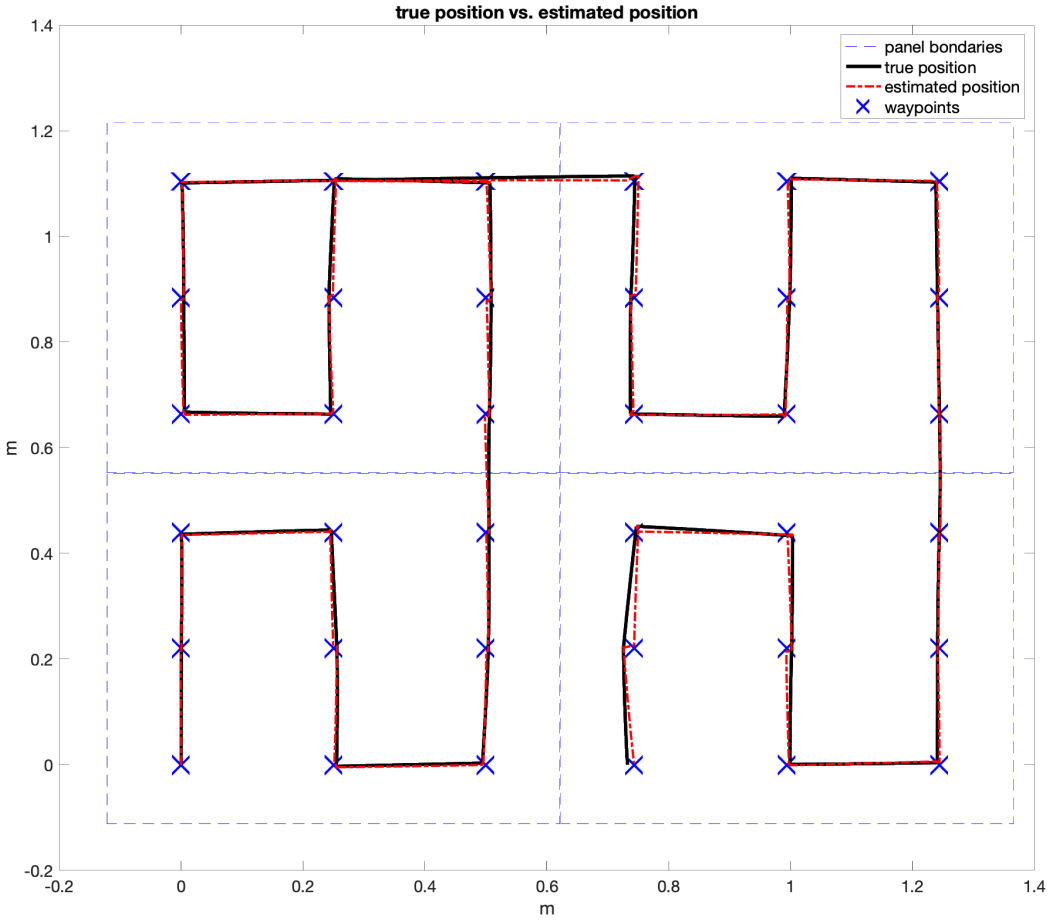


Figure 2.9: The robot's true positions and the estimated positions in the simulation with normal noise level.

evidently larger than those from the first run, with 92.5% of the errors fall below 0.01 m, mean error equals to 0.0039 m and largest error equals to 0.0317 m.

## 2.4 Summary

This chapter introduces the localization algorithm, including how the Kalman Filter is fused together with the occasional ultra-accurate position measurement from Etalon system, how the algorithm deals with asynchronous measurement, and how the robot computes the angle of entering panel when crossing from

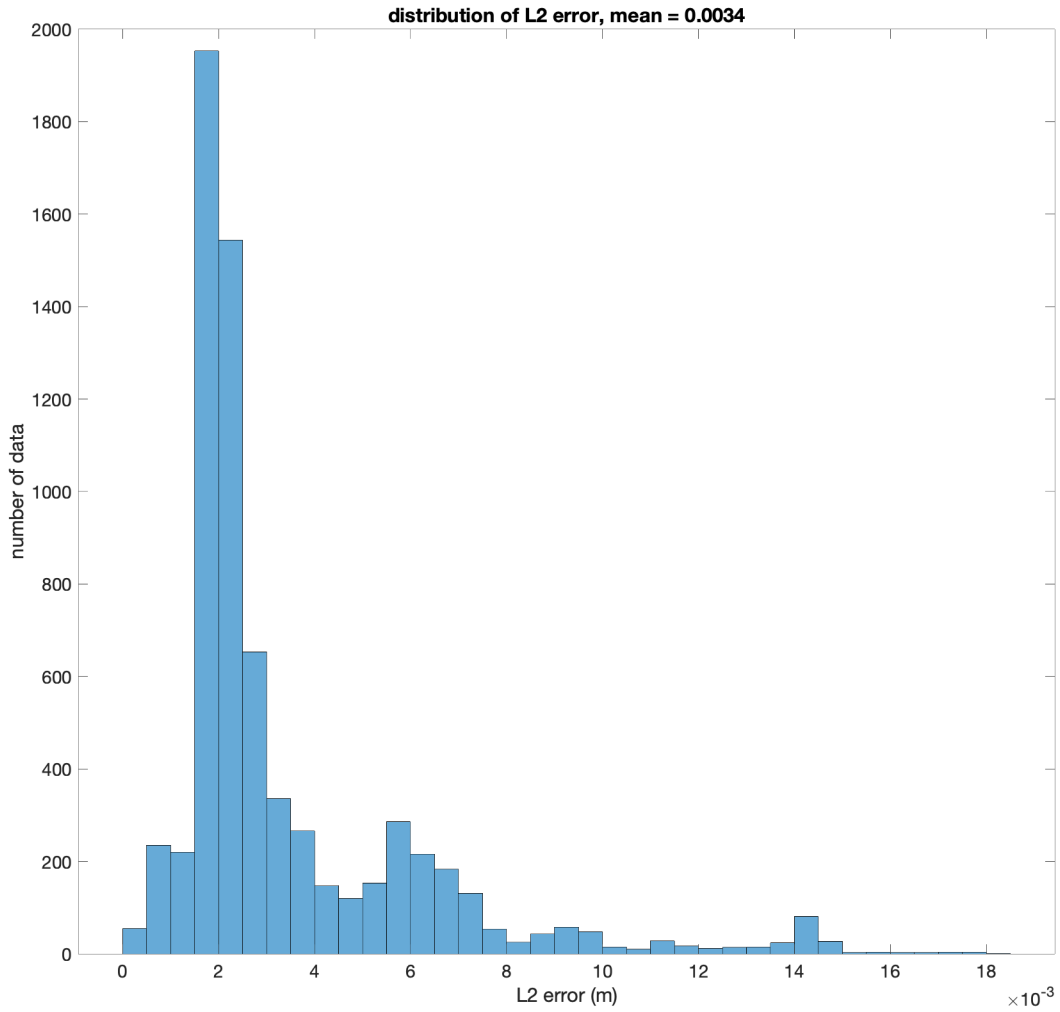


Figure 2.10: The L2 error between the true positions and the estimated positions.

one panel to another, and use it to eliminate the estimation error in orientation. The chapter also demonstrates the performance of the localization algorithm in the simulation software under two noise levels, and provides statistical analysis of the localization errors. Now that a trustworthy localization is in place, the control program can make use of this knowledge to determine the robot's next move (i.e the robot's speeds) under different situations. Next chapter would discuss the details of the motion control scheme.

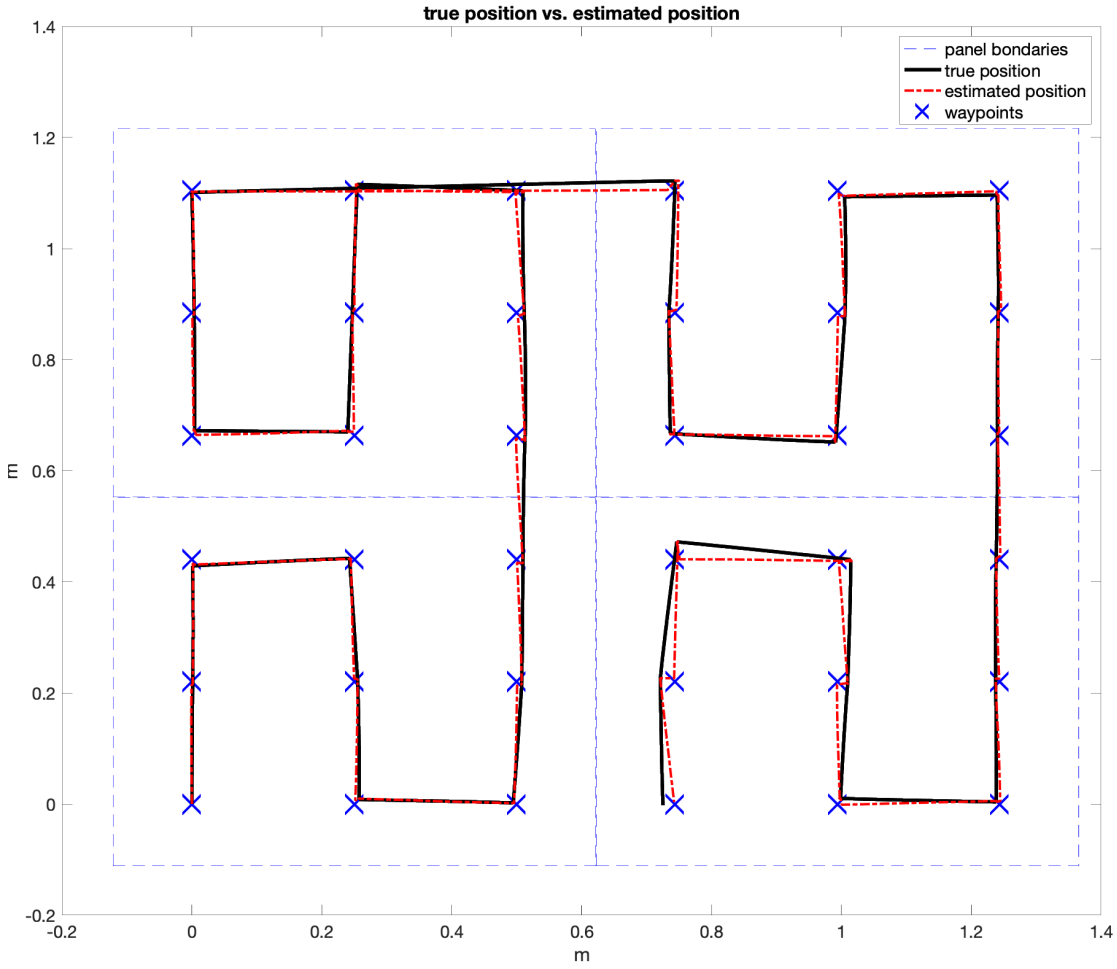


Figure 2.11: The robot's true positions and the estimated positions in the simulation with twice as much noise level.

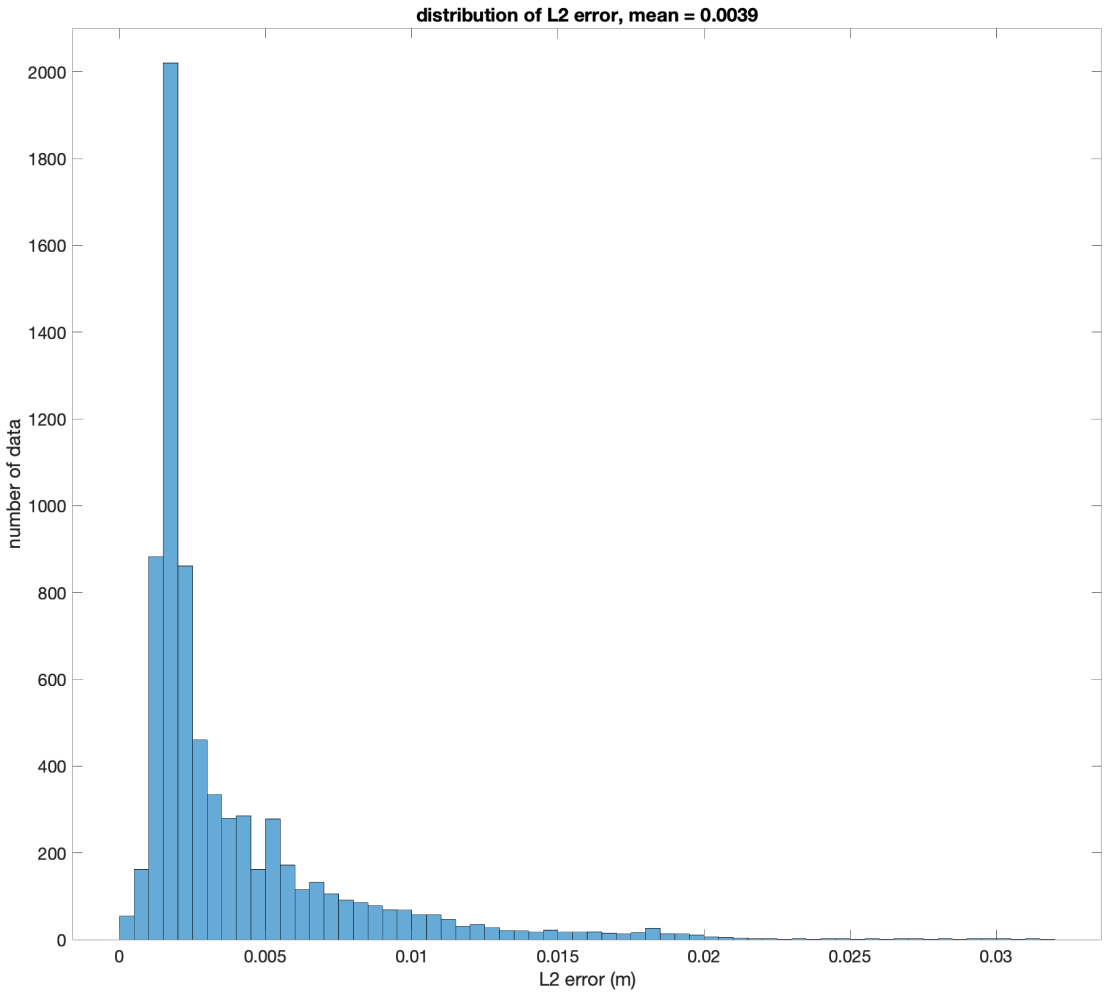


Figure 2.12: The L2 error between the true positions and the estimated positions with twice as much noise level.

CHAPTER 3

# MOTION CONTROL

## 3.1 Overview

The main purpose of the differential-drive robot is to place the puck at each measurement point, where the robot needs to stop so that the Etalon system can measure its position. Moreover, when operating on the mirror, the robot should respond to more complicated situations; for example, in the case of emergencies (e.g. power outage, earthquake, fire alarm, etc.) the entire operation needs to be aborted, and in the danger of falling-off the mirror the robot needs to back-off from the edge. Therefore, a motion control scheme is needed so that the robot would receive the correct drive command under different situations.

The motion control scheme consists of three parts: the main loop process that implements waypoint tracking logic and PID controllers to compute the speed command that needs to be sent to the robot through TCP communication, the danger loop process that runs in parallel to the main loop and continuously checks for dangerous conditions using sensor data and emergency signal from the observatory, and the on-board low-level control that takes the speed command from the off-board computer to compute the voltage command to the DC motors. Moreover, the main-loop process is also implemented with path planner that selects and generates a path consisted of a series of waypoints based on the robot's operation status, and also moniters the current waypoints for the waypoint tracking logic to use. The flowchart in figure 3.1 provides an overview of the relationships among different parts of the control scheme. The following sections would discuss each parts of the control scheme in details. The chapter

would also analyze the result and the performance of the motion control scheme in the simulation.

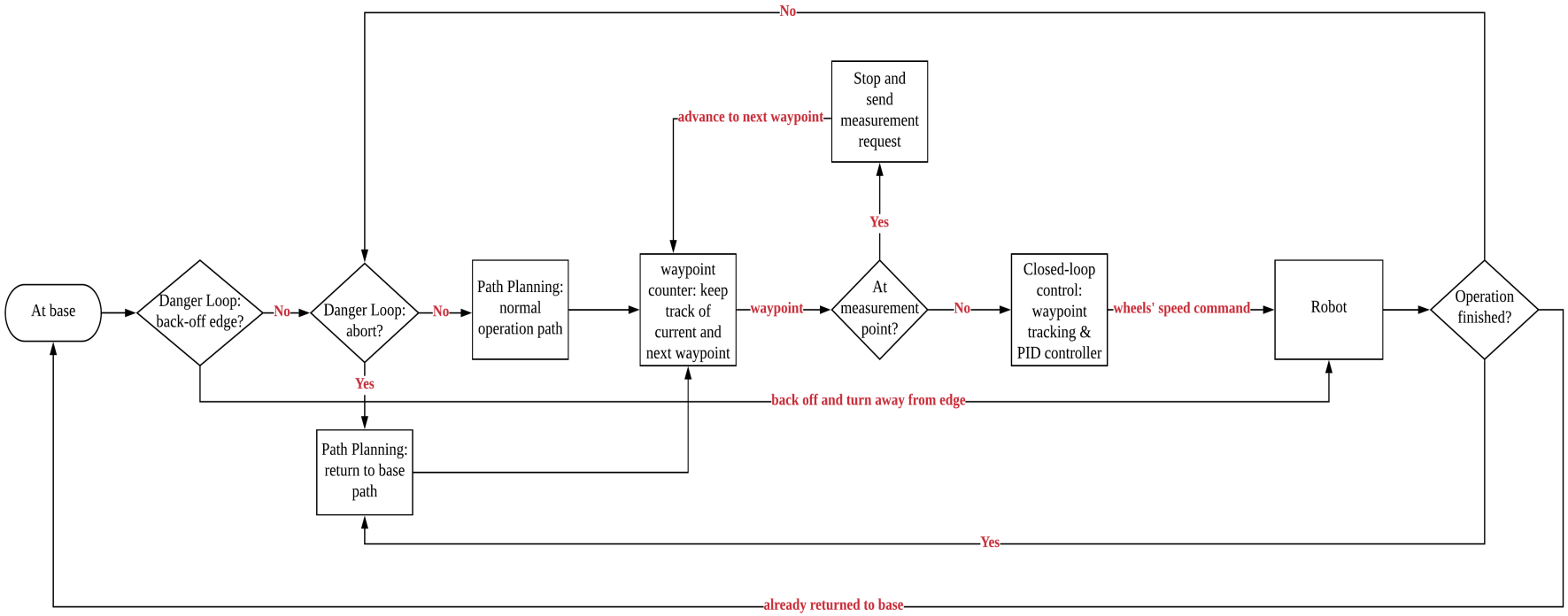


Figure 3.1: The overview of the motion control scheme.

## 3.2 Path Planning

When moving on the mirror, the robot always tracks a waypoint from either a pre-defined path during the normal operation, or a return-to-base path when the operation is aborted. One of the two paths are fed into the path planner, which keep track the current and the next waypoint until all the waypoints in the path are exhausted. The nine waypoints on each panel are numbered according to figure 2.2, while a panel's location on the mirror is described by its row and column position on the mirror. The traversing order of the waypoints on different panels are different to prevent the robot from making excessive turning, which would inject more disturbances in the orientation estimate. More precisely, there are four types of traversing order depending on the row and column that a panel is located on the mirror, as shown in figure 3.2. For a panel with its location described by $(r_i, c_j)$, where $1 \leq i \leq 9, 1 \leq j \leq 9,\ \ i, j \subseteq Z$, the traversing order is described by table 3.1.

| remainder of i,j divided by 2 | mod(j,2) == 1 | mod(j,2) == 0 |
|---|---|---|
| mod(i,2) == 1 | order 1 | order 4 |
| mod(i,2) == 0 | order 2 | order 3 |

Table 3.1: Traversing order of each panel depending on its row and column location in the mirror.

## 3.3 Closed-Loop Control

As shown in the overview of control scheme in figure 3.1, close-loop control is used to track the waypoints in the planned path defined above. The reason to use closed-loop control instead of open-loop control is that the robot should

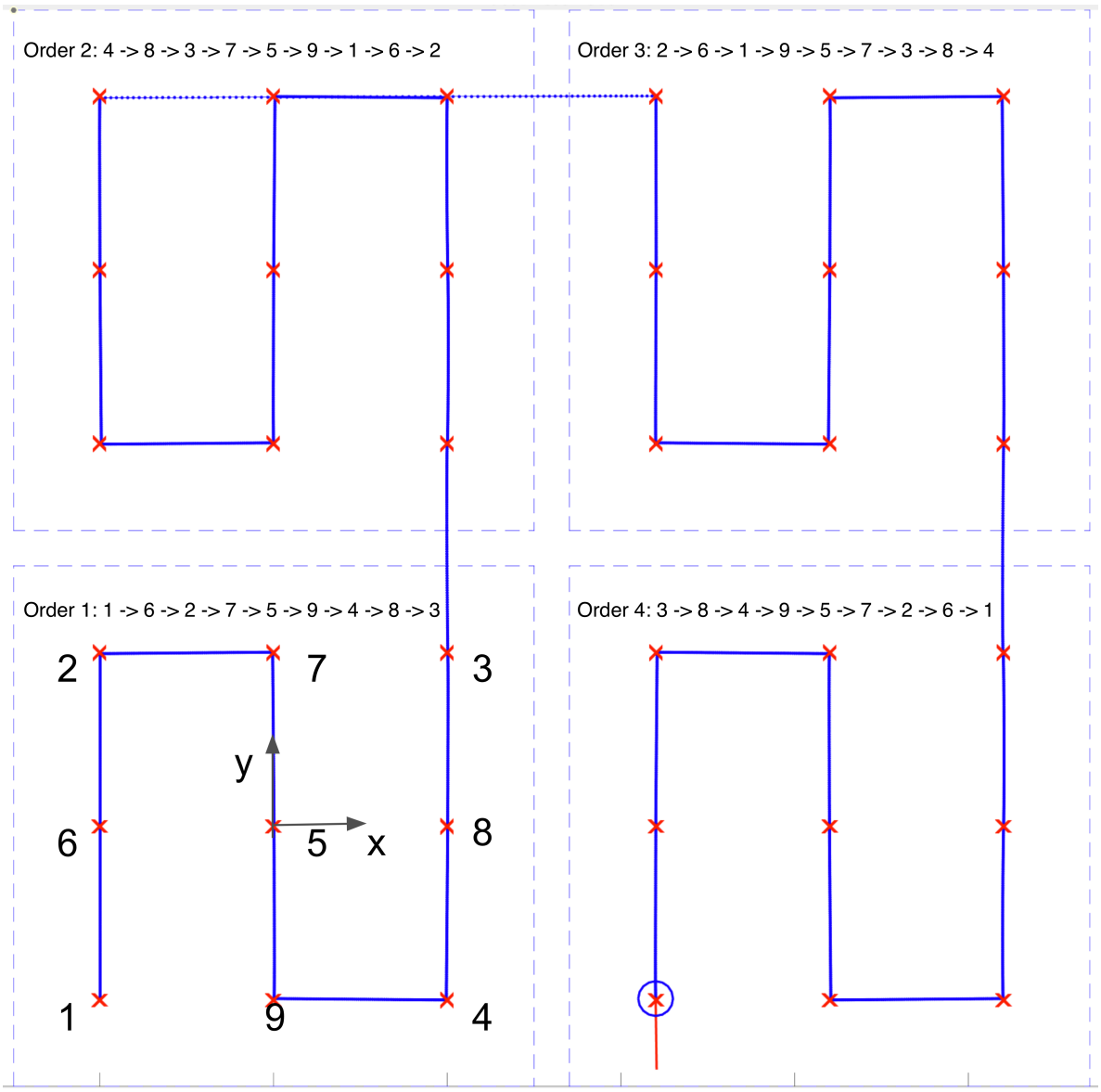


Figure 3.2: The four types of traversing order in simulation.

reach the measurement point with very high precision, while the disturbances during its operation, such as wheels' slip and the vibration of the fans, are not negligible.

More precisely, the states of the robot that need to be controlled are the forward velocity and the angular velocity in the panel frame. A good strategy is to let the robot first align its orientation with the waypoint as best as possible,

and then head to the target. This turn-and-move method is well suited for this project because the robot has limited turning radius, especially at the edge of the mirror to prevent the robot from driving off the mirror by making a large turn. Another benefit of the turn-and-move method is that it simplifies the control of the forward velocity and angular velocity, since the two velocities can be separately controlled.

The forward velocity and the angular velocity are separately controlled by two PID controllers, where the errors fed into the controllers are computed using the waypoint tracking logic. The control output is the left and right wheels' speeds which are then sent to the on-board computer and passed as the reference input to another low-level PID controller that computes the voltage inputs to the DC motors. The entire close-loop control strategy can be summarized in figure 3.3.

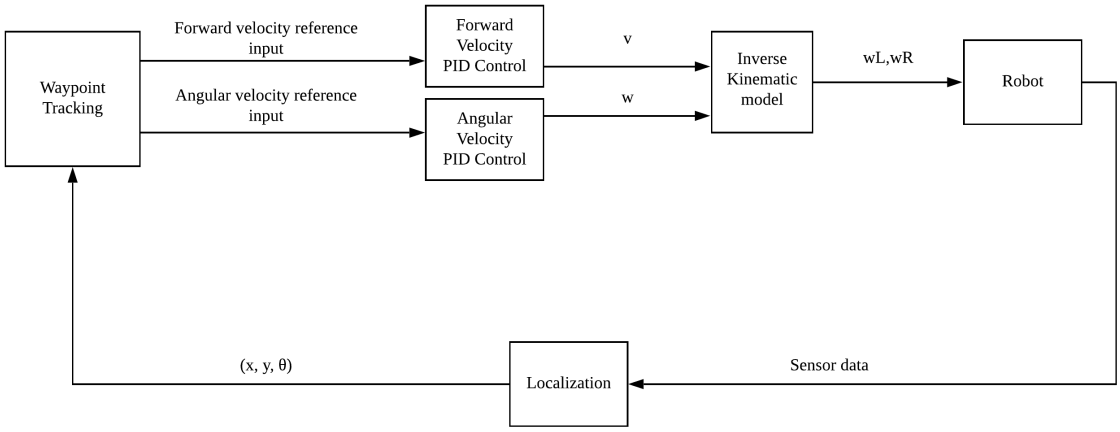


Figure 3.3: Flowchart of close-loop control strategy.

### 3.3.1 Waypoint Tracking

The robot's forward velocity and angular velocity are separately controlled by two PID controllers, and the PID controller needs an error as input, which is the differences between a set-point and feedback data. The purpose of waypoint tracking logic is to compute the errors to feed into the two PID controllers. For the forward velocity, the error is computed based on the distance between the current waypoint position and the robot's current position in panel frame as shown in equation 3.1,

$$e_v = \sqrt{(x_{waypoint} - x_p)^2 + (y_{waypoint} - y_p)^2} \tag{3.1}$$

and the result is the error that is fed into the PID controller to compute the forward velocity, as shown in the block diagram in figure 3.5. The range of the error $e_v$ is from 0 to $0.744m$, where $0.744m$ is the maximum distance the robot would travel from one waypoint to another when crossing two panels. For the angular velocity, the error is computed based on the angular displacement between the robot's current orientation in the panel frame ($\theta_p$) and the angle between the robot and the waypoint ($\beta$) as shown in equation 3.2,

$$\begin{aligned} e_w &= \beta - \theta_p \\ \beta &= arctan(x_{waypoint} - x_p, y_{waypoint} - y_p) \end{aligned} \tag{3.2}$$

The above equation can also be visualized in figure 3.4. Similarly, the result is fed to a separate PID controller to control the robot's angular velocity. The range of the error is from 0 to 3.14 *rad*, since the maximum turn the robot would make is 180°.

The forward velocity and the angular velocity are then used to compute the robot's left and right wheels' speeds with the inverse kinematic model given in equation 3.3.

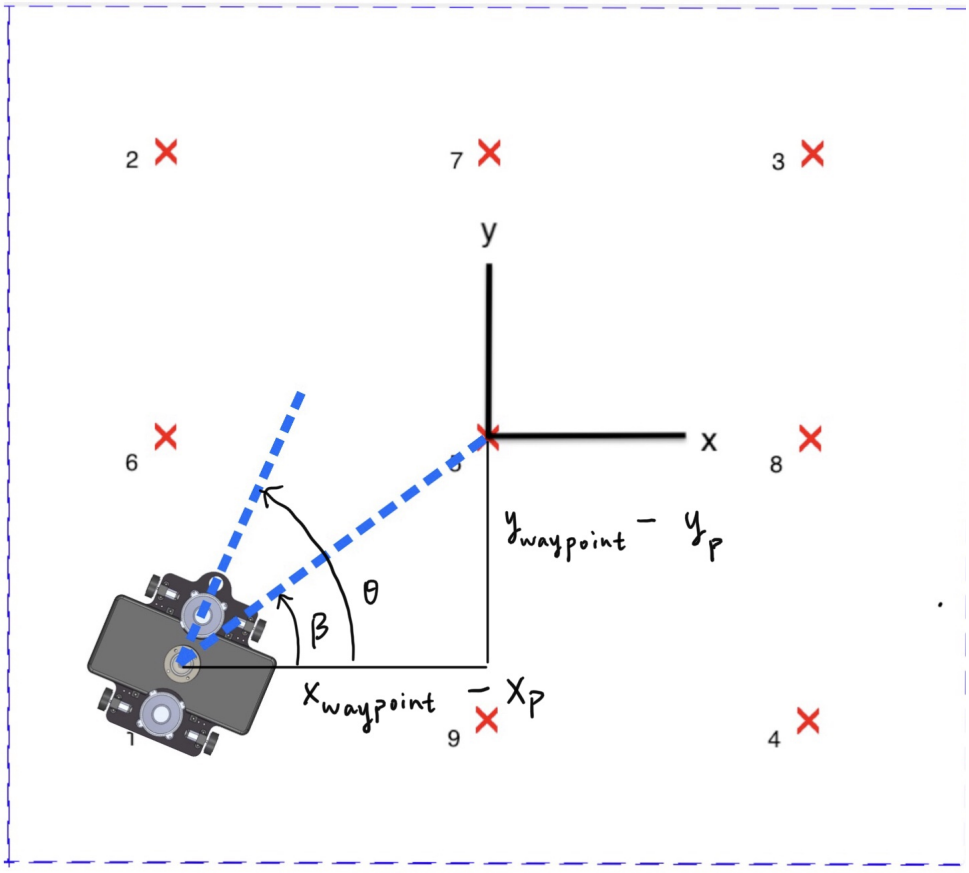


Figure 3.4: Angular displacement between the robot's orientation $\theta$ and the angle between the robot and waypoint $\beta$.

$$\begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix} = \begin{bmatrix} 1/r & L/2r \\ 1/r & -L/2r \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.3}$$

where:

r = the wheel's radius, $[m]$

L = the width of the wheel base, $[m]$

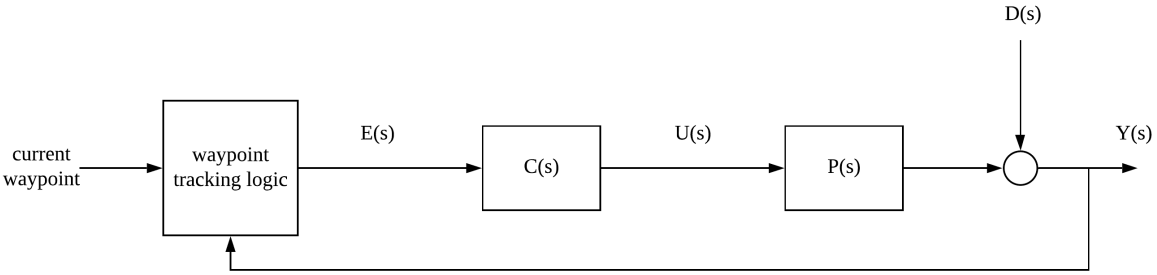


Figure 3.5: The block diagram of the PID control.

### 3.3.2 PID Controller

A PID controller $C(s)$ can be defined by its three control gains, as shown by the transfer function in equation 3.4. To analyse and tune the control gains, a plant model, $P(S)$, is needed to describe the system that needs to be controlled. Since the inputs to the robot are velocity commands and the observed outputs are the robot's position and orientation, the relationship between the input and output can be modelled as a first order system, which is given in equation 3.4. By modelling the system as first order system, the delays such as the communication delay and the delay from actuator response can be lumped into the time constant.

$$\begin{aligned} P(s) &= \frac{1}{Ts+1} \\ C(s) &= K_p + \frac{K_i}{s} + K_d s \end{aligned} \tag{3.4}$$

In equation 3.4, $K_p, K_i, K_d$ are the gains of the PID controller, and $T$ is the time constant of the first order system. There are several factors that should be considered when tuning the PID gains. First, the controller should satisfy the precision requirement, which implies that the steady-state error should be zero; ideally overshoot should also be zero so that the robot does not move back and

forth around the target (i.e the waypoint). Second, the controller should provide good rejection to the high frequency disturbance from fans' vibration. Third, the control effort should not exceed the constraints of the DC motor. Since the ranges of errors for forward velocity control and angular velocity control are different as described in the previous section, the control gains should be tuned accordingly for the two PID controllers.

Figure 3.6 shows the bode plot of the open-loop system for forward velocity, with $K_p = 0.5, K_i = 0.5, K_d = 0$. The infinite gain margin and 120° phase margin show that the system is stable. The infinite gain at low frequency shows that there is no steady-state error and could provide good disturbance rejection, while the downward slope at high frequency provide good noise rejection. Moreover, the large phase margin would account for the potential communication delay and generate only very small overshoot. The root locus plot of the closed-loop system in figure 3.7 also verifies that the system is stable since there aren't poles on the right-hand plane, and the pole at infinity shows that the system would be stable with arbitrarily large gains given that the gains are practically achievable by the actuators. Figure 3.8 shows the bode plot of the open-loop system for angular velocity, with $K_p = 0.1, K_i = 0.05, K_d = 0.01$. Similarly, the gain margin and phase margin shows that the system is stable, and there is good disturbance rejection and good noise rejection and no steady-state error. The root locus plot in figure 3.9 also verifies that the system is stable and has zero overshoot.

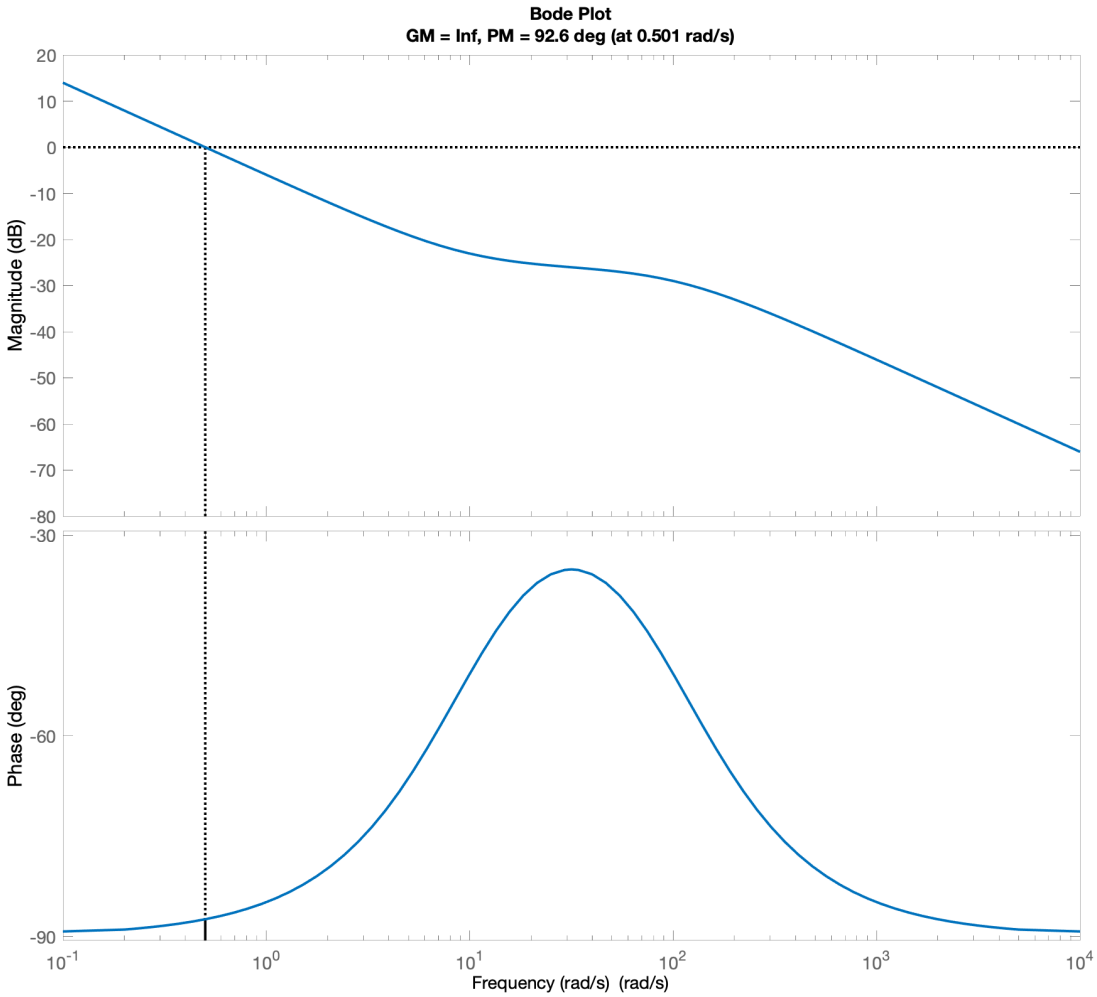


Figure 3.6: The bode plot of forward velocity open-loop system.

## 3.4 Danger Loop

There are many situations that the robot could encounter when operating on the mirror, such as driving near the edge of the mirror due to malfunctioning sensors, and various emergency events. Therefore, it's necessary to look out for those possible events based on the sensor data returned from the on-board sensors and the emergency signals sent by the observatory, so that the main loop process could send out the correct speed commands to the robot. The danger loop is designed for that purpose; it is a separate process that gathers all the

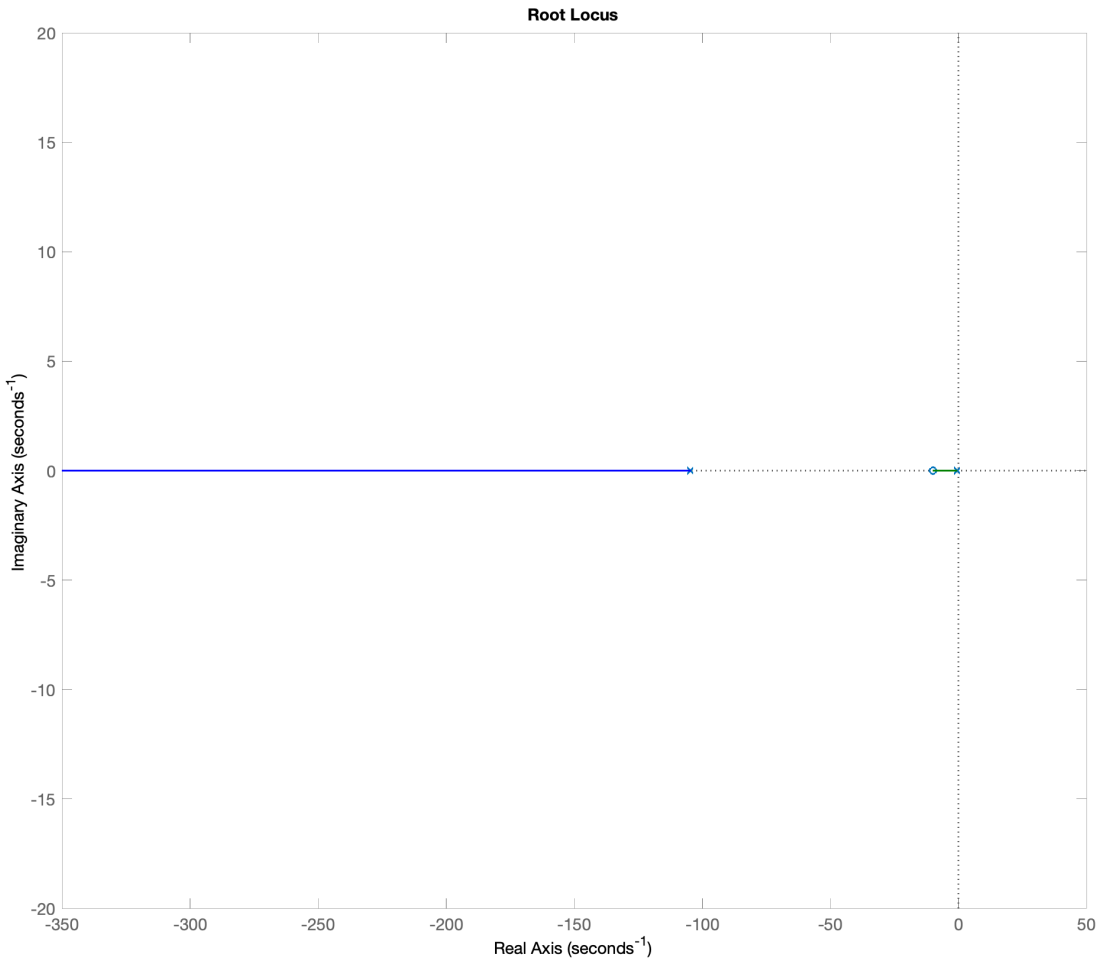


Figure 3.7: The root locus of forward velocity closed-loop system.

sensor data and the emergency signals to determine what situation the robot is about to encounter and inform the main loop process by sending the corresponding event code, which would instruct the robot with the corresponding control action. The reason that the danger loop runs on an independent process is that it could potentially check for the danger situations at a higher loop frequency than the main loop process, which could be extended with computationally heavy programs (such as different localization algorithms) in the future. Table 3.2 shows the events and its corresponding event code, and also the actions that the main loop implements. Details about how each event is determined are given in the following subsections.

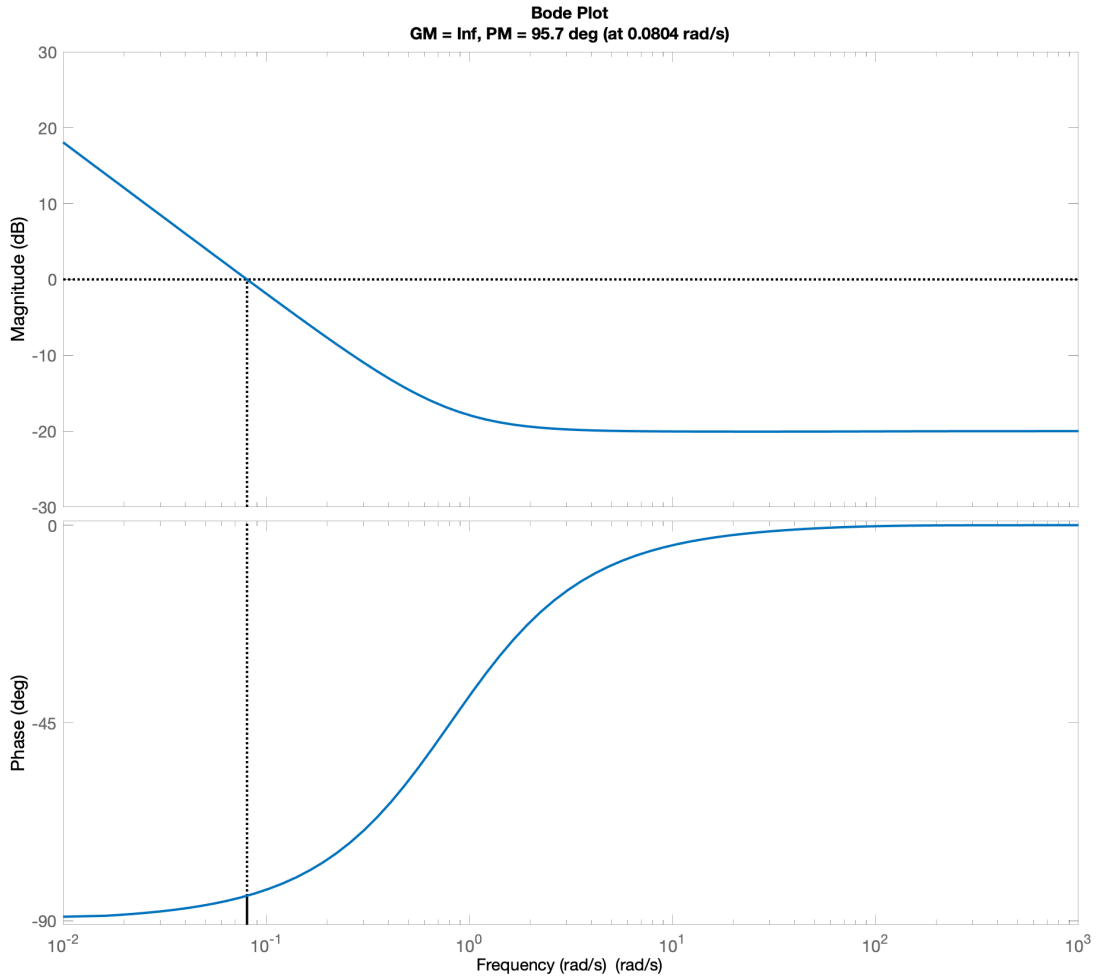


Figure 3.8: The bode plot of angular velocity open-loop system.

| Event code | Event | Actions |
|---|---|---|
| 0 | normal operation | tracking current waypoint |
| 1 | near mirror edge | backing off and turning away from the edge |
| 2 | operation aborted | returning to base |

Table 3.2: Event code with corresponding event and actions.

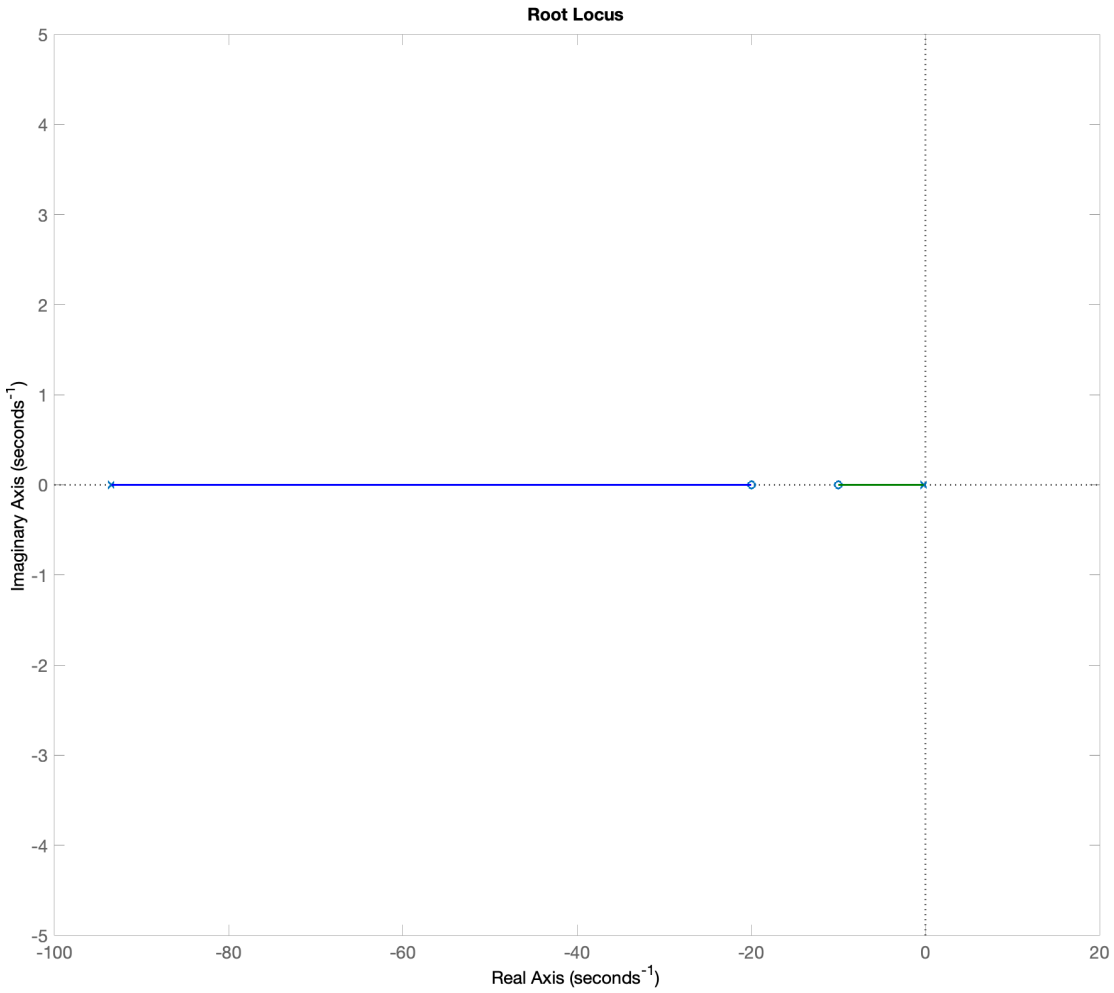


Figure 3.9: The root locus of angular velocity closed-loop system.

### 3.4.1 Near the mirror's edge

In the case of sensor malfunctioning, the localization algorithm could produce an estimate with large error, which could potentially cause the robot to drive toward the mirror's edge. To prevent the robot from falling off the edge, the danger loop uses the four edge sensors positioned around the robot, as shown in figure 2.6 in the previous chapter, to determine if the robot is near the edge. Essentially, the analog signal from an edge sensor needs to be compared with a threshold value to determine if the edge sensor is on/off the aluminum panel,

which is already being processed at the on-board computer. Therefore, the input to the edge detection part of the danger loop is a list of 0/1s indicating on/off the panel.

However, the edge sensor also would be temporarily off the panel when the robot crosses from one panel to another, since there is a 1 *cm* gap in-between each panel. Therefore, the edge detection algorithm should distinguish whether the robot is crossing the panels or is actually near the mirror's edge. The edge detection algorithm uses the fact that the edge sensors would be off the panel for a very short amount of time when the robot crosses the panels, since the robot travels at an average speed of 0.1 m/s. Therefore, the algorithm can compare the amount of time the edge sensors stay off-panel with an predefined threshold time; if the edge sensors are off the panel for more than the threshold time, then the algorithm would determine that the robot is near the mirror's edge and send the corresponding event code. A good estimate of the threshold time can be derived by dividing the 1 *cm* gap distance by 0.1 $m/s$, which would equal to 0.1 *second*.

Once the event code is sent to the main loop process, the main loop would instruct the robot to back-off by sending a negative forward velocity command to the robot, and then turn away from the edge, before resuming the normal operation. This process can be visualized in the overview flowchart in figure 3.1.

### 3.4.2 Operation aborted

When encountering emergency situations, such as an earthquake and fire alarm, the operation is aborted. The danger loop would listen on the designated channel for the emergency signal from the observatory. Once the emergency signal is picked up, the danger loop sends the corresponding event code to the main loop process, which would instruct the robot to return to base. The main loop process would first generate a path from the currently tracking waypoint to the base waypoint. The ”abort path” also consists of a series of waypoints and would substitute the normal operation path in the path planner. The main loop process would then compute the speed commands based on the new waypoints until all waypoints in the path planner are exhausted. This process can be visualized in the overview flowchart in figure 3.1.

## 3.5 Simulation and Result

As described in the previous chapter, the simulation is carried out with two levels of sensor noise. For each simulation run, the performance of the motion control scheme is measured by the L2 error between the robot's position when it stops at a waypoint and the actual waypoint position, as given by equation 3.1, and then the percentage of the errors that fall below 0.01 $m$ is calculated, since the project's requirement states that the robot should reach within one centimeter of the measurement points. For each of the two noise levels, the simulation is carried out 5 times and the errors of reaching 180 waypoints are recorded. For the normal noise level, figure 3.10 shows that 88.89% of the waypoints are reached within the 1 $cm$ error limit, with mean error equals to 0.0064 $m$ and

largest error equals to 0.0149 *m*. For the large noise level, figure 3.11 shows that 83.33% of the waypoints are reached within the 1 *cm* error limit, with mean error equals to 0.0071 *m* and largest error equals to 0.0429 *m*.

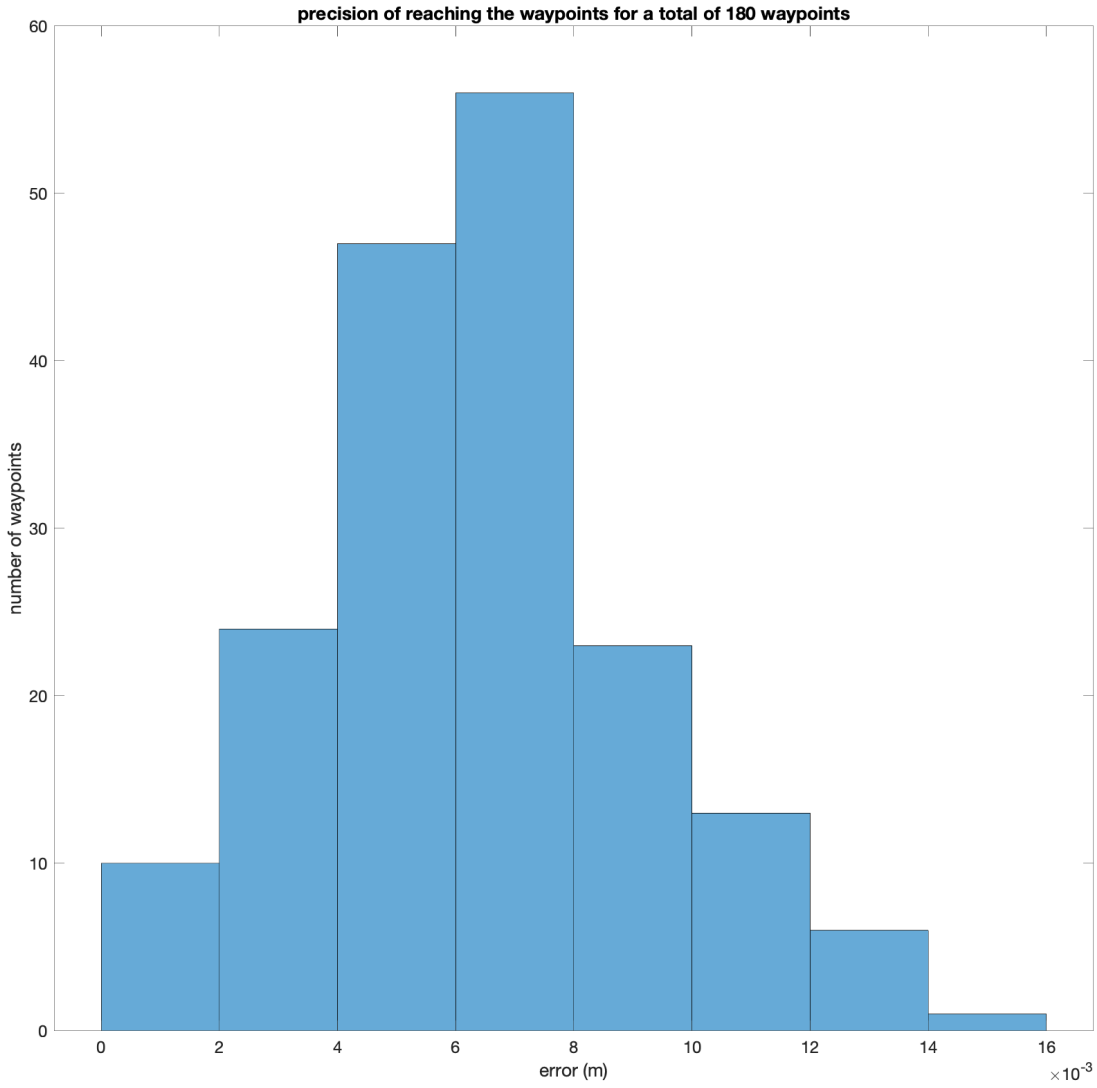


Figure 3.10: The errors of reaching the waypoints in five simulation runs with normal noise level. A total number of 180 data is recorded.

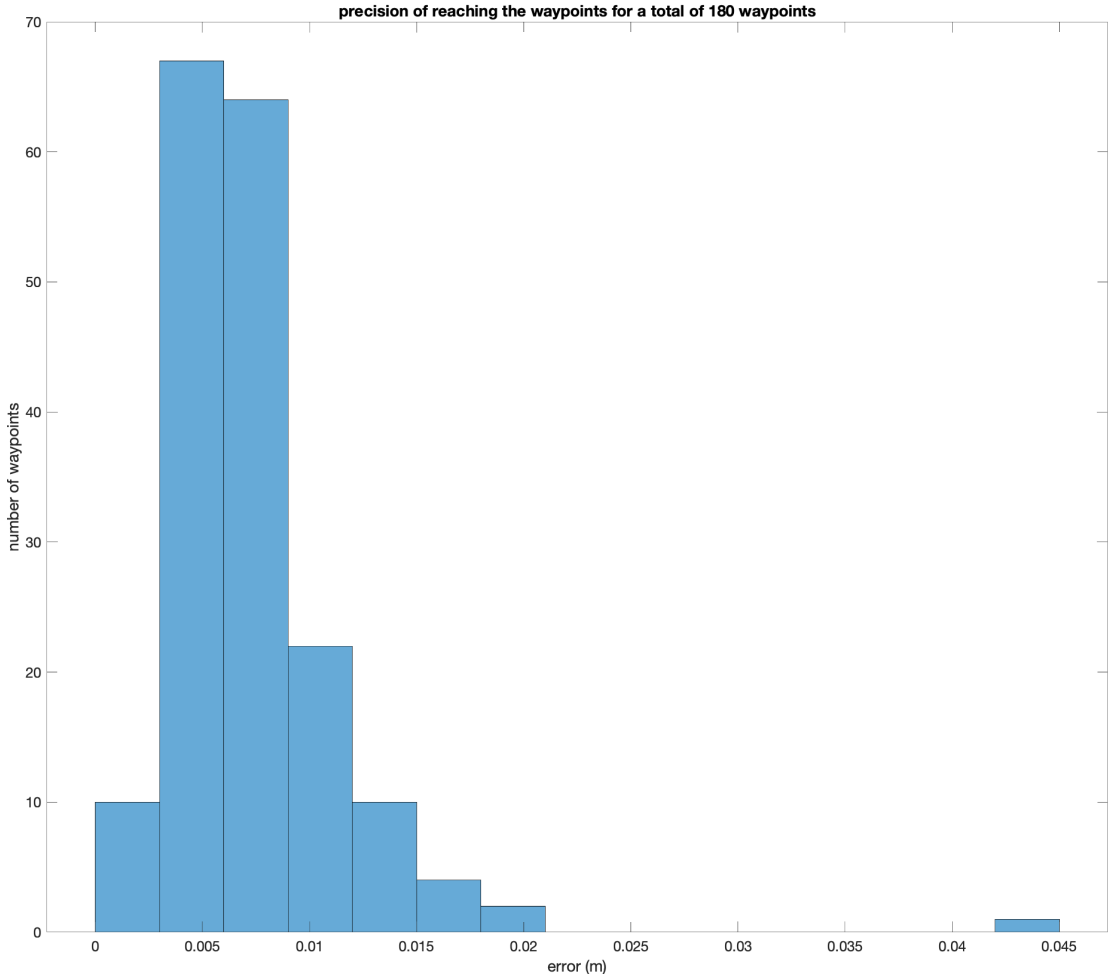


Figure 3.11: The errors of reaching the waypoints in five simulation runs with twice as much noise level. A total number of 180 data is recorded.

## 3.6 Summary

This chapter introduces the motion control scheme, including the closed-loop control that computes the speed commands sent to the robot when tracking a waypoint, and the danger-loop process that informs the main loop process of different situations, so that the main loop process can choose the correct action. The simulation results demonstrate the effectiveness of the motion control scheme, that nearly 90% of the waypoints are reached within the 1 *cm* the precision requirement.

# CHAPTER 4

# CONCLUSION AND FUTURE WORK

This thesis addresses the some of functional requirements of the CCTA-p project through the designs of a localization algorithm and a motion control scheme. For the design of the localization algorithm, the thesis presents a novel approach by fusing inertial measurement unit and encoder data with the Etalon external measurement data through Kalman Filter, which increases the localization precision to the degree of centimeter, a level of precision that is difficult to satisfy for many indoor localization problems. The thesis also presents the design of a motion control scheme, which handles different situations that the robot would encounter when operating on the mirrors and computes the correct speed commands using closed-loop control with waypoint-tracking logic and PID control. The thesis then demonstrates the effectiveness of the designs through the simulation software, and analyzes the performances and errors of the designs. In general, the designs have good performances and are able to satisfy the project's functional requirements.

Future work could improve the tracking accuracy further using different localization algorithms such as Sigma-point Filter. Future work could improve the overall efficiency of the operation by employing a better motion control strategy, providing that the robot has more powerful motors to realize the control actions. Future work could also study a more accurate dynamic model of the robot that also models other factors such as the frictions between the aluminum surface and the tyre, and the vibration causing by the fan, which could be employed in the prediction step of the Kalman Filter, or used as the plant model in the control design. The simulation could be improved with better representation

of the robot, and the robot's behaviour could become more realistic by injecting the random disturbances into the "DataLoop". Finally, the implementations should be tested together with the actual hardware to study how well the algorithms perform in the real operation. The software implementations are based on Object-Oriented Design (OOD) such that the software are flexible to be modified with different algorithms and extended with additional functionalities to satisfy any changes in the project's requirement in the future.

APPENDIX A

# SOURCE CODE

The source code for the simulation software, which also includes the implementations of the localization algorithm and the motion control scheme is in the GitHub repository https://github.com/SIOSlab/CCATpRobot.

## BIBLIOGRAPHY


[1] Basic principles of inertial navigation, seminar on inertial navigation systems. Technical report, Tampere University of Technology, 2018.

[2] P. Bahl and V. N. Padmanabhan. Radar: an in-building rf-based user location and tracking system. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, volume 2, pages 775–784 vol.2, March 2000.

[3] Y. Bar-Shalom, X.R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. Wiley, 2004.

[4] Wan Mohd. Yaakob Wan Bejuri, Mohd Murtadha Mohamad, Maimunah Sapri, and Mohd Adly Rosly. Performance evaluation of mobile u-navigation based on GPS/WLAN hybridization. *CoRR*, abs/1210.3091, 2012.

[5] CCAT-prime. Ccat: About ccat, 2020.

[6] Zijie Chen, Hyunji Kim, and Alex Zhou. Mechanical design for ccat-p wall climbing robot. Master's thesis, Cornell University, 2019.

[7] Stephen Parshley. Ccatp robotic puck cu-mae. presentation, 2017.

[8] Stephen Parshley. Optical design update. Technical Report P-TSSS-MEM-0010-B, CCAT-p, 2017.

[9] Bosch Sensortec. *BNO055 Intelligent 9-axis absolute orientation sensor data sheet*. Bosch Sensortec.

[10] S. Thrun, W. Burgard, D. Fox, and R.C. Arkin. *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents series. MIT Press, 2005.

[11] Eric W Weisstein. $l2$-norm, 2021.

[12] Wikipedia. Indoor positioning system, 2020.

[13] M. A. Youssef, A. Agrawala, and A. Udaya Shankar. Wlan location determination via clustering and probability distributions. In *Proceedings of the*

*First IEEE International Conference on Pervasive Computing and Communications, 2003. (PerCom 2003).*, pages 143–150, 2003.

[14] Moustafa A Youssef, Ashok Agrawala, and A Udaya Shankar. Wlan location determination via clustering and probability distributions. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003.(PerCom 2003).*, pages 143–150. IEEE, 2003.