10-5-2017

Implementation of an Autonomous Small-scale Car with Indoor Positioning using UWB and IMU

Alvin Marquez
University of Windsor

Implementation of an Autonomous Small-scale Car with Indoor Positioning using UWB and IMU

By

Alvin Marquez

A Thesis
Submitted to the Faculty of Graduate Studies
through the Department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Science
at the University of Windsor

Windsor, Ontario, Canada

2017

Implementation of an Autonomous Small-scale Car with Indoor Positioning using UWB and IMU

By

Alvin Marquez

Approved By:

J. Urbanic
Department of Mechanical, Automotive and Materials Engineering

---

M. Khalid
Department of Electrical and Computer Engineering

K. Tepe, Adviser
Department of Electrical and Computer Engineering

September 7^{th}, 2017

Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

Robotics have had a major impact in the current generation as they have a wide range of uses in manufacturing and automation; therefore, researching new technology related to robotics is currently at a high demand. Indoor robotics, such as automatic guided vehicles or humanoids, is a section of robotics that are mobile and need accurate positioning in order to navigate properly. Thus, research into indoor positioning systems (IPS) has become an interesting research topic to be able to provide a standard in indoor positioning.

This thesis tests an ultrawideband (UWB) based IPS and fuses the data from an inertial measurement unit (IMU) using an extended Kalman filter (EKF). The testing platform was implemented using Robot Operating System (ROS) and a Beaglebone Black as the microcontroller for the sensors. However, the main processing was done on a separate laptop. As a result, a proposed smoothing technique was able to provide consistent velocity commands to the vehicle platform without affecting the data output rate of the UWB based IPS. In line-of-sight (LOS) conditions and a travel length of about 13 m, the best results produced an error of only 0.111 m at the final point, and an error of up to 0.603 m during travel.

I dedicate my thesis to myself for wanting to finish my academic studies.

Acknowledgements

I would like to give special thanks to my supervisor, Dr. Kemal Tepe, for his support throughout the span of this research. I also appreciate the time and assistance of my committee members, Dr. Mohammed Khalid and Dr. Jill Urbanic.

Contents

List of Tables

List of Figures

Abbreviations

BBB	**B**eagle**b**one **B**lack
DC	**D**irect **C**urrent
EKF	**E**xtended **K**alman **F**ilter
GPS	**G**lobal **P**ositioning **S**ystem
IMU	**I**nertial **M**easurement **U**nit
I2C	**I**nter-**I**ntegrated **C**ircuit
IP	**I**nternet **P**rotocol
IPS	**I**ndoor **P**ositioning **S**ystem
LOS	**L**ine-**o**f-**s**ight
MEMS	**M**icro**e**lectro**m**echanical **System**
NLOS	**N**o **L**ine-**o**f-**s**ight
PC	**P**ersonal **C**omputer
ROS	**R**obot **O**perating **S**ystem
SLAM	**S**imultaneous **L**ocalization **a**nd **M**apping
UKF	**U**nscented **K**alman **F**ilter
URI	**U**niform **R**esource **I**dentifier
USB	**U**niversal **S**erial **B**us
UWB	**U**ltra**w**ide**b**and
WLAN	**W**ireless **L**ocal **A**rea **Network**

Chapter 1

Introduction

1.1 Motivation

The purpose of this thesis is to continue research regarding the use of Ultrawideband (UWB) as an Indoor Positioning System (IPS). In [1], a study was done to mitigate NLOS using geometry based methods. However, the experimental calculations were done offline, which means that data was recorded beforehand and simulations were run on MATLAB afterwards. Thus, it could not conclude whether the use of an UWB based IPS could be applicable for real-time applications such as indoor robotics or vehicles in parking garages.

Accordingly, the implementation of a small-scale vehicle platform should help determine whether an UWB based IPS is viable for indoor robotics. The calculations should be done in real-time, which means that the data will be processed as it arrives compared to offline calculations from a previously recorded test. As a start, the main goal is to be able to autonomously guide a vehicle through a known map by using UWB with the help of an Inertial Measurement Unit (IMU).

Moreover, autonomous vehicles have become a major research topic for automotive and technology companies. As such, the race to achieve the first vehicle with Level 5 autonomy is crucial. The leader in self-driving technology will definitely have a large impact in the automotive industry, shaping how future vehicles will be

designed and created. Therefore, a small-scale vehicle platform allows for low-cost testing of autonomous vehicle technologies, which can then be scaled to actual autonomous vehicles in the future. Google Trends shows that the 'Autonomous car' topic reached its peak interest around September 2016 over the last five years as seen in Fig. 1.1 [2]. Accordingly, this research hopes to help further research related to autonomous vehicle technology.

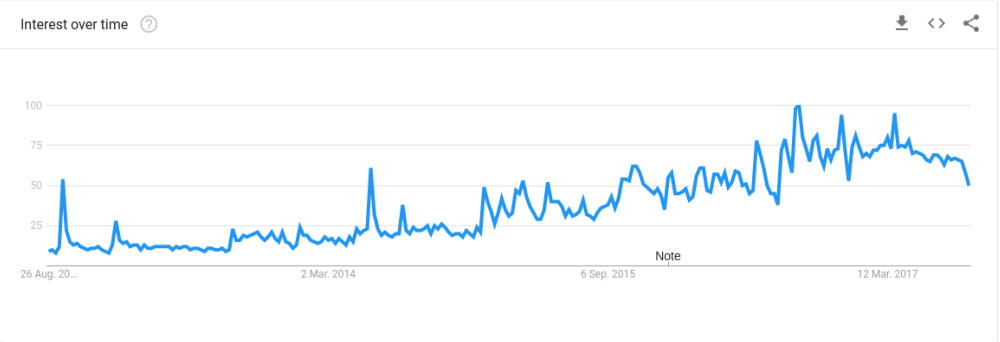


FIGURE 1.1: Google Trends results over a five year time period for Autonomous Car.

1.2 Problem statement

Currently, there is no standard wireless IPS that is used for reliably estimating the absolute position of an object or person. Some of the problems with IPS is having to deal with the harsh and constantly changing indoor environments. This includes people, furniture or electronics that can interfere with the wireless signals transmitted. Additionally, the continuous evolution of technology in the recent years has made robotics become an interesting topic of research due to being able to perform repeated tasks reliably. However, indoor robotics that are designed to autonomously navigate themselves require a reliable positioning system in order to reach their desired destination. As such, a wireless IPS will be needed to provide the absolute position of a robot before being able to navigate. The applications of an IPS can also scale to guiding vehicles inside a parking garage as Global Positioning Systems (GPS) lose reliability when vehicles are inside concrete parking garages.

1.3 Thesis Contribution and Limitation

This thesis includes the design and implementation of a small vehicle platform using UWB, IMU, Robot Operating System (ROS), and Extended Kalman Filter (EKF) to provide a testing scenario for an indoor robot with positioning. The goal of this thesis is to physically test the viability of using an UWB based IPS for indoor robotics. However, this thesis does not include the analysis of other hardware used for wireless IPS and does not include bias error correction when there is NLOS for the UWB tags and anchors, which is studied in [1] and [3].

1.4 Thesis Outline

There is a total of six chapter for this thesis. Chapter 1 introduces the motivation of the research topic and its possible contribution to ongoing and future technology. Then, Chapter 2 explains the sensors and software used for position estimation, as well as related research works. Moreover, Chapter 3 describes the components selected and the implementation of the vehicle platform. Next, Chapter 4 contains the methods and processing done to improve the positioning. Furthermore, Chapter 5 shows the test results and evaluates the data before and after sensor fusion. Lastly, Chapter 6 provides a conclusion for the experiment and recommendations for future work.

Chapter 2

Background

2.1 Indoor Positioning System

2.1.1 Infrastructure vs. Infrastructure Free

IPS can be categorized into either one of the following categories: infrastructure based positioning or infrastructure free positioning. The main difference between these is that infrastructure based positioning requires additional setup of hardware before being able to provide an accurate positioning system. As a result, infrastructure based systems can lead to an increased initial costs compared to infrastructure free. Some examples of infrastructure based positioning systems include Real-Time Kinematic GPS, Zigbee, and UWB. On the other hand, infrastructure free systems require no additional setup and these systems can often be deployed immediately. Wifi can be considered as an infrastructure free positioning system as Wifi is most likely to be found in a vast majority of buildings in North America.

2.1.2 Ultrawideband

UWB operates in a wide frequency range of $3.1 - 10.6$ GHz; thus, it allows for flexible operation when other known wireless devices are operating nearby such as Wifi. In this experiment, UWB positioning is done by using two-way ranging of time of arrival. Two-way ranging means that when the tag sends a signal to the anchor, another signal is sent back to the tag to determine the total travel time. This method only requires that time stamps be recorded on the UWB tag and does not need to be synchronized between the other anchors. Other methods of positioning is further explained in past works by Mati [1].

After determining the distance between the three anchors and tag using two-way ranging, trilateration can then be applied to produce an estimated absolute position. Trilateration in this case operates by using the points of intersection of the three spheres, where the radius is equivalent to the received distance. Ideally, this should give exactly two points of intersection since the vehicle platform is on the ground, while the three anchors are elevated. In Fig. 2.1, the two points of intersection would be one going into the page, and another one going out of the page with equivalent distance from the center. However, only the 2D coordinates are needed for this case since the vehicle platform will only be navigating on a 2D map. If the height component is needed for other work, a fourth anchor will be required to determine the proper height.

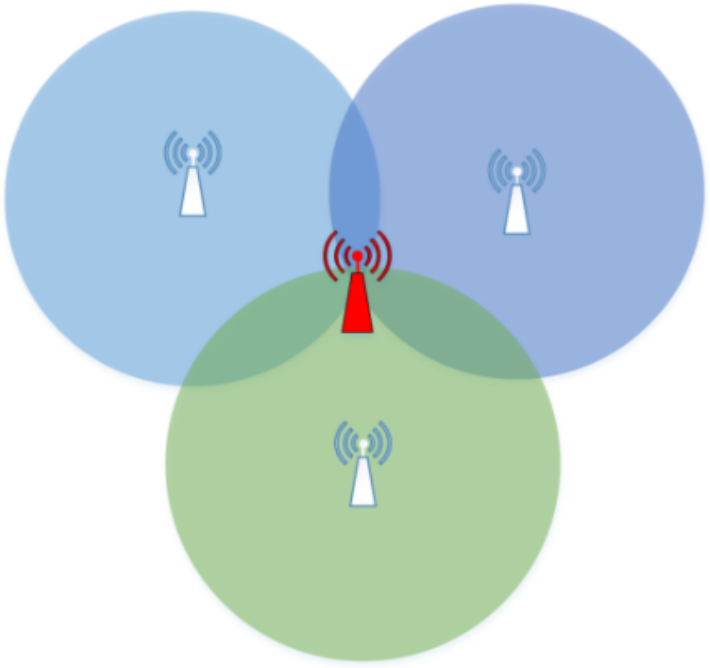


FIGURE 2.1: An ideal example diagram of trilateration for 2D positioning.

2.2 Inertial Measurement Unit

Nowadays, IMUs can come as small as the size of a Canadian quarter and are referred to as microelectromechanical systems (MEMS). These MEMS can operate with minimal power requirements and can fit into many other devices such as cellphones, vehicles, and video game controllers. Cellphones can then be used for tracking user movements such as steps, vehicle safety systems can respond to sudden stops or movements, and video game controllers can use the IMU as another way of controlling game movement. Thus, their ability to operate with fast response times and accurate measurements make IMUs useful for this experiment.

2.2.1 Accelerometer

One of the measured values from the IMU is acceleration and measured in g, the gravity vector. Originally, a stationary IMU should always have a reading equivalent to the earth's gravitational pull of about 9.8 m/s. Fortunately, sensor

fusion allows some IMUs to filter out the gravitational pull and only output the linear acceleration that is needed for typical vehicle movements. More details about the chosen IMU and its sensor fusion is discussed in Chapter 3.

2.2.2 Magnetometer and Gyroscope

The other two measured values from the IMU are magnetic field in μT and angular velocity in deg/s. The magnetometer responsible for the magnetic field detects the magnetic poles of the Earth, and other magnetic objects nearby. However, it is best to avoid changing the field around the magnetometer after calibration is done to produce the best results. Ideally, if the magnetometer detects only the magnetic poles of the Earth and is calibrated properly, it will provide the most accurate readings of the true North direction. On the other hand, the gyroscope responsible for the angular velocity measure the rotation about an axis. A representation of gyroscope readings are useful for an airplane in flight as the airplane would need to know its proper Euler angles to fly properly. In this case, the vehicle platform only requires the rotation about one axis. However, the Euler angles can only be as accurate as its original position as the angular velocity can only provide relative orientation and not absolute orientation. More details regarding the magnetometer and gyroscope can be found in Chapter 3.

2.3 Kalman Filter

The Kalman filter is a filtering technique commonly used for multiple sensors, containing some noise or known variance, and fusing them together in an attempt to achieve a more accurate measurement than one sensor alone. The noise is usually represented as process noise, such as wind or something cause by the sensor's surrounding during operation, and known variance can be represented as the measuring accuracy of the sensor given by its datasheet. One of its common uses is for guidance and positioning of vehicles by fusing its location information,

GPS or IPS in this case, with another sensor that can produce a relative location, such as an IMU. The main requirements for the Kalman filter are that the sensor measurements should have a Gaussian distribution, be a linear system, and must be compared in the same measurement domain. Another feature that makes the Kalman filter popular is that it is recursive, requiring only the last state and present set of data measurements for its input, and allows it to be computed in real-time [4].

2.3.1 Extended Kalman Filter

As mentioned previously, the Kalman filter is great for linear systems and measurements in the same domain, but cannot properly predict nonlinear systems without some modifications. As a result, the extended and unscented Kalman filters were designed to deal with nonlinear systems. The UKF specializes in constant changes where linearizing the data would cause worse prediction, such as differential steering when driving. If the data was to be linearized, the tangent of the curves would not be a good representation of the actual turn. On the other hand, the EKF specializes in linearizing the data and works best for the maneuverability of the vehicle platform for this experiment. More details about the vehicle platform is presented in Section 3.1.5.

Consequently, the EKF is used for this experiment and the acceleration received from the IMU will be linearized and then fused with the positioning values received from the UWB based IPS. Balzer provides a detailed implementation of the Kalman filter using 2D acceleration and 2D position as input [5]. As such, Balzer's implementation is as follows.

First, the Kalman filter can be split into the prediction phase and the correction phase. The prediction phase requires the state vector matrix to represent what is being tracked, and the error covariance from the initial state uncertainty and process noise covariance. Accordingly, the prediction phase can seen in Eq. 2.1 and Eq. 2.2.

Projecting the state ahead.

$$x_{k+1} = Ax_k + Bu_k \tag{2.1}$$

Projecting the error covariance ahead.

$$P_{k+1} = AP_kA^T + Q \tag{2.2}$$

The state vector of the system is represented in Eq. 2.3.

$$x_k = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ \ddot{x} \\ \ddot{y} \end{bmatrix} \tag{2.3}$$

where:

- x: is the x-component of position
- y: is the y-component of position
- $\dot{x}$: is the x-component of velocity
- $\dot{y}$: is the y-component of velocity
- $\ddot{x}$: is the x-component of acceleration
- $\ddot{y}$: is the y-component of acceleration

Next, the formal definition for motion is given by Eq. 2.4 with the dynamics of the Egomotion, A and Eq. 2.5 after removing the control input, u.

$$x_{k+1} = A \cdot x_k + B \cdot u \tag{2.4}$$

$$x_{k+1} = \begin{bmatrix} 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} xy\dot{x}\dot{y}\ddot{x}\ddot{y} \end{bmatrix}_k \tag{2.5}$$

Now, the measurement matrices based on acceleration from the IMU and position from the UWB based IPS are represented by Eq. 2.6 and Eq. 2.7.

$$y = H \cdot x \tag{2.6}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot x \tag{2.7}$$

where:

x: is the state

H: is the measurement

Afterwards, the initial state uncertainty, P, measurement covariance, R and process noise covariance, Q can be modeled by Eq. 2.8, Eq. 2.9, and Eq. 2.10. The matrices are simplified with 0 on the non diagonals as the sensors used in this case have independent variances. The measurement matrix, R, was also simplified to only four rows since only the x-component and y-component of acceleration and position are measured.

$$P = \begin{bmatrix} px & 0 & 0 & 0 & 0 & 0 \\ 0 & py & 0 & 0 & 0 & 0 \\ 0 & 0 & p\dot{x} & 0 & 0 & 0 \\ 0 & 0 & 0 & p\dot{y} & 0 & 0 \\ 0 & 0 & 0 & 0 & p\ddot{x} & 0 \\ 0 & 0 & 0 & 0 & 0 & p\ddot{y} \end{bmatrix} \tag{2.8}$$

$$R = \begin{bmatrix} rx & 0 & 0 & 0 & 0 & 0 \\ 0 & ry & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & r\ddot{x} & 0 \\ 0 & 0 & 0 & 0 & 0 & r\ddot{y} \end{bmatrix} \tag{2.9}$$

$$Q = \begin{bmatrix} px & 0 & 0 & 0 & 0 & 0 \\ 0 & py & 0 & 0 & 0 & 0 \\ 0 & 0 & p\dot{x} & 0 & 0 & 0 \\ 0 & 0 & 0 & p\dot{y} & 0 & 0 \\ 0 & 0 & 0 & 0 & p\ddot{x} & 0 \\ 0 & 0 & 0 & 0 & 0 & p\ddot{y} \end{bmatrix} \tag{2.10}$$

The Kalman filter can now enter the correction phase after measurements and their corresponding covariance matrix, R, are received. Finally, the correction phase can be seen in Eq. 2.11, Eq. 2.12, and Eq. 2.13.

Computing the Kalman Gain.

$$K_k = P_k H^T (H P_k H^T + R)^{-1} \tag{2.11}$$

Updating the estimate via measurement using z_k.

$$x_k = x_k + K_k(z_k - H x_k) \tag{2.12}$$

Updating the error variance with measurement matrix, H, and identity matrix, I.

$$P_k = (I - K_k H)P_k \tag{2.13}$$

2.4 Robot Operating System

ROS is an open source meta-operating system that runs on Unix operating systems, such as Linux Ubuntu. Similar to a regular operating system, it can provides low-level device control and message-passing between processes, but it is not designed to be the main operating system of a device. Since ROS is open source, it includes many online repositories from other organizations or community of developers, including industrial applications [6]. As described in [7], "ROS runtime graph is a peer-to-peer network of processes (potentially distributed across machines) that are loosely coupled using the ROS communication infrastructure. ROS implements several different styles of communication, including synchronous RPC-style communication over services, asynchronous streaming of data over topics, and storage of data on a Parameter Server."

2.4.1 ROS Nodes, Topics, and Messages

ROS nodes are processes that usually does some computation and are able to communicate with each other. Some examples for this experiment are one node responsible for the IMU data, another node responsible for the UWB data, and another node responsible for processing both sensor data. They can communicate through the use of topics, RPC services, or a parameter server. In order for nodes to communicate, a unique topic, service, or server name must be assigned by the node. Afterwards, any node in the system can publish or subscribe to that topic name. Publishing to a topic is similar to sending data to that topic name, while subscribing is similar to receiving the data. Each topic must be assigned a proper message structure, either custom built or one of the ROS messages. For example, there is an already built ROS message for IMU that contains linear acceleration,

angular velocity, and orientation data. It is important to note that the node publishing and subscribing to the topic must use the same message structure.

2.4.2 Navigation Stack

The navigation stack is a collection of ROS packages that receives input data from certain sensors and outputs safe velocity commands to the vehicle [8]. A ROS package is a method used by ROS in bundling software; thus, it can contain multiple ROS nodes, a ROS-independent library, or other software of similar use. These sensors include, but are not limited to, lidar, laser rangefinders, and cameras. Furthermore, the navigation stack contains multiple ROS packages that can be used together to achieve full autonomy. For example, planner packages such as base_local_planner, dwa_local_planner, and global_planner can be used for pathfinding. The package map_server will handle updates made to all maps, and move_base will handle movement of the robot. Fig. 2.2 shows an overview diagram of how the navigation stack works [7].

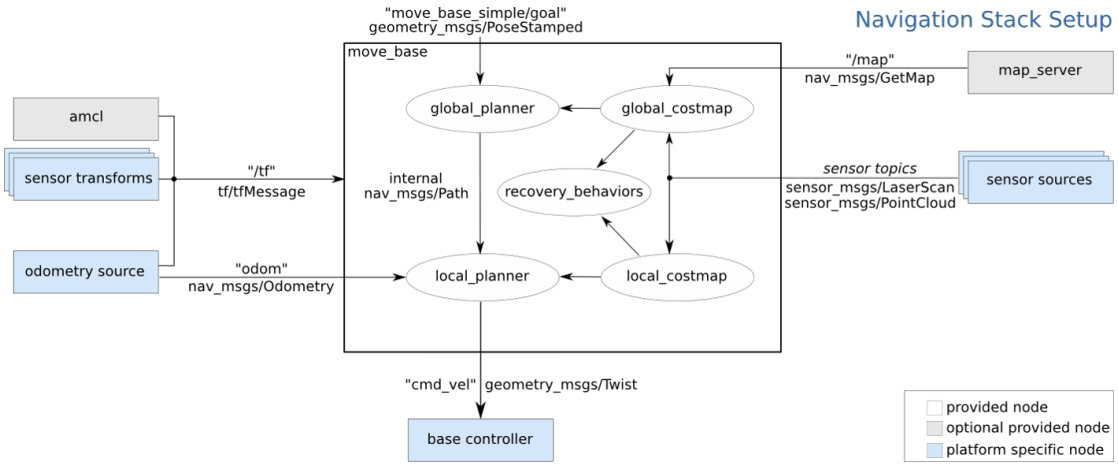


FIGURE 2.2: An overview diagram of ROS topics and nodes for the navigation stack.

2.5 Related Works

Some related works regarding the use of indoor positioning include industrial uses for asset tracking. For example, Zebra Technologies uses UWB for asset tracking

in manufacturing or warehouse management environments [9]. Ubisense is using a system called 'Smart Factory' that helps manufacturers optimize efficiency with the use of asset tracking [10]. Another industrial use is for navigating a robot for painting a floor layout as mentioned in [11].

Similar indoor positioning systems include an UWB and MEMS based indoor navigation for pedestrians. Through the use of multiple observation data such as angles of arrival, time differences of arrival, accelerations, angular velocities, and magnetic fields, Renaudin, Merminod, and Kasser were able to produce at best position errors below 1 meter with a 0.1 meter variance [12]. Similarly, Evennou and Marx studied an advanced integration of Wifi and inertial navigation systems to produce a mean error of 2.56 meter using a Kalman filter, and mean error of 1.53 meter using a particle filter with an inertial navigation system [13]. Additionally, Yuan, Xiyuan, and Qinghua used an iterated EKF in forward data processing of the Extended Rauch-Tung-Striebel smoothing to produce a position error of 3.50 -3.73cm [14]. Likewise, Jun, Guensler, and Ogle studied different methods of GPS distance smoothing and concluded that the Kalman filter and their modified Kalman filter produced better results than least squares spline approximation and a Kernel-based smoothing method [15]. Yavari also studied UWB positioning with LOS and NLOS conditions and produced at best an average positioning accuracy of 10.97 cm and 51.96 cm, respectively [3].

ROS has also been helpful in research topics regarding indoor robotics. Foote created the ROS tf library that allows the user to keep track of multiple coordinate frames over time [16]. Furthermore, Marder-Eppstein et al. designed a robust navigation system in a typical indoor office environment that had a robot navigate itself for 26.2 miles without human intervention [17]. In addition, Garimort, Hornung, and Bennewitz also designed navigation for a humanoid with dynamic footstep plans [18]. This paper also accounts for the ability for humanoid robots to step over certain obstacles compared to wheeled robots that would have to navigate around.

Chapter 3

Platform Hardware and Software Implementation

3.1 Component Selection and Evaluation

This section explains why specific hardware and software components were used for the experiment. As mentioned in Section 1.1, the motivation of this project is to continue working on the Decawave UWB evaluation kit and its possible applications as an accurate indoor positioning system. Accordingly, the Decawave TREK1000 evaluation kit will be used for the positioning system and can be seen in Fig. 3.1. Comparisons with other wireless positioning systems are further discussed in [1]. Next, the software and programming language to be used was the first major decision to be made.

FIGURE 3.1: Decawave TREK1000 evaluation kit used as 3 anchors and 1 tag configuration.

3.1.1 Software and Programming Language

Originally, the plan was to just write the entire project using a single Python script since Python is easily accessible, large community for troubleshooting, and large support for drivers and other libraries. Unfortunately, this could lead to some bottleneck issues depending on the computer's processor that is being used and may not be as reliable when running real-time calculations. Fortunately, ROS was found to be compatible with both C++ and Python language and allows for multiple processes, or Python scripts, to run simultaneously and communicate with each other. ROS communicates by sending structured messages using ROS nodes and topics as mentioned in Section 2.4. Additionally, ROS already includes built in sensor messages such as Odometry for positioning and IMU message for IMUs. This was the major deciding factor for utilizing ROS for this platform. Finally, ROS Indigo was chosen for being the latest long term support version when starting this project.

3.1.2 Master Computer

The master computer for this experiment does not have very high requirements besides being compatible with Ubuntu 14 and ROS Indigo. Accordingly, a Lenovo Yoga 2 Pro model was used with an Intel Core i5-4210 CPU @ 1.70 GHz x 4 processor with integrated graphics. It runs Ubuntu 14 64-bit with 8GB of RAM. This computer was chosen as it was already owned, acts as a minimal requirement for specifications, and the project did not need to do any known heavy computing.

3.1.3 Microcontroller

The microcontroller to be used for this experiment required it to be compatible with Debian 8, have I2C bus lines, Wi-fi connectivity, USB port, and bonus for having a readily available motor controller (this avoids creating a custom board and leads to better scalability as it keeps all the header pins available). The operating system must be Debian 8 as it is needed for the installation of ROS Indigo, as chosen earlier in Section 3.1.1. Additionally, the I2C bus lines is for the IMU or other future sensors, and the USB port is for the Decawave UWB tag, Wi-fi adapter, or other future sensors. The two major candidates for the microcontroller to be used were the Beaglebone Black or the Raspberry Pi 2 Model B as they are both known to be compatible with running Debian.

The comparison of the technical specifications of the BBB and the Raspberry Pi 2 Model B can be seen in Table 3.1. The processing power of both microcontrollers were relatively similar with a difference of only 100 MHz, so it was a major deciding factor. However, the BBB only had half the RAM of the Raspberry Pi 2, but this can be compensated by using a portion of the microSD card as a swap space to also give it a RAM of 1GB. Furthermore, the BBB model had no integrated WLAN adapter, but could also be compensated by using a low cost Wi-fi dongle. It was also found that both microcontrollers are capable of running a Debian operating system and both microcontrollers had support for a motor controller that can handle four separate DC motors. The most significant factor in deciding the

microcontroller came down to the pin count and experience. The larger GPIO pin count for the microcontroller ensured that the platform would be easily scalable with multiple sensors for future use. Based on experience from past projects and peers, the BBB seemed adequate enough for the experiment. As a result, the BBB was chosen as the microcontroller to be used.

Table 3.1: BBB vs Raspberry Pi 2

	Beaglebone Black	**Raspberry Pi 2 Model B**
Price	$ 84.31	$ 48.99
Processor	1 GHz	900 MHz
RAM	512 MB	1 GB
Wireless LAN	Not included	Integrated 802.11n
Power Source	5V @ 2A DC Jack	5V Micro-USB @ 2A
Pin count	92 (65 GPIO)	40 (26 GPIO)
Operating System	Debian compatible	Debian compatible
Motor Controller	Seeed Motor Bridge Cape	Adafruit DC Motor Hat

3.1.4 Inertial Measurement Unit

The IMU requirements for this experiment were to give an accurate measurement of linear acceleration, magnetic field, and relative angular velocity. The linear acceleration will be used in the EKF for predicting the relative position of the vehicle, while the magnetic field and relative angular velocities will be used together in calculating the current heading of the vehicle. There were three IMUs that were studied for this experiment: Xsens MTi 10-series [19], Adafruit 9-DOF IMU Breakout [20], and Adafruit BNO055 Absolute Orientation Sensor [21].

Table 3.2 shows a comparison of some specifications of each IMU. The compatible output interfaces with the chosen microcontroller were USB or I2C and all three IMUs were able to supply either interface. In addition, the input voltages for all three IMUs could be provided by the BBB as well. Although the ranges for the accelerometer and gyroscope values differ between the three IMUs, each IMU can provide an acceptable operating range, (< 5 g for linear acceleration, and $<$ 450 dps for angular velocity). Likewise, the output frequencies for the three IMUs

seemed sufficient enough for the project, (< 100 Hz should be able to provide real-time measurements). Unfortunately, the Adafruit 9-DOF IMU does not contain its own sensor fusion output compared to the other two IMUs. Also, the Xsens MTi-10 series lacks magnetic field readings and has a very steep price compared to the two Adafruit IMUs. The magnetic field readings are needed in order for the vehicle platform to determine its true North based on the magnetic poles. As a result, the Adafruit BNO055 Absolute Orientation Sensor was the chosen IMU in terms of performance and price compromise.

Table 3.2: Inertial Measurement Units Comparisons

	Adafruit 9-DOF IMU	**Adafruit BNO055**	**Xsens MTi-10**
Price	$ 19.95	$ 47.23	$ 1214.85
Output Type	I2C	I2C	UART, USB
Input Voltage	3V3 or 5V	3V3 or 5V	3V3 or 4.5-34V
Accelerometer Range	±2/±4/±8/±16 g	±2/±4/±8/±16 g	∼ ±5/±15 g
Gyroscope Range	±250/±500/±2000 dps	±125 to 2000 dps	±450/±1000 dps
Magnetic Field	Yes	Yes	Not available
Output Frequency	Up to 400 Hz	100 Hz	Up to 2000 Hz
Sensor Fusion	No	Yes	Yes

3.1.5 Vehicle Platform

The vehicle platform was assembled using aluminum parts as its chassis with four separate DC motors to control each wheel. By being able to control the motors separately, the vehicle platform will be able to rotate in place by having half of the motors spin forward and the other half spin backward. Each DC motor was rated at 0.263 N.m with a current of 0.52 A. Table 3.4 shows the other specifications for the vehicle platform. Additionally, plywood was used as platforms to hold the microcontroller, USB hub, and UWB tag atop the vehicle. This was chosen as plywood is a low cost and easily accessible material, and allows for multiple

TABLE 3.3: Vehicle Specification

Wheel Diameter	120mm
Wheel Width	60mm
Body Length	270mm
Body Width	280mm
Motor Rated Voltage	12V DC
Motor Load Speed	100RPM
Motor Rated Speed	90RPM
Motor Gearbox Length	19mm
Motor Rated Current	0.52A
Motor Rated Torque	0.263N.m
Motor Maximum Torque	0.597N.m

platforms to be stacked on top. Plywood also makes it easier to drill through compared to the sturdier aluminum chassis. Furthermore, a small plastic pipe was attached to the side of the vehicle in order to elevate the IMU away from the magnetic field disturbance produced by the DC motors. Accordingly, different views of the vehicle platform can be seen in Fig. A.1, Fig. A.2 and Fig. A.3.

3.1.6 Power Supply

After choosing the BBB as the main microcontroller and its corresponding motor controller, Seed Motor Bridge Cape, the power requirements for the platform are 5V @ 2A for the BBB and 12V DC for the motor controller. The BBB power must be supplied through a 5.5 mm/2.1 mm DC barrel jack and must be able to handle currents of up to 2A, due to the power requirement for Wi-fi and other peripherals. To satisfy the requirements, a custom built USB 2.0 A male to 5.5 mm/2.1 mm DC barrel jack connector was created using 18AWG wires. The 18AWG wires were used to guarantee that up to 2A can be used for power transmission following the AWG guide in [22]. As for the main power source for the BBB, a 20000mAh portable battery pack with rated 2.4A output per USB port was purchased to ensure that the BBB can operate with its peripherals properly. For the motor controller, the 12V DC power supply will be used with PWM to control the speed of the four separate DC motors. As a result, two sets of 12V NiMH rechargeable battery packs and a NiMH charger were purchased to allow for multiple uses.

Additionally, a set of Tamiya battery connector plugs and sockets were used to connect the battery pack to the motor controller. The connections to the power supplies can be seen in Fig. A.1 and Fig. A.2.

3.2 Device Specifications

In order for the vehicle platform to function properly, certain specifications must first be met. Table 3.4 shows the specifications needed for the master PC and the chosen microcontroller based on the previous components mentioned. Firstly, the master PC and the BBB must both be connected to the same local network. Internet is not necessary for this project to operate; however, the experiment setup in this paper requires Internet to connect to ntp time servers to synchronize the time between the master PC and the BBB. It is recommended to disable the devices from automatically setting time from the Internet to ensure that the devices do not try to connect to other time servers. This experiment uses the time servers of the University of Windsor for time synchronization.

Table 3.4: Device Specifications

	Master PC	**BBB**
Power	Computer dependent	up to 10W at 5V
Wi-fi	Yes	Yes
Operating System	Ubuntu 14	Debian 8
ROS Version	Indigo	Indigo
roscore	Running	Not running
ROS_MASTER_URI	<IP_ADDR>:11311	Master PC's URI
Network/IP Address	xxx.xxx.xxx.1-254	xxx.xxx.xxx.1-254
Subnet Mask	255.255.255.0	255.255.255.0
IMU	Not connected	Connected via I2C 2 bus
Decawave UWB Tags	Not connected	Connected via USB

Moreover, the operating systems may vary for each device, but it must be compatible with the selected ROS distribution. Roscore is used as the main process for communication between ROS nodes and should be run on the master PC, while the BBB should export the corresponding ROS Master URI to communicate successfully. For this experiment, the two main sensors used will be the UWB tag

for providing an absolute position and the BNO055 IMU for providing linear acceleration, magnetic field readings, and relative angular velocity. The UWB tag is connected to the BBB via USB, while the IMU is connected via I2C 2 bus line on the BBB. Accordingly, an overview of the connections can be seen in Fig. 3.2.

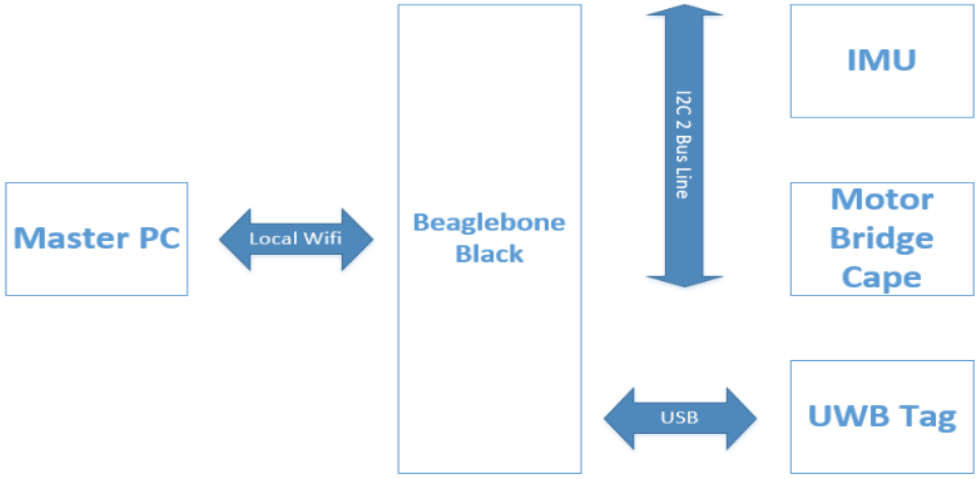


FIGURE 3.2: An overview diagram of the project connectivity.

3.3 Sensor Interface

This section explains the interface between the sensors and the BBB to receive raw data for processing. However, the processing and application of filters will be explained in greater detail in Chapter 4.

3.3.1 Ultrawideband

The UWB tag connected to the BBB via USB can be accessed using the Python serial library. The initial code received from the past research work at the University of Windsor Wicip Lab used the ttyACM0 port at a baud rate of 9600 to read the output of the UWB tag and store it in a text file. Then, another Python script was used for the trilateration calculation after reading the corresponding

text file. However, this method is not ideal for real-time applications since it requires having to open and close the text file in every iteration. As a result, the files were modified to work better for real-time calculations.

Instead of reading the output data from a text file, the trilateration calculations were placed inside a function that takes multiple input parameters as seen in Algorithm 1. The parameters, $dist\#$, represent the three distance values between the UWB tag on the vehicle platform and the three stationary anchors. The following $x\#, y\#$ parameters are the coordinate pairs for the known positions of the UWB anchors. Finally, the function returns the absolute position of the UWB tag as a set of tuples for further processing. This modification allows for the trilateration function to be called in the same iteration that the distance values are read without the need to open, write or read, and close a file object. Furthermore, if the raw distance values or calculated coordinates need to be recorded for future research work, a separate Python script can simply subscribe to the corresponding ROS topic and not have to interfere with the real-time positioning calculations.

Algorithm 1 Trilateration Function

1: **function** TRILAT($dist0, dist1, dist2, x0, y0, x1, y1, x2, y2$) ▷ Calculates position using trilateration with received distance and anchor coordinates
2: Apply trilateration formula
3: **return** (x_pose, y_pose)
4: **end function**

3.3.2 Inertial Measurement Unit

Next, the Adafruit BNO055 sensor uses the I2C bus lines and includes a Python library to help interface with the sensor [21]. The sensor can be powered via 3.3V or 5V and connected to the default I2C bus line of the BBB. Using Fig. B.2 as a reference, P9_1-2 can be used as the common ground, P9_7-8 for 5V power supply, and P9_19-20 for the corresponding I2C 2 bus line. However, the Python library for the BNO055 defaults to the I2C 1 bus line, (which may be compatible with older versions of Debian), and does not match the configuration of the BBB with Debian 8 due to an applied kernel update. Thus, it was required to alter the given

driver file to correctly match the appropriate I2C bus line for data to be read properly. Fig. 3.3 shows the BNO055 breakout board and its relative size.

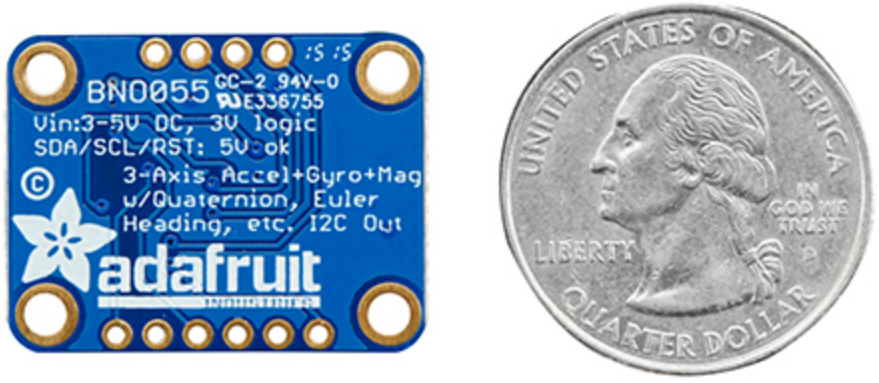


FIGURE 3.3: Adafruit BNO055 breakout board compared with an American quarter as reference.

Afterwards, the Adafruit BNO055 sensor can provide multiple types of data, including raw readings and sensor fused data. All data available from the sensor are Euler angles for roll, pitch, yaw (in °), orientation as a quaternion, sensor temperature (in °C), magnetometer data (in μT), gyroscope data (in deg/s), accelerometer data (in m/s^2), linear acceleration data (in m/s^2), and gravity acceleration data (in m/s^2). For this experiment, the important data needed are Euler angles, linear acceleration, magnetometer data, and gyroscope data. At first, the orientation as a quaternion was supposed to be used to properly match the ROS IMU sensor message structure. However, sending the quaternion orientation as a ROS message would always throw an error that the values were not normalized, most likely due to rounding limitations. As a result, the Euler angles were read instead and then transformed into a quaternion orientation using the ROS transforms library. Moreover, the linear acceleration is chosen for estimating the vehicle platform's position because the acceleration due to gravity is already filtered out, preventing it from generating unnecessary noise. Lastly, the magnetometer and gyroscope data are fused together to accurately track the current orientation of the vehicle platform. The magnetometer allows for finding its orientation after power up, while the gyroscope excels when there is magnetic field disturbance around the sensor.

Furthermore, the Adafruit BNO055 sensor also came with its own manual on how to properly calibrate the acceleration, gyroscope, and magnetometer data [23]. Additionally, the calibrated registers can be saved after successful calibration to avoid having to go through the physical calibration process again. Similar to Section 3.3.1, the raw values before processing can also be recorded using the same method without interfering with the rest of the program. It is worth noting that this experiment is done in 2D only and only require the use of the x and y linear acceleration.

3.4 Experimental Setup

This section briefly explains the setup and scenario for the experiment. Figure 3.4 shows an overview diagram of the placement of the UWB anchors and the starting point of the vehicle platform containing the UWB tag. The location for the tests were done in the hallways outside the Wicip Lab. The floor is tiled and the tests done avoids any major ramps or slants. It is important to note that the anchors are elevated to a height of 1.5 m by using tripods, while the vehicle platform continues to operate on ground level. Table 3.5 shows the respective anchor coordinates and the output frequencies for the IMU and IPS used in the experiments. The IPS frequency was kept constant throughout the experiments, but the IMU output data frequencies were modified to study the process noise of the accelerometer. The original output frequency of the accelerometer was at 30 Hz, but was concluded to be too noisy to be useful in predicting position. As a result, the 30 Hz acceleration values received were averaged over two different time windows for the tests. The first set of tests took the average of acceleration values over 1/3 seconds resulting in a frequency output of 3 Hz, while the second set of tests took the average of acceleration values over 1 second resulting in a frequency output of 1 Hz.

FIGURE 3.4: Experimental setup of experiment in the hallway (not to scale).

TABLE 3.5: Experiment Parameters

Parameter	**Value**
Anchor 0	(2.196 m, 19.840 m)
Anchor 1	(3.380 m, 0.000 m)
Anchor 2	(0.620 m, 0.000 m)
Anchor height	1.5m above ground
IMU Frequency	1.0 - 3.0 Hz
IPS Frequency	3.5 Hz

Chapter 4

Methodology and Description of the Work

4.1 Position Estimation

The vehicle platform consists of two sensors that produces a position estimate: the UWB based IPS and the IMU. Process for how each system estimates the position can be found in Section 2.1 and Section 2.2. On the other hand, this section will explain the data processing done in an attempt to improve the problems of the current system.

4.1.1 Smoothing IPS

The first problem encountered was the measurement noise of the UWB tag when stationary. After the navigation goal and path finding is done by the ROS navigation stack, it sends a safe velocity command to guide the vehicle platform through the found path. However, every new measurement from the UWB tag would cause the current position to change as seen by the multiple points in Fig. 4.1. As a result, the velocity commands from the navigation stack constantly change with

every new measurement, making it impossible to autonomously guide the vehicle platform.

At first, a single threshold parameter was set to not update the current position unless the new position received from the UWB tag was larger than the threshold. However, this process made the results look more discrete (as the map only updates when the vehicle moves past the threshold distance) rather than the desired continuous feel of tracking. Another method attempted was calculating the average of the received measurements over 1 second, but this just resulted in delaying the time the next position would change to every 1 second rather than 3.5 times a second.

Therefore, the proposed solution was to use a moving window average with a desired threshold distance. By using a moving window average, the output frequency of the data can continue to remain at 3.5 Hz and the only delay would be the initial time it takes to fill the window when starting. Calculating the average of the window also helps to mitigate some outliers during operation. Additionally, the use of the threshold distance helps reduce the measurement noise by not accepting small position changes. As a result, the average of the window would not be changed unless the measurement values are large changes, which is more likely to happen when the vehicle platform is in motion. The final result can be seen in Fig. 4.1, where it drastically reduces the measurement noise to an acceptable stationary position for consistent velocity commands.

Algorithm 2 Moving Window Average with Threshold Distance

procedure SMOOTH_IPS(x_pose, y_pose) ▷ Smoothens the output coordinates
  if $buffer\_window$! =$window\_size$ **then**
    append coordinates to $buffer\_window$
  else if length($buffer\_window$) == $window\_size$ AND distance($new\_pose - avg\_pose$) > $threshold\_distance$ **then**
    delete oldest $buffer\_window$ values
    append new values to $buffer\_window$
  end if
  avg_pose_x = average of x values in $buffer\_window$
  avg_pose_y = average of y values in $buffer\_window$
end procedure

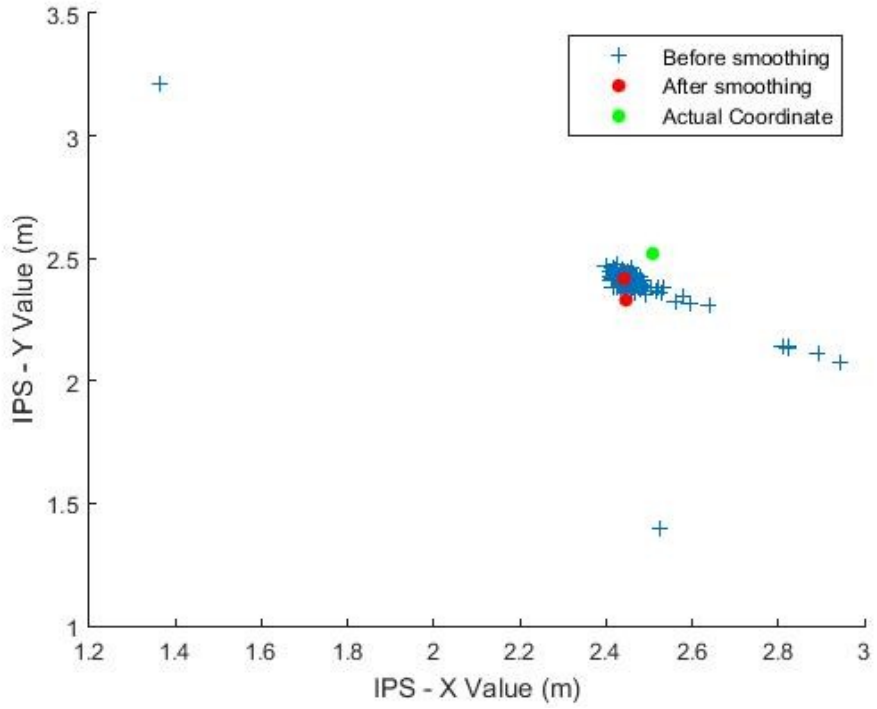


FIGURE 4.1: Plot comparing the output coordinates before and after Algorithm 2.

4.1.2 Processing IMU Data

Fortunately, the Adafruit BNO055 sensor chosen for this experiment had its own sensor fusion done by an on-board processor. As such, the fused data was able to estimate the proper orientation of the vehicle platform accurately. However, the problems that occurred during the tests were interpreting high output frequencies of the linear acceleration, interpreting linear acceleration after turning, and calibrating the linear acceleration to zero.

As mentioned previously in Section 3.4, the linear acceleration data received at 30 Hz made it very difficult to produce a good estimate of the vehicle platform's relative position. Accordingly, the IMU output was reduced to 3 Hz and 1 Hz in an attempt to make the linear acceleration have a better prediction of the vehicle platform's position.

Secondly, the reference direction of the IMU when turning does not cause the direction of the linear acceleration to change. As a reference, Fig. 4.2 shows the

coordinate frames used for the experiment. For example, the base_link frame represents the vehicle platform and the base_imu represents the IMU. If the base_link frame (front of the vehicle platform) and the base_imu (x-linear acceleration) are both originally facing North and a 90° clockwise turn was made, the problem was that the interpreted x-linear acceleration would cause the position to be updated as moving in the North direction rather than East. The problem was solved by using ROS transforms to properly track the rotational change of the base_imu relative to the base_link. Now, every time a linear acceleration value is received from the base_imu, the change in orientation is applied to the linear acceleration before being interpreted by the base_link frame.

Next, the measurements of the linear acceleration had to be further calibrated to match its position on the vehicle. Although the Adafruit BNO055 sensor's onboard processor does a good job of filtering out acceleration due to gravity, there was still some small acceleration values that were being read when the vehicle platform was stationary. As a result, a calibration method was applied to correct the small linear acceleration values being detected when the vehicle platform is stationary. The calibration works by collecting linear acceleration values for the first few seconds and determining the maximum, minimum and average values over the time period. For this experiment, a calibration time of 4 seconds was found to be adequate in determining the minimum and maximum values. During calibration, the vehicle should remain stationary to best represent a zero acceleration scenario. After the maximum, minimum and average values are recorded, all future values that are between the minimum and maximum values return a value of $0 m/s^2$. For values outside of this range, the calibration method will return the difference of the value and the average value from calibration. As a result, this calibration method should avoid publishing most acceleration values while the vehicle platform is stationary. Algorithm 3 shows the calibration function used for linear acceleration in the x-direction. Likewise, the same method can be applied to the other dimensions as needed.

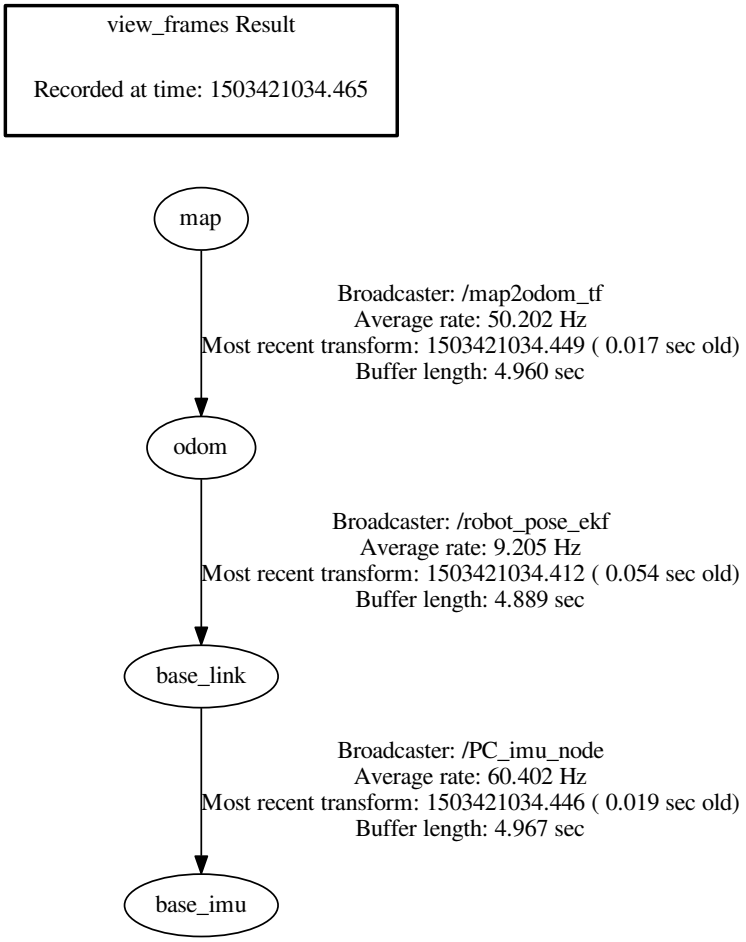


FIGURE 4.2: ROS transform frames used for the experiment.

Algorithm 3 Calibrating IMU to zero out when stationary.

1: **function** SENSOR_CALIBRATE(x_accel) ▷ Returns 0 if linear acceleration is within the calibrated min/max.
2: **if** $x\_accel >= x_{min}$ and $x\_accel <= x_{max}$ **then**
3: **return** 0.0
4: **else**
5: **return** $x\_accel - x_{avg}$
6: **end if**
7: **end function**

4.1.3 Extended Kalman Filter

The EKF is a modification of the original Kalman Filter as previously described in Section 2.3. In this experiment, the 'robot_pose_ekf' from the ROS navigation stack was utilized. The 'robot_pose_ekf' has four possible inputs with the following topic names: odom, imu_data, vo, and gps. As for outputs, it publishes a topic named 'robot_pose_ekf/odom_combined' and a transformation between the odom frame and base_link frame. As described in [8], the 'robot_pose_ekf' uses the relative pose differences rather than the absolute poses of the received data from the sensor. For example, if the vehicle platform moves from coordinate position (3, 4) to (3, 8), it is interpreted as a movement of 4 units rather than using the final position of (3, 8). As more sensors are added into the 'robot_pose_ekf', it becomes impossible to compare absolute position between different sensor reference frames. Furthermore, the covariance of the sensors are used rather than the covariance of the EKF pose for future calculations, as the EKF pose covariance would be infinitely increasing as more sensor data is received. Lastly, the timing mentioned in [8] explains that there must be a measurement from at least two different sensors before it produces an output. After receiving the required measurement data, the 'robot_pose_ekf' produces an estimated position based on the latest time where both sensor data is available. For sensor data that have different output rates, interpolation is done in order to calculate the measurement at the earlier time.

For this experiment, the input data used were 'imu_data', 'vo', and 'gps'. The 'imu_data' uses the ROS message structure for an IMU found in [8], while the 'vo'and 'gps' both use the ROS message structure for Odometry. The 'imu_data' receives a quaternion orientation, 3D angular velocity, and 3D linear acceleration, each having its own 3x3 covariance matrix. A sample covariance matrix for the 'imu_data' can be seen in Eq. 4.1. Note that since the variance in one direction does not influence the other directions, only the diagonal have non-zero values. Furthermore, since acceleration in the z-direction is not used, it should be given a very large variance to ensure that it does not influence the EKF improperly.

$$covariance\_matrix = \begin{bmatrix} x & 0 & 0 \\ 0 & y & 0 \\ 0 & 0 & z \end{bmatrix} \tag{4.1}$$

where:

x: is the x-component for angular velocity, linear acceleration, or rotation

y: is the y-component for angular velocity, linear acceleration, or rotation

z: is the z-component for angular velocity, linear acceleration, or rotation

Moreover, the 'vo' receives a 2D pose estimate equivalent to displacement integrated from the linear acceleration, and the 'gps' receives a 2D pose estimate from our UWB based IPS. Accordingly, both 2D pose estimates use the ROS Odometry message which follows a 3D pose structure with both position and orientation. Thus, this message results in a 6x6 matrix and a similar covariance matrix can be seen in Eq. 4.2. In order to publish only the 2D pose estimate, higher variance values should be given for the z_pose, x_rot, y_rot, and z_rot since 'vo' and 'gps' cannot accurately measure these components.

$$covariance\_matrix = \begin{bmatrix} x_{pose} & 0 & 0 & 0 & 0 & 0 \\ 0 & y\_pose & 0 & 0 & 0 & 0 \\ 0 & 0 & z\_pose & 0 & 0 & 0 \\ 0 & 0 & 0 & x_{rot} & 0 & 0 \\ 0 & 0 & 0 & 0 & y_{rot} & 0 \\ 0 & 0 & 0 & 0 & 0 & z_{rot} \end{bmatrix} \tag{4.2}$$

where:

x_{pose}: is the x-coordinate for positioning

y_{pose}: is the y-coordinate for positioning

z_{pose}: is the z-coordinate for positioning

x_{rot}: is the rotation about the x-axis

y_{rot}: is the rotation about the y-axis

z_{rot}: is the rotation about the z-axis

As mentioned earlier in Section 2.3, the covariance matrices have a major impact on how the sensor data will be treated. Higher variances lead to a less trusted sensor data, while lower variances will cause the EKF to trust the according sensor data more. Therefore, these covariance matrices represent the measurement and process noise for each sensor, and will need to be modified to achieve a desirable fused output.

Chapter 5

Testings and Results

The values for the following tests were recorded by subscribing to the corresponding ROS topic and storing the values in a separate file, which was then plotted by using MATLAB. This method ensures that the data logging process avoids as much interference with the real time processing of the data. Measured values were recorded by attaching a small action camera on the vehicle platform and the use of coloured markers on the floor as seen in Fig. 5.1. However, the start of recording the test values and starting the camera were done manually so it may account for some milliseconds in timing error. Lastly, the command sent to the vehicle platform was a single 'go forward' command calculated to be about 0.23m/s, and was stopped after approximately reaching the 16 m mark of the experimental setup.

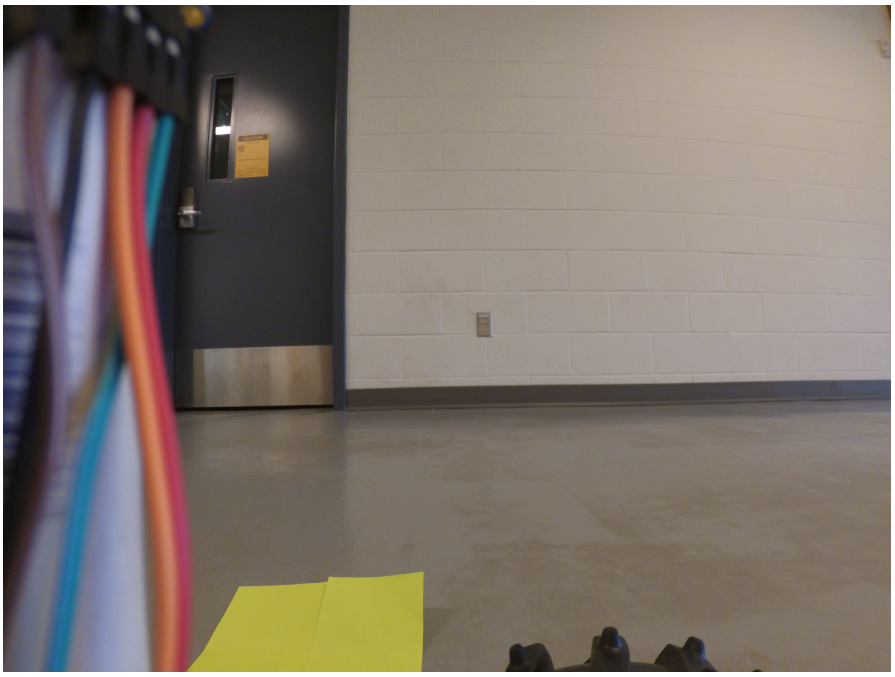


FIGURE 5.1: Test markers as seen by the camera for known positions.

5.1 Preliminary Tests

The preliminary results seen in Fig.C.1, Fig. C.2, and Fig. C.3 were used as a reference to determine how the 'robot_pose_ekf' would treat the initial covariance matrices of Eq. 5.1 and Eq. 5.2.

$$displacement\_from\_imu = \begin{bmatrix} 0.001 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 & 0 & 0 \\ 0 & 0 & 99999 & 0 & 0 & 0 \\ 0 & 0 & 0 & 99999 & 0 & 0 \\ 0 & 0 & 0 & 0 & 99999 & 0 \\ 0 & 0 & 0 & 0 & 0 & 99999 \end{bmatrix} \quad (5.1)$$

$$displacement\_from\_ips = \begin{bmatrix} 0.00001 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.00001 & 0 & 0 & 0 & 0 \\ 0 & 0 & 99999 & 0 & 0 & 0 \\ 0 & 0 & 0 & 99999 & 0 & 0 \\ 0 & 0 & 0 & 0 & 99999 & 0 \\ 0 & 0 & 0 & 0 & 0 & 99999 \end{bmatrix} \tag{5.2}$$

5.2 Test using IMU Average Output at 3 Hz and IPS Output at 3.5 Hz

It was observed that the linear acceleration with reference to the front and back of the vehicle is not as reliable as the positioning from UWB. On the other hand, the linear acceleration with reference to the side of the vehicle is more accurate, especially since the vehicle is not able to move sideways. Therefore, the following tests were completed by using a constant IPS covariance seen in Eq. 5.3 and a varying covariance for the IMU as seen in Eq. 5.4, Eq. 5.5, and Eq. 5.6.

5.2.1 Test Run 1

First test with modified IMU covariance.

$$displacement\_from\_ips = \begin{bmatrix} 0.001 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 & 0 & 0 \\ 0 & 0 & 999999 & 0 & 0 & 0 \\ 0 & 0 & 0 & 999999 & 0 & 0 \\ 0 & 0 & 0 & 0 & 999999 & 0 \\ 0 & 0 & 0 & 0 & 0 & 999999 \end{bmatrix} \tag{5.3}$$

$$displacement\_from\_imu = \begin{bmatrix} 0.00001 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 999999 & 0 & 0 & 0 \\ 0 & 0 & 0 & 999999 & 0 & 0 \\ 0 & 0 & 0 & 0 & 999999 & 0 \\ 0 & 0 & 0 & 0 & 0 & 999999 \end{bmatrix} \tag{5.4}$$

TABLE 5.1: Test 1: IMU Output @ 3 Hz and IPS Output @ 3.5 Hz

	Measured Values (meter)	**IPS (meter)**	**EKF (meter)**
Start	(2.153, 3.185)	(2.039, 3.036)	(2.039, 3.036)
Point 1	(2.129, 5.932)	(2.165, 6.178)	(1.977, 6.274)
Point 2	(2.112, 7.859)	(2.103, 8.087)	(1.992, 8.450)
Point 3	(2.089, 10.402)	(1.757, 10.644)	(2.012, 10.940)
End	(2.053, 14.512)	(1.316, 14.462)	(1.972, 14.436)

TABLE 5.2: Test 1: Error in positioning using measured values.

	IPS only error (meter)	**EKF error (meter)**
Start	0.187	0.187
Point 1	0.249	0.374
Point 2	0.228	0.603
Point 3	0.411	0.544
End	0.739	0.111

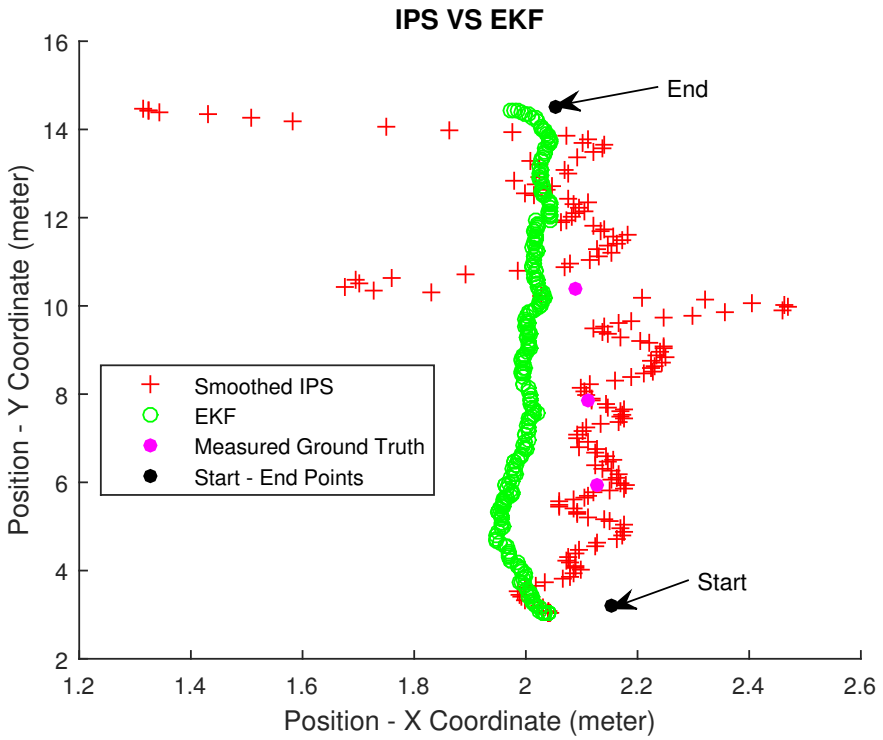


FIGURE 5.2: Comparison of coordinate position before and after filtering for Test 1.

FIGURE 5.3: Plot of x-coordinates versus time for real-time comparison.

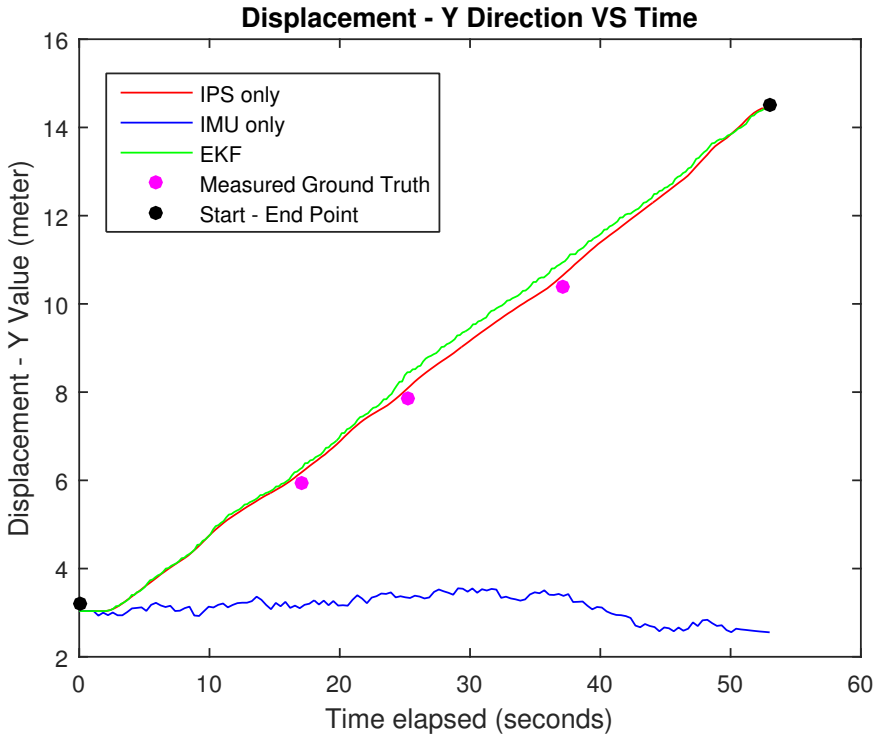


FIGURE 5.4: Plot of y-coordinates versus time for real-time comparison.

5.2.2 Test Run 2

Second test with modified IMU covariance.

$$displacement\_from\_imu = \begin{bmatrix} 0.00001 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 & 0 & 0 \\ 0 & 0 & 999999 & 0 & 0 & 0 \\ 0 & 0 & 0 & 999999 & 0 & 0 \\ 0 & 0 & 0 & 0 & 999999 & 0 \\ 0 & 0 & 0 & 0 & 0 & 999999 \end{bmatrix} \tag{5.5}$$

TABLE 5.3: Test 2: IMU Average Output @ 3 Hz and IPS Output @ 3.5 Hz

	Measured Values (meter)	**IPS (meter)**	**EKF (meter)**
Start	(2.153, 3.185)	(2.103, 3.056)	(2.103, 3.056)
Point 1	(2.220, 5.932)	(2.236, 6.199)	(2.255, 6.146)
Point 2	(2.267, 7.859)	(2.378, 7.794)	(2.312, 7.699)
Point 3	(2.329, 10.402)	(2.217, 10.375)	(2.333, 10.259)
End	(2.462, 15.830)	(2.663, 15.815)	(2.134, 15.815)

TABLE 5.4: Test 2: Error in positioning using measured values.

	IPS only error (meter)	**EKF error (meter)**
Start	0.138	0.138
Point 1	0.496	0.217
Point 2	0.129	0.166
Point 3	0.115	0.143
End	0.202	0.328

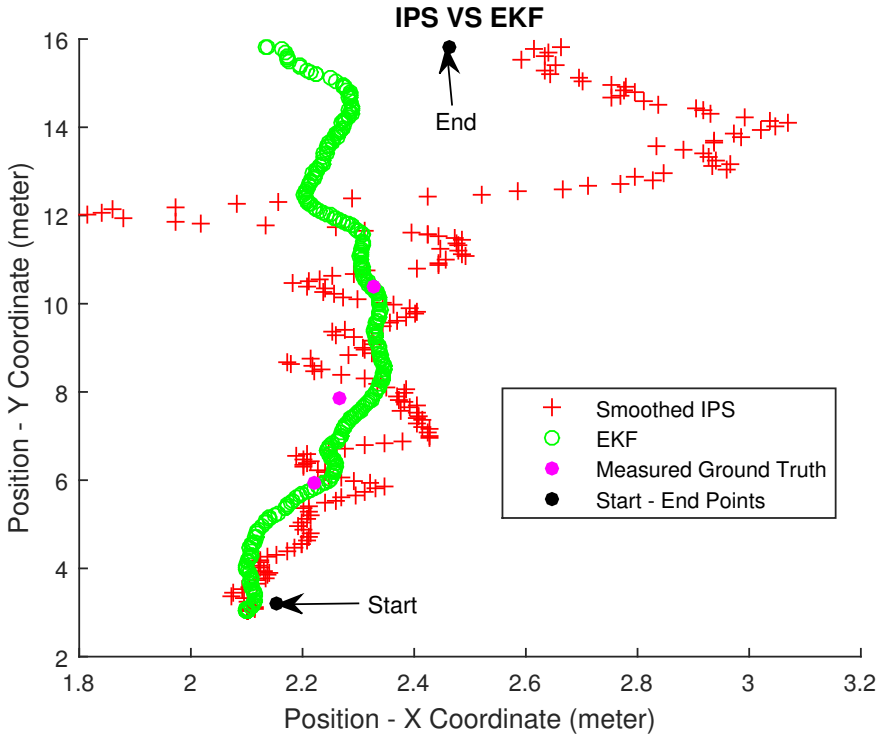


FIGURE 5.5: Comparison of coordinate position before and after filtering for Test 2.

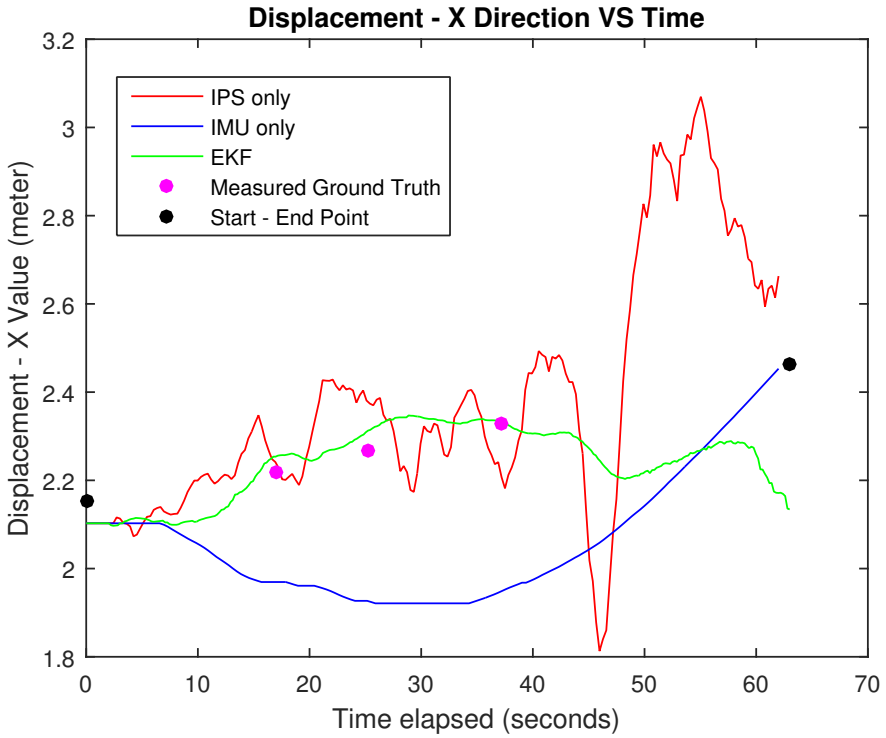


FIGURE 5.6: Plot of x-coordinates versus time for real-time comparison.

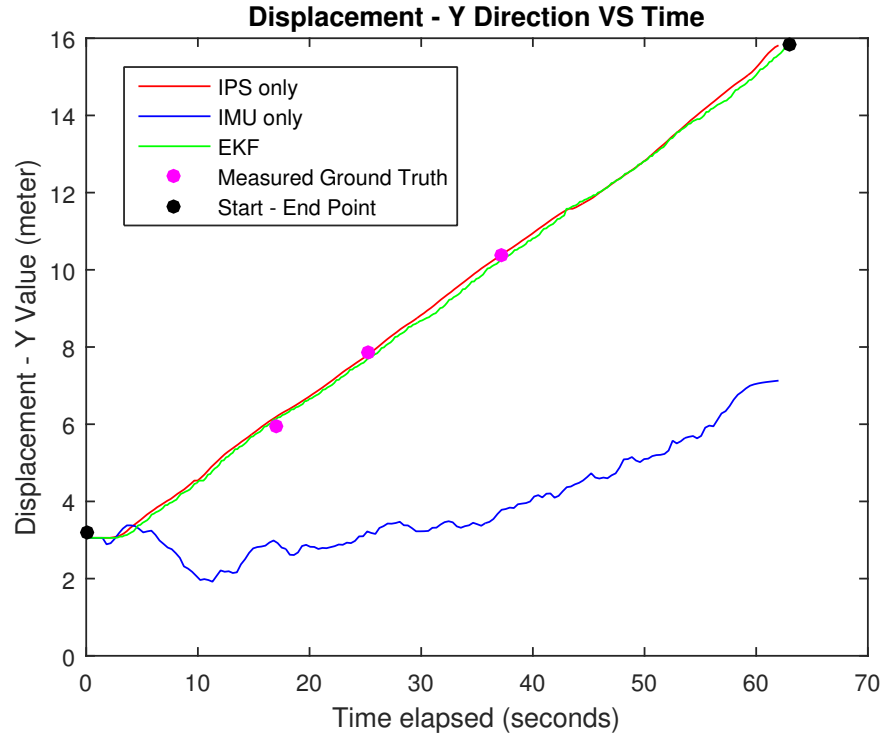


FIGURE 5.7: Plot of y-coordinates versus time for real-time comparison.

5.2.3 Test Run 3

Third test with modified IMU covariance.

$$displacement\_from\_imu = \begin{bmatrix} 0.00001 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 & 0 & 0 \\ 0 & 0 & 999999 & 0 & 0 & 0 \\ 0 & 0 & 0 & 999999 & 0 & 0 \\ 0 & 0 & 0 & 0 & 999999 & 0 \\ 0 & 0 & 0 & 0 & 0 & 999999 \end{bmatrix} \tag{5.6}$$

TABLE 5.5: Test 3: IMU Average Output @ 3 Hz and IPS Output @ 3.5 Hz

	Measured Values (meter)	**IPS (meter)**	**EKF (meter)**
Start	(2.153, 3.185)	(2.022, 3.006)	(2.022, 3.006)
Point 1	(2.114, 5.932)	(2.088, 5.454)	(2.042, 5.252)
Point 2	(2.086, 7.859)	(2.109,7.059)	(2.066, 6.800)
Point 3	(2.049, 10.402)	(2.145, 9.286)	(2.116, 8.886)
End	(1.973, 15.751)	(2.049, 15.745)	(1.662, 15.744)

TABLE 5.6: Test 3: Error in positioning using measured values.

	IPS only error (meter)	**EKF error (meter)**
Start	0.222	0.222
Point 1	0.479	0.684
Point 2	0.800	1.059
Point 3	1.120	1.517
End	0.077	0.311

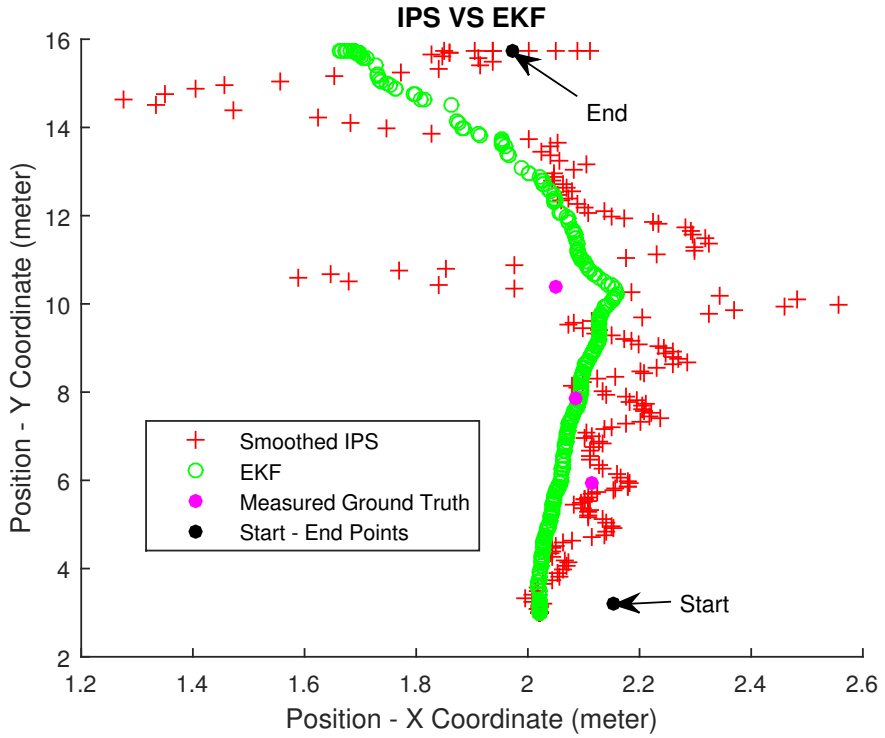


FIGURE 5.8: Comparison of coordinate position before and after filtering for Test 3.

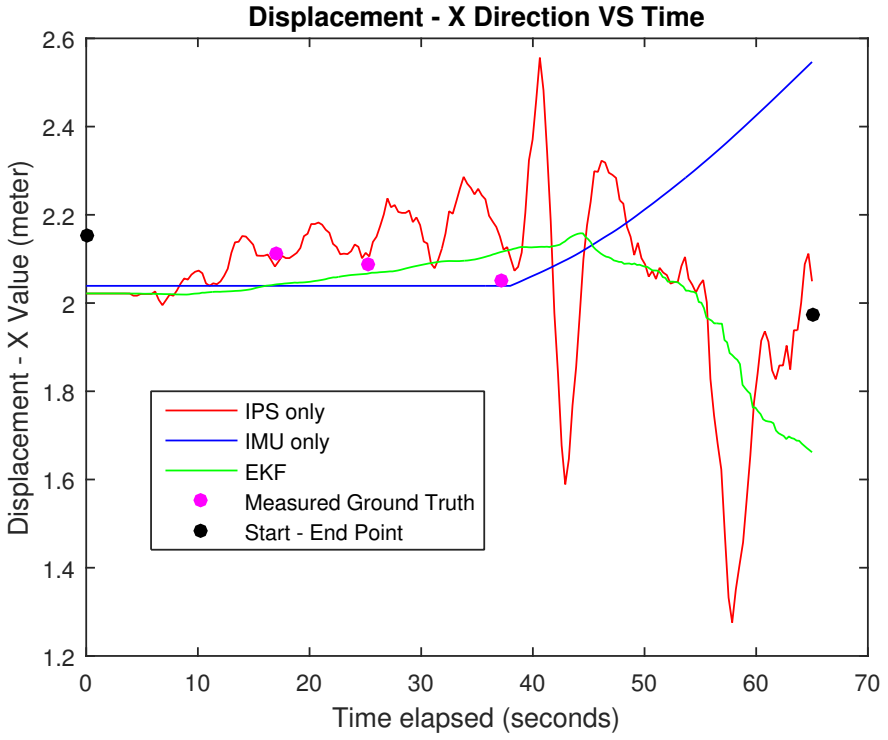


FIGURE 5.9: Plot of x-coordinates versus time for real-time comparison.

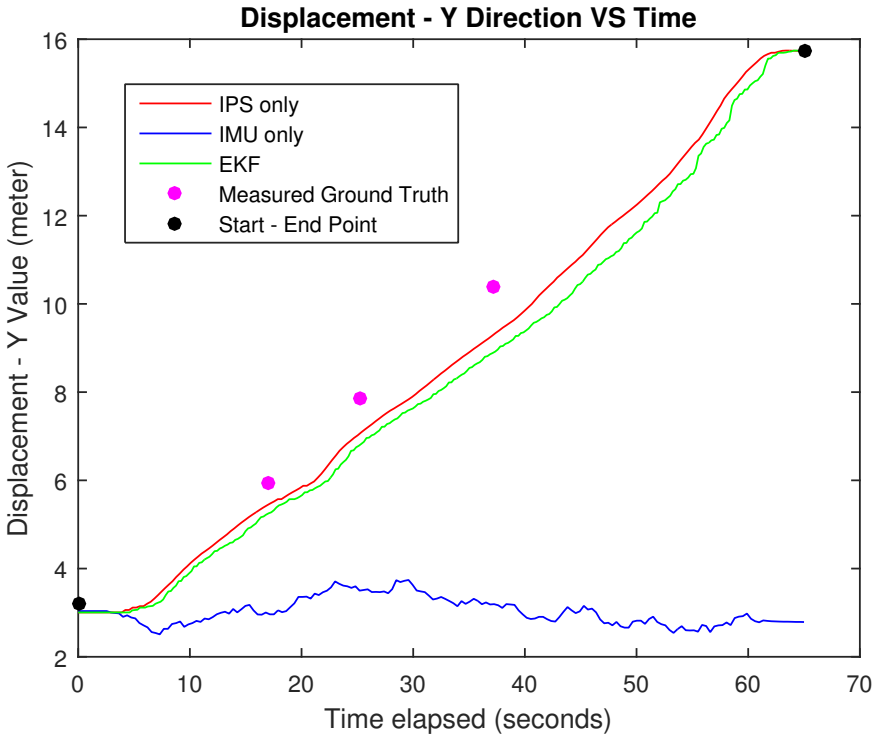


FIGURE 5.10: Plot of y-coordinates versus time for real-time comparison.

5.3 Test using IMU Output at 1 Hz and IPS Output at 3.5Hz

This section uses the same parameters as Section 5.2, but the IMU average output is reduced to 1 Hz. Thus, the following test uses the same covariances as Eq. 5.3 and Eq. 5.4. Further tests were also done with Eq. 5.5 and Eq. 5.6, but were excluded due to the results being inconsistent; thus, they were considered as outliers. As seen in Fig. 5.13, the lower output rate of the IMU produced a reasonable displacement until about the 25 second mark, where the displacement no longer became reliable. However, it can also be observed that once the vehicle stopped around the 55 second mark, the displacement values began to flat line as expected.

TABLE 5.7: Test 4: IMU Average Output @ 1 Hz and IPS Output @ 3.5 Hz

	Measured Values (meter)	**IPS (meter)**	**EKF (meter)**
Start	(2.153, 3.185)	(2.104, 2.971)	(2.104, 2.971)
Point 1	(1.927, 5.268)	(2.187, 5.112)	(2.122, 4.933)
Point 2	(1.641, 7.896)	(2.370, 7.836)	(2.040, 7.719)
Point 3	(1.383, 10.269)	(1.964, 10.181)	(2.046, 9.934)
Point 4	(1.118, 12.701)	(1.964, 12.471)	(2.050, 12.234)
End	(0.791, 15.711)	(0.456, 15.720)	(1.595, 15.719)

TABLE 5.8: Test 4: Error in positioning using measured values.

	IPS only error (meter)	**EKF error (meter)**
Start	0.215	0.215
Point 1	0.303	0.388
Point 2	0.731	0.436
Point 3	0.588	0.743
Point 4	0.877	1.042
End	0.335	0.804

5.3.1 Test Run 4

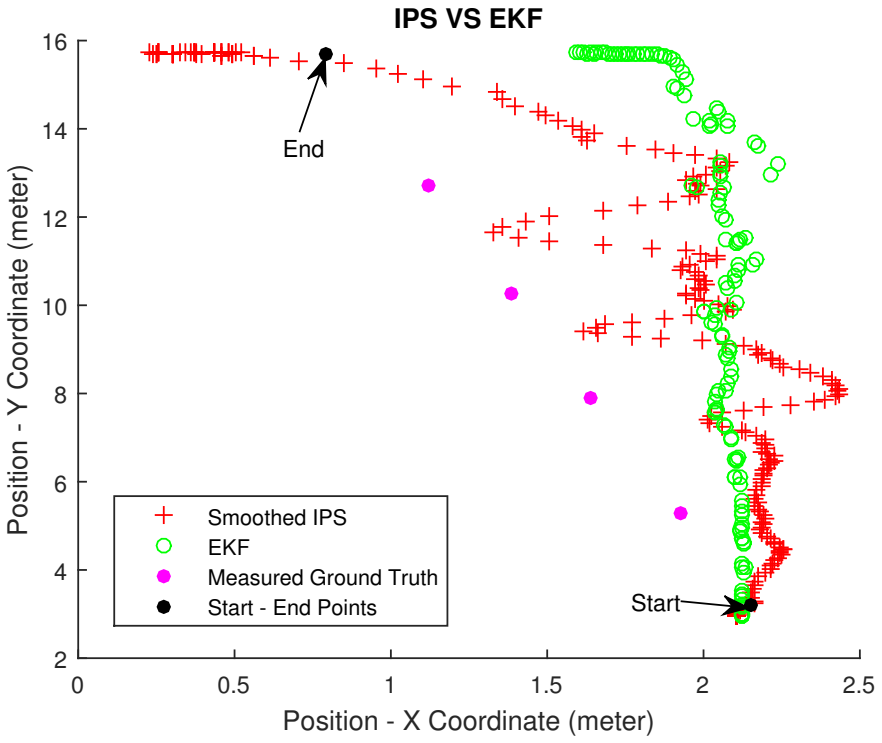


FIGURE 5.11: Comparison of coordinate position before and after filtering for Test 4.

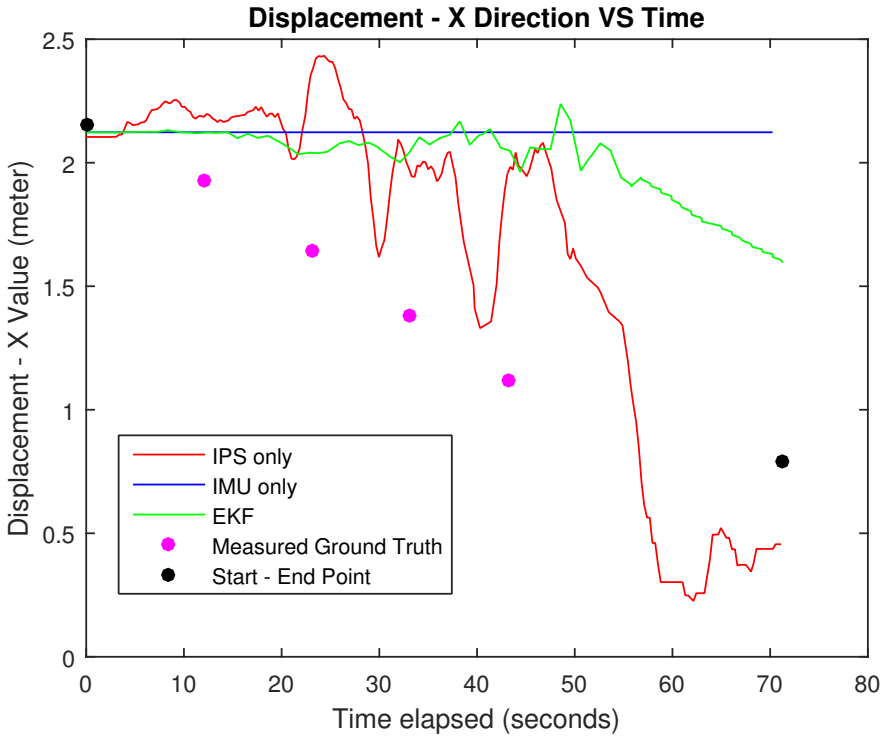


FIGURE 5.12: Plot of x-coordinates versus time for real-time comparison.

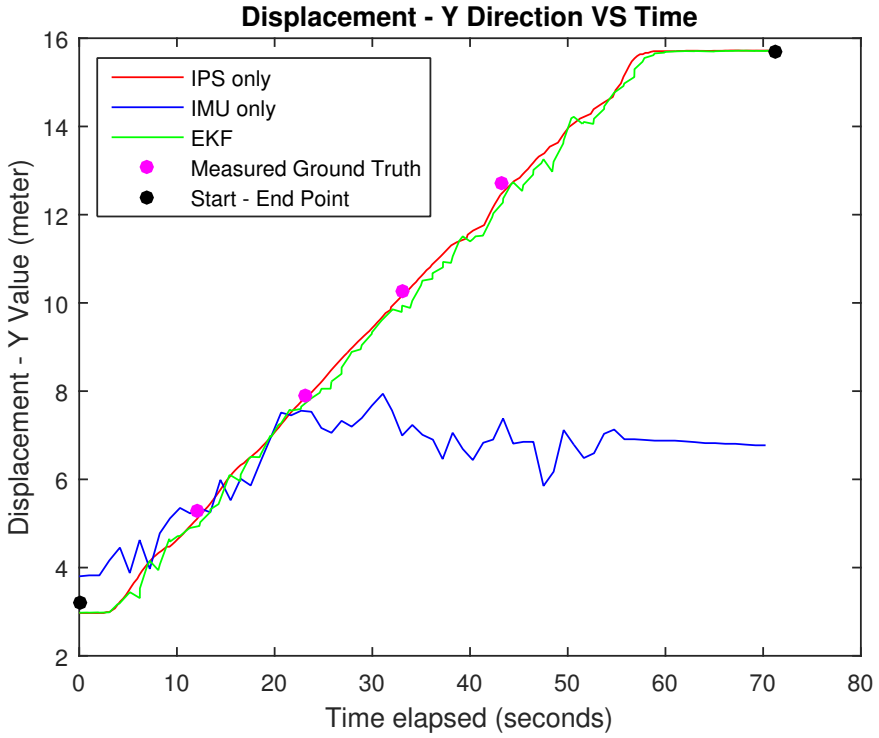


FIGURE 5.13: Plot of y-coordinates versus time for real-time comparison.

After reviewing all the test results, Section 5.2.1 produced the best results based on the error distance in the end of the run and the closest resemblance of a linear travel path in Fig. 5.2. Unfortunately, the measured points during run time was insufficient to provide a reasonable conclusion after seeing the overall plot. After travelling about 10 m in one direction, the values received from the UWB based IPS drastically increased its deviation compared to the EKF, and also had no more measured markers for further comparison. Additionally, the errors in the EKF model are also caused due to the higher initial error as seen by the plots. The higher initial error and use of relative displacement of the EKF rather than the absolute position of the UWB based IPS could explain the high EKF errors during travel time.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

All in all, this thesis was able to design and implement a relatively low cost vehicle platform that can receive velocity commands and react accordingly, as well as providing a better position estimate through the use of an EKF. The relatively low costs can be linked with the use of a lower cost IMU as compared to the Xsens IMU, as well as using the Decawave UWB for positioning as compared to the more expensive lidar technology commonly used for navigating robots. Additionally, the costs for the UWB transceivers will also be reduced once the evaluation kit is no longer needed. However, this design was focused more for testing scenarios, and the platform should be redesigned to be more robust for industrial uses.

The best results were achieved by using the covariances in Eq. 5.3 and Eq. 5.4. Since the 'robot_pose_ekf' only accepted one covariance matrix for each input measurement, the covariance matrices used represents the combined uncertainty. Moreover, the EKF excelled in using the best components of each measurement to provide the most accurate position estimate possible. By modifying the EKF model to have less reliability on displacement due to acceleration in one direction, and have higher reliability on its other more reliable component, the experiment was able to produce an accurate IPS. However, it is difficult to conclude the

accuracy of the system without being able to calculate the root mean squared error for the tests due to the lack of ground truth values.

6.2 Future Work

This thesis has just begun the actual implementation of a vehicle platform for the University of Windsor Wicip Lab and has more to improve on. Firstly, an improved test scenario could include an overhead cameras that can track the vehicle's ground truth values at least once a second. This will allow for a more accurate calculation of accuracy for the UWB based IPS. Next, the current vehicle platform design is not the most robust design as more focus was placed on the electronic components and sensor measurements. The weight of the battery being placed on the vehicle may have influenced the vehicle to head towards unwanted directions, as it was not able to head directly straight after a few meters. Next, a larger test space with obstacles will further test the scalability and stability of this project. Accordingly, the addition of obstacles will also increase NLOS conditions, causing the UWB based IPS to fluctuate. There is still ongoing research in the University of Windsor to be able to classify NLOS conditions during real-time operation, and once available, the corresponding covariance matrix for the IPS can be adjusted based on LOS or NLOS. Furthermore, the fusion with the IMU can also be improved by using a higher quality IMU such as the Xsens series mentioned in Section 3.1.4, or using known states of the vehicle (going forward, stop, etc.) to modify the covariance of the IMU accordingly. Lastly, addition of other sensors such as lidar would greatly increase the accuracy of the positioning system. A lidar can provide excellent relative position through the use of light sensors and SLAM algorithms, while IPS can provide excellent absolute positions, making these two sensors a perfect combination.

Appendix A

Vehicle Platform

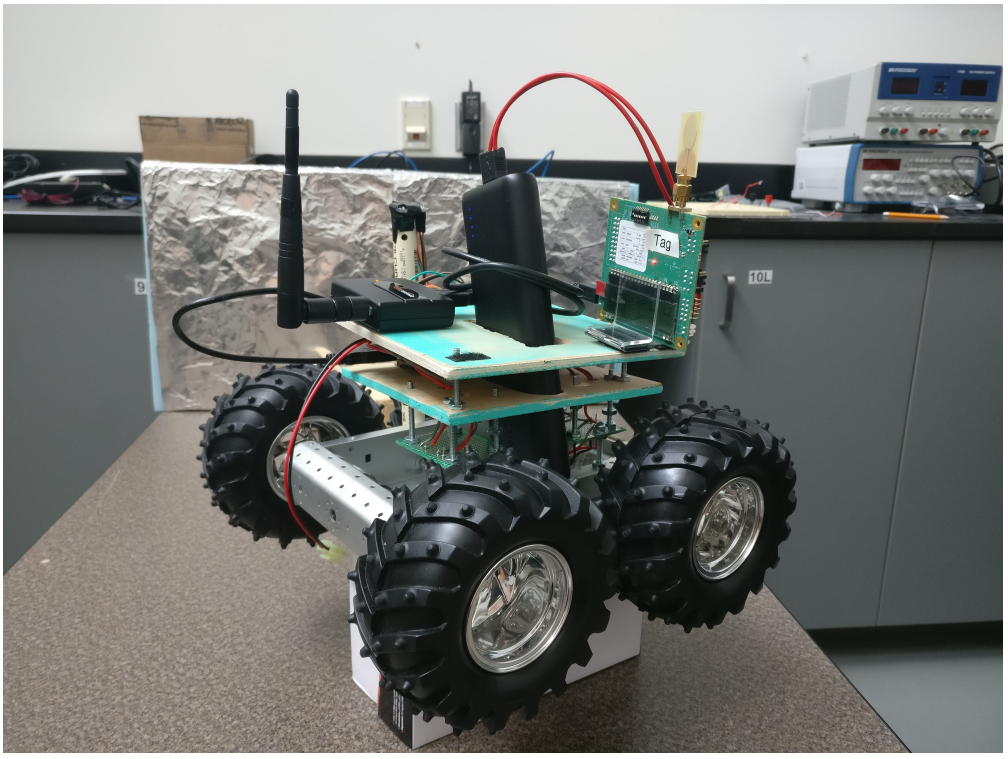


FIGURE A.1: Vehicle platform high angle left side view.

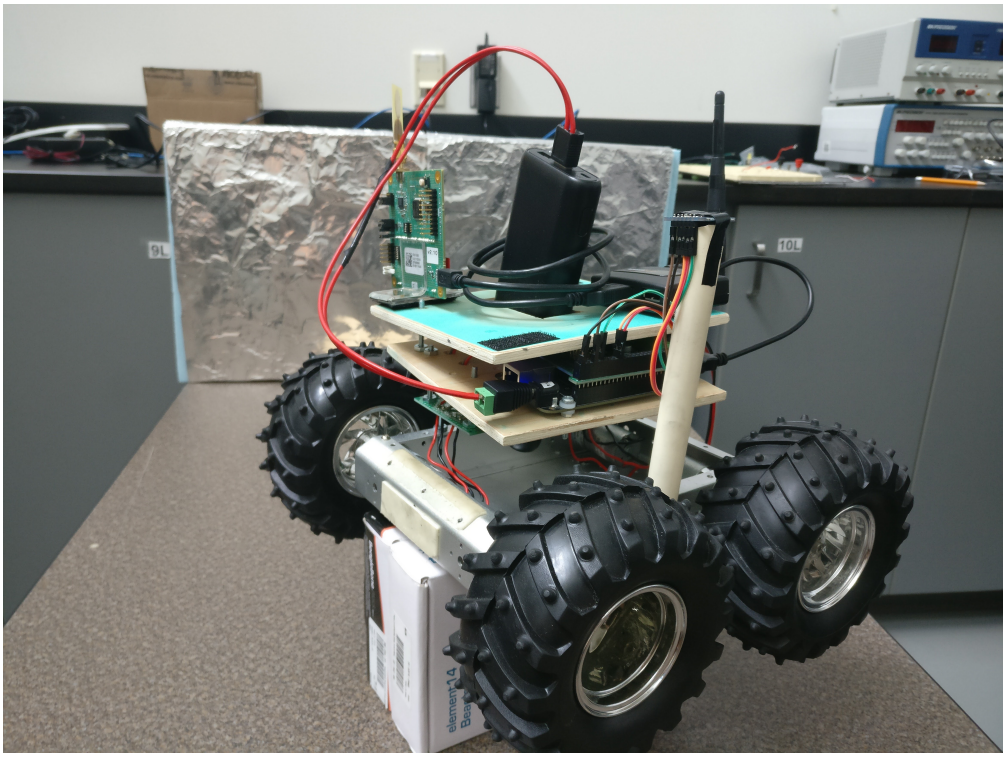


FIGURE A.2: Vehicle platform high angle right side view.

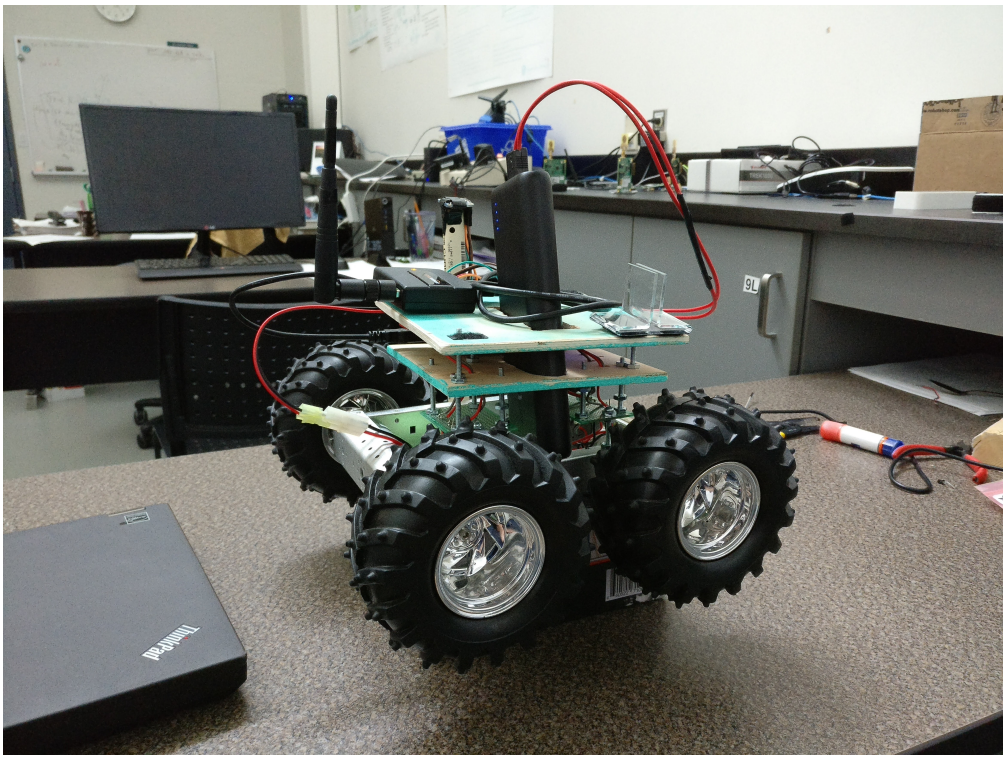


FIGURE A.3: Vehicle platform low angle view.

Appendix B

Beaglebone Black Header Pinout

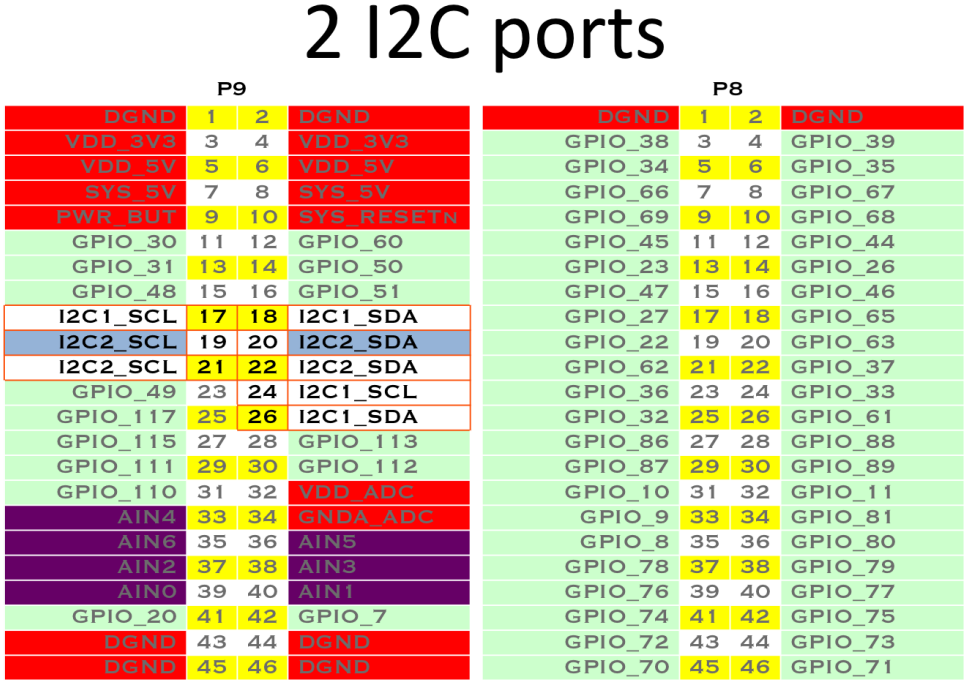
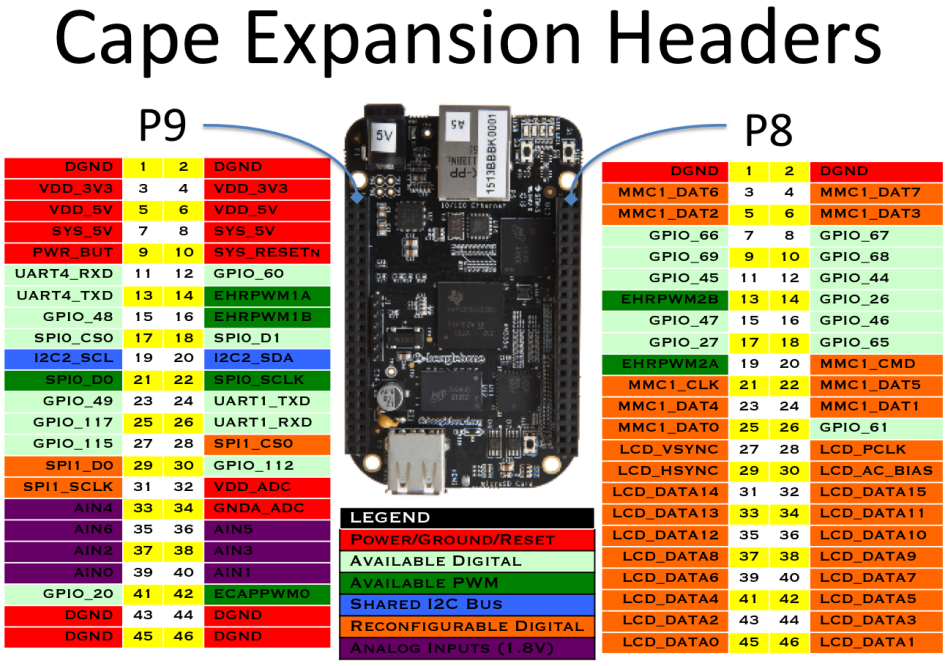


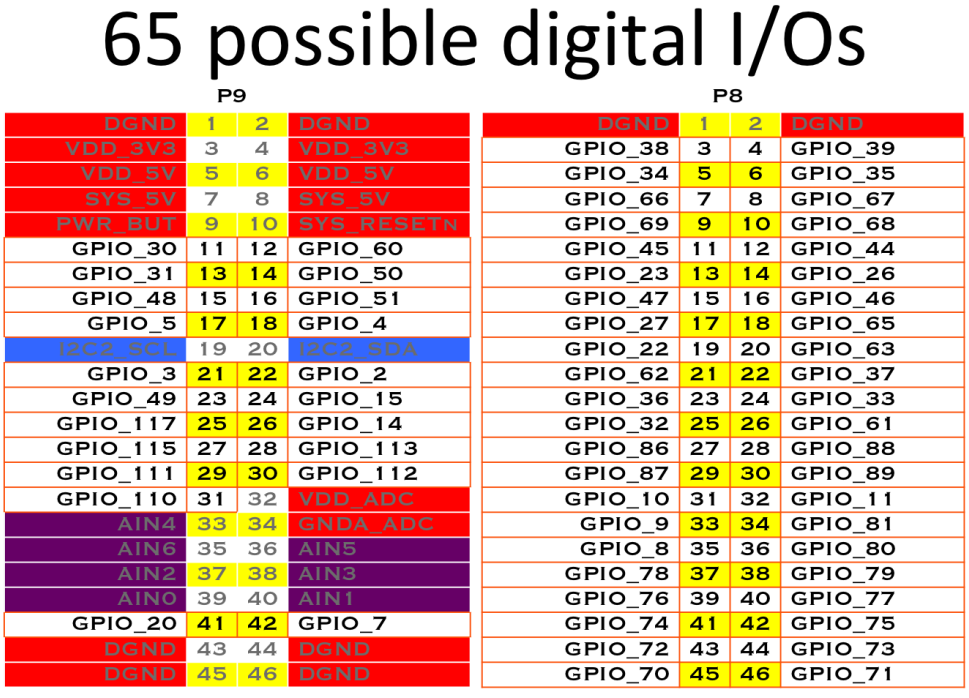
FIGURE B.1: BBB header pins for I2C ports.



DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETn
UART4_RXD	11	12	GPIO_60
UART4_TXD	13	14	EHRPWM1A
GPIO_48	15	16	EHRPWM1B
SPI0_CS0	17	18	SPI0_D1
I2C2_SCL	19	20	I2C2_SDA
SPI0_D0	21	22	SPI0_SCLK
GPIO_49	23	24	UART1_TXD
GPIO_117	25	26	UART1_RXD
GPIO_115	27	28	SPI1_CS0
SPI1_D0	29	30	GPIO_112
SPI1_SCLK	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	ECAPPWM0
DGND	43	44	DGND
DGND	45	46	DGND

DGND	1	2	DGND
MMC1_DAT6	3	4	MMC1_DAT7
MMC1_DAT2	5	6	MMC1_DAT3
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
EHRPWM2B	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
EHRPWM2A	19	20	MMC1_CMD
MMC1_CLK	21	22	MMC1_DAT5
MMC1_DAT4	23	24	MMC1_DAT1
MMC1_DAT0	25	26	GPIO_61
LCD_VSYNC	27	28	LCD_PCLK
LCD_HSYNC	29	30	LCD_AC_BIAS
LCD_DATA14	31	32	LCD_DATA15
LCD_DATA13	33	34	LCD_DATA11
LCD_DATA12	35	36	LCD_DATA10
LCD_DATA8	37	38	LCD_DATA9
LCD_DATA6	39	40	LCD_DATA7
LCD_DATA4	41	42	LCD_DATA5
LCD_DATA2	43	44	LCD_DATA3
LCD_DATA0	45	46	LCD_DATA1

FIGURE B.2: BBB cape expansion headers.



DGND	1	2	DGND
VDD_3V3	3	4	VDD_3V3
VDD_5V	5	6	VDD_5V
SYS_5V	7	8	SYS_5V
PWR_BUT	9	10	SYS_RESETn
GPIO_30	11	12	GPIO_60
GPIO_31	13	14	GPIO_50
GPIO_48	15	16	GPIO_51
GPIO_5	17	18	GPIO_4
I2C2_SCL	19	20	I2C2_SDA
GPIO_3	21	22	GPIO_2
GPIO_49	23	24	GPIO_15
GPIO_117	25	26	GPIO_14
GPIO_115	27	28	GPIO_113
GPIO_111	29	30	GPIO_112
GPIO_110	31	32	VDD_ADC
AIN4	33	34	GNDA_ADC
AIN6	35	36	AIN5
AIN2	37	38	AIN3
AIN0	39	40	AIN1
GPIO_20	41	42	GPIO_7
DGND	43	44	DGND
DGND	45	46	DGND

DGND	1	2	DGND
GPIO_38	3	4	GPIO_39
GPIO_34	5	6	GPIO_35
GPIO_66	7	8	GPIO_67
GPIO_69	9	10	GPIO_68
GPIO_45	11	12	GPIO_44
GPIO_23	13	14	GPIO_26
GPIO_47	15	16	GPIO_46
GPIO_27	17	18	GPIO_65
GPIO_22	19	20	GPIO_63
GPIO_62	21	22	GPIO_37
GPIO_36	23	24	GPIO_33
GPIO_32	25	26	GPIO_61
GPIO_86	27	28	GPIO_88
GPIO_87	29	30	GPIO_89
GPIO_10	31	32	GPIO_11
GPIO_9	33	34	GPIO_81
GPIO_8	35	36	GPIO_80
GPIO_78	37	38	GPIO_79
GPIO_76	39	40	GPIO_77
GPIO_74	41	42	GPIO_75
GPIO_72	43	44	GPIO_73
GPIO_70	45	46	GPIO_71

FIGURE B.3: BBB header pins for GPIOs.

Appendix C

Preliminary Results

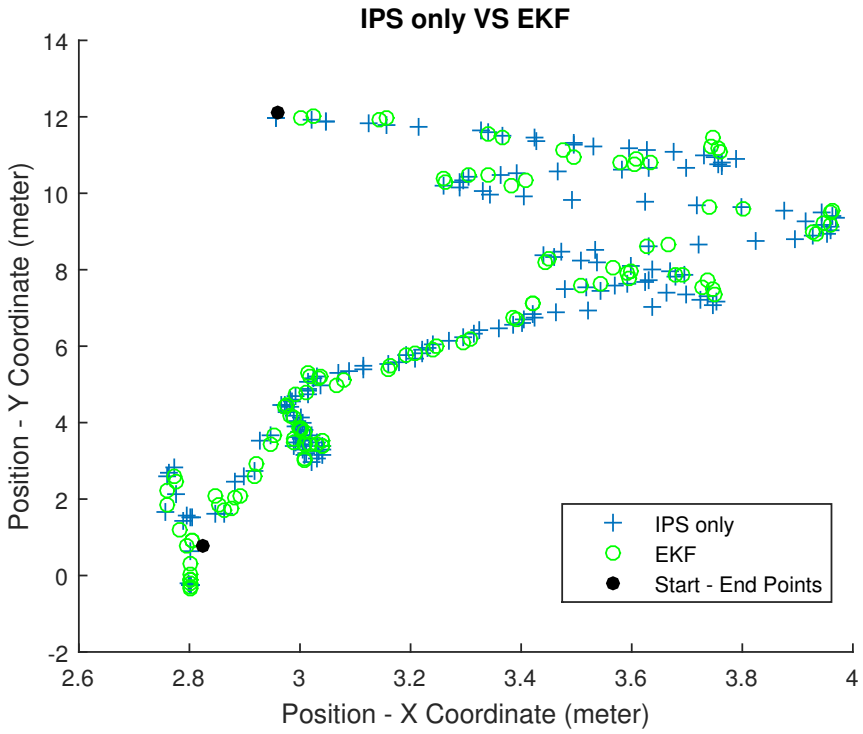


FIGURE C.1: Preliminary results for testing EKF.

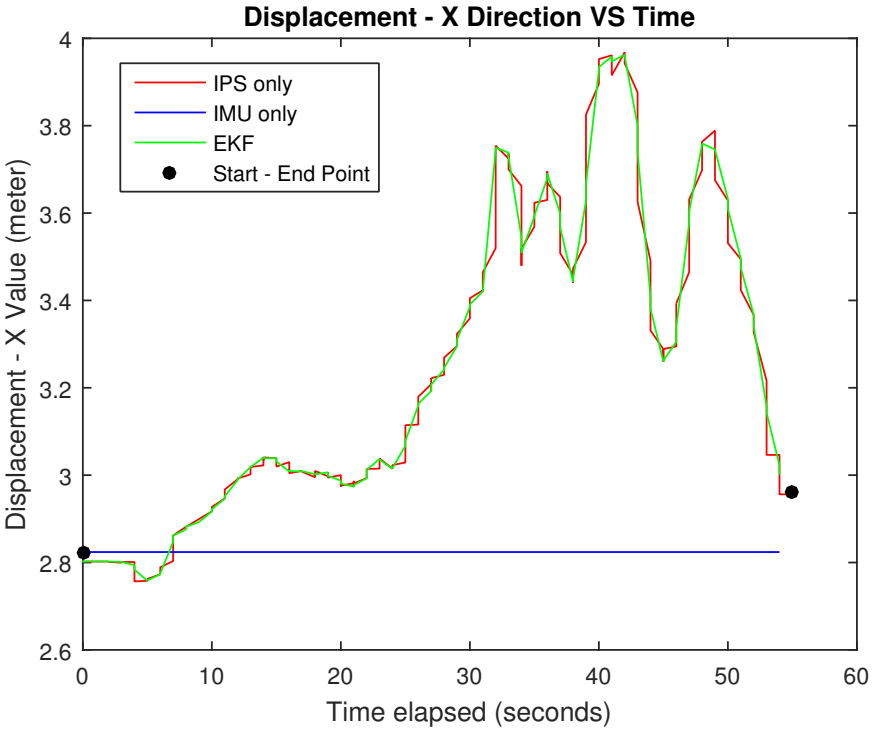


FIGURE C.2: Preliminary results for testing x-component.

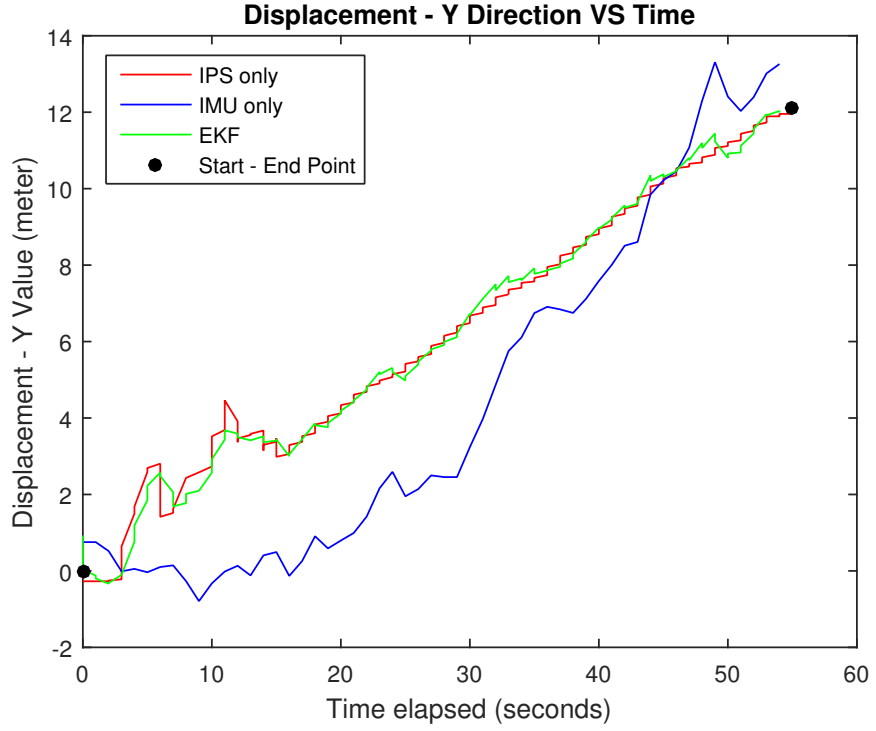


FIGURE C.3: Preliminary results for testing y-component.

Bibliography

[1] M. Mati, "Identification & Mitigation of NLOS Information for UWB based Indoor Localization," Master's thesis, Universty of Windsor, 2016.

[2] "Autonomous car - Explore - Google Trends." `https://trends.google.com/trends/`, Accessed August 12, 2017.

[3] M. Yavari, "Indoor real-time positioning using ultra-wideband technology," Master's thesis, University of New Brunswick.

[4] R. Faragher, "Understanding the basis of the kalman filter via a simple and intuitive derivation [lecture notes]," *IEEE Signal Processing Magazine*, vol. 29, no. 5, pp. 128–132, 2012.

[5] "balzer82/Kalman: Some Python Implementations of the Kalman Filter." `https://github.com/balzer82/Kalman`, Accessed August 12, 2017.

[6] "ROS-Industrial." `rosindustrial.org`, Accessed August 12, 2017.

[7] "Ros.org — Powering the world's robots." `http://www.ros.org/`. [Online; Accessed August 2017].

[8] "Documentation - ROS Wiki." `wiki.ros.org`, Accessed August 2017.

[9] "Zebra makes businesses as smart and connected as the world we live in.." `https://www.zebra.com/us/en.html`, Accessed February 22, 2017.

[10] "Ubisense." `http://ubisense.net`, Accessed February 22, 2017.

[11] J. McDonald, R. Multani, A.-B. Al-Rfouh, and M. Alamer, "Proposition for the Design of an Autonomated Floor Layout System." University of Windsor Capstone Final Report, 2016.

[12] V. Renaudin, B. Merminod, and M. Kasser, "Optimal data fusion for pedestrian navigation based on uwb and mems," in *Position, Location and Navigation Symposium, 2008 IEEE/ION*, pp. 341–349, IEEE, 2008.

[13] F. Evennou and F. Marx, "Advanced integration of wifi and inertial navigation systems for indoor mobile positioning," *EURASIP Journal on Advances in Signal Processing*, vol. 2006, pp. 1–12, 2006.

[14] Q. L. Yuan Xu, Xiyuan Chen, "Autonomous integrated navigation for indoor robots utilizing on-line iterated extended rauch-tung-striebel smoothing," *Sensors 2013*, no. 13, pp. 15937–15953, 2013.

[15] J. Jun, R. Guensler, and J. Ogle, "Smoothing methods to minimize impact of global positioning system random error on travel distance, speed, and acceleration profile estimates," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1972, pp. 141–150, 2006.

[16] T. Foote, "tf: The transform library," in *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, Open-Source Software workshop, pp. 1–6, April 2013.

[17] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *International Conference on Robotics and Automation*, 2010.

[18] J. Garimort, A. Hornung, and M. Bennewitz, "Humanoid navigation with dynamic footstep plans," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), May 2011.

[19] "MTi 10-series - Products - Xsens 3D motion tracking." `https://www.xsens.com/products/mti-10-series/`, Accessed August 12, 2017.

[20] "LSM303DLHC Datasheet." https://cdn-shop.adafruit.com/datasheets/LSM303DLHC.PDF, Accessed August 12, 2017.

[21] "BNO055 Datasheet." https://cdn-shop.adafruit.com/datasheets/BST_BNO055_DS000_12.pdf, Accessed August 12, 2017.

[22] "American Wire Gauge table and AWG Electrical Current Load Limits with skin depth frequencies and wire breaking strength." https://www.powerstream.com/Wire_Size.htm, Accessed August 12, 2017.

[23] "Device Calibration — Adafruit BNO055 Absolute Orientation Sensor — Adafruit Learning System." https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/device-calibration, Accessed August 12, 2017.

Vita Auctoris

Alvin Marquez graduated from Holy Names High School in Windsor, Ontario and continued to pursue further studies at the local university, University of Windsor. He graduated University of Windsor with a B.A.Sc. in Electrical Engineering with Co-op and continued on to pursue a M.A.Sc. in University of Windsor.