UTILIZING ORIENTATION ESTIMATION FROM TRILATERATED POSES OVER TIME TO IMPROVE RO-EKF SLAM

by

David Grabowsky

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

Charlotte

2018

Approved by:

_____

Dr. James Conrad

_____

Dr. Andrew Willis

_____

Dr. Robert Cox

©2018
David Grabowsky
ALL RIGHTS RESERVED

ABSTRACT

DAVID GRABOWSKY. Utilizing orientation estimation from trilaterated poses over time to improve RO-EKF SLAM. (Under the direction of DR. JAMES CONRAD)

A variety of robotics applications involve placing a robotic agent in an unknown environment. The argent must then track its location while simultaneously building a map of the unknown environment. This is termed to Simultaneous Localization and Mapping (SLAM) problem. A subset of the SLAM problem entails environments where the only data available from the environment to use for localization and mapping is range information. This is termed the Range Only Simultaneous Localization and Mapping (RO-SLAM) problem. Algorithms that solve SLAM and RO-SLAM use landmarks in the environment to assist with localizing the robotic agent. SLAM algorithms have the advantage of being able to utilize range and orientation of an agent to a landmark to localize while pure RO-SLAM algorithms can only utilize the range. The goal of this work is to infer the orientation from the range only data and apply that inferred orientation to the RO-SLAM algorithm. In this work, a series of range measurements to a nonholonomic constantly moving robotic agent will be used to trilaterate the position of the agent. A series of the trilatered positions gathered over time will be used to estimate the robotic agents orientation. Two RO-SLAM algorithms will be run, the algorithms will be identical with the exception that one algorithm runs the orientation estimation method, and the other does not. The results of these algorithms will then be compared to determine if the orientation estimation method resulted in an increases in performance of the RO-SLAM algorithm.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. James Conrad for giving me the opportunity to contribute to the research being conducted within his lab. Dr. Conrad provided insight and direction that developed my passion for research and experimentation. Without the opportunities presented by Dr. Conrad I would not be in the position I am today.

I would also like to thank one of my committee members, Dr. Andrew Willis. Dr. Willis taught several of the classes which imparted information that proved to be fundamental to this research. In addition, Dr. Willis generously provided a simulation environment and foundation of code that greatly aided with the progress of this research. In addition, Dr. Willis was also willing to go out of his way to assist with problems/questions that were generated by this research. The value of his contributions can not be understated as without it the difficulty and time required to get the research to this points would have been doubled, if not more.

I would also like to thank my final committee member Dr. Robert Cox. Dr. Cox was one of the first true exemplars of academia that I encountered in my career at UNCC. His passion for his students and the work he conducts inspired me to set higher goals for myself and started me on my path in academia.

Finally, I would like to thank Dr. Sam Shue. Dr. Shue provided hours of hands on assistance and it was his research conducted under Dr. Conrad that led to the conception of the idea for this research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Over the last two decades autonomous mobile robotics has grown into a not only a sizable research field but also prolific commercial and industrial market [6]. Companies continue to invest in autonomous mobile robotics as a way of reducing costs and improving performance. Of paramount importance to autonomous mobile robotics is the ability to understand and move throughout the environment surrounding it. This involves the tasks of localization and mapping. Localization refers to the robot recognizing its position in relation to objects in the environment around it. Simultaneously, the robot is also expected to build a map of its environment as it moves throughout it. This is termed the Simultaneous Localization and Mapping Problem (SLAM). The problem is refereed to as an "chicken or the egg problem" [7]. The reason for this being that in order for the robot to create a map it must understand its location in an environment, in order to track its location it must be able to examine a map to determine changes of location in relation to the environment. A multitude of algorithms have been presented as solutions to this problem [8]. The plethora of solutions surrounding the general SLAM problem has led to research focusing on improving the performance of existing methods.

Most SLAM algorithms can be broken up into a series of prediction and measurement steps. Robot motion control information, such as translation and rotational velocities, is used to predict how the robot will move based on its previous pose. Measurements from landmarks are used to estimate the current pose and correct the pose estimation from the prediction. Landmarks are a factor of critical important for SLAM algorithms. Landmarks refer to objects in an environment which can be observed and distinguished such as planar surfaces or colored balls. These landmarks

form the basis for a the map of the environment and are an important factor in the robot localization process. A variety of sensors, such as cameras or laser range finders, are used to distinguish and acquire data from landmarks. A subset of these sensors are known as range only (RO) sensors, meaning that the only data they receive from the landmarks are the ranges to these landmarks. This range only restriction results in the SLAM problem termed Range Only-Systematic Localization and Mapping (RO-SLAM). There are a number of advantages associated with the range only restriction, such as the decrease in sensor information that must be processed. For example, RGB cameras utilized as sensors for non RO-SLAM have a number of pixels associated with a image. Those pixels will need to be processed to determine which pixels in the image should be used as a landmark. With range only sensors the case is often that the data utilized to obtain the range reading, such as wireless signal strength, requires much less processing since the signal strength will contain less data points than the data points present in an image. In addition, non RO-SLAM methods that operate inside buildings will often utilize corners or planar surfaces as landmarks. An issue with this is that buildings have a such a high number of corners and planar surfaces that the memory required to store all landmarks becomes an issue. Devices that produce range only sensor readings are typically much less numerous than corners and planar surfaces, thus the memory required to store the location of all of the devices in an environment is much smaller.

Many environments utilize range only sensors to solve the localization problem, such as in underwater environments where line of sight is not reliable so ranging information derived from signal transmissions are used instead [9]. Methods that solely utilize RO data from landmarks are denoted as RO-SLAM methods. In some cases, metrics such as the angle of arrival of a signal [10] can be used to determine the angle to the origination of the signal, this work will specifically focus on methods where only the ranging information is available. The reason for this is that in order

for metrics such as angle of arrival to be tracked, costly and more advanced hardware must be used. The restriction of RO data from landmarks means that these landmarks must often be specifically manufactured and placed throughout an environment. The exact conditions by which the landmarks are placed vary; in some cases the exact location of the landmark may be known ahead of the initial SLAM process, in others it might not.

Range only measurements do not natively provide information on the robots orientation to the object responsible for the range measurement. In environments where RO-SLAM must be solved, the only direct orientation information comes from sources such as the integration of wheel encoder information. Wheel encoders are used to track to rotation of a motor. However wheel encoders are known to slip which leads to a build up of error in the integration. Slippage occurs when a parts of a rotation occur but a change in position does not. This is a common problem in rough terrains such as uneven or sandy environments [11]. The range only measurements are utilized by RO-SLAM algorithms to account for this, however if the range only measurements contain error that is erratic then this could lead to a degradation in performance of the algorithm. This is a common occurrence in indoor environments where range information determined from wireless transmission is affected by factors such as multipath fading [12]. However, if a method were to utilize the range only measurements to estimate the robots orientation at a finite point in time and apply that to the RO-SLAM algorithm as well, then this could be help to accommodate for the introduction of the noise and lead to an improvement in the performance of the RO-SLAM method.

1.1 Objective of this Work

The lack of native orientation measurements in RO-SLAM algorithms means that the estimates produced are solely based upon prediction from control information and updates from range only measurements. This limitation results in RO-SLAM algo-

rithms lacking measurement orientation information that their SLAM counterparts are able to utilize, potentially leading to a decrease in performance of the RO-SLAM algorithm. The objective of this work is to increase the performance of an RO-SLAM algorithm through estimating the orientation of a robot from a collection or range measurements gathered over time. Then applying that orientation estimation as another measurement to the RO-SLAM algorithm without violating the restriction of landmarks providing range only data.

1.2 Contribution

This paper presents a novel implementation of the Range Only Extended Kalman Filter (RO-EKF) SLAM algorithm which incorporates an orientation measurement of the robot estimated from a history of trilatered robot positions. The novel implementation is termed modified RO-EKF. As the robot traverses the environment trilateration is used to estimate the position of the robot from the range measurements provided by landmarks. A history of these positions is retained over time. After the history of positions has passed a threshold, a least squares approximation with a second degree polynomial is applied to estimate the trajectory of the robot from the most recent samples in the approximation. The orientation measurement will then be used to update the orientation that is estimated by the modified RO-EKF SLAM algorithm. The end result of this research will show that the modified RO-EKF SLAM algorithm outperforms the standard RO-EKF SLAM algorithm under the constraints of range only information and noisy range data produced by the natural multipath of an indoor environment.

1.3 Organization

This thesis is organized into five chapters. Chapter 1 provides the motivation for this work as well as a brief introduction of the topics relevant to the work. Chapter 2 covers background information concerning SLAM, Extended Kalman Filtering, range

only Extended Kalman Filtering, linear regression, and trilateration. Chapter 3 covers the equipment, environment, and algorithms used to conduct this work. Specifically it will lay out the algorithm used to generate the orientation measurement estimations and the modified RO-SLAM method algorithm. Chapter 4 covers the results of this work. Finally Chapter 5 focuses on drawing conclusions from the results and defining future work.

CHAPTER 2: Background

2.1 SLAM

The main purpose of SLAM is broken into two parts. The first is to provide some device, such as a robot, with a map of the environment. The second is to simultaneously provide the device its position within the map. This information is used by a variety of applications such as obstacle avoidance and path planning algorithms. Thus, the information is a cornerstone for any application seeking to achieve autonomy as location and mapping information are basic dependencies for such applications.

What makes SLAM so attractive for commercial and industrial applications is that a minimum amount of prior work is necessary to allow the device to function in most environments. Currently, many environments are specifically modified to allow a device to localize, such as by using magnetic strips on the ground for a robot to follow visual color coded signs to be pre-identified as landmarks, or an already existing map. SLAM removes the need to modify the environment since the mapping and localization will be done without any prior data about the environment.

The lack of prior information means that SLAM has a problem initializing. How can the devices position be estimated if the device is not certain of its location on the map? How can the map be generated if the device does not know where it is? Thus a cycle is created as the map and device position are dependent on one another. The purpose of solutions to SLAM is to create an optimum estimation of the position and map given measurement inputs.

Solutions to simultaneous localization and mapping have seen continued development and refinement [8]. Many of these solutions can be differentiated between two

factors: the sensors, and the estimation method. Commonly implemented sensors, such as LIDAR [13], are used to identify landmarks and determine obstacles. Cameras have also been used for landmark identification and map construction [14] in environments where more processing power is available. The filtering method is how the landmark data, uncertainty, the map, and the robot motion are combined into an estimation of reality. Filtering methods include Extended Kalman Filter SLAM, Graph SLAM, Particle Filter SLAM, and Fast SLAM, to name a few [7]. The filtering methods each take a unique approach to handling SLAM and have their own strengths and weaknesses. Research has been conducted to create an objective way to benchmark and compare these methods despite their differences in operation [15].

2.1.1 EKF SLAM

As the name suggests, the Extended Kalman Filter SLAM refers to the utilization of the Extended Kalman Filter to solve the SLAM problem. It is one of the most common implementations of SLAM and is used in a variety of robust situations. It is capable of being adapted for multiple types of landmarks and offers quick computation speed as only the prior state must be stored. At its core EKF is a recursive Bayesian filter that which models with additive Gaussian noise. To fully understand the EKF SLAM algorithm the following sections will proceed to introduce the Kalman Filter, the Extended Kalman Filter, and finally the Extended Kalman Filter SLAM algorithm used for range only applications.

2.1.1.1 Kalman Filter

The Kalman Filter is a type/category of Bayesian Filter and is a technique for filtering and predicting linear Gaussian Systems [7]. The primary components of the Kalman Filter are the state vector, which represents elements such as the position of an agent and landmarks and can be seen in more detail in Equation 2.10, and the covariance matrix, which represents how strongly correlated variables in the state

vector are and can be seen in more detail in Equation 2.11. The process Kalman Filter can be broken into two parts: The prediction and the measurement phase. The prediction phase updates a state vector, which is composed of elements such as the position and orientation, and covariance matrix values, according to a specific state transition model. Thus predicting how a variable will change over a time period as well as how the uncertainty in the variable will change. The measurement phase then uses a sensor measurement to correct the values from the prediction phase, while also accounting for uncertainties in the measurement. These two phases are then run continuously in a cycle as represented by Figure 2.1.

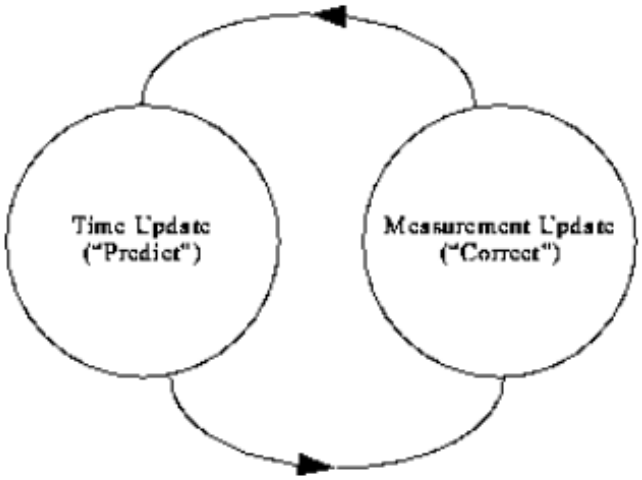

Figure 2.1: The cycle of the Kalman filter. The predict phase projects the current estimate of the state vector forward in time. The measurement phase updates the projected state vector according to a measurement made at that point in time [1].

2.1.1.2 The Prediction Phase

The prediction phase is represented by two equations. The first denotes how the state of the filter, represented by state vector variable x, is updated. The second denotes how the covariance matrix, represented by variable P, is updated.

$$x_{k|k-1} = F_k x_{k-1|k-1} + B_k u_k \tag{2.1}$$

Where $x_{k|k-1}$ is the estimate of the state variables at time step k, given the previous estimate at $k-1$, F_k is the state transition matrix at time step k, $x_{k-1|k-1}$ is the state variable last time step, B_k is the control input model, and u_k is the control input.

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \tag{2.2}$$

The $P_{k|k-1}$ is the covariance matrix at step k given the previous estimate at $k-1$, $P_{k-1|k-1}$ is the covariance matrix of the previous iteration, and Q_k is the covariance matrix of the process noise, which represents how much noise is generated in the transition between states [3].

2.1.1.3 Measurement Phase

The measurement phase seeks to correct error in the prediction utilizing a sensor measurement. To begin the error between the estimation and state is calculated.

$$y_k = z_k - H_k x_{k|k-1} \tag{2.3}$$

Where y_k is the error vector, z_k is the measurement vector, and H_k is the observation model, which maps the state variable vector into the space of the measurement vector.

Next, the innovation covariance matrix, S_k, is calculated. This relates the covariance of the state to the measurement vector, integrating the noise from the measurement.

$$S_k = H_k P_{k|k-1} H_k^T + R_k \tag{2.4}$$

Where R_k is the covariance matrix of the noise of the measurement vector, z_k .

Then the Kalman gain, which specifies the degree to which the measurement is incorporated into the new state, is calculated by multiplying the current state covariance matrix, the observation model, and the innovation matrix:

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \tag{2.5}$$

The Kalman gain is then used to correct the current state estimation, $x_{k|k}$:

$$x_{k|k} = x_{k|k-1} + K_k y_k \tag{2.6}$$

Finally, the covariance matrix, $P_{k|k}$, is then updated, by scaling the covariance values associated with the observation model by the value determined by the Kalman gain through the following equation:

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{2.7}$$

Where I is the identity matrix. The filter processes multiple iterations during which the state variable x is estimated via the state transition model and measurement. The covariance matrix is also updated to reflect the variance, or confidence, of the state.

The condensed representation of the Kalman Filter algorithm can be seen in algorithm 1.

2.1.1.4 Extended Kalman Filter

There are many modifications to the Kalman Filter that allow for non-linear systems to be accounted for, such as the Extended Kalman Filter. The Extended Kalman Filter seeks to linearize any non-linear system via first-order Taylor series expansion of the state transition and observation models [7]. This linearizion process leads to the addition of two new variables, the Jacobian of F and H. These Jacobians con-

Algorithm 1 The Kalman Filter $(x_{k-1|k-1}, P_{k-1|k-1}, u_k, z_k)$

KF_Prediction:
$x_{k|k-1} = F_k x_{k-1|k-1} + B_k u_k$
$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k$
KF_Measurement:
$y_k = z_k - H_k x_{k|k-1}$
$S_k = H_k P_{k|k-1} H_k^T + R_k$
$K_k = P_{k|k-1} H_k^T S_k^{-1}$
$x_{k|k} = x_{k|k-1} + K_k y_k$
$P_{k|k} = (I - K_k H_k) P_{k|k-1}$
return $x_{k|k}, P_{k|k}$

tain the first order derivative of a vector function of several variables and define the curvature of the linearized function models. This in turn describes the probability spread during the prediction and measurement phase. The Jacobians are defined as:

$$F_{k-1} = \frac{\partial f}{\partial x}|_{x_{k-1|k-1}, u_k} \tag{2.8}$$

$$H_k = \frac{\partial h}{\partial x}|_{x_{k-1|k-1}} \tag{2.9}$$

The state transition matrix and observation matrix are replaced by nonlinear functions which describe the state transitions and observation models, f and h, in the prediction and measurement phases [3]. Based on these functions the Jacobians are calculated and the values of F and H change depending on the time of linearization. The Extended Kalman Filter algorithm which incorporates the lineraization into the Kalman Filter can be seen in Algorithm 2

The ultimate goal of the algorithm is to accurately track the robot pose: (x,y,θ) and the location of the landmarks (x_1,y_1,...,x_n,y_n). During the prediction phase, the state vector, containing the robot pose (x,y,θ) is updated based on the state transition model and control input. The control input is determined from the odometry

Algorithm 2 The Extended Kalman Filter $(x_{k-1|k-1}, P_{k-1|k-1}, u_k, z_k)$

EKF_Prediction:
$x_{k|k-1} = f(x_{k-1|k-1}, u_k)$
$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k$
EKF_Measurement:
$x_{k|k-1} = h(x_{k-1|k-1}, z_k)$
$S_k = H_k P_{k|k-1} H_k^T + R_k$
$K_k = P_{k|k-1} H_k^T S_k^{-1}$
$x_{k|k} = x_{k|k-1} + K_k y_k$
$P_{k|k} = (I - K_k H_k) P_{k|k-1}$ **return** $x_{k|k}, P_{k|k}$

information attained from the wheel encoders. When a landmark measurement is received, the distance to that landmark is used to correct the state vector updated in the prediction phase. An important part of this process is the weight, or trust, that is applied during prediction and measurement phases. For example, if the odometry readings are known to be inaccurate, they will be trusted less, leading to higher uncertainty in pose as the prediction phase continues, and if the landmark measurements are known to be accurate, then they will be trusted more, meaning that the state vector resulting from the prediction steps will be updated to better reflect the measurement and the uncertainty in the state will decrease as a result. The covariance matrix reflects trust/uncertainty of and between the variables stored in the state vector. This process is illustrated in Figure 2.2 where a robot, represented by the solid line triangle, records the location of three landmarks, represented by stars. The robot then moves some distance, and based upon odometry the prediction step updates the state vector to where the robot is estimated to be, represented by the dashed line triangle. Next, new measurements to the landmarks are recorded and the update phase uses these measurements to correct the location of the robot. This figure does not graphically represent the uncertainty of the pose, but it serves to illustrate the general process of the algorithm.

The EKF SLAM algorithm has some additional processes that are required for estimating the system state. One of those process is data association. In this case, data

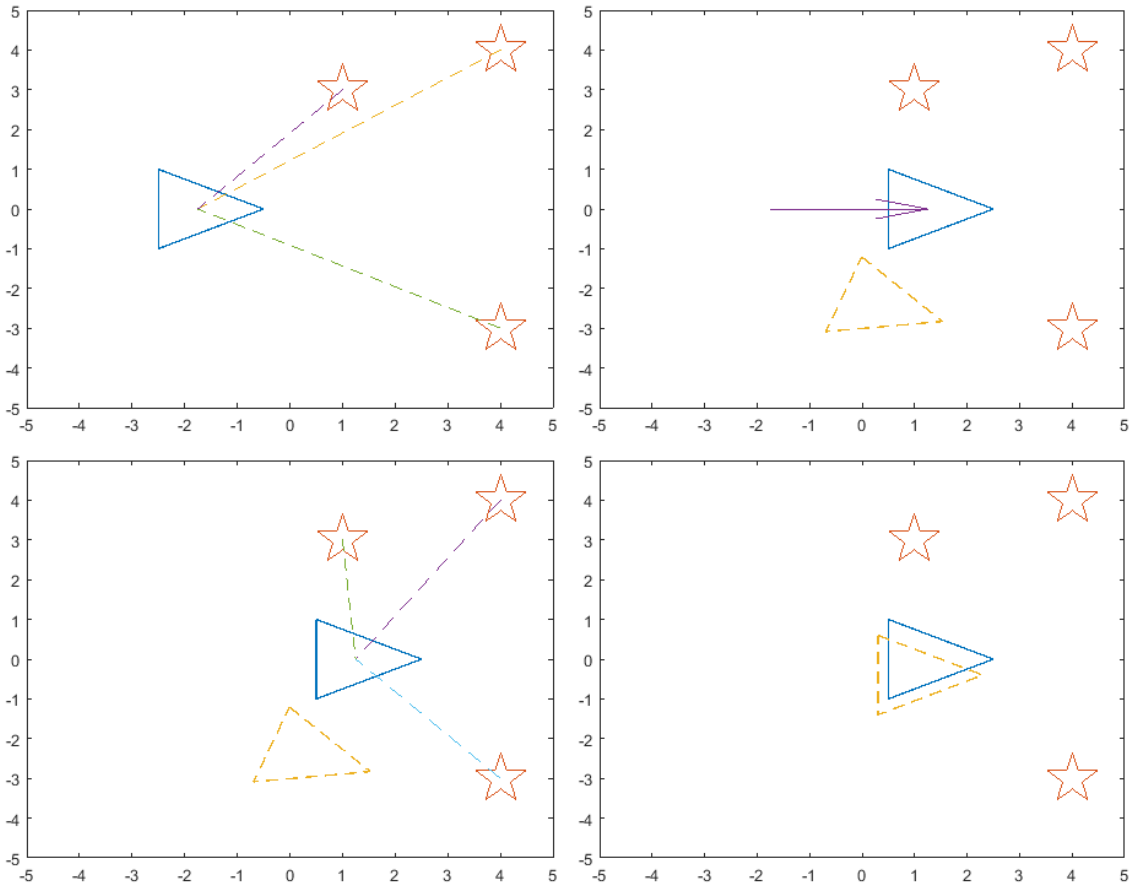

Figure 2.2: EKF Prediction and Measurement Phase based on [2]. Top left: Device (triangle) takes a measurements of distance and angle to three landmarks (stars). Top right: Physical device translates to the right (solid triangle), based on control information the tracked position of the device (dotted triangle) erroneously translates. Bottom left: Physical device measures distance and angle to three landmarks. Bottom right: the tracked position is updated to more closely reflect the physical position.

association refers to correctly identifying the landmark from which a measurement is gathered. For example, say a RGB camera is being used to measure the distance to landmarks, and the landmarks are represented by colored spherical balls. A red colored ball is observed and denoted as landmark 1. As the camera moves a second red colored ball is observed at position 2 and is denoted as landmark 2. Now the camera re-observes landmark 1, the EKF SLAM algorithm must correctly associate the current measurement with landmark 1 and not with landmark 2. The data association problem refers to the misidentification of measurements with landmarks, and can result from observations of similar landmarks. This is illustrated in Figure 2.3, where the blue circle represents the camera, the triangle represents the cameras cone of vision, and the red circles represent the landmarks. A measurement of L1 is made and associated with L1. Then a measurement of L2 is made an incorrectly associated with L1.

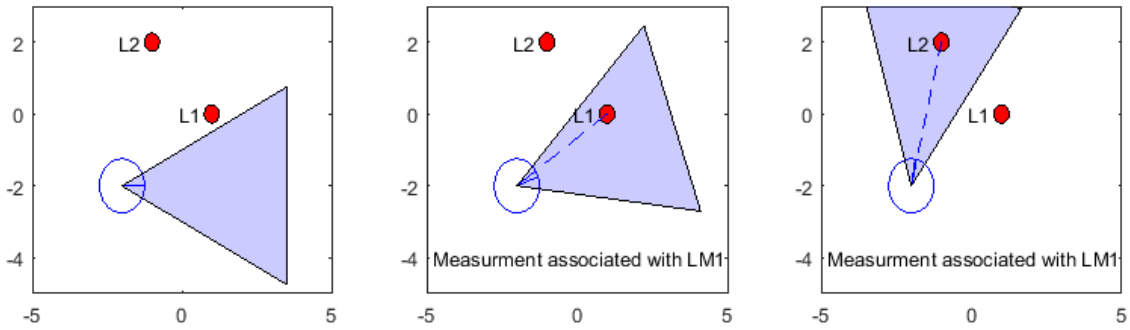

Figure 2.3: Illustrating the problem with data association. Left: Device has no landmarks in field of view. Middle: Device gets a measurement to L1 and correctly associates it with L1. Right: Device gets a measurement to L2 and incorrectly associates it with L2.

Misassociation can occur for a variety of reasons, such as the similarity of landmarks or the poor criteria used for landmark classification. Processes must often be developed that are specific for the type of landmark, such as discerning colored balls, or extremely robust for general environments, such as laser scans using corners as landmarks.

2.1.2 EKF SLAM Algorithm

The EKF SLAM algorithm utilized has several inputs, the first of which are odometry readings. These readings can come from a variety of sources such as wheel encoders, accelerometers, or visual based odomentry methods [16]. Odometry readings are not exclusive to EKF SLAM, in fact they are the fundamental sensory information used for almost all SLAM algorithms. However, these readings contain error as a result of factors such as noise and wheel slippage, meaning that over long periods of time the accumulation of odometry data drifts from the actual path of the object being tracked. Eventually this accumulated error will become large enough to make the odometry erroneous when estimating location. To account for this error, the EKF SLAM algorithm utilizes a second input, landmark measurements. These measurements are used to correct the error in tracking that occurs from the afore mentioned factors. In the following sections the details of the EKF SLAM algorithm will be covered, specifically relating to the vectors and matrices that compose the algorithm referenced to in Algorithm 2.

2.1.2.1 The State Vector

The state vector (x) contains the current state of the robot and the current state of tracked landmarks. The state vector is extremely important as it is used as the basis for any map to be generated. The vector is used for two-dimensional EKF SLAM in this case, which is composed of variables along an x and y plane. Specifically, the state vector contains the estimate of the robot's pose (x,y), the robot's orientation (θ), and location of each landmark (x,y).The state vector appears as seen in Equation 2.10.

$$x = \begin{bmatrix} x_r \\ y_r \\ \theta_r \\ x_1 \\ y_1 \\ \vdots \\ x_n \\ y_n \end{bmatrix} \tag{2.10}$$

Where x_r, x_y are the position of the robot, and θ_r is the robots orientation. x_i, x_y represent the i_{th} landmark and n represents the number of landmarks. The number of landmarks in the state vector is dynamic, meaning that new landmarks are continually appended to the state vector as they are discovered. This requires each matrix in the algorithm to expand with the state vector, which causes an exponential increase in the memory utilized by the algorithm.

2.1.2.2 The Covariance Matrix (P)

The covariance matrix stores the covariance of each state variable in relation with each other [3]. The matrix can be represented by as follows:

$$P = \begin{bmatrix} Var(x_r) & Cov(x_r, y_r) & Cov(x_r, \theta_r) & Cov(x_r, x_1) & Cov(x_r, y_1) & \dots & Cov(x_r, x_n) & Cov(x_r, y_n) \\ Cov(y_r, x_r) & Var(y_r) & Cov(y_r, \theta_r) & Cov(y_r, x_1) & Cov(y_r, y_1) & \dots & Cov(y_r, x_n) & Cov(y_r, y_n) \\ Cov(\theta_r, x_r) & Cov(\theta_r, y_r) & Var(\theta_r) & Cov(\theta_r, x_1) & Cov(\theta_r, y_1) & \dots & Cov(\theta_r, x_n) & Cov(\theta_r, y_n) \\ Cov(x_1, x_r) & Cov(x_1, y_r) & Cov(x_1, \theta_r) & Var(x_1) & Cov(x_1, y_1) & \dots & Cov(x_1, x_n) & Cov(x_1, y_n) \\ Cov(y_1, x_r) & Cov(y_1, y_r) & Cov(y_1, \theta_r) & Cov(y_1, x_1) & Var(y_1) & \dots & Cov(y_1, x_n) & Cov(y_1, y_n) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ Cov(x_n, x_r) & Cov(x_n, y_r) & Cov(x_n, \theta_r) & Cov(x_n, x_1) & Cov(x_n, y_1) & \dots & Var(x_n) & Cov(x_n, y_n) \\ Cov(y_n, x_r) & Cov(y_n, y_r) & Cov(y_n, \theta_r) & Cov(y_n, x_1) & Cov(y_n, y_1) & \dots & Cov(y_n, x_n) & Var(y_n) \end{bmatrix} \tag{2.11}$$

The diagonal values of the matrix store the variance of each state variable, such as the robot pose or landmark positions. The off diagonal values of the matrix store the covariance of a variables in relation to the horizontally and vertically located diagonal matrix values. As the algorithm iterates these values reflect the systems' confidence in the current estimation of the state. With a perfect system these values would be expected to decrease and converge.

To further discusses the P matrix, it is better to break it into subsections, as represented in figure 2.4.

Subsection A of the P matrix contains a $3x3$ covariance of the robots position and orientation. Subsection B contains a $2x2$ covariance of the first landmarks position. Subsection C contain the covariance of the final landmark. Subsection D contains the covariance of the robots pose in relation with the first landmark, subsection E is the transpose of this covariance. Subsection F contains the covariance of the first landmark in relation to the final landmark, subsection G is the transpose of this covariance.

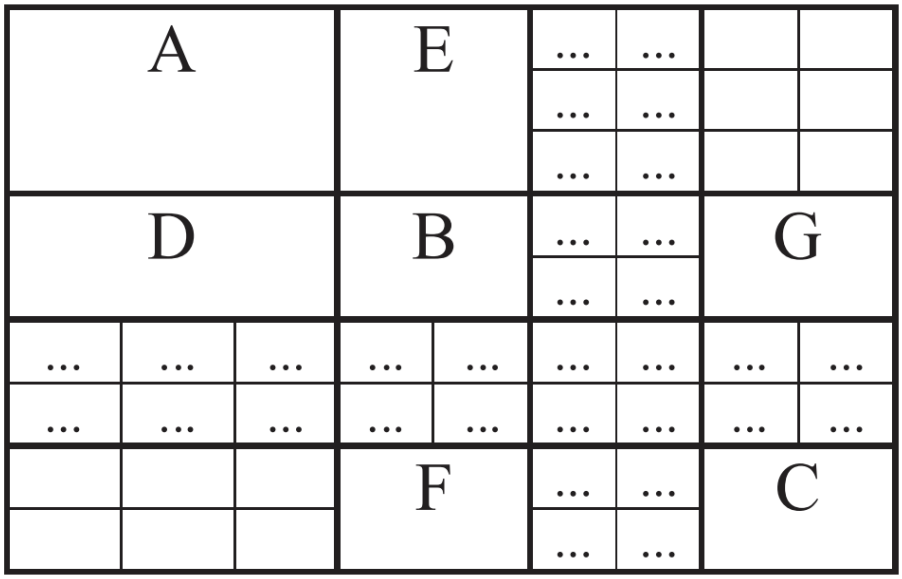

Figure 2.4: Illustration of the sub-sections of the covariance matrix [2]

2.1.2.3 The Prediction Model and the Jacobian

The prediction model for EKF SLAM is a non-linear function that estimates state variables for the next iteration [3]. For two dimensional mobile robotics the prediction model is defined as:

$$f(x_{k-1|k-1}, u_k) = x_{k-1|k-1} + \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \end{bmatrix}^T \begin{bmatrix} -\frac{v_t}{\omega_t} sin(\theta) + \frac{v_t}{\omega_t} sin(\theta + \omega_t \Delta t \\ \frac{v_t}{\omega_t} cos(\theta) - \frac{v_t}{\omega_t} cos(\theta + \omega_t \Delta t) \\ \omega_t \Delta t \end{bmatrix} \tag{2.12}$$

$$f(x_{k-1|k-1}, u_k) = \begin{bmatrix} x_r & y_r & \theta_r & x_1 & y_1 & \dots & x_n & y_n \end{bmatrix}^T + \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \end{bmatrix}^T \begin{bmatrix} -\frac{v_t}{\omega_t} sin(\theta) + \frac{v_t}{\omega_t} sin(\theta + \omega_t \Delta t \\ \frac{v_t}{\omega_t} cos(\theta) - \frac{v_t}{\omega_t} cos(\theta + \omega_t \Delta t) \\ \omega_t \Delta t \end{bmatrix} \tag{2.13}$$

The current iteration of the filter is defined as k, the current orientation of the robot in the state vector is defined as θ, the linear velocity of the robot is defined as v_t, the rotational velocity is defined as w_t, and the time interval over which the linear and rotational velocity's were measured is defined as Δt. The transposed matrix multiplied with the state prediction matrix is used to calculate the state prediction matrix into state vector form. In addition, the transposed matrix ensures that the landmark values in the state prediction are all zero, as they are not affected by the prediction model. The control vector is defined as u_k. It contains the linear and rotational relocites as such:

$$u_k = \begin{bmatrix} v_t \\ \omega_t \end{bmatrix} \tag{2.14}$$

Since $f(x_{k-1|k-1}, u_k)$ is a non-linear function, it must be linearized in order to be used in the prediction stage.

$$F_k = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \end{bmatrix}^T \begin{bmatrix} -\frac{v_t}{\omega_t} sin(\theta) + \frac{v_t}{\omega_t} sin(\theta + \omega_t \Delta t \\ \frac{v_t}{\omega_t} cos(\theta) - \frac{v_t}{\omega_t} cos(\theta + \omega_t \Delta t) \\ \omega_t \Delta t \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \end{bmatrix} \tag{2.15}$$

The Jacbian of the motion model is defined as F_k The matrices with 1's along the diagonal are size 3x3N+3 where N is the number of recorded landmarks. This is used to get the linearized model into a form where 1's are present along the diagonals of the matrix and the linearized motion equations are in the top 3x3 section of the F_k matrix [3]. When F_k is multiplied with P the terms representing the landmarks will be unaffected. With the Jacobian included, the prediction step appears as follows.

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \tag{2.16}$$

This is of the same form as equation 2.2, except F_k represents the Jacobian from equation 2.8.

2.1.2.4 The Observation Model and the Jacobian

The observation model is utilized to get measurements into the form of the state vector. When a landmark is measured the range and orientation of the robot to the landmark with respect to the base frame of the robot is recorded. However, the state vector stores landmarks as x and y coordinates. To get the stored landmarks into a form that can be compared to the measurements the observation model is utilized. Once converted by the observation model, $h(x_{k|k-1}, u_k)$, the error between the landmarks predicted value and the measurement, this eventually results in the Kalman gain which describes how much the filter should correct the state estimates based on the measurements.

The observation model is defined as:

$$h(x_{k|k-1}, u_k) = \begin{bmatrix} (\sqrt{(x_r - x_i)^2 + (y_r - y_i)^2} \\ tan^{-1}(\frac{y_i - y_r}{x_i - x_r}) - \theta) \end{bmatrix} = \begin{bmatrix} range \\ bearing \end{bmatrix} \tag{2.17}$$

and is utilized during the beginning of the measurement phase when the error between the measurement and state is taken. The robots (x,y) coordinates form the state vector are defined as x_r, y_r. The ith detected landmark (x,y) coordinates are also determined by the state vector and defined as x_i, y_i. The robots orientation, defined as θ comes from the state vector as well. The Jacobian H is obtained by taking the first order partial derivative of $h()$ and appears as such:

$$H = \begin{bmatrix} \frac{x_r - x_i}{r} & \frac{y_r - y_i}{r} & 0 \\ \frac{y_r - y_i}{r^2} & \frac{x_r - x_i}{r^2} & -1 \end{bmatrix} \tag{2.18}$$

The change in range with respect to x is seen in $H(1,1)$ and the change in bearing with respect to x is see in $H(2,1)$. The change in range with respect to y is seen in $H(1,1)$ and the change in bearing with respect to y is see in $H(2,1)$. The last column is used to show the change in rotation, since the orientation of the robot has no impact on the range to a landmark $H(1,3)$ is set to 0.

H must now be reshaped depending on which landmark has been detected. Given that H is of the form:

$$H = \begin{bmatrix} A & B & C \\ D & E & F \end{bmatrix} \tag{2.19}$$

Then the H matrix within the filter will have contents seen in 2.5, using the 2nd landmark as an example:

X_r	Y_r	T_r	X_1	Y_1	X_2	Y_2	X_3	Y_3
A	B	C	0	0	-A	-B	0	0
D	E	F	0	0	-D	-E	0	0

Figure 2.5: Jacobian of the observation model [2] [3].

The first three columns contain the full H matrix. In the columns containing the 2nd landmark, only the first two columns are used. This is because the orientation of the landmarks are not part of the state variables. The landmark values are also negated due to the change of perspective from the robot's base frame to the coordinates of the detected landmark [3].

2.1.2.5 The Kalman Gain

The Kalman gain is used to determine how much a measurement should be used to correct the state estimate. The Kalman gain is defined as follows:

$$K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1} \tag{2.20}$$

P is the covariance matrix, H is the Jacobian of the observation model, and R is the measurement error covariance matrix. The form of the Kalman gain matrix is seen in Figure 2.6 The robot's position orientation, and time, are denoted as x, y,

x_r	x_b
y_r	y_b
t_r	t_b
$x_{l,r}$	$x_{l,b}$
$y_{l,r}$	$y_{l,b}$
...	...
...	...
$x_{n,r}$	$x_{n,b}$
$y_{n,r}$	$y_{n,b}$

Figure 2.6: Kalman gain matrix, K [2].

and t. The coordinates of the landmarks are denoted as x_n and y_n where n is nth landmark observed. The subscript r is used to denote the first column as representing the range of each variable The subscript b is sued to denote the second column as representing the bearing of each variable. The Kalman gain is then used to correct the state estimates via the following two equations:

$$x_{k|k} = x_{k|k-1} + K_k y_k \tag{2.21}$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{2.22}$$

2.1.2.6 Noise Covariance Matrix

To model the uncertainty of the state prediction and measurement values, the Q and R matrices are utilized. The Extended Kalman Filter assumes that the noise associated with measurement and predictions are zero mean Gaussian distributed. The variance of the Gaussian distribution is assigned to the Q and R matrices. The variance is typically determined through empirical testing and is application/variable specific. The Q matrix is associated with the prediction phase and represents the variance from the control inputs. The Q matrix must be the same size as the covariance matrix in order for equation 2.16 to compute. The R matrix represents the variance of the measurement. Often, the R matrix is a two by two matrix with the variance of the range to the landmark as the first diagonal element and the variance of the bearing to the landmark as the second element. A sensor such as a laser range finder would use the diagonals as described where the resolution of the sensors is assigned to the first value of the diagonal and the step size of the sweep is assigned to the second diagonal [3].

2.1.3 Range Only EKF SLAM

The EKF SLAM algorithm in section 3.1.2 uses measurements of landmarks that contain the range and orientation of the current robot pose relative to the landmark. However, when devices such as wireless beacons are utilized as landmarks, the measurements only contain information pertaining to the range between landmarks and robot pose [17]. Thus these measurements only have a meaningful influence on pose of the tracked robot, while the orientation of the tracked robot is unaffected. Algorithms that utilize this measurement format are referred to as range-only SLAM (RO-SLAM).

The EKF RO-SLAM algorithm is largely the same as the EKF SLAM algorithm

with the exception of the measurement phase. The first difference comes from the innovation vector y_k. Previously y_k was denoted as

$$y_k = z_k - h(x_{k|k-1,u_k})$$

which can be expressed as:

$$y_k = \begin{bmatrix} d_m \\ b_m \end{bmatrix} - \begin{bmatrix} d_e \\ b_e \end{bmatrix} \tag{2.23}$$

Where d_m is measured distance to the landmark, b_m is the measured bearing to the landmark, d_e is the expected distance to the landmark based on the pose of the robot and landmark in the state vector, and b_e is the expected bearing to the landmark based on the state vector. Without a bearing measurement to the landmark the second row of the innovation matrix must be removed as no relationship between measured and expected bearing can be inferred. The innovation vector becomes:

$$y_k = \begin{bmatrix} d_m \end{bmatrix} - \begin{bmatrix} d_e \end{bmatrix} \tag{2.24}$$

With no bearing measurement the Jacobian must, H must also be reevaluated:

$$H = \begin{bmatrix} \frac{x_r - x_i}{r} & \frac{y_r - y_i}{r} & 0 \end{bmatrix} \tag{2.25}$$

Which reprsented as:

$$H = \begin{bmatrix} A & B & C \end{bmatrix} \tag{2.26}$$

Leads to the new Jacobian of the range only observation model: Thus allowing the

X_r	Y_r	T_r	X_1	Y_1	X_2	Y_2	X_3	Y_3
A	B	C	0	0	-A	-B	0	0

Figure 2.7: Jacobian of the range only observation model

EKF SLAM algorithm measurement phase to appropriately operate with range only information.

2.1.4 Orientation measurement estimation

An additional measurement for the modified RO-EKF algorithm proposed by this work is that of an measurement of the robots orientation. This orientation measurement is obtained through trajectory fitting by applying a least squares fit polynomial of degree two to a set of trilaterated robot position measurements. This orientation is then applied to RO-EKF as another measurement phase. The measurement vector for the new measurement phase is identical to the EKF measurement vector described in Section 2.1.2.4, and is defined as the range to a landmark and the bearing to a a landmark. The observation model also follows the EKF observation model as defined in Equation 2.17 and the EKF Jacobian as defined by Equation 2.18.

2.2 Least Squares Fitting

The least squares fit of polynomial degree 2 utilizes linear regression. Linear regression is used to find a line that best fits a set of data [18]. The line is defined as:

$$y = a_0 + a_1 * x^1 + a_2 * x^2 + ... + a_k * x^k \tag{2.27}$$

Where k is the degree of the polynomial of the line being fit to the data set x, and a is the coefficients of that polynomial. The line is considered the best fit if it minimizes

the sum of square errors as presented here:

$$E^2 = \sum_{i=1}^{n} [y_i - (a_0 + a_1 * x_i^1 + a_2 * x_i^2 + ... + a_k * x_i^k)] \tag{2.28}$$

E is the error, x_i is the i_{th} value of the x portion of the data set being considered, y_i is i_{th} the value of the y portion of the data set being considered, and n is total number of data values in the data set. Thus E will be minimized for values of a_0 through a_k with which the partial derivatives with respect to the coefficients equal zero:

$$\begin{bmatrix} \frac{\delta E^2}{\delta a_0} = -2 \sum_{i=1}^{n} [y_i - (a_0 + a_1 * x_i^1 + a_2 * x_i^2 ... + a_k * x_i^k)] = 0 \\ \frac{\delta E^2}{\delta a_1} = -2 \sum_{i=1}^{n} [y_i - (a_0 + a_1 * x_i^1 + a_2 * x_i^2 ... + a_k * x_i^k)] * x_i^1 = 0 \\ \vdots \\ \frac{\delta E^2}{\delta a_k} = -2 \sum_{i=1}^{n} [y_i - (a_0 + a_1 * x_i^1 + a_2 * x_i^2 ... + a_k * x_i^k)] * x_i^k = 0 \end{bmatrix} \tag{2.29}$$

Which can be expanded as:

$$\begin{bmatrix} \sum_{i=1}^{n} y_i = a_0 * n + a_1 \sum_{i=1}^{n} x_i + ... + a_k \sum_{i=1}^{n} x_i^k \\ \sum_{i=1}^{n} x_i y_i = a_0 \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^2 + ... + a_k \sum_{i=1}^{n} x_i^{k+1} \\ \cdots \\ \sum_{i=1}^{n} x_i^k y_i = a_0 \sum_{i=1}^{n} x_i^k + a_1 \sum_{i=1}^{n} x_i^{k+1} + ... + a_k \sum_{i=1}^{n} x_i^{2*k} \end{bmatrix} \tag{2.30}$$

Which can then be put into matrix form:

$$\begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \\ \vdots \\ \sum_{i=1}^{n} x_i^k y_i \end{bmatrix} = \begin{bmatrix} n & \sum_{i=1}^{n} x_i & \cdots & \sum_{i=1}^{n} x_i^k \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & \cdots & \sum_{i=1}^{n} x_i^{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} x_i^k & \sum_{i=1}^{n} x_i^{k+1} & \cdots & \sum_{i=1}^{n} x_i^{2*k} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix} \tag{2.31}$$

Since this is a Vandermonde matrix [19] the matrix for a least squares fit can be composed as:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & ... & x_1^k \\ 1 & x_2 & x_2^2 & ... & x_2^k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & ... & x_n^k \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix} \tag{2.32}$$

In this form the coefficients $a_0, a_1, ..., a_k$ can be solved for.

$$\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix} = \left(\begin{bmatrix} 1 & x_1 & x_1^2 & ... & x_1^k \\ 1 & x_2 & x_2^2 & ... & x_2^k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & ... & x_n^k \end{bmatrix}^T \begin{bmatrix} 1 & x_1 & x_1^2 & ... & x_1^k \\ 1 & x_2 & x_2^2 & ... & x_2^k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & ... & x_n^k \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & x_1 & x_1^2 & ... & x_1^k \\ 1 & x_2 & x_2^2 & ... & x_2^k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & ... & x_n^k \end{bmatrix}^T \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \tag{2.33}$$

Thus, a polynomial of degree k can be selected in order to find the line that bests fits a set of data points by minimizing the square error of the between the fit and the points.

The selection of the degree of the polynomial depends on the what information is desired from the data points. In this research the data points represent the trajectory via the x and y position of the trilaterated range measurement of the robot over time. The motion of the robot can be described as translation with a linear and angular

velocity. In other words the motion of the robot is controlled by control vector composed of a translation velocity and rotational velocity at any instantaneous point in time.

$$u_t = \begin{bmatrix} v_t \\ w_t \end{bmatrix} \tag{2.34}$$

Where w is the rotation velocity, v is the translation velocity, u is a control vector composed of the two velocities, and t is an instantaneous point in time. Assuming the control is noise free and that the robot maintains a constant u between two t intervals, then the motion can be desribed as circular with a radius of:

$$r = |\frac{v_t}{w_t}| \tag{2.35}$$

Assuming the pose of the robot is stored in a vector formated as:

$$p = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{2.36}$$

Where x, y, and θ are the location of the robot in a global coordinate frame. Then the center of the circle resulting from the translation and rotational velocity at a point in time can be defined as:

$$x_c = x - \frac{v}{w} sin(\theta) \tag{2.37}$$

$$y_c = y - \frac{v}{w} cos(\theta) \tag{2.38}$$

Where x_c and y_c are the center of the circle, x, y, and θ are the pose of the robot, and

v and w are the velocity of the robot. Thus, after applying translation and rotational velocity for one time instance, defined as Δ_t the change in the robots pose can be expressed as:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x_c + \frac{v}{w} sin(\theta + w\Delta_t) \\ y_c - \frac{v}{w} cos(\theta + w\Delta_t) \\ w\Delta_t \end{bmatrix} \tag{2.39}$$

Which can be further defined as:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} -\frac{v}{w} sin(\theta) + \frac{v}{w} sin(\theta + w\Delta_t) \\ \frac{v}{w} cos(\theta) - \frac{v}{w} cos(\theta + w\Delta_t) \\ w\Delta_t \end{bmatrix} \tag{2.40}$$

Where x', y', and θ' are the pose of the robot after applying the translational and rotational velocities. However, in reality there is noise that is applied to the control vector, thus it is more accurately represented as:

$$\begin{bmatrix} \hat{v} \\ \hat{w} \end{bmatrix} = \begin{bmatrix} v \\ w \end{bmatrix} + \begin{bmatrix} \epsilon_{\alpha_1 v^2 + w^2 \alpha_2 v^2} \\ \epsilon_{\alpha_3 v^2 + w^2 \alpha_4 v^2} \end{bmatrix} \tag{2.41}$$

Where $\hat{v}$ and $\hat{w}$ are the control vector after a zero-mean error variable, ϵ, is introduced with a variance of b^2 that is proportional to the command velocities. The values of α are selected to model the accuracy of the control. The less accurate, the larger the α.

The introduction of this noise leads to a degeneracy in the currently defined motion model. The control noise affects the radius of the circle defined by equation 2.35 and the distance traversed. However, the assumption that the trajectory is circular is not affected. Thus, the motion model must be generalized by assuming that the robot performs an additional rotation, Υ, after translating to its final $[x', y', \theta']$ pose. The

result being that θ' is computed as:

$$\theta' = \theta + \hat{w} + \Upsilon\Delta_t \tag{2.42}$$

Where Υ is defined as

$$\upsilon = \alpha_5 v^2 + w^2\alpha_6 v^2 \tag{2.43}$$

With a_5 and a_6 defined as additional noise that is specific to the robot. This leads to the final motion model represented as:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} -\frac{\hat{v}}{\hat{w}}sin(\theta) + \frac{\hat{v}}{\hat{w}}sin(\theta + \hat{w}\Delta_t) \\ \frac{\hat{v}}{\hat{w}}cos(\theta) - \frac{\hat{v}}{\hat{w}}cos(\theta + \hat{w}\Delta_t) \\ \hat{w}\Delta_t + \Upsilon\Delta_t \end{bmatrix} \tag{2.44}$$

Based on the motion model defined, the robots motion can be described at any point in time as having an instantaneous center of curvature. Considering the robot is constrained to have a linear velocity to rotational velocity ratio of at least 0.35 meters, this means that the minimum radius for the circle represented by the instantaneous center of curvature of the robot turn is 0.35 meters. Additional constraints were placed on the motion, these consisted of ensuring that during a two second time period the ratio of linear to rotational velocity could only either increase or decrease by a maximum of .1. The arc-length of the minimum turn of the robot during the two second time period, as constrained by statements above, has the additional constraint of that all points along the arc-length of the turn are unique. Therefore, given the possible trajectories that can be formed from these constrains, a polynomial of degree two will best describe the trajectory or the robot over the specified sampling period. However, the robot motion contains noise leading to erroneous readings, thus the coefficients that minimize the error for this polynomial must be solved for. This can

be expressed as:

$$\begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} x_i^2 y_i \end{bmatrix} = \begin{bmatrix} n & \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i^3 \\ \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} \tag{2.45}$$

2.3 Trilateration

Trilateration for wireless localization is the process by which at least three wireless devices at known positions in two dimensional space produce a range measurement to a fourth wireless device at an unknown position in two dimensional space, then from those range measurements the location of the fourth device is estimated. Trilateration on its own has been used to solve the localization problem in numerous situations [20, 21]. Under ideal conditions (exact range measurements) the location of the fourth device will be precisely calculated. This is done by treating each range measurement as a radius for a circle centered at the location of the wireless device that produced each measurement. The point of intersection for these three radii can then be used to determine the location of the unknown device. Figure 2.8 illustrates this process. Mathematically this ideal process can be described as follows [4].

$$r_1^2 = (x_{D_1} - x_{D_4})^2 + (y_{D_1} - y_{D_4})^2 \tag{2.46}$$

Where D_1 represents device one with its known location represented in a Cartesian coordinate system as x_{D_1} and y_{D_1}. The device with an unknown location is represented as D_4 with its unknown x and y location represented as x_{D_4} and y_{D_4}. r_1 represents the estimated Euclidean distance between the two locations, such as the

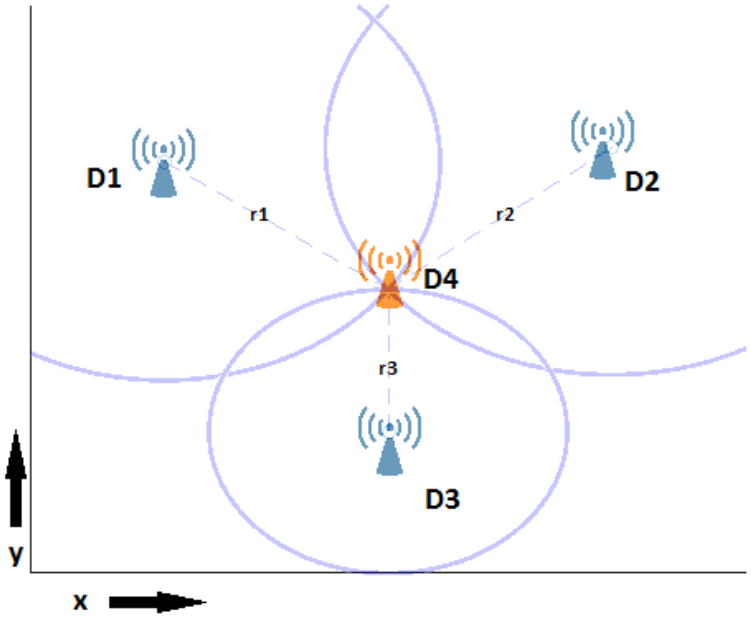

Figure 2.8: Wireless devices D1, D2, and D3, receive range measurements r1, r2, and r3, respectively, to the location of an unknown device. These ranges are represented as the radii of circles and the point of intersection between the three is the location of the unknown device [4].

range measurements described previously. The equation can be rearranged as follows:

$$(x_{D_1} - x_{D_4})^2 + (y_{D_1} - y_{D_4})^2 - r_1^2 = 0 \tag{2.47}$$

Each device with a known position and range measurement can also be represented in this form which leads to the formation of the following system of equations:

$$\begin{bmatrix} (x_{D_1} - x_{D_4})^2 + (y_{D_1} - y_{D_4})^2 \\ (x_{D_2} - x_{D_4})^2 + (y_{D_2} - y_{D_4})^2 \\ (x_{D_3} - x_{D_4})^2 + (y_{D_3} - y_{D_4})^2 \end{bmatrix} - \begin{bmatrix} r_1^2 \\ r_2^2 \\ r_3^2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.48}$$

The solution to this system of equations will yield the values for x_{D_4} and y_{D_4}, the location of the device being localized. However, if any error is introduced to the range measurements, such as from signal multi-path, then a scenario where no intersection

between all radii may occur, such as in Figure 2.9.

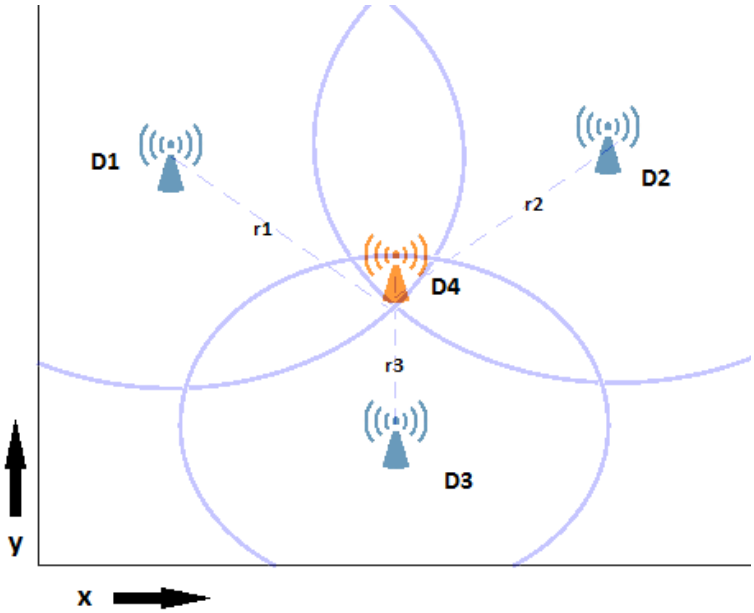

Figure 2.9: Wireless devices D1, D2, and D3, receive slightly erroneous range measurements r1, r2, and r3, respectively, to the location of an unknown device. Due to the erroneous range readings there is no solution for the location of the unknown device D4 [4].

In any practical environment it is unacceptable to operate with the requirement of near perfect data. This holds true for wireless ranging as well. Factors such as multi-path wave propagation, which is when a signal reaches a receiver by two or more paths, can generate erroneous range measurements. The scale of this error varies based on environmental conditions such as inside buildings where reflective materials is common place [12]. Since it is unlikely for a solution to exist under these conditions, Equation 2.48 is re-written as follows:

$$\begin{bmatrix} (x_{D_1} - x_{D_4})^2 + (y_{D_1} - y_{D_4})^2 \\ (x_{D_2} - x_{D_4})^2 + (y_{D_2} - y_{D_4})^2 \\ (x_{D_3} - x_{D_4})^2 + (y_{D_3} - y_{D_4})^2 \end{bmatrix} - \begin{bmatrix} r_1^2 \\ r_2^2 \\ r_3^2 \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = E \tag{2.49}$$

Where e_1 represents the error for node D_1, and E is a vector of containing the error

of each node. With the equation in this form, the solution is finding an x_{D_4} and y_{D_4} coordinate that will minimize the error vector E [20].

2.4 Non-Linear Least Squares Trilateration

Non-linear least squares estimation is utilized to minimize the sum of squares of distance errors. The minimization of the sum of square errors can be represented as the minimization of the following function:

$$F(\theta) = F(x, y) = \sum_{i=1}^{n} f_i(x, y)^2 \tag{2.50}$$

with f_i as

$$f_i(x, y) = f_i(\theta) := d_i(\theta) - r_i = \sqrt{(x - x_i)^2 + (y - y_i)^2} - r_i \tag{2.51}$$

Where r_i represents the estimated distance between devices and n is the total number of devices. Since $F(\theta)$ is non-linear the first order partial derivative is taken with respect to x and y to linearize it. Differentiation with respect to x produces:

$$\frac{\partial F(x, y)}{\partial x} = 2 \sum_{i=1}^{n} \frac{\partial f_i(x, y)}{\partial x} = 2 \sum_{i=1}^{n} \frac{\partial d_i(x, y)}{\partial x} \tag{2.52}$$

Differentiation with respect to y produces:

$$\frac{\partial F(x, y)}{\partial y} = 2 \sum_{i=1}^{n} \frac{\partial f_i(x, y)}{\partial y} = 2 \sum_{i=1}^{n} \frac{\partial d_i(x, y)}{\partial y} \tag{2.53}$$

To find the a solution which minimizes the squared error, ∇F must be solved for:

$$\nabla F = 2 J^T f = 0 \tag{2.54}$$

As ∇F is the linearized vector of partial derivatives and J is the Jacobian defined as:

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \vdots & \vdots \\ \frac{\partial f_i}{\partial x} & \frac{\partial f_i}{\partial y} \end{bmatrix} \tag{2.55}$$

f is the error function vector defined as:

$$f = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \tag{2.56}$$

With n as the total number of wireless devices. Which yields the following for ∇F:

$$\nabla F = 2 \begin{bmatrix} \sum_{i=1}^{n} \frac{(x-x_1)\partial f_i}{\partial x} \\ \sum_{i=1}^{n} \frac{(y-y_1)\partial f_i}{\partial x} \end{bmatrix} \tag{2.57}$$

The derivatives of F with respect to x and y has now been formulated. To solve for ∇F Newton's method can be applied.

2.5 Newton's Method

Newtons method successively searches for better approximations to the roots of a real function. Newtons method begins as:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{2.58}$$

Where x_0 is an initial guess for the approximation of the roots of the function f defined over a set of real number, and x_1 is the improved approximation. The function f divided by its derivative produces a tangent line of the function. The root for that

function is found from the x intercept of the tangent line. As the tangent x intercept approaches the true x intercept of the function the approximation to the root of the function becomes more accurate. This process can then be iteratively repeated as follows:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{2.59}$$

until a desired threshold of accuracy is achieved as depicted by Figure 2.10.

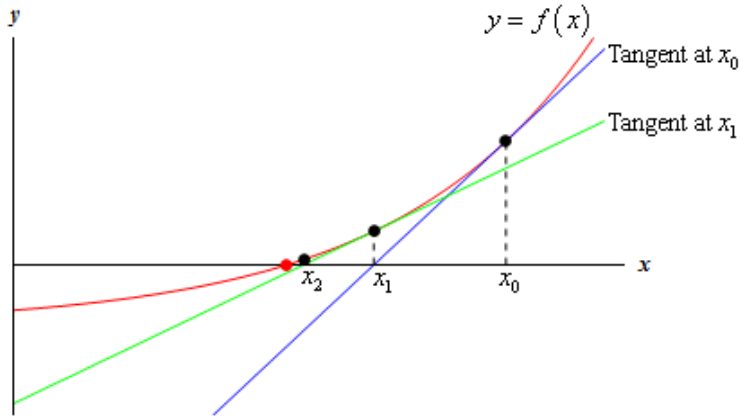

Figure 2.10: Examples of Newtons method iteratively solving for the best approximation of the function. The approximation x_1 falls close to the actual function than the initial approximation x_i [5]

Newtons method must now be defined for vector functions:

$$x_{n+1} = x_n - [J(x_n)]^{-1} f(x_n) \tag{2.60}$$

Where x is the current set of parameters of the current iteration k, $f(x_n)$ is the function to be minimized, and $J(x_n)$ is the Jacobian of the function to be minimized. The function to be minimized for two dimensional trilateration is the first order derivative of the squared error function described in equation 2.54.

$$x_{n+1} = x_n - \left[J(x_n)^T J(x_n) \right]^{-1} J(x_n)^T f \tag{2.61}$$

Where n is current iteration, x is the set of parameters, $f()$ is the error function vector from equation 2.54, and J is the Jacobian of $f()$.

2.6 Ultra-wide band

Ultra-wide band (UWB) refers to the 7.5 GHz frequency band between 3.1-10.6 GHz that was made available for commercial applications in the United States in early 2002. Several mandates have been set by the Federal Communication Commission [22], such as UWB devices occupying at least 500 MHz of bandwidth and setting the Equivalent Isotropically Radiated Power limit to -41 dBm/Mhz, make it so that UWB emission levels are extremely small. This results in UWB signals having a low probability of interfering with other radio systems. In addition, UWB signals have high resilience to the effect of multi-path interference due to its wide bandwidth nature, making UWB devices an excellent choice for operating in environments with high amount of multi-path interference, such as indoors. Finally, conventions on data transmission encoding for UWB result in a power consumption reduction per transmission which makes UWB an ideal candidate to be implemented on low power embedded systems [23].

All of the afore mentioned factors make UWB an attractive option to use for RO-SLAM applications, especially in environments where networks that utilize wireless sensors readings are being implemented [24]. In environments where line of sight is impeded by factors such as dust or smoke any visual based SLAM methods will falter [25]. SLAM methods that utilize UWB will still be able to operate as the dust/smoke filled environment will have little affect on the UWB signal.

CHAPTER 3: Experimental Setup

The experiments were conducted in two environments denoted as the laboratory and atrium. In both environments identical pieces of equipment and algorithms were used. During the experiments the robot would navigate a predefined course. The constraint placed upon the experimentation was that the robot must at all times with a positive linear velocity.

3.1 Equipment

The equipment used during testing is primarily composed of four elements: pozyx nodes/anchors, the Kobuki Turtlebot 2, a laptop used for data transmission and Turtlebot commands, and a computer used for data processing.

3.1.1 Pozyx anchors and nodes

Throughout the experiments conducted four pozyx anchors were used, as pictured in Figure 3.1. These pozyx anchors utilized two way ranging in conjunction with a pozyx node, as seen in Figure 3.2, to produce range measurement between the node and anchor. The anchors were mounted on pvc pipe at a height of .901 meters as seen in Figure 3.3. The anchors and node utilized two way ranging to measure the distance between anchor and node. The anchors and nodes operated at a central frequency 6489.6MHz with a 500 MHz bandwidth. The bitrate of transmissions was set to 850 kbps.

3.1.2 Turtlebot

The Kobuki Turtlebot 2 was used as the robotic base for transporting the pozyx node. The node was mounted to the top of the turtlebot at a height of 0.4064 meters. The node and turtlebot were connected to a laptop that utilized the robot operating

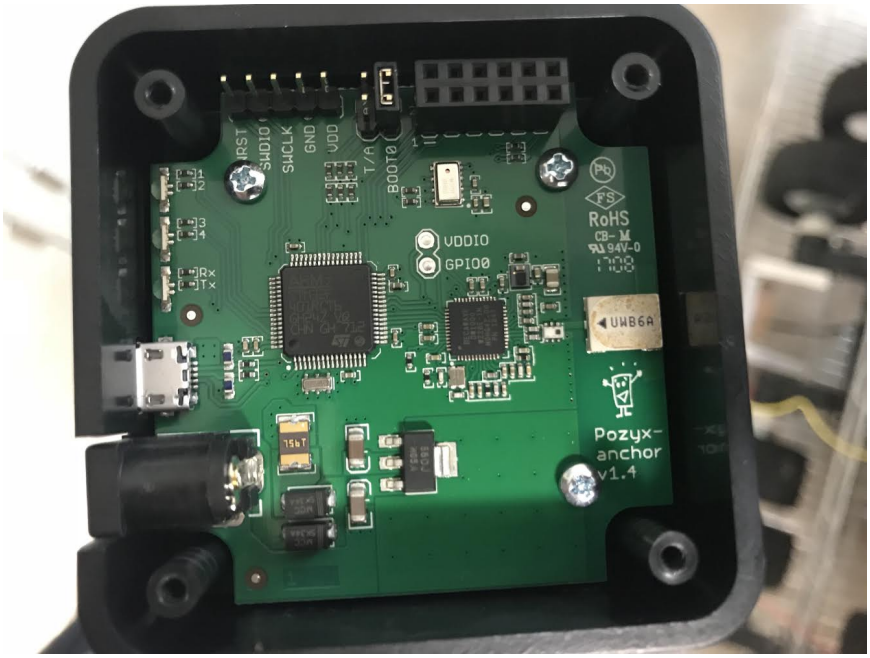

Figure 3.1: Pozyx anchor.

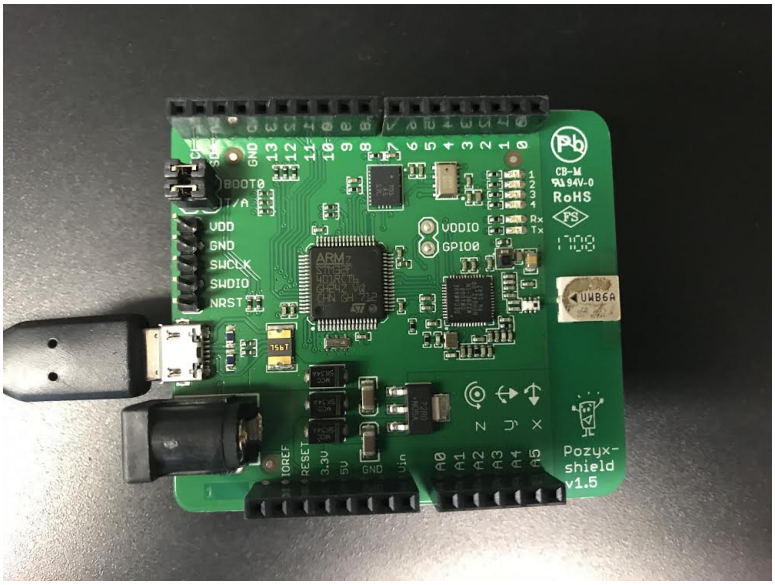

Figure 3.2: Pozyx node.

system to communicate between itself and another computer whose primary function was data analysis and computation. The Turtlebot, pozyx node, and laptop are seen in Figure 3.4.

Figure 3.3: Pozyx anchors mounted on poles.

Figure 3.4: Kobuki Turtlebot 2 with mounted pozyx node and the laptop responsible for communications.

3.2 Laboratory environment

The dimensions of the laboratory environment must be established in order to fully understand the experiment being conducted. Four pozyx anchor nodes, with hardware

addresses 6937, 6940, 6935, and 6955, were placed at locations as seen in Figure 3.5. It is noted that the remainder of this work will refer to the hardware addresses as Ids. The exact locations can be seen in Table 3.1. Twenty three points were placed throughout the lab environment and were used to record the exact position of the robot as it navigated a predefined path, these points are referred to as waypoints. Figure 3.5 visualizes the location of these waypoints. The number associated with each waypoints refers to the order by which the points were traversed by the robot. The exact locations of each point are seen in Table 3.2. To visualize the multitude of equipment and metallic objects in the laboratory environment pictures were taken from locations near each anchor, as through the schematic in Figure 3.5, the pictures of the physical environment are seen in Figure 3.6.

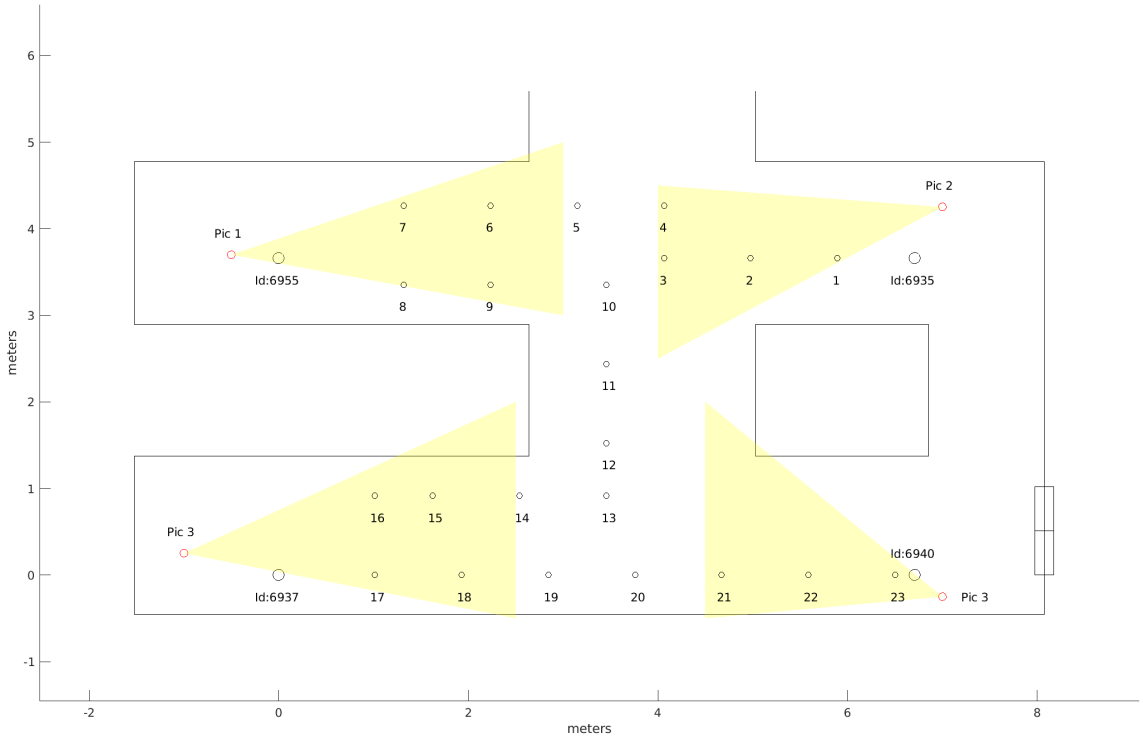

Figure 3.5: Plotted map of laboratory environment. Four anchor nodes denoted by their Ids are large black circles. Four picture were taken, as seen in Figure 3.6, red circles denote the approximate location and field of view from where the pictures were taken. Small black circles numbered 1 through 23 represent the way-points the robot traverses.

Figure 3.6: Visual pictures of the lab at each anchor. The field of view each picture encompasses is roughly depicted in Figure 3.5.

Table 3.1: The x and y location of each anchor node as seen in figure 3.5

Anchor ID	x(meters)	y(meters)
6937	0.00	0.00
6940	6.71	0.00
6935	6.71	3.66
6955	0.00	3.66

Table 3.2: The x and y location of each waypoint as seen in Figure 3.5

Waypoint Num.	x(meters)	y(meters)
1	5.8928	3.6576
2	4.9784	3.6576
3	4.0640	3.6576

17	1.0160	0.0000
18	1.9304	0.0000
19	2.8448	0.0000
20	3.7592	0.0000
21	4.6736	0.0000
22	5.5880	0.0000
23	6.5024	0.0000

3.2.1 Lab multipath effect

The laboratory environment was expected to contain a large amount of multipath noise due to the various metallic structures and the equipment running through out the lab. To verify the noise of the range measurements in the experimental environment the following tests were conducted. In the laboratory environment the robot was placed at several different location as seen in Figure 3.7 and 3.8. Roughly 250 range measurements were then taken with respect to each anchor at these locations.

The error between the actual range and measured range was computed for each anchor at each location. The results are displayed through the histograms in Figure 3.9, 3.10, 3.11, and 3.12.

The mean and standard deviation of the difference between the actual range and the measured range for each anchor at each location is tabulated in Table 3.3. The table shows how the difference in location affects the error of the ranging. The differences maintain a roughly Gaussian shape, however the mean shifts by a substantial amount based on the location of the tag. In addition, the shift is not equally reflected by each anchor when the location is changed. For example, at location 1 anchor 6935 has a mean range difference of 0.548 meters. At location 2 the range difference becomes 0.563. Between the two locations there is a range difference increase of 0.015 meters.

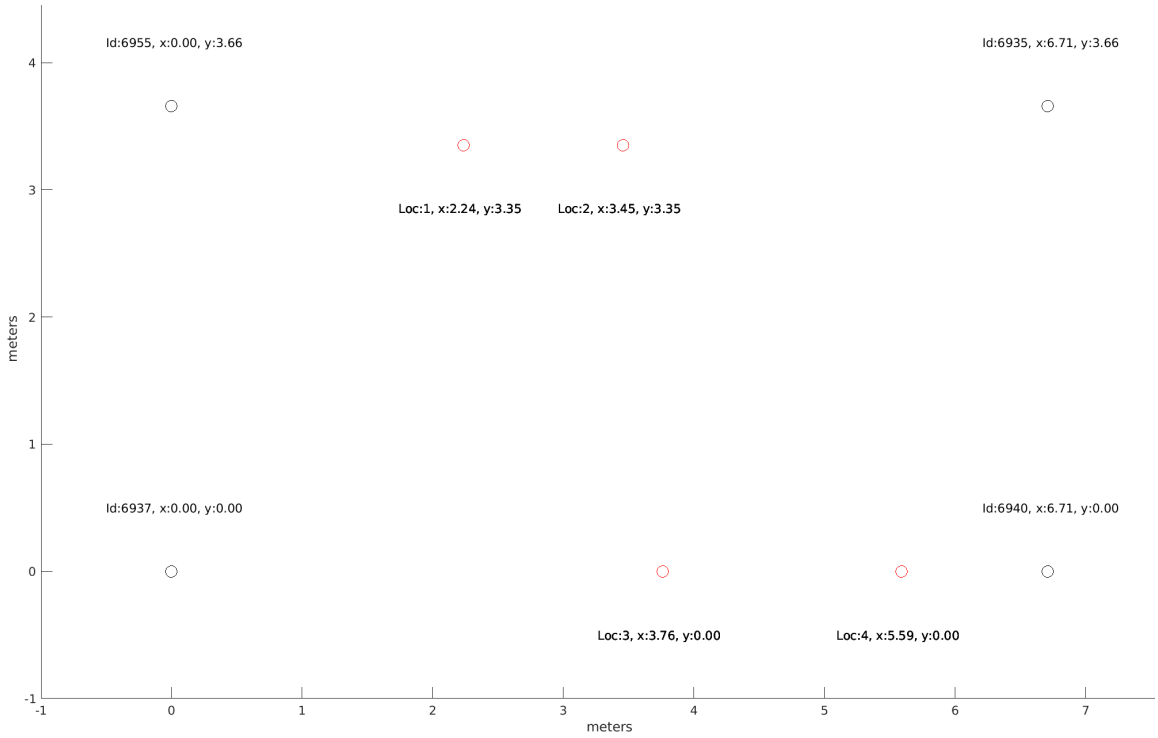

Figure 3.7: Location of anchors and locations where measurements were taken in the lab environment

At location 1 anchor 6937 has a mean range difference of 0.023 meters, at location 2 anchor 6937 has a mean range difference of 0.354 meters. There is a difference increase of 0.331 meters. Thus, between the two locations anchor 6935 has a change of 0.015 meters and anchor 6937 has a change of 0.331 meters. This exemplifies the effect of multipath in the lab environment. There is a similar correlation between the standard deviation of the range differences between two locations, although it is not as severe as the difference between means.

Table 3.3: The mean and standard deviation of the difference between the actual and measured range for each anchor at each location in the laboratory environment

Location	Anchor ID	Mean of Range Diff (m)	Std. Dev. of Range Diff
1	6937	0.023	0.0368
1	6940	1.605	0.0290
1	6935	0.548	0.0254

1	6955	0.856	0.0388
2	6937	0.354	0.0320
2	6940	1.216	0.0317
2	6935	0.563	0.0290
2	6955	0.879	0.0316
3	6937	0.502	0.0247
3	6940	0.852	0.0297
3	6935	0.482	0.0269
3	6955	-0.425	0.0370
4	6937	-3.261	0.0345
4	6940	1.695	0.0281
4	6935	1.170	0.0596
4	6955	-1.823	0.0254

3.3 Atrium environment

The dimensions of the atrium environment must be described to understand the experimentation that takes place in said environment. Four anchor nodes, with Ids 6937, 6940, 6935, and 6955, utilizing pozyx devices were placed at locations as seen in Figure 3.13. The exact locations can be seen in Table 3.4. Fifteen points were placed throughout the lab and were used to record the exact position of the robot as it navigated a predefined path, these points are referred to as waypoints. Figure 3.13 visualizes the location of these waypoints. The number associated with each waypoints refers to the order by which the points were traversed by the robot. The exact location of each point is seen in Table 3.5. To visualize the open area of the atrium enviorment in comparison to the laboratory pictures were taken from locations near two anchors, as seen in Figure 3.13, these pictures are seen in Figure 3.14.

Figure 3.8: The figure visually shows the locations of the robot in Figure 3.7 from which ranges were measured. 1) Location 1 taken from the perspective of anchor id 6955, 2) Location 2 taken from the perspective of anchor id 6955, 3) Location 3 taken from the perspective of anchor id 6937, 4) Location 4 taken from the perspective of anchor id 6937.

Table 3.4: The x and y location of each anchor node in the atrium as seen in Figure 3.13

Anchor ID	x(meters)	y(meters)
6937	4.05	0.00
6940	0.00	-0.91
6935	-1.83	4.27
6955	4.57	5.52

Table 3.5: The x and y location of each waypoint as seen in Figure 3.13

Waypoint Num.	x(meters)	y(meters)
1	1.8288	0

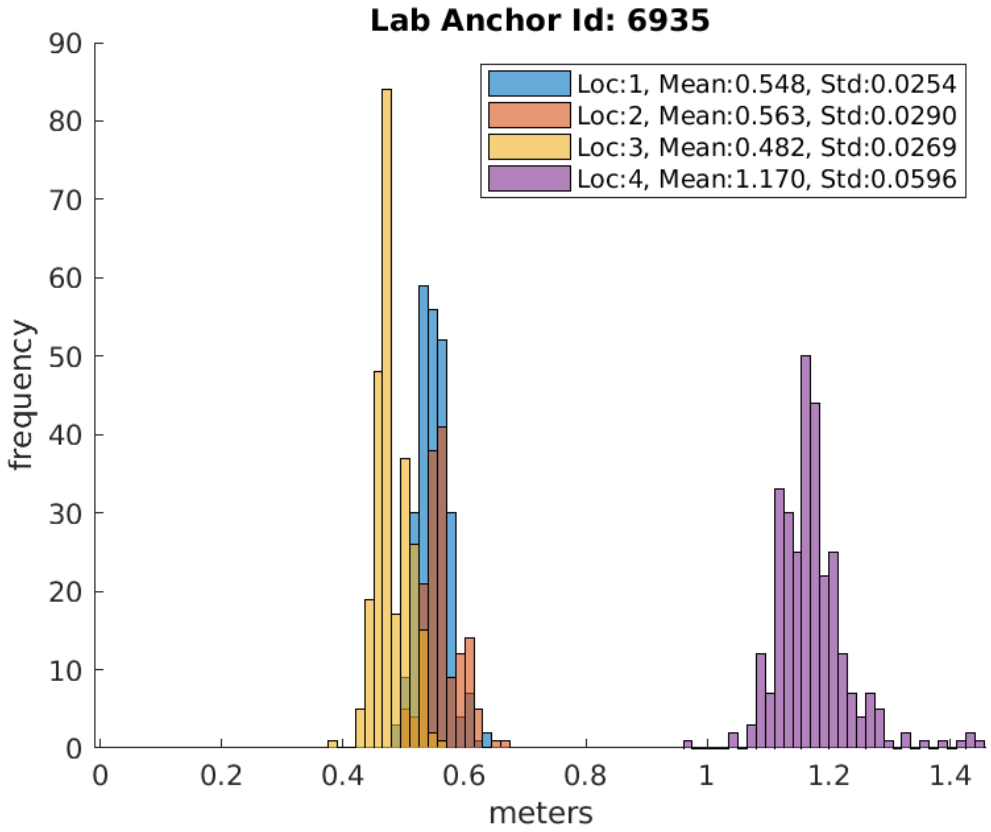

Figure 3.9: Histogram visualizing the error between the actual range and measured range for Anchor 6935 over the three locations as seen in Figure 3.7

3.3.1 Atrium multipath effect

The atrium environment was expected to be less affected by multipath due to the how much more open and unobstructed the space was. To verify the noise of the range measurements in the environment the following tests were conducted. In the atrium environment the robot was placed at several different location as seen in Figure 3.16 and 3.15. Roughly two hundred and fifty range measurements were then taken with respect to each anchor at these locations.

The error between the actual range and measured range was computed for each anchor at each location. The results are displayed through the histograms in Figure 3.17, 3.18, 3.19, and 3.20.

The mean and standard deviation of the difference between the actual range and the

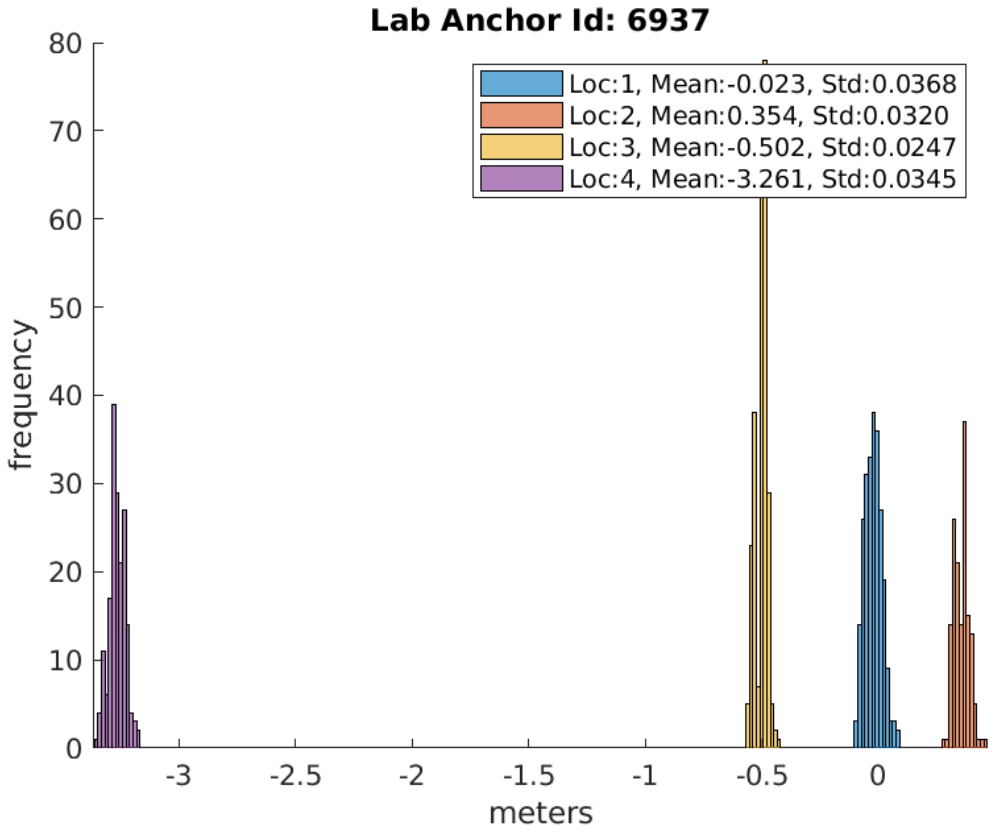

Figure 3.10: Histogram visualizing the error between the actual range and measured range for Anchor 6937 over the three locations as seen in Figure 3.7

measured range for each anchor at each location is tabulated in Table 3.6. The mean of the range differences in Table 3.6 is consistently closer to the value of zero than the range differences in Table 3.3. The table shows that in the atrium environment the effect of multi-path is much less pronounced than in the lab environment. This is a result of the open space and lack of reflective objects and equipment between anchors. It should also be noted that the standard deviation of the range differences is much more consistent between between locations compared to the standard deviation between locations in the laboratory environment.

Table 3.6: The mean and standard deviation of the difference between the actual and measured range for each anchor at each location in the atrium environment

Location	Anchor ID	Mean of Range Diff (m)	Std. Dev. of Range Diff

1	6937	-0.208	0.0319
1	6940	0.002	0.0338
1	6935	0.192	0.0806
1	6955	-0.238	0.0285
2	6937	-0.042	0.0294
2	6940	-0.037	0.0305
2	6935	-0.197	0.0257
2	6955	-0.037	0.0240
3	6937	-0.235	0.0265
3	6940	-0.198	0.0270
3	6935	-0.034	0.0279
3	6955	-0.060	0.0262

3.3.2 Algorithm

Before describing the algorithm utilized in this experiment, the communication between devices is first addressed. A laptop mounted on the turtlebot receives odomentry information from the Turtle, it communicates this odometry information to a processing computer via ROS over a Wireless Fidelity (Wi-Fi) network. The laptop also receives range information about each pozyx anchor from the pozyx node, it communicates this information to the processing computer as well. The processing computer utilizes the received information to run the modified RO-EKF algorithm and the traditional RO-EKF algorithm and track the estimated position of the node mounted to the robot. These algorithms were written in the Matlab language and run using the Matlab software. Finally, a human user inputs commands to the processing computer to control the movement of the turtle bot. These commands, termed control info, are transmitted to the laptop via ROS over wi-fi and are then communicated

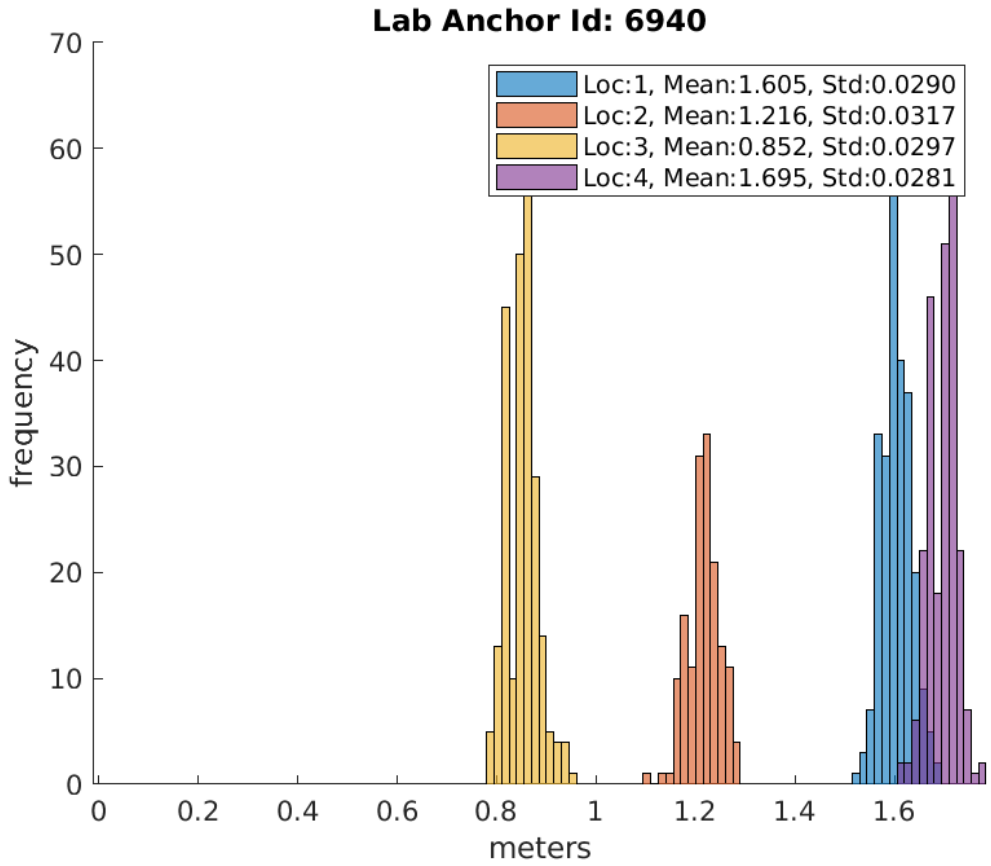

Figure 3.11: Histogram visualizing the error between the actual range and measured range for Anchor 6940 over the three locations as seen in Figure 3.7

by the laptop to the Turtlebot.

The processing computer runs the modified RO-EKF algorithm that is used to generate the position estimate of the robot. The modified RO-EKF algorithm functions very similarly to the traditional RO-EKf algorithm. After the algorithm is initialized the typical prediction and measurement phase are run. A prediction is made whenever the control information is registered and a measurement update is done whenever range information is received from a pozyx anchor. The additional elements of the modified RO-EKF algorithm consist of the trilaterate step and the orientation estimate step. The trilaterate step is run whenever at least three range measurement have been received from a pozyx anchors with compact time stamps. The trilaterated

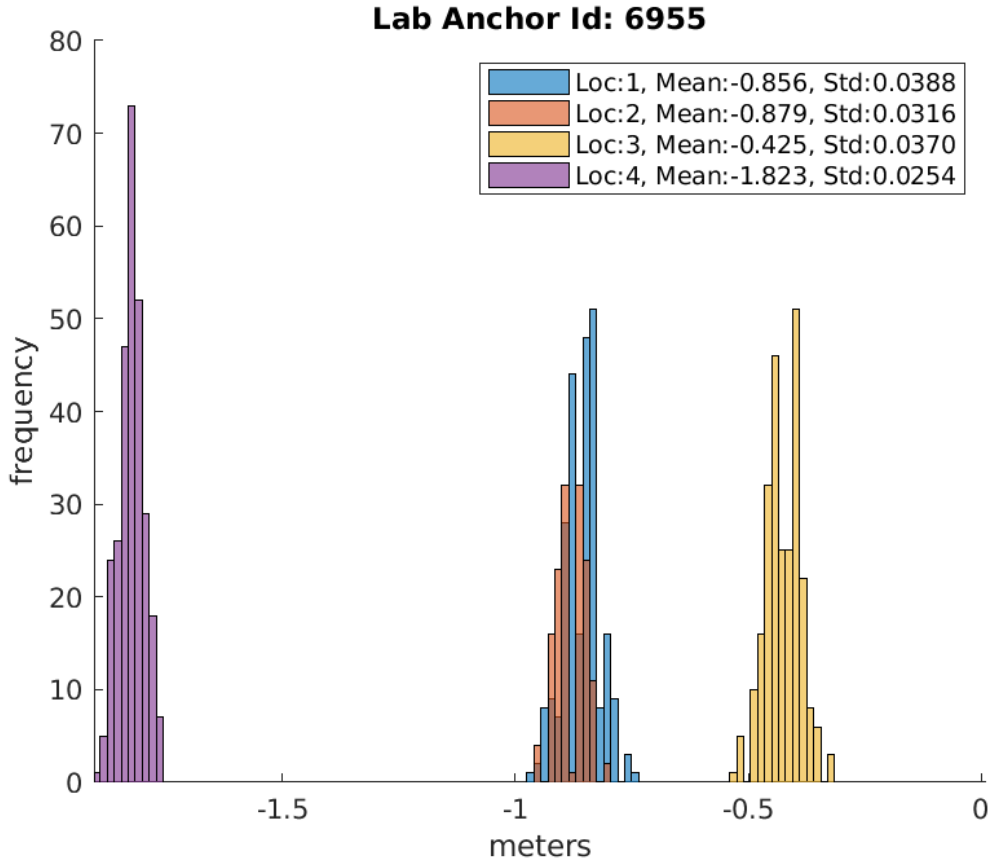

Figure 3.12: Histogram visualizing the error between the actual range and measured range for Anchor 6955 over the three locations as seen in Figure 3.7

positions are then stored. Once the number of stored positions passes a threshold, 75 for the experiments performed, the orientation estimation step is run. The orientation estimation method simply performs a second order least squares fit along the x and y dimensions of the trilaterated positions over time. The angle between each point of the line fit to the trilaterated positions is calculated and a summed with a weighting that decreases the older the time stamp of the trilaterated position. The estimated orientation is then used as a in conjunction with the range measurement to run the measurement phase and update the orientation stored in the modified RO-EKFs state vector as described in Section 2.1.4. A visual representation of this process can be seen in Figure 3.22.

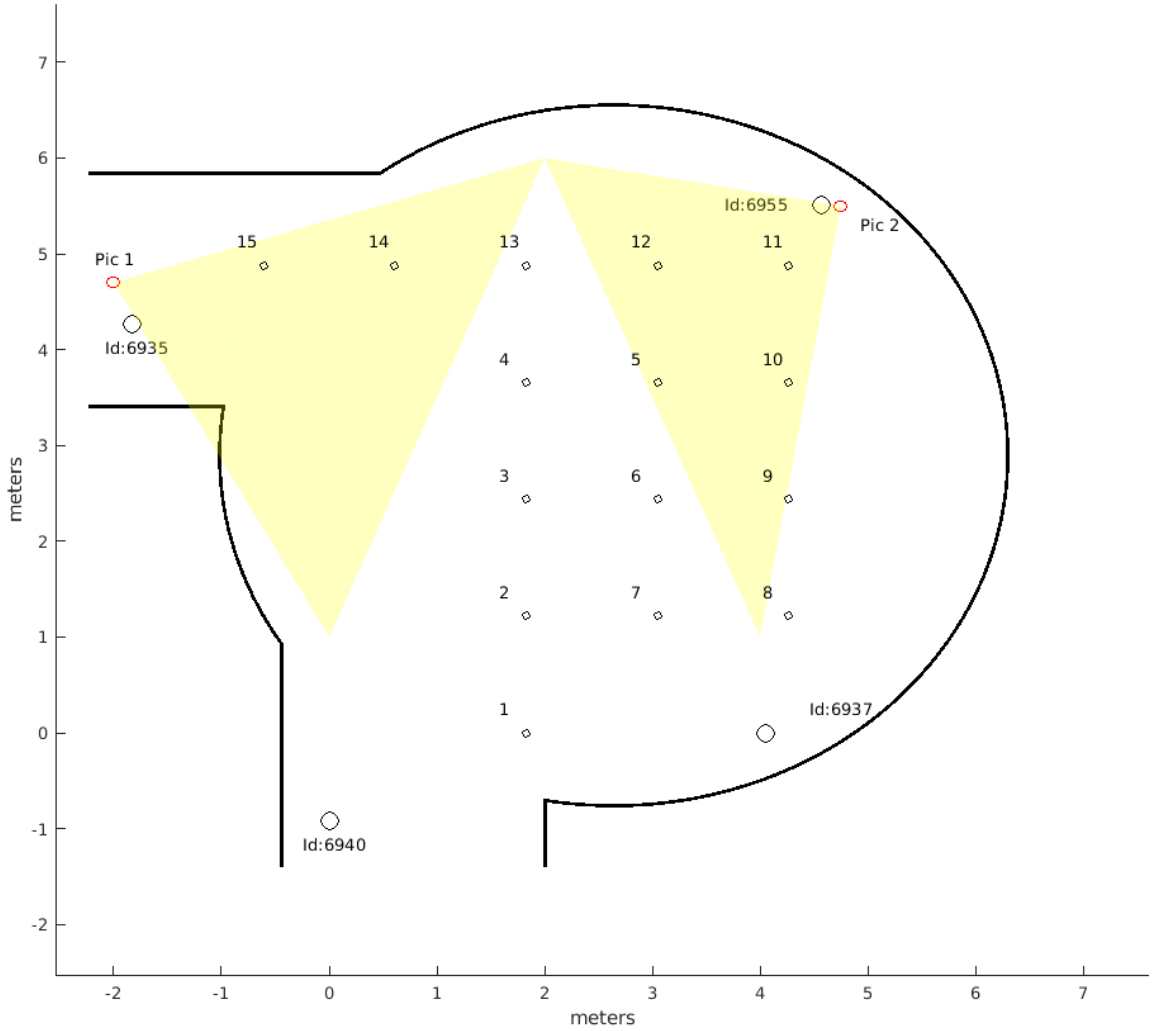

Figure 3.13: Plotted map of atrium environment. Four anchor nodes denoted by their Ids are large black circles. Two pictures were taken, as seen in Figure 3.14, red circles denote the approximate location and field of view from where the pictures were taken. Small black circles numbered 1 through 15 represent the way-points the robot traverses.

Figure 3.14: Visual pictures of the atrium at each anchor. The field of view each picture encompasses is roughly depicted in Figure 3.13.

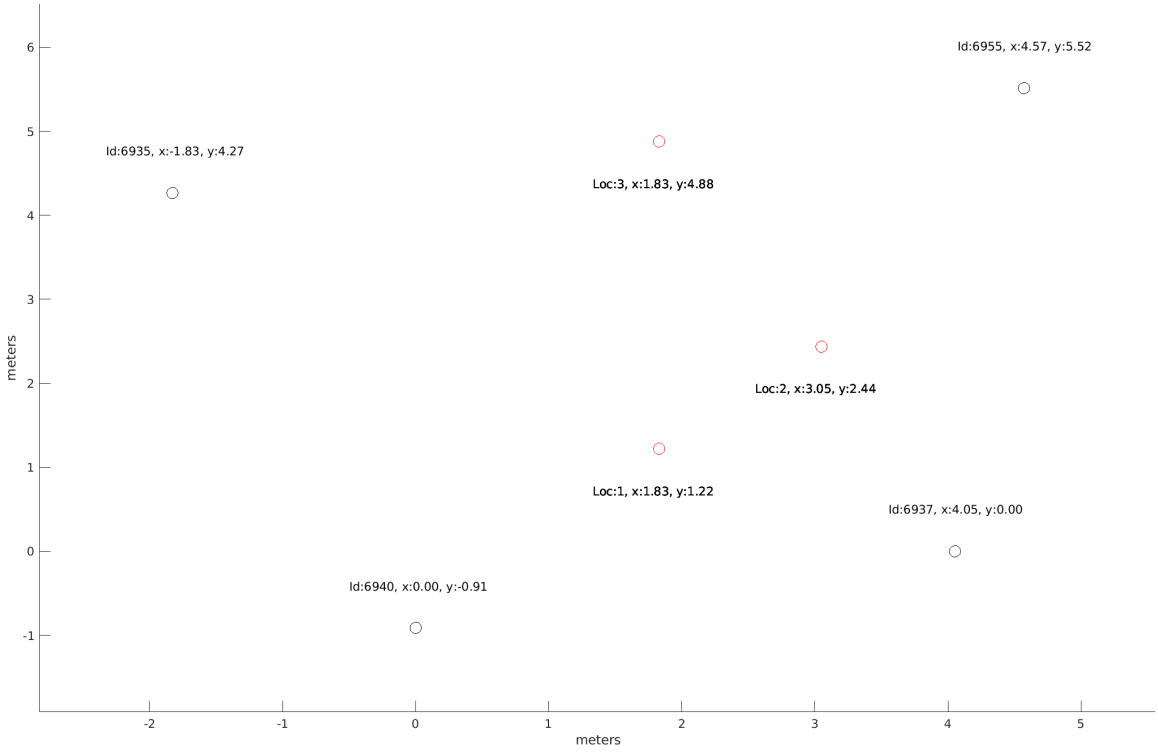

Figure 3.15: Positions of anchors and locations where measurements were taken in the atrium environment

Figure 3.16: Picture of location 1, 2, and 3 in the atrium taken from the perspective of anchor 6940.

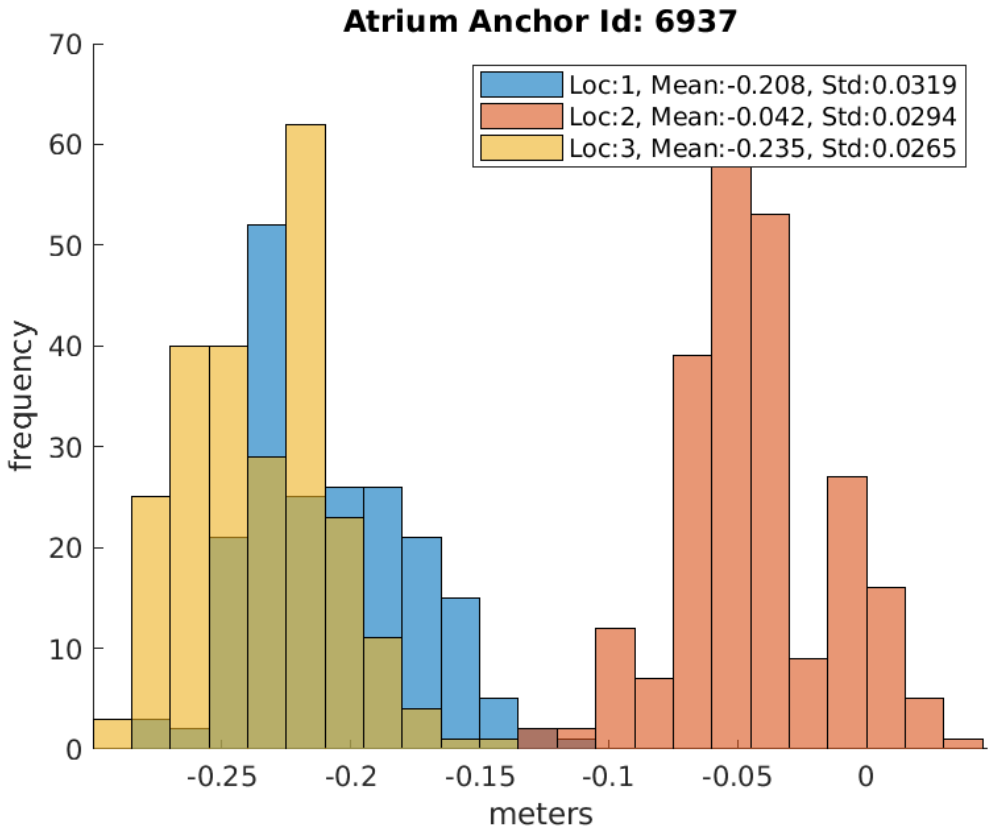

Figure 3.17: Histogram visualizing the error between the actual range and measured range for Anchor 6937 over the three locations as seen in Figure 3.16

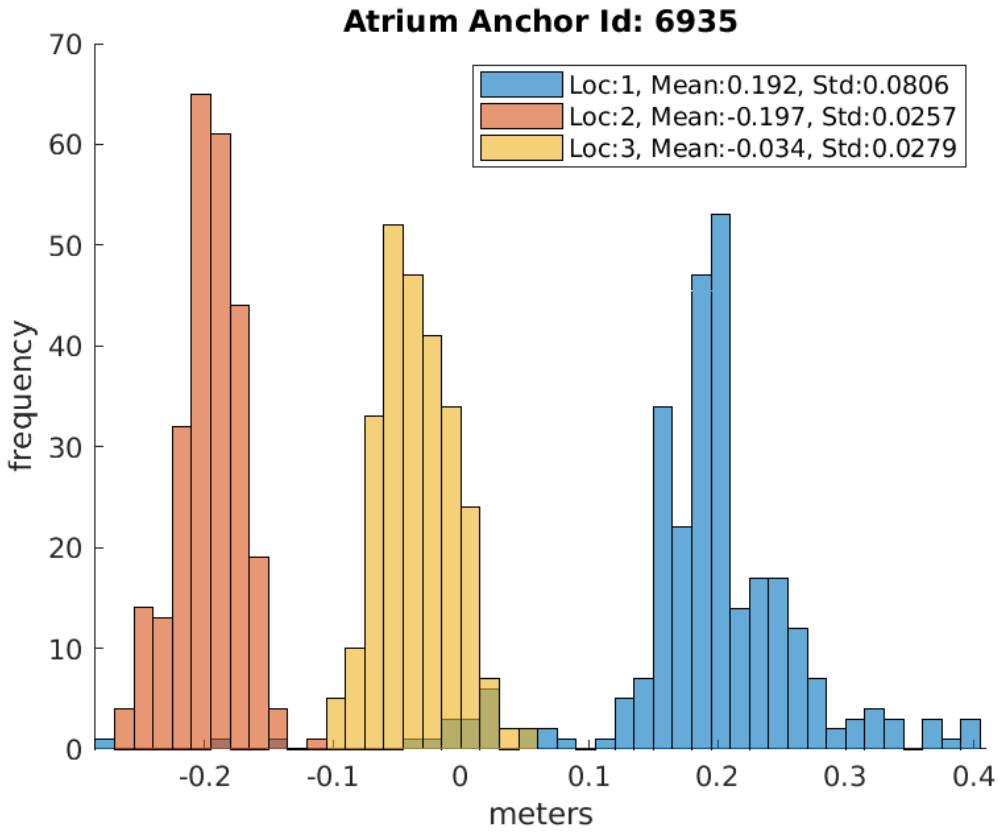

Figure 3.18: Histogram visualizing the error between the actual range and measured range for Anchor 6935 over the three locations as seen in Figure 3.16

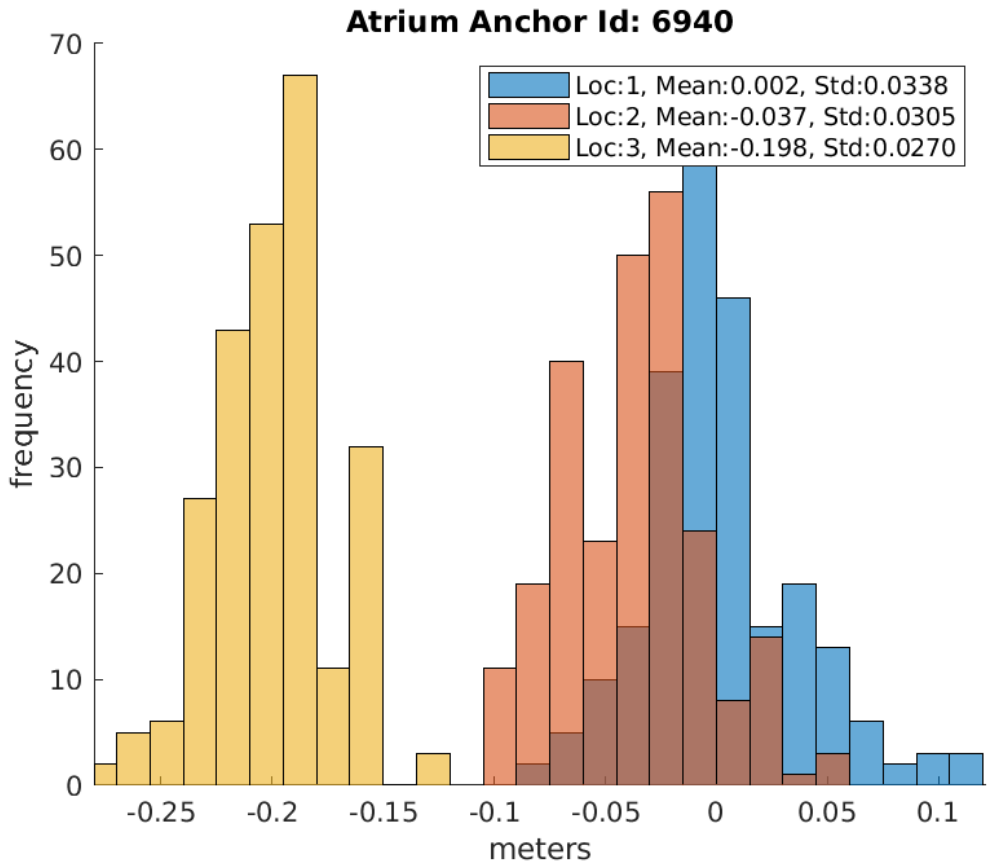

Figure 3.19: Histogram visualizing the error between the actual range and measured range for Anchor 6940 over the three locations as seen in Figure 3.16

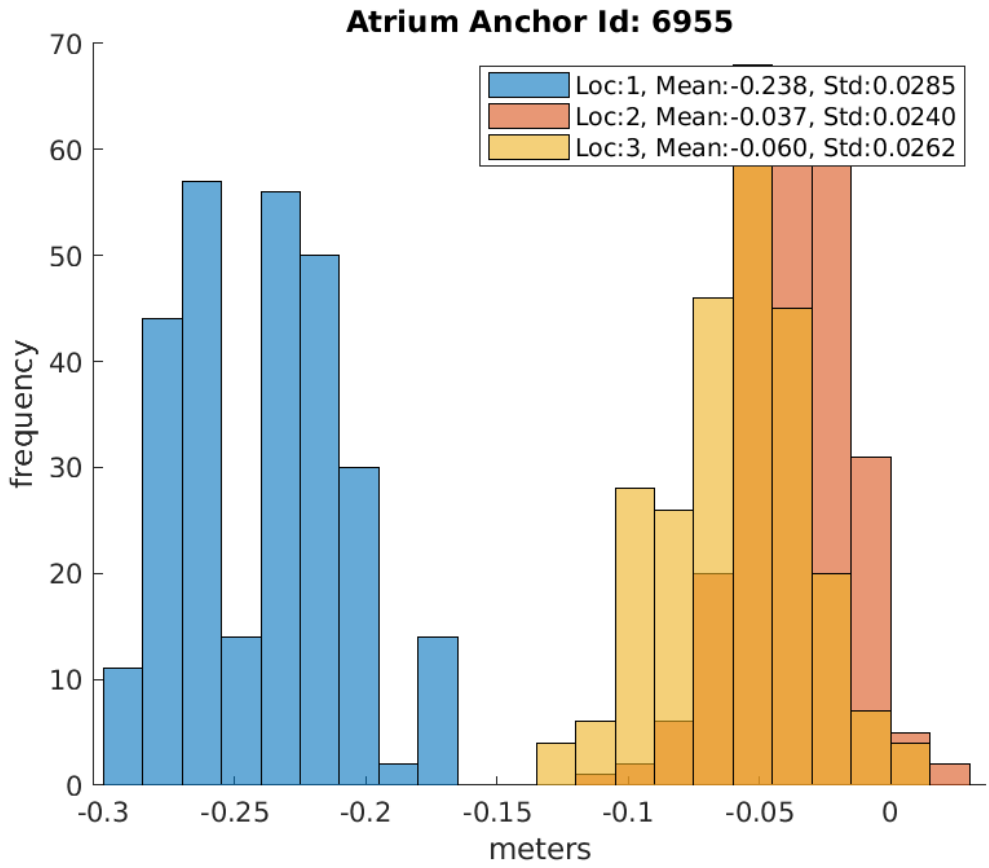

Figure 3.20: Histogram visualizing the error between the actual range and measured range for Anchor 6955 over the three locations as seen in Figure 3.16

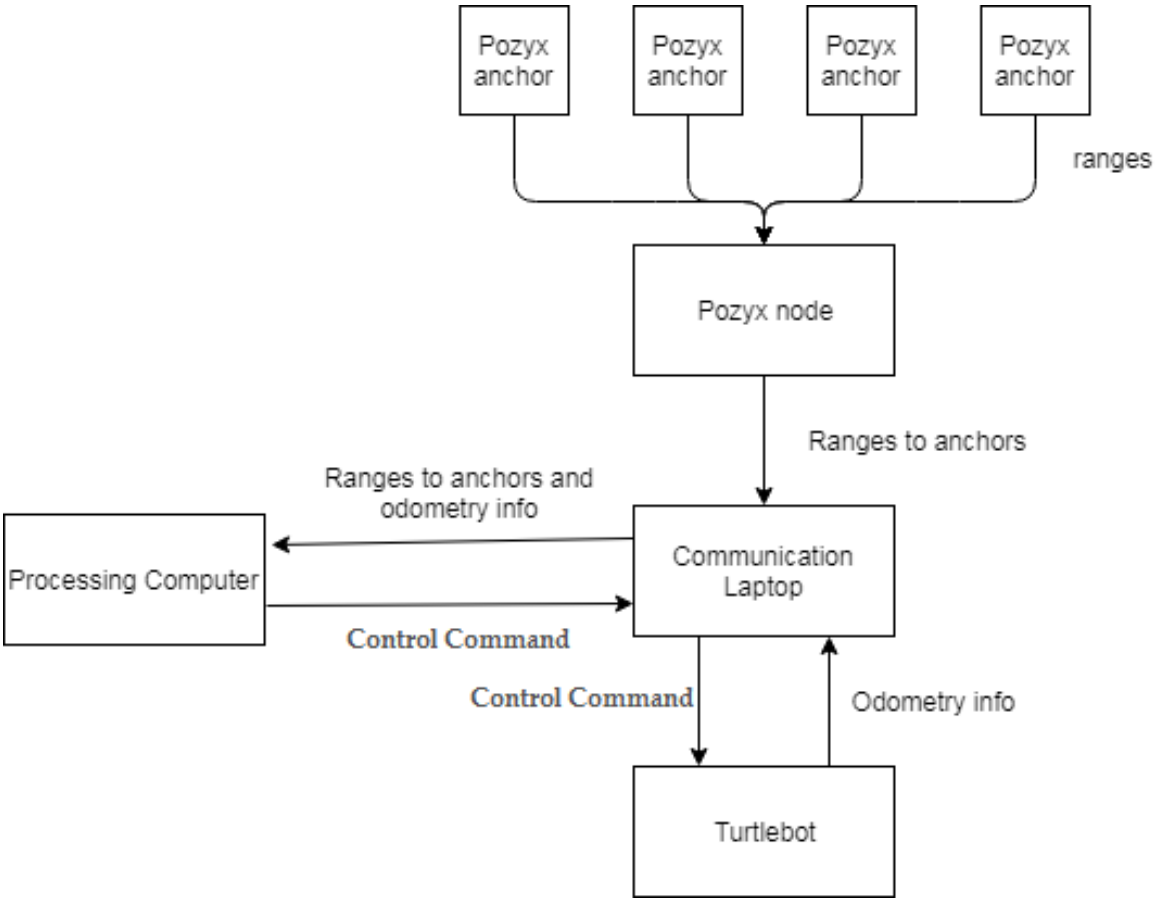

Figure 3.21: Communication between various devices throughout the experiment

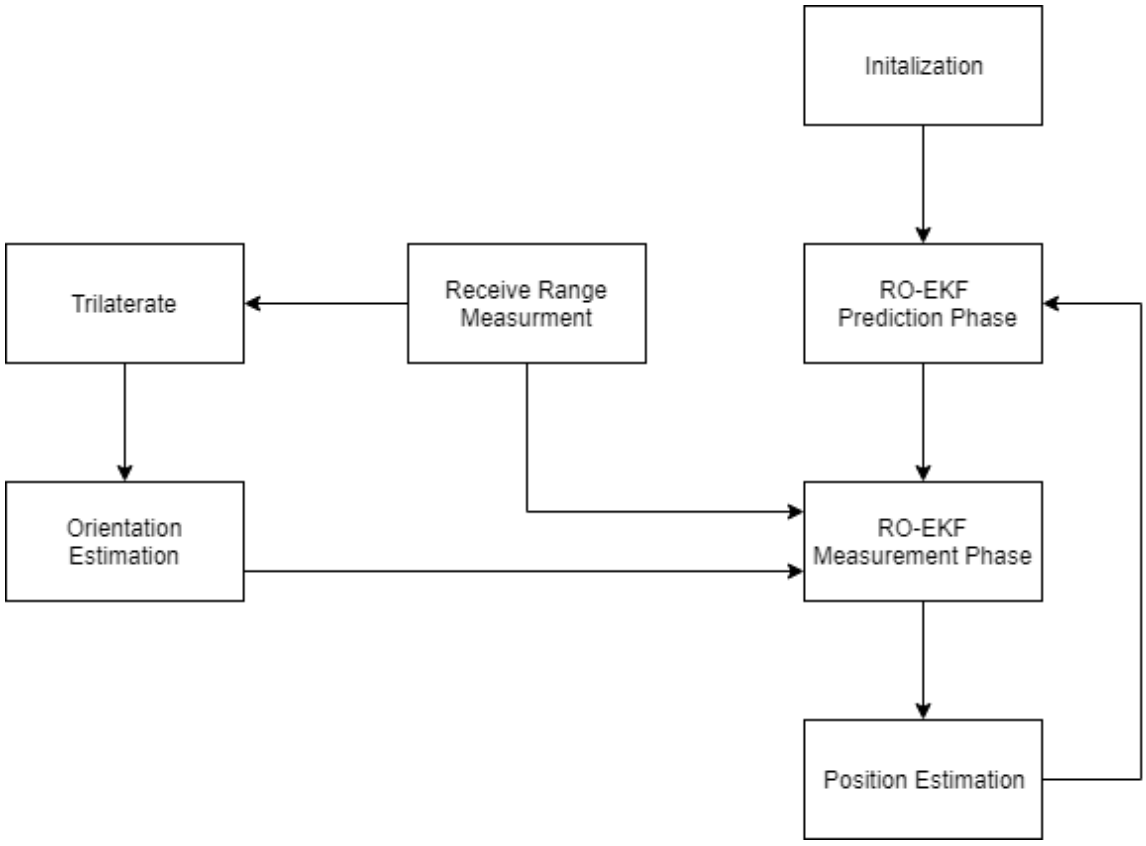

Figure 3.22: Flow chart of the modified RO-EKF algorithm

CHAPTER 4: RESULTS

4.1 Lab Environment Results

The results in Table 4.1 and 4.2 were recorded through four trials. During each trial the robot would navigate the environment as seen in Figure 3.5 by moving directly from one waypoint to the next in numeric order. The robot would start at position x=6.25 meters and y= 3.658 meters with an orientation of pi radians. The time during which the robot crossed a waypoint would be recorded and used to compare with the estimated position of the robot at that same moment in time. The final distance between the actual and estimated landmark position for each method was recorded in Table 4.2. For each trial 75 of the most recent trilateration data samples were used to estimate the orientation of the robot.

Table 4.1: The mean of the Euclidean distance between the actual robot position and the robot position estimated by the modified RO-EKF method, EKF method, and odometry in the laboratory environment at each waypoint. Also referenced to as the position error.

Trial Num.	Modified RO-EKF	RO-EKF	Odometry
1	0.694 m	0.713 m	1.772 m
2	0.756 m	0.792 m	1.674 m
3	0.803 m	0.866 m	1.759 m
4	0.635 m	0.682 m	1.242 m

Table 4.2: The mean of the Euclidean distance between the actual anchor positions and the anchor positions estimated by the modified RO-EKF method and EKF method in the laboratory environment.

Trial Num.	Anchor: 6937		Anchor: 6940		Anchor: 6935		Anchor: 6955	
	Modified RO-EKF	RO-EKF	Modified RO-EKF	EKF	Modified RO-EKF	RO-EKF	Modified RO-EKF	RO-EKF
1	0.4513 m	0.8034 m	0.2910 m	0.2848 m	0.4874 m	0.7195 m	0.1369 m	0.3341 m
2	0.5775 m	0.8341 m	0.2640 m	0.3598 m	0.2893 m	0.4511 m	0.1734 m	0.4029 m
3	0.5775 m	0.8341 m	0.2640 m	0.3596 m	0.2893 m	0.4511 m	0.1734 m	0.4029 m
4	0.5127 m	0.3711 m	0.3177 m	0.0911 m	0.2293 m	0.3801 m	0.2428 m	0.3566 m

4.2 Atrium Environment Results

The results in Table 4.3 and 4.4 were recorded through three trials. During each trial the robot would navigate the environment as seen in Figure 3.13 by moving directly from one waypoint to the next in numeric order. The robot would start at position x=1.219 meters and y= 0.000 meters with an orientation of 0 radians. The time during which the robot crossed a waypoint would be recorded and used to compare with the estimated position of the robot at that same time as seen in Table 4.3. The final distance between the actual and estimated landmark position for each method was recorded in Table 4.4. For each trial 75 of the most recent trilateration data samples were used to estimate the heading of the robot.

Table 4.3: The mean of the Euclidean distance between the actual robot position and the robot position estimated by the modified RO-EKF method, EKF method, and odometry in the atrium environment at each waypoint in meters. Also referenced to as the position error.

Trial Num.	Modified RO-EKF	RO-EKF	Odometry
1	0.327 m	0.257 m	0.802 m
2	0.656 m	0.521 m	0.851 m
3	0.482 m	0.430 m	0.187 m

Table 4.4: The mean of the Euclidean distance between the actual anchor positions and the anchor positions estimated by the modified RO-EKF method and EKF method in the atrium environment.

Trial Num.	Anchor: 6937		Anchor: 6940		Anchor: 6935		Anchor: 6955	
	Modified RO-EKF	RO-EKF	Modified RO-EKF	RO-EKF	Modified RO-EKF	RO-EKF	Modified RO-EKF	RO-EKF
1	0.0739 m	0.1965 m	0.0430 m	0.1734 m	0.1490 m	0.2764 m	0.1630 m	0.2796 m
2	0.1364 m	0.4396 m	0.5875 m	0.9540 m	0.2304 m	0.4348 m	0.2913 m	0.8839 m
3	0.5553 m	0.4252 m	0.3073 m	0.1924 m	0.3545 m	0.3473 m	0.2483 m	0.2459 m

CHAPTER 5: CONCLUSIONS

The work presented has led to the conclusion discussed in Section 5.1 and inspired future research objectives as discussed in Section 8.2.

5.1 Conclusion

A key distinction between SLAM and RO-SLAM is the fact that with pure RO-SLAM the landmarks utilized to build the map and estimate the robots position only provide range measurements. In particular, the measurement vector presented by RO-EKF only contains the range measurement to the landmark. The goal of this work was to improve the performance of the RO-EKF algorithm when solving RO-SLAM by inferring the orientation of the robot from a set of range measurements received, and then applying that to RO-EKF. The orientation was inferred by using trilateration to determine the robots position from the range measurements produced by pozyx devices. During the time that these range measurements were received the robot was required to function as a nonholonomic robot with a positive linear velocity at all times. Once a set of trilaterated positions were obtained a least squares approximation with a second degree polynomial was used to estimate the trajectory of the robot in time. Utilizing the most recent points of the trajectory estimation the orientation or the robot was estimated. The orientation was then used in conjunction with the ranging measurement as a single measurement to run the measurment phase of the RO-EKF algoirhtm. The algorithm utilizing the orientation estimation was termed modified RO-EKF. Modified RO-EKF was then run simultaneously with traditional RO-EKF in a laboratory and atrium environment over several trials.

The results of the experiment in the laboratory environment show that the addi-

tion of the orientation measurement estimation technique resulted in an increase of performance over the method that did not utilize the orientation measurement estimation technique. However, the increases in performance of the modified RO-EKF method over the traditional RO-EKF was slight. Over the four trails the modified RO-EKF method estimated the robots position to be an average of 0.04125 meters closer to the actual position of the robot than what traditional RO-EKF estimated. The modified RO-EKF method was seen to estimate with an average 2.6471 percent more accuracy than traditional RO-EKF in the laboratory environment. The results from individual trials are tabulated in Table 5.1.

Table 5.1: The percent difference between the mean odometry position estimation error and the mean Modified RO-EKF/RO-EKF position estimation error in the trails conducted in the lab.

Trial Num.	Modified RO-EKF	RO-EKF
1	60.8352 %	59.7630 %
2	54.8387 %	52.6882 %
3	54.3491 %	50.7675 %
4	48.8728 %	45.0886 %

Table 5.2: The difference in meters between the mean odometry position estimation error and the mean Modified RO-EKF/RO-EKF position estimation error in the trails conducted in the lab.

Trial Num.	Modified RO-EKF	RO-EKF
1	1.078 m	1.059 m
2	0.918 m	0.882 m
3	0.956 m	0.893 m
4	0.607 m	0.560 m

The results of the experiment in the atrium environment show that the inclusion of the orientation measurement estimation technique was a detriment to the method that utilized said technique. In the atrium environment the modified RO-EKF method had an average decrease in performance of 17.4665 percent over the three trails as compared to traditional RO-EKF. In meters the accuracy of the modified RO-EKF method was an average 0.08567 meters less accurate than traditional. The details of this can be seen in Table 5.3.

Table 5.3: The percent difference between the mean odometry position estimation error and the mean Modified RO-EKF/RO-EKF position estimation error in the trails conducted in the atrium.

Trial Num.	Modified RO-EKF	RO-EKF
1	59.2269 %	67.9551 %
2	22.9142 %	38.7779 %
3	-1.5775 %	-1.2994 %

Table 5.4: The difference in meters between the mean odometry position estimation error and the mean Modified RO-EKF/RO-EKF position estimation error in the trails conducted in the atrium.

Trial Num.	Modified RO-EKF	RO-EKF
1	0.475 m	0.545 m
2	0.195 m	0.333 m
3	-0.295 m	-0.243 m

The primary difference between the two environments where these experiments were conducted was the presence, or lack of presence, of multipath noise. Under conditions of high multipath the results show that the orientation measurement estimation technique increased the performance of RO-EKF. An explanation for this

result is that in order for the RO-EKF to function optimally the noise present in measurements should be Gaussian distributed with a mean of zero. In the presence of high multipath, such as in the lab environment as demonstrated by Table 3.3, the performance of RO-EKF decreases as the noise is not a mean zero Gaussian distribution. In this environment however, the orientation estimated from the trilaterated positions must have had a noise that was closer to a mean zero Gaussian than that of the range measurements. Thus, resulting in the slight increase in performance. This reasoning holds with the atrium environment as it is seen that the range measurement noise in said environment is much more consistently close to a mean zero Gaussian.

5.2 Future Work

One of the factors observed while conducting this research was that of how the number of samples affected the results of the orientation measurement estimation. The focus of the research was not to perfect this orientation measurement estimation, but merely to see if such a technique could increase performance, thus the number of samples was chosen at a value that produced consistent results. However, in the future it would be very interesting to explore if the number of samples could be dynamically set based upon external factors in order to optimize the estimation. For example, the control information could be examined and based upon the speed of the robot the number of samples used for trilateration could be increased or deceased. Or perhaps even based upon the methods, such as fingerprinting, the samples could be changed according the the amount of interference in an environment.[26]

REFERENCES

[1] Y. Turchin, A. Gal, and S. Wasserkrug, "Tuning complex event processing rules using the prediction-correction paradigm," *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems - DEBS '09*, no. January 2009, p. 1, 2009.

[2] S. Riisgaard and M. R. Blas, "SLAM for Dummies: A tutorial approach to Simultaneous Localization and Mapping by the 'dummies'," 2003.

[3] S. Shue, *Utilization of Wirless Signal Strength for Mobile Robot Localization in Indoor Enviorments*. PhD thesis, University of North Carolina at Charlotte, 2017.

[4] S. Farahani, *ZigBee Wireless Networks and Transceivers*. 2008.

[5] "Calculus i - newton's method." `http://tutorial.math.lamar.edu/Classes/CalcI/NewtonsMethod.aspx`. (Accessed on 04/15/2018).

[6] D. Scaradozzi, S. Zingaretti, and A. Ferrari, "Simultaneous localization and mapping (SLAM) robotics techniques: a possible application in surgery," *Shanghai Chest*, vol. 2, no. 1, 2018.

[7] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT Press, 2010.

[8] J. Aulinas, Y. Petillot, J. Salvi, and X. Lladó, "The SLAM problem: A survey," in *Frontiers in Artificial Intelligence and Applications*, vol. 184, pp. 363–371, 2008.

[9] V. Chandrasekhar, W. K. Seah, Y. S. Choo, and H. V. Ee, "Localization in underwater sensor networks," *Proceedings of the 1st ACM international workshop on Underwater networks - WUWNet '06*, p. 33, 2006.

[10] R. Peng and M. Sichitiu, "Angle of Arrival Localization for Wireless Sensor Networks," *2006 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks*, vol. 1, no. C, pp. 374–382, 2006.

[11] G. Reina, L. Ojeda, A. Milella, and J. Borenstein, "Wheel slippage and sinkage detection for planetary rovers," *IEEE/ASME Transactions on Mechatronics*, vol. 11, no. 2, pp. 185–195, 2006.

[12] D. Puccinelli and M. Haenggi, "Multipath Fading in Wireless Sensor Networks: Measurements and Interpretation," *Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing*, pp. 1039–1044, 2006.

[13] R. A. Hewitt and J. A. Marshall, "Towards intensity-augmented SLAM with LiDAR and ToF sensors," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem, pp. 1956–1961, 2015.

[14] K. Çelik, S. J. Chung, and A. Somani, "Mono-vision corner SLAM for indoor navigation," *2008 IEEE International Conference on Electro/Information Technology, IEEE EIT 2008 Conference*, pp. 343–348, 2008.

[15] W. Burgard, C. Stachniss, G. Grisetti, B. Steder, R. Kümmerle, C. Dornhege, M. Ruhnke, A. Kleiner, and J. D. Tardós, "A comparison of SLAM algorithms based on a graph of relations," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pp. 2089–2095, 2009.

[16] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1, pp. I–652–I–659 Vol.1, 2004.

[17] J. A. Djugash, S. Singh, G. Kantor, W. Zhang, and J. Djugash, "Range-Only SLAM for Robots Operating Cooperatively with Sensor Networks," no. May, pp. 2078–2084, 2006.

[18] "Least squares fitting–polynomial – from wolfram mathworld." `http://mathworld.wolfram.com/LeastSquaresFittingPolynomial.html`. (Accessed on 04/21/2018).

[19] "Vandermonde matrix – from wolfram mathworld." `http://mathworld.wolfram.com/VandermondeMatrix.html`. (Accessed on 04/22/2018).

[20] W. S. Murphy Jr and H. Willy, *Determination of a Position in Three Dimensions Using Trilateration and Approximate Distances*. PhD thesis, Colorado School of Mines, 1999.

[21] F. Thomas and L. Ros, "Revisiting trilateration for robot localization," *IEEE Transactions on Robotics*, vol. 21, no. 1, pp. 93–101, 2005.

[22] F. C. Commission, "Revision of Part 15 of the Commission's Rules Regarding Ultra-Wideband Transmission Systems," *First Report and Order in ET ...*, no. FCC02-48, pp. 1–118, 2002.

[23] T. K. K. Tsang and M. N. El-Gamal, "Ultra-wideband (UWB) communications systems: An overview," *3rd International IEEE Northeast Workshop on Circuits and Systems Conference, NEWCAS 2005*, pp. 381–386, 2005.

[24] S. Shue and J. M. Conrad, "A survey of robotic applications in wireless sensor networks," *2013 Proceedings of IEEE Southeastcon*, pp. 1–5, 2013.

[25] T. DeiBler and J. Thielecke, "UWB SLAM with rao-Blackwellized Monte Carlo data association," *2010 International Conference on Indoor Positioning and Indoor Navigation, IPIN 2010 - Conference Proceedings*, no. September, pp. 15–17, 2010.

[26] E. Olson, J. J. Leonard, and S. Teller, "Robust range-only beacon localization," *IEEE Journal of Oceanic Engineering*, vol. 31, no. 4, pp. 949–958, 2006.