Indoor Localization For Robots Using HTC VIVE Lighthouse Positioning System

by

© Lavaanantha Thayaparendran

A Project Report submitted to the

School of Graduate Studies

in partial fulfillment of the

requirements for the degree of

Master of Science

Supervisor: Dr. Andrew Vardy

Department of Computer Science

Memorial University of Newfoundland

August 2021

St. John's Newfoundland

Abstract

Recent advancements in virtual reality technology have brought us affordable consumer VR devices. One such system is the HTC Vive lighthouse. The lighthouse system was initially developed as a tracking technology for virtual reality gaming applications. Due to its attractive use-cases in robotics, it is gaining attention as an off-the-shelf tracking system for collecting ground truth position data. It offers a wide field-of-view, high-resolution head-mounted display, and a room-scale tracking system for an affordable price.

Indoor localization is one of the critical challenges in autonomous robotics. There are many ways to achieve localization, and each has it is pros and cons. Various studies show that HTC Vive could provide precision in millimeter magnitude and accuracy ranging from 3 to 10 millimeters[1]. Therefore, we intended to study the lighthouse's functionality and see the possibility of applying it for indoor localization to implement a faster and better solution. Our research studies and results conclude that the HTC Vive lighthouse positioning system could be a reliable solution for indoor robot localization. With further development, we could more improve the pose estimation accuracy.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Dr. Andrew Vardy for choosing me into his team and giving me an opportunity to work on this project, offering all the required materials and access to his robotics lab.

I also would like to thank Dr. Oscar De Silva for his support and for offering VR trackers for my project work. Brian and Greg for providing the Oscilloscope on time and for the advice on my circuit design. This project was mainly adopted from the work done by Alexender, and I would like to thank him for his help and support throughout this project.

I am also very grateful for the knowledge and experience that I gathered from my friends and for the brilliant work developed by the VIVE team,

and of course, to my family, for their endless love and support from far, far away.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

Ever since humanity started to explore the world, it has looked for a reliable way for navigation. It began with simple landmark reference points for local travels, lately perfected towards celestial navigation for global journeys. When humanity finally conquered space, it deployed a global system for high accuracy localization. Autonomous robots face the same localization problem because, to be able to navigate with precision, they first need to know their position.

Several localization methods have been proposed over the years and refined according to the accuracy requirements and the navigation environment. Some approaches are meant for high precision local navigation, while others provide an approximate global position.

A robot capable of moving safely and accurately in a dynamic environment can have innumerous applications, ranging from simple delivery tasks to advanced swarm robotics. Besides improving productivity by performing repetitive tasks with precision and speed, robots can also act as coworkers, helping humans perform their jobs more efficiently and reducing the overall production costs.

1.2 Project

Several studies have examined the human vision of actively moving and exploring observers with the advancement of virtual reality(VR) technologies. VR setups allow participants to move in computer-controlled environments that can be manipulated to test hypotheses, which are hard to examine systematically in a real-world environment. Studies show that, independently moving observers may differ in significant ways from a stationary observer, and it reveals the difference in in-depth perception between static and moving observers.

The HTC Vive is a consumer device in the market used in general for VR games. It has a motion capture technology designed for virtual reality applications. Motion capture is the process of determining the absolute position and orientation of object in real-time [3]. It has scope in many areas like gaming, medical, film, engineering and most importantly in robotics. HTC Vive might be a small step for the virtual reality industry, but it's a big step for autonomous robotics [4].

The HTC Vive system includes a head-mounted device called a lighthouse that emits synchronized laser sweeps followed by an IR flash. The trackers or receiving module uses photodiodes to measure the light pulse timings to estimate the horizontal and vertical angles to lighthouses from the observer on the field. The tracking system uses the angle measurements taken from a bundle of photodiodes to estimate the pose using a method similar to Angle-of-Arrival [5].

1.3 HTC VIVE Overview

The HTC Vive consists of a VR tracker that acts as a receiver and infrared laser emitter units known as a lighthouse. In this project setup, the tracker was replaced by an integrated circuit with a photodiode to detect the IR pulses and laser sweeps. The Vive's tracker works based on the inside-out principle, without the need for any external cameras. Vive's tracking technology uses two laser emitters. These two boxes send out horizontal and vertical infrared laser sweeps alternatingly, spanning 120° in each direction. On the surface of the receiver, the circuit was integrated with photodiodes that indicate when the IR pulse and laser hits them. The time difference at which the laser hits the various photodiodes allows recovery of the position and orientation of the observer. The current tracked position and inertial measurement units update the orientation through dead reckoning as it allows for much higher update rates[6].

Based on the inertial measurements, the lighthouse units serve to correct and limit the build-up in error inherent in the dead reckoning at a rate of 120 Hz. If both lighthouses' view is interrupted, the Vive reports unchanging position and orientation values instead of updating. In general, the Vive allows physical movement within an area limited to 4 x 4 m. There are no limits to the range over which the Vive delivers position and orientation data. Various performance test results show that the tracking quality seemed equally good even in the largest where the lighthouse units were placed 7.45 m apart [3].

After mounting the lighthouse units, it is necessary to perform a calibration routine to know the position of lighthouse modules. Vive controllers are used for calibration by placing them close together on the floor in the center of the track space to know the ground and its orientation. Furthermore, the field is set up at this stage by defining the orientation and origin of the Vive's coordinate system in the X-Z plane.

1.4 Motivation and Objective

Indoor localization is one of the key issues in an autonomous robotics application, and HTC Vive seems to solve this age-old problem. When it comes to the robotics, the HTC Vive system provides compelling means of getting ground truth data. Compared to other competing technologies (especially computer vision-based), it is straightforward to set up and use. The implementation of this project includes a photodiode receiver circuit (tracking system) integrated with a robot that navigates the system to

identify the position and orientation for the robot based on lighthouse laser sweeps. In general, such localization is done with cameras and computer vision algorithm, which is a computationally costly process. By using the lighthouse positioning method, we could build a simple and effective solution for indoor localization.

1.5 Dissertation Outline

The remainder of this report is divided into seven chapters. Chapter 2 provides an overview and a brief history of the main localization approaches for the mobile robot platforms. Chapter 3 lists the usage of hardware components and the software components for the implementation of this project. Chapter 4 details the methodology that covers the main installation parts of the HTC Vive lighthouse, designing and building IR receiver circuit and mathematical model of the pose estimation algorithm. The project results and outputs are presented in chapter 5. In chapter 6, we discuss the overall summary of this project with experimental findings and results. Finally, chapter 7 presents a conclusion of this project and suggestions for future work.

Chapter 2

Approaches to Localization

Indoor robot localization is being a challenging area for a long time in robotics research and application. Over the years, different approaches were designed and applied for localization. When choosing the localization method, one should consider the precision requirements and the environment in which the robot is designed to operate, and all have their pros and cons. Some methods need external support systems to estimate the position, and others can be entirely autonomous, allowing the robot to localize it without any external dependencies. When designing a localization solution, the two fundamental issues that have to be considered are representing uncertain information about the environment and the robot pose.

Before working on any localization methods, it is good to know about the existing approaches and its pros and cons to make the design decisions well-founded. Also, some systems have a limited range, while others can only be effective in open spaces.

Moreover, several of these methods were designed to cope with errors in the localization sensors and tolerate temporary malfunctions. The below sections will introduce some of the most used self-localization systems that can be employed to estimate a robot's pose.

2.1 Proprioceptive Methods

Proprioceptive localization also known as a robot egocentric localization method which does not depend on the perception or recognition of external landmarks. These methods are immune to poor lighting, bad weather or extreme environmental conditions that may impact sensors such as cameras or lasers[7]. In the proprioceptive method, the localization system depends on its internal information. The robot process the internal operation and motion to update the current position . In this case, the robot does not rely on any external support system to operate. Clearly, this method does not update their position estimation based on the environment state; there can be a significant cumulative error. Proprioceptive methods are usually combined with exteroceptive systems to maintain an accurate robot pose estimation as a best practice.

2.1.1 Odometry

The odometry method is about estimating the current pose by integrating the robot's velocity over time. The robot velocity can be calculated by measuring the number

of rotations of the wheels (Optical encoders best fit for this measurement). This method can provide viable approximated positions. Vision-based odometry is a robust technique as it allows a vehicle to localize itself robustly by using a stream of images captured by a camera mounted on robots [8].

Dead reckoning is an extension of the odometry method. It is a typical standalone technique that sailors used in old times before the development of satellites. The travelled distance and direction are incrementally integrated relative to known start location to determine the current position[9]. It uses the acceleration and angular velocity parameters to improve the localization estimations. To improve the position estimation and to provide the robot orientation, other sensors can also be used, such as an accelerometer and gyroscope.

2.2 Exteroceptive Methods

The exteroceptive approach is about using a range of sensors to study the environment and retrieve the necessary details to perform the localization. Exteroceptive sensors can measure distances to known coordinates in order to provide absolute information. It often exists biases that affect these measurements because of particular environmental conditions or because of an inaccurate knowledge of the beacon's positions[10]. Generally, it is a more reliable and accurate method, as it periodically takes environment input and estimates the pose. It helps to reduce the cumulative errors.

2.2.1 Trilateration Methods

Trilateration is one of the geometric methodologies used to determine the absolute or relative position using distance from known points. Those distances are measured indirectly by using the Received Signal Strength (RSS) or Time of Arrival (ToA) parameters[11]. For 3D localization, it usually involves the intersection of four or more spheres, in which their radius is the distance to known positions.

2.2.1.1 Global Navigation Satellite System (GNSS)

Global Navigation Satellite System (GNSS) such as the Global Positioning System (GPS) , allows 3D positioning on Earth using a trilateration method. In these systems, a set of satellites broadcasts a radio signal with their position information and the time of the message sent. Using this data and knowing the speed of the radio waves, the distances to the satellites can be computed, with at least three satellites distances, the 3D position can be calculated.

Given that the correct measurement of the distances to the satellites relies on the accurate computation of the elapsed time between the message dispatch and its reception, it is critical that both the satellites and the receiver have synchronized clocks. This is achieved by using high precision atomic clocks in the satellites, and a clock reset technique in the receiver [12]. This reset method relies on the fact that three satellite distances will only have a valid location if the receiver's clock is synchronized. With this knowledge, the receiver can compute the necessary corrections

to reset its internal clock to match the satellite's time.

2.2.1.2 Differential Global Positioning System (DGPS)

The DGPS system works by installing reference base stations with known coordinates to improve the accuracy of the GPS position[13]. These stations are fixed and provide information about the corrections that can be made to the satellite signals in order to improve the localization precision.

These corrections are useful to mitigate some of the ambient interference that the satellite signals face. The interference can range from simple signal reflection in the environment landscape to the more complex interactions with the atmosphere, which can change the speed and path of the radio signals. The computation of the corrections is based on the fact that these stations are fixed, and as such, they can compare the location given by the satellite signals with their known location. With this position differential, the appropriate corrections can be calculated and broadcasted to the GPS receivers.

2.2.1.3 Assisted Global Positioning System (AGPS)

Assisted Global Positioning System (AGPS) systems are a common method used to speed up the Time To First Fix (TTFF) of a GPS receiver. They usually rely on the cellphone network to provide location estimation and signal corrections. This information can significantly reduce the TTFF when few satellites are visible, or their signal is very weak and only temporarily available[14].

2.2.1.4 Signal Strength Geolocation Methods

Signal strength geolocation, also known as fingerprinting localization, is an approximate method that can be used to calculate relative positions. It relies on the analysis of the signal attenuation from a given access point (like a Wi-Fi router or cellphone tower), to estimate the distance. With enough access points (usually four), an approximate position can be computed.

This type of distance estimation can be useful for indoor navigation but requires a propagation model of the signal and the environment. If these models are not accurate, then the localization precision of these methods will be very low. Though this method is less accurate than the more recent global localization systems (such as GPS), still it can be used without human-made infrastructures[15]. As such, it is a viable solution in case of temporary disruption of the GPS signal.

2.2.2 Celestial Navigation

Celestial navigation relies on observing stars, planets, or other reference objects to calculate the latitude and longitude. On Earth, position is represented by a latitude/longitude pair. Every celestial body can be projected onto the terrestrial surface by considering the line that goes from the center of Earth to the body. The point of intersection between the terrestrial surface and this line is the geographic position (GP), of the body[16]. The calculation of the position on the surface of the Earth using celestial navigation is similar to trilateration, but in this case, angles are used

instead of distances. These angles (delta) are measured between the Earth's horizon and the center of the celestial object. Having the delta and knowing the relative position of the Earth to the reference object, along with the Greenwich hour time, it is possible to calculate a circle of position.

Having at least three circles of position, the latitude and longitude can be computed. Although this method is less accurate than the more recent global localization systems (such as GPS), it can be used without human-made infrastructures. As such, it is a viable solution in case of temporary disruption of the GPS signal.

2.2.3 Landmark Methods

Landmark methods can be used to perform relative localization and are very useful to reduce the required information for navigation. In general, multiple landmarks are proposed for the mobile robot position calculation, because single landmark can imprecise due to the noise error. The robot measures the landmark direction while moving and the target landmarks can change depending on the robot status. The robot determines its position by the directions of the landmarks and its moving distance[17]. In these methods, a database of markers/environment geometry is stored along with its location, and when the robot recognizes one of these markers, it corrects its proprioceptive method's measures. It is useful for environments that have unique geometry in key positions of the navigation map.

2.2.4 Point Cloud Methods

Point cloud localization methods can be used to perform relative localization by finding the best point cloud match between the environment and the known map. These methods require a 2D or 3D representation of the environment and tend to be used in conjunction with proprioceptive methods (to estimate movement) and probabilistic methods (when the point cloud acquisition location is not known)[18].

One of the most used algorithms for 3D point cloud matching is the iterative closest point. It is an iterative algorithm that finds the translation and rotation transformation that minimizes the distances of the corresponding points on both clouds. It finishes when the matching error is below a given value, the matrix transformation between iterations has a translation/rotation below the specified thresholds or when the maximum allowed iterations is reached.

2.2.5 Probabilistic Methods

Probabilistic localization approaches identify the probabilities of robots to be in a specific position. The measurement errors can affect the sensor data. Therefore, the probability of a robot in a specific configuration can only be computed[19]. Probabilistic methods aim to reduce the impact of sensor accumulated errors or even temporary malfunctions by using Bayesian estimations and Markov processes. However, the Bayesian rule is used for perception update

2.2.5.1 Monte Carlo Localization (MCL)

Monte Carlo Localization (MCL) (also known as a particle filter) is a global localization algorithm that estimates the robot position and orientation by analyzing and adjusting the distribution and weights of state particles in a given environment. It is a computationally efficient method while retaining the ability to show arbitrary distributions. To approximate the probability distributions, it uses sampling-based methods, in a way that places computation as needed[20].

It starts by randomly distributing the state particles on the map, and over time it changes their position and weight according to new sensor readings. The probable location of the robot will be in the area of the map that has the largest cluster of state particles. It is faster, more accurate, less memory-intensive than earlier grid-based methods, and much easier to implement.

2.2.5.2 Kalman Filters

Kalman filters are probabilistic algorithms that estimate a given system state even when it is affected by noise or other errors. They perform linear quadratic estimations to achieve optimal results and be efficiently implemented in real-time systems. They are recursive algorithms based on Marcov processes, and as a result, they only need to know the current system state in order to perform measurement corrections[21].

The Extended Kalman Filter (EKF) is a variant of the Kalman filter, designed to handle non-linear systems by performing linear approximations to the state variables.

These approximations may lead to divergence in the estimations, and as such, the EKF can not guarantee optimal results[22]. The Unscented Kalman Filter (UKF) is another variant of the Kalman filter that was designed for highly non-linear systems. It usually achieves better results than EKF due to its unscented transform.

For the particular case of localization, these algorithms start with an initial estimation of the system state, and for each new position (computed from the sensors data), they predict the estimated robot location (according to the Bayes estimation model and the Gaussian distribution of errors), and then update their system's internal model (mean and covariance) to incorporate the system evolution.

2.2.5.3 Perfect Match

The Perfect Match is an efficient self-localization algorithm that is based on modeling the quality of several measures and minimizing the matching error. Its main goal is to minimize the localization error by carefully analyzing the known map and selecting the most probable current position using a gradient descent approach[23]. To improve tracking accuracy, the algorithm also uses a stochastic weighted approach. With the proper configuration, it can achieve a localization accuracy similar to the particle filter while using about ten times less computations.

2.2.6 Simultaneous Localization And Mapping (SLAM)

Simultaneous Localization And Mapping (SLAM) is a very effective approach to either explore unknown environments or update a known map. It relies on both proprioceptive and exteroceptive methods to perform localization and mapping[24]. It is also very useful to make the robot navigation more robust in dynamic environments, in which the topology of the world may change considerably over time.

There are numerous approaches to perform SLAM. Some optimized for exploration and others for map improvement. SLAM can be based on the Kalman filter, but there are far more sophisticated methods. For map correction and improvement, several probabilistic methods can be employed according to the precision required. For high accuracy 3D mapping, the ICP algorithm can be used to build accurate point clouds of the environment[25].

2.3 Lighthouse Positioning System

The Lighthouse system was initially developed as a tracking system for virtual reality gaming applications. Lately, it was focused on by robotics researchers due to its affordable price and attractive use-cases in robotics. In 2017 a team of researchers demonstrated a project call Crazyflie at the ICRA (International Conference on Robotics and Automation) conference in Montreal. The Carzyflie project is an application of the HTC VIVE lighthouse positioning system for the localization of

swarm robots. Crazyflie project is also the main inspiration for the initiation of this project.

The Crazyflie project that was demonstrated in 2017 was designed for a swarm cluster of micro unmanned aerial vehicles (UAV). When flying a cluster of robots in a small volume, the position estimation accuracy is important. The Craziflie project shows how the Lighthouse positioning system helped to achieve the required accuracy [26]. Crazyflie uses the crossing beam method, which was developed by the open-source community [2][3] for lighthouse version 1, and later, another research team extended it to use it for lighthouse version 2 [26]. With the crossing beam method, it requires two lighthouse base stations for pose estimation, but they proposed, Extended Kalman Filter (EKF) lighthouse measurement model enables the Crazyflie to fly with only one base station [27].

Chapter 3

Relevant Software and Hardware Technology

3.1 Hardware Setup

The whole project setup was an integration of a set of software and hardware components. HTC VIVE base station is the main hardware that we used for position calculation and the mobile robot that we used to test and demonstrate this project's scope. We already discussed about the HTC VIVE base station in this paper with sufficient information.

Any robot can be used for the application of this project, but it should move in a 2D frame as we are calculating 2D coordinates only. We used Pololu 3Pi Robot(Figure 3.1) (currently, we have a newer version of this robot called 3Pi+ 32U4 robot). It is a

complete, high-performance mobile platform with two micro gear motors, few sensors and other embedded components. For the scope of this project, we need to access only the motors to drive the robot. We need an AVR ISP programmer to upload the program and 4 AAA batteries to power up the board.

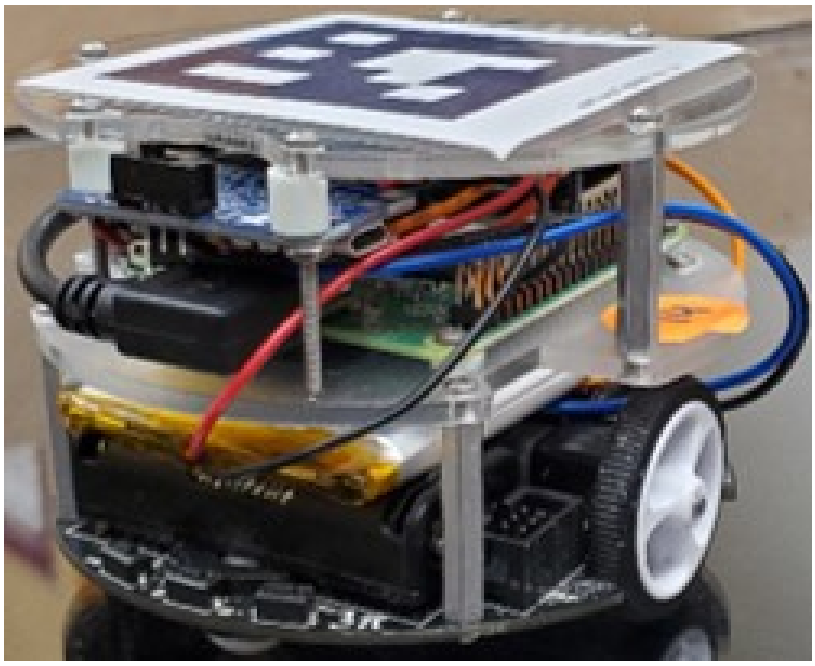

Figure 3.1: Pololu Robot

To make it wirelessly accessible, we used Raspberry Pi 3 (Model A+) (Figure 3.2) to communicate with Pololu. Raspberry Pi is attached with the AVR ISP programmer that connects the Pololu. As it is all integrated, any program can be written to Pololu from Raspberry pi through serial gateway communication via AVR ISP programmer. Here the Raspberry Pi acts as a staging server, so if anyone wants to access the Pololu, first should log in to Raspberry Pi, then feed the program to Pololu to drive

the robot. As the Raspberry Pi 3 model has a wifi module, it is wirelessly accessible using a secure shell host (SSH).

Figure 3.2: Raspberry Pi 3 (Model A+)

As discussed earlier, the HTC Vive emits IR pulses and laser sweeps, so we need a receiver to detect those IR signals. This is the second important part of the project to build the IR receiver circuit to detect and send it to process the IR signals. It was mounted on top of the robot. So that, while the robot moves on field, the receiver circuit helps to find the robot's current position. The circuit is pretty simple but need to consider a few points. The HTC Vive will emit IR pulse and laser sweep in very short time intervals, so the IR receiver should be able to capture it. Most

of the IR receiver's field of view is only at the line of sight towards the IR emitter. But in our case, the robot will be in motion over the defined 2D plane, so the IR receiver should have a wide-angle of view to capture the IR emission from the base station. Considering all these facts, the circuit was designed with a BPV22NF IR photodiode which has an optical IR filter 790-1050nm, high sensitivity, speed, and most importantly 120° field of view. To cover the detection zone as much as possible, we deployed three photodiodes to cover the whole top hemisphere field of view. IR photodiodes produces a very small current, which needs to be amplified before feeding to the processing module; for that, TLV2462IP opamp was used. Following that, a simple highpass filter was added to filter out background illumination level changes. As one opamp IC has two of them, both the amplifier and the highpass filter can be designed in one chip. A completed circuit module and the schematic diagram was shown in below figure 3.3.

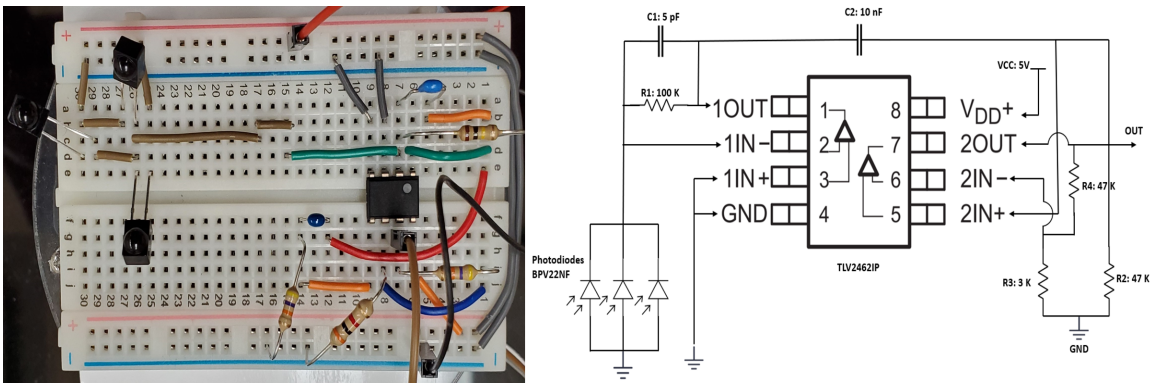

Figure 3.3: IR Receiver Circuit and Schematic

Another hardware component used in this project was the Teensy microcontroller. As already mentioned, Pololu was used for a robot controller, and the program can be executed from Raspberry Pi. Basically, here we are, running the program from Raspberry Pi and sending the speed parameters to Pololu. We also need to do position calculation by processing the IR signals detected by the receiver circuit. The Teensy microcontroller was used here to process those IR signals from the receiver circuit output. So that can facilitate dedicated computing resources for position calculation. The Teensy is a small-scale USB-based microcontroller. It does not need any additional hardware for programming; can simply program it via a USB port (figure 3.4). The Teensy 3.2 model was used for this project, but any advanced version also would work. From the Teensy, the processed position output can be sent to Raspberry Pi via USB or Teensy's TX1/RX1 UART ports can be used to feed the output instead of USB. To avoid integration complexity, the position calculation also can be done it Raspberry Pi, but not recommended for high-performance requirement applications.

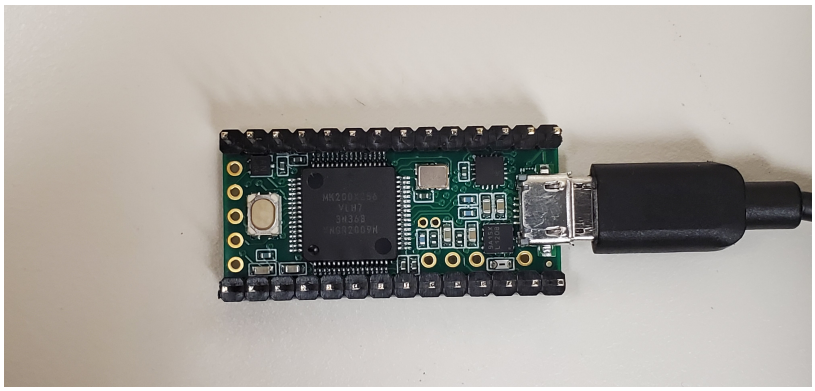

Figure 3.4: Teensy Microcontroller

3.2 Software Setup

Programming was done on two different platforms for this project. Teensy microcontroller can be programmed in C/C++ language; for that Teensyduino IDE was used. It can be installed in Windows machine to write the code and the IDE will support compiling and uploading the binary file (.hex file) to the Teensy board. In Teensy, the program should be written to output the current position values to Raspberry pi via USB or Tx/Rx UART interface. On the other hand, the robot control program can be written in Raspberry Pi in python language. For the development on Raspberry Pi, VScode IDE can be used from a Windows machine. From VScode, we can SSH to Raspberry Pi and access its file system to do the development.

For the pose estimation, we need the translation and the rotation of base stations, which can be described in various ways, including as a rotation matrix (could also also use Euler angles or a quaternion). To get those parameters we need to install steamVR application on our system and connect the VR trackers for the calibration and to get base station parameters. SteamVR is a gaming application platform, therefore we need a system with at least 8GB ram, i5/i7 processor and storage space of 8GB. Before starting to build the pose estimation model, it is advisable to setup the steamVR application with VR trackers. It helps for any debugging or troubleshooting.

Chapter 4

Methodology

The implementation of the localization solution using the lighthouse positioning system consists of few stages. Initially, we need to set up the lighthouse units with the appropriate channel configuration. A clear operational understanding of the lighthouse is important to write the software code. To detect and process the IR pulses and laser sweeps, we need a receiver circuit and it needs to be integrated into the robot. Finally, a software program is required to perform a pose estimation for localization and to drive the robot.

4.1 HTC Vive Lighthouse Setup

HTC Vive lighthouse units are the fundamental units of this localization solution; therefore, they need to be installed before any other implementations. Before installing the units, we need to know how it works. We already discussed the func-

tionality of the HTC Vive units in previous chapters. Basically, it emits an IR synchronization pulse, and a laser plane sweeps in vertical and horizontal directions. The sync pulses are used to synchronize time, so they have to simultaneously hit all reachable sensors of all tracked objects. For that, they use a wide-angle light source like LEDs as they can flood the entire tracked volume with light. The base station parts that generate this light pulse are called Omnidirectional Optical Transmitter (OOTX).

Even though it is recommended to use two base stations in the AR gaming environment, we are using a single base station unit to determine the feasibility to implement a localization solution, and that is the whole idea of this project. Using two base stations reduces the pose estimation update frequency, by using a single base station, it can be improved. The base station works in three different channels. If we are using two units, keep one as channel A and B or B and C or A and C. When using only one unit, always set the channel to A or B, C belongs to the slave channel, and it always waits for the master channel (A or B) to respond. So channel C will not respond when using a single base station. Channels can be configured by pressing the push button behind the lighthouse device.

Laser sweeps are emitted, followed by the IR sync pulse. If there are two base stations, only one will cast the laser sweep at a time, and it will be coordinated among the base station units during the sync pulse communication. The vertical laser sweep goes bottom to top, as shown in figure 4.1. From the ABCD 2D plane

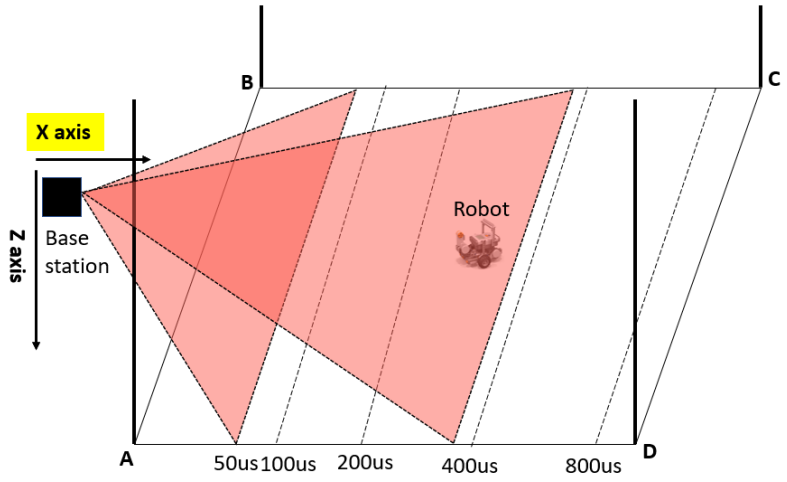

Figure 4.1: Vertical Laser Sweep

perspective, the vertical laser sweep moves from AB line to DC line. For the position estimation, we need to measure the time it takes for the laser sweep to hit the robot (IR receiver mounted on top of the robot) after the sync pulse. Based on the robot's position along the X-axis, the impact time may differ. That time difference gives the coordinate estimation of the robot on X-axis.

The horizontal laser sweep goes left to right, as shown in figure 4.2. From the ABCD 2D plane perspective, the horizontal laser sweep moves from line BC to line AD. Based on the robot's position along the Z-axis, the laser hit time will differ, and this time difference can be mapped to coordinate values along the Z-axis

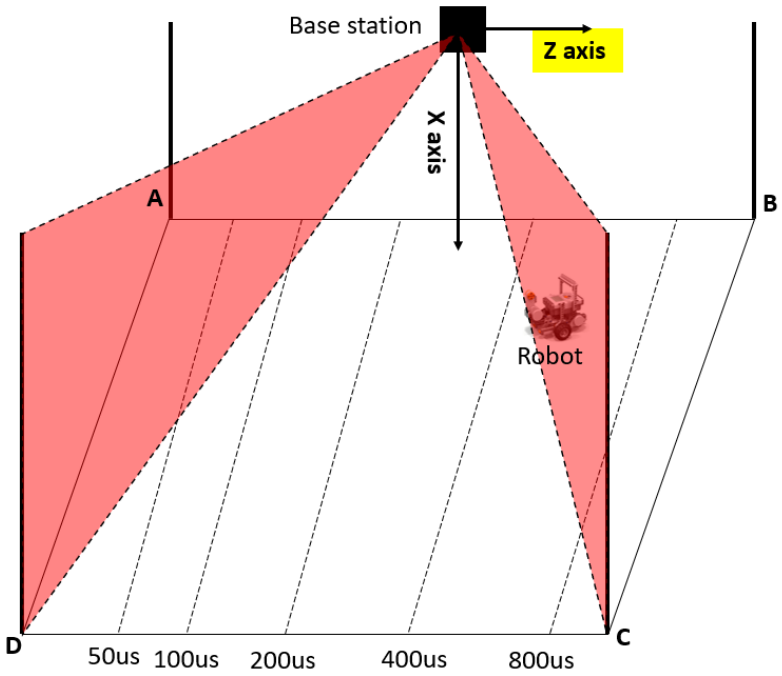

Figure 4.2: Horizontal Laser Sweep

Finally, the lighthouse needs to be mounted in the field of view of the robot. It is recommended to place the lighthouse not less than two meters in height and not more than 5 x 5 meters of moving zone for better accuracy. If we need a broad active region, then we should place more lighthouse units. When using two or more base stations, it has to be mounted at the corners of the room at line of sight to each other (for synchronization) and tilt downwards around 30 to 40 degree angle. As we are using a single lighthouse base station for this localization application, we propose to place the unit facing the robot moving plane in top view, as shown in figure 4.3

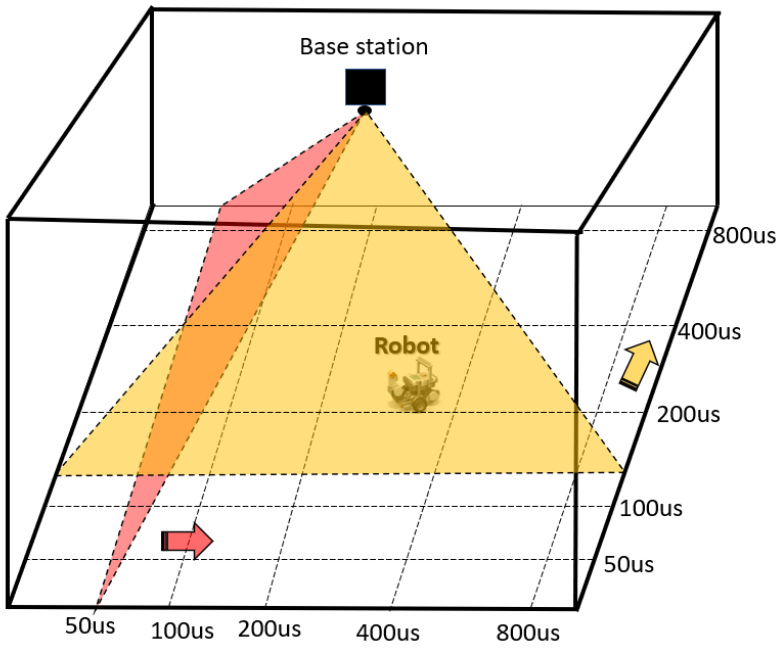

Figure 4.3: HTC Top Mounting

4.2 IR Receiver Circuit

In HTC Vive based AR gaming, users wear a headset and tracking gadgets that act as receivers to detect the IR pulses and laser sweeps from the lighthouse. To replace that, we need an IR receiver circuit for tracking. Three photodiodes were used in 120-degree formation in the horizontal plane to cover the whole top hemisphere field of view to build the circuit. The IR photodiode outputs a minimal current, so we need to amplify it before feeding it to a processing module. The amplified output was fed into a simple highpass filter to eliminate the background illumination level changes. Below figure 4.4 shows the complete receiver circuit functional diagram.

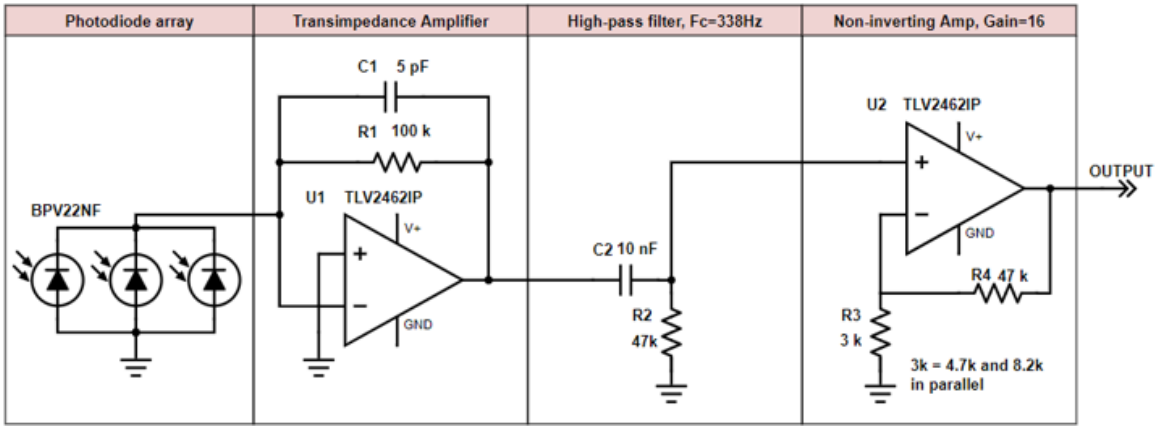

Figure 4.4: IR Receiver Circuit functional Diagram [2]

Below table 4.1 shows the bill of material required to build the IR receiver circuit

Table 4.1: IR Receiver Circuit BOM

Part	Model	Count
D1, D2, D3	BPV22NF/BPV23NF	3
U1/U2 (Op-amp)	TLV2462IP	1
Board	Perma-proto	1
C1	5pF	1
C2	10nF	1
R1	100k	1
R2, R4	47k	2
R3	3k	1

Once we are done with the circuit, now we can see how it sees the IR pulses and laser sweeps from the lighthouse base station. It is important to know this pattern as we need to write a program to detect them, classify them, and estimate the robot's position. Power up the circuit, place it under the lighthouse's field of view and channel it with the oscilloscope at the output to see the detected signal pattern. The figures 4.5 show the expected circuit output for single base station and dual base stations.

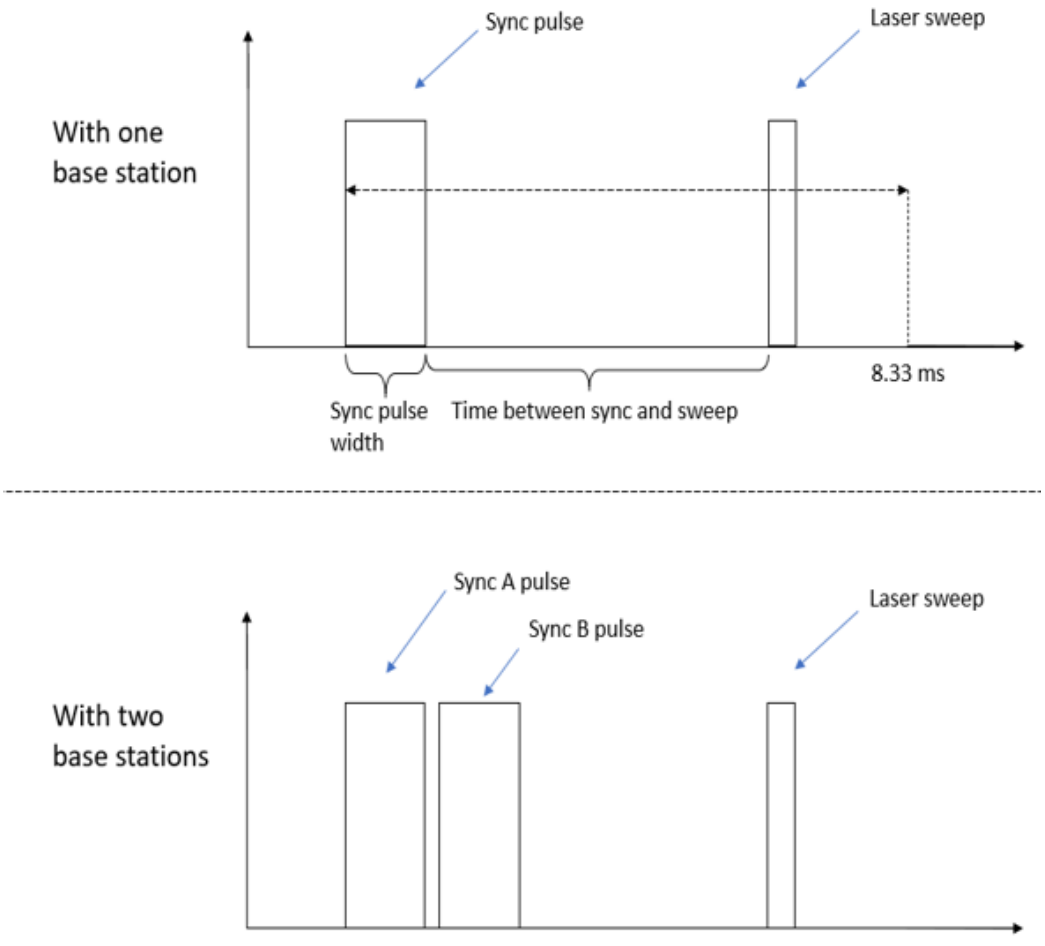

Figure 4.5: IR Pulse Classification

The sync pulse length is wider than the laser sweep, and this pulse width measurement can be used to classify the detected pulse as sync pulse or laser sweep. The above image shows that the rising edge signal implies starting a new rotation period for one of two rotors in the base station. The sync pulse length is used to identify which rotor is going to start the laser sweep next. If we are using two base stations, it tells which unit is going next as well; also it holds the information about the base station that sent this sync pulse. Below table 4.2 shows the details about how the sync pulse width can be decoded to get useful information about the laser sweep orientation and base station information.

Table 4.2: Sync Pulse Decoding

Skip	Data	Axis	Length(ticks)	Length (µs)
0	0	0	3000	62.5
0	0	1	3500	72.9
0	1	0	4000	83.3
0	1	1	4500	93.8
1	0	0	5000	104
1	0	1	5500	115
1	1	0	6000	125
1	1	1	6500	135

The length column in the above table shows the possible pulse width lengths for synchronization pulse. It can be mapped to length as the number of ticks, where each length represents a combination of 3-bit states. For the given sync pulse length, the best match of 3-bit state can be found using the below equation,

[skip,data,axis] = (length - 2501) / 500

In the 3-bit state, the axis bit determines the rotation axis (vertical/horizontal) of the laser sweep that follows the sync pulse. The skip bit determines whether the rotor will skip this cycle. It is important to consider when using two base stations, as only one will emit the laser sweep in one cycle, so the other will skip it. The data bits of consecutive sync pulses of a base station are concatenated to yield a data structure called OOTX Frame.

4.3 Robot Setup and Integration

The receiver circuit needs to be connected to the processing module for the position estimation. To avoid the complex integration process, we directly connect the IR receiver circuit to the Raspberry Pi on the robot and did the IR pulse processing. But it is recommended to use an external processing module for pose estimation calculation. Teensy can be powered with USB or through any regulated voltage supply of 5v by connecting it to the power input ports of Teensy. Once the Teensy is powered up, it can be used to power the receiver circuit. Now the receiver circuit output should be connected to the Teensy via a digital input/output port.

The processed data of position values from Teensy need to be fed into the robot controller for localization. If the robot controller has a USB port, then it can be communicated via USB, or Teensy has Tx/Rx port that can be connected to the robot controller and pass the position values via serial communication. The below figure 4.6 shows the integration setup of the receiver circuit, Teensy and robot controller.

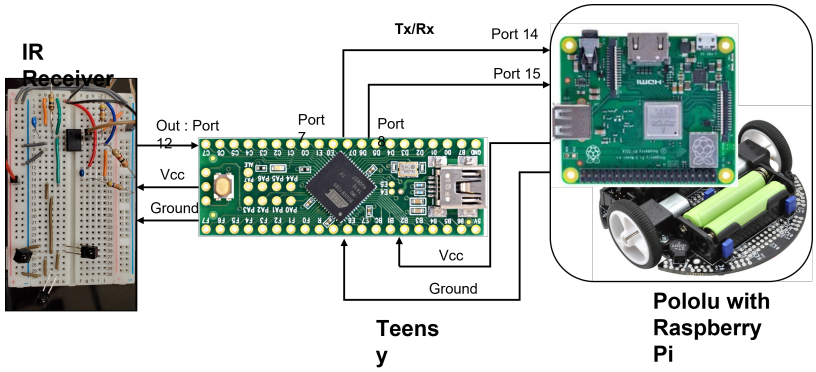

Figure 4.6: Proposed Solution Integration

4.4 Pose Estimation Algorithm

The pose estimation algorithm depends on three main parts, pulse detection, pulse classification and pose estimation. The pulse detection is to detect the IR pulses from the lighthouse base unit. After that, the detected pulse should be classified as synchronization pulse or laser sweep, and the pulse classification part will do that. The output of the pulse classification also gives us the time offset values for laser sweep, and that will be used for the pose estimation.

The IR receiver circuit detects the IR and outputs 1 or 0 to the controller (Raspberry Pi). For pulse detection, we need to detect the rising edge and the falling edge. Also, it should start the timer at the rising edge till the falling edge and record the time. This time implies the pulse length, and that is needed for pulse classification. In pulse classification, we can decode few data from the recorded pulse length, such as whether it is a sync pulse or sweep pulse; if it is a sync pulse, it also tells whether the following laser sweep is vertical or horizontal. Any sync pulse length will be around 60 microseconds to 135 microseconds, and anything below 10 microseconds will be laser sweep.

Once the laser sweep offset is calculated, it can be applied to the position estimation algorithm. We used a cross beam algorithm for the position estimation, and basically, it is applicable for two base stations. In this project, we adopted that and modified it to use it with a single base station. The cross beam algorithm's idea was to find the sensor hitting the angle of laser sweep and plotting two 3D lines from each base station. The intersection of those two lines gives the nearest point/position of the sensor.

We know the time offset between sync pulse and laser plane sweep pulse. The laser sweep plane makes half rotation (180°) in each cycle. Therefore we can calculate the sweep plane angle of arrival with equation 1. Here, the time offset is the time laser beam takes to hit the sensor, $8333\mu s$ is one cycle period and $4000\mu s$ is approximated mid value of one cycle.

$$angle = (time\ offset\ -\ 4000) \times \frac{\Pi}{8333} - - - (1)$$

Calculate 3d planes from stations: Now we need to calculate the 3D planes from the stations. Let us assume that the single base station itself is centred in origin, and Z-axis is facing the opposite direction of the base station's field of view, and Y is up and, X is left (see figure 4.7).

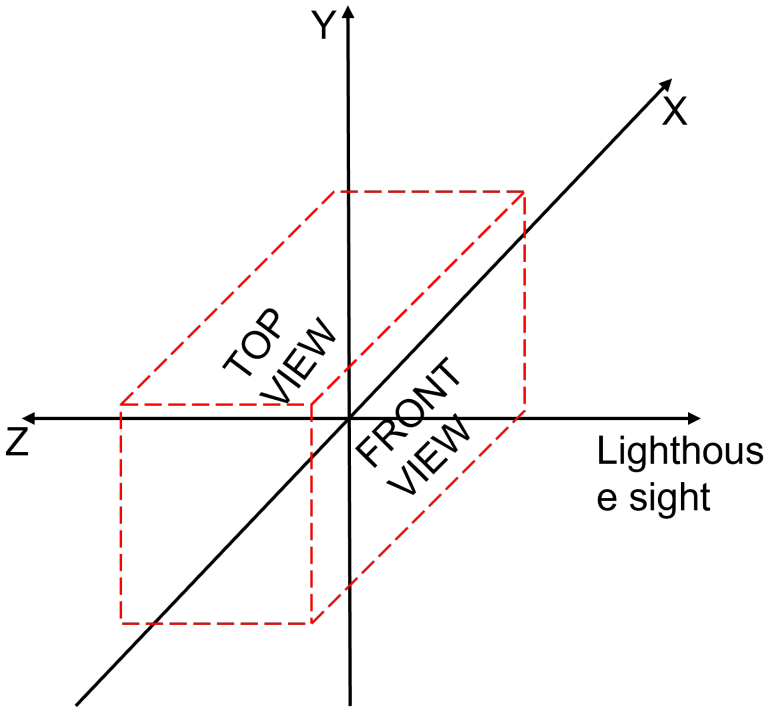

Figure 4.7: Base Station In 3D Coordinates

The horizontal laser plane rotates around the X-axis, and the vertical plane rotates around Y-axis (see figure 4.8).

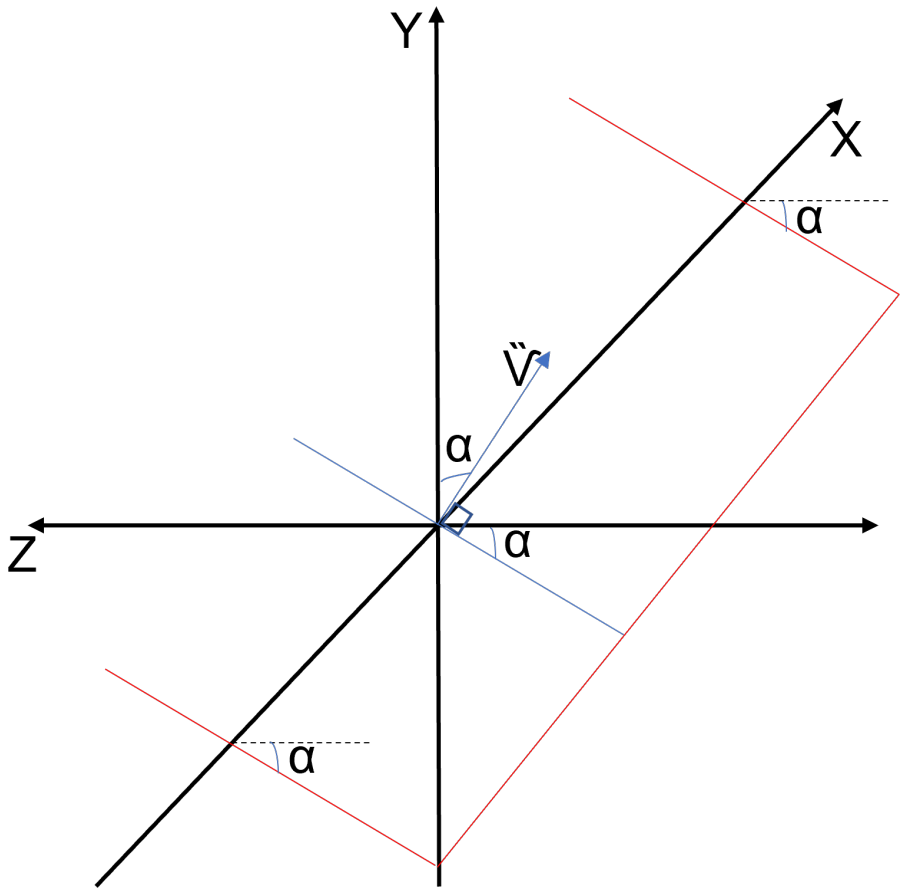

Figure 4.8: Horizontal Laser Sweep Plane

In vector space, a 3D plane can be defined by a point and a normal vector V. In horizontal laser sweep, the plane rotates around X-axis, and it is normal vector V lies on the YZ plane, and it rotates with the same rotation angle as the plane. From the following formulas, we can get the XYZ coordinates of normal vectors. The respective horizontal and vertical plane angles(α) should be calculated from equation 1.

$$Horizontal\,Normal\,Vector = (0,\ Cos(\alpha),\ Sin(\alpha)) \quad ---(2)$$

$$Vertical\,Normal\,Vector = (Cos(\alpha),\ 0,\ -Sin(\alpha)) \quad ---(3)$$

A 3D line can be described by a point and a vector (U). To get the 3D line from lighthouses, we need to intersect the planes. As a fact, this line lie on both planes of two laser sweeps at the moment of detection. The line vector U should be perpendicular to both horizontal normal vector and vertical normal vector. So that we can calculate the U by a cross product of both normal vectors as shown in equation 4.

$$U = Horizontal\,Normal\,Vector \times Vertical\,Normal\,Vector --- (4)$$

As we now know, the 3D lines from each lighthouse reference frame, it could be converted to global reference frame later. To convert the line vector frame, multiply it by the station rotation matrix (R is the lighthouse rotation matrix).

$$U_{global} = R \times U --- (5)$$

Once we know the two lines from each base stations, intersection of those lines gives us the position of the sensor/robot. For that, we need to compute two points on those two lines that are the closest to each other. For example, below equations 6 and 7 shows two lines from two base stations.

$$P(s) = P0 + (s \times U1_{global}) --- (6)$$

$$Q(t) = Q0 + (t \times U2_{global}) --- (7)$$

Wheres, s and t are scalar parameters, P0, Q0 - lighthouse origin points and U_{global} should be calculated separately for each base stations ($U1_{global}$ and $U2_{global}$).

Scalar parameter calculation:

$$W0 = P0 - Q0(difference\ between\ line\ origins)$$

$$a = U1_{global} \cdot U1_{global}(dot\ products)$$

$$b = U1_{global} \cdot U2_{global}$$

$$c = U2_{global} \cdot U2_{global}$$

$$d = U1_{global} \cdot W0$$

$$e = U2_{global} \cdot W0$$

$$d = ac - bb\ (denominator\ must\ be\ >\ 0\ for\ non-parallel\ lines)$$

$$s = (be - cd)/d\ (parameter\ of\ closest\ point\ on\ first\ line\ to\ the\ second\ line)$$

$$t = (ae - bd)/d\ (same\ for\ second\ line)$$

$$Estimated\ Position = \frac{P(s) + Q(t)}{2}(middlepoint) - - - (8)$$

P(s) and Q(t) are the closest points on corresponding lines to the other line. Computing the middle coordinate of those two points gives us the estimated position (equation 8). Distance between those points gives us the quality of the solution. This algorithm is known as cross beam algorithm [2].

The above model with cross beam algorithm works only with two or more base stations. But the main challenge we took in this project was to implement a localization solution with single base station. The Bitcraze team who implemented the Crazyflie with lighthouse positioning system, recently published a research paper [26] and they proposed, using the extended Kalman filter (EKF) on position estimation, enables robots to run with only one base station[27]. But practically, we could not implement the proposed model.

So, we came up with our own solution with a simple mathematical model to do the pose estimation with a single lighthouse unit. To implement this we need to make sure the base station is mounted at top facing the robots downwards as shown in figure 4.3. From that perspective, we can see that the vertical sweep scans one axis and the horizontal sweep scans the other (X,Y).

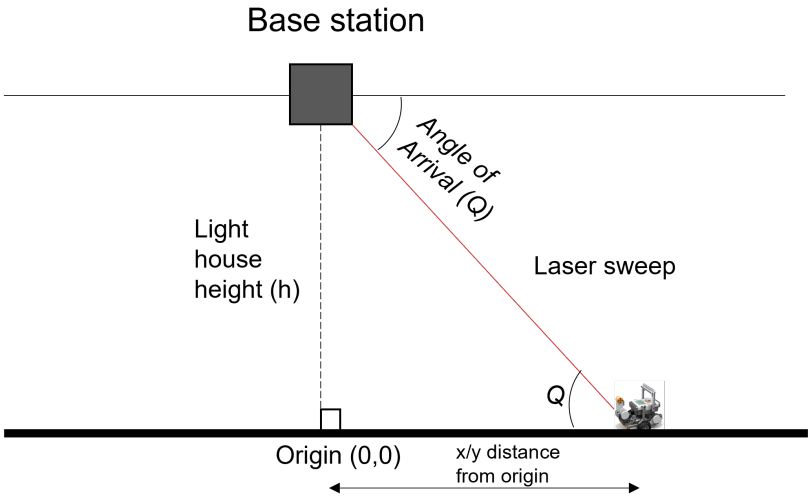

Figure 4.9: Pose Estimation with Single Lighthouse

We know the duration of one cycle (8.33ms), and it rotates through 180° degrees during one cycle. From that, we can calculate the angular velocity (ω) of rotating laser beam drums.

$$\omega = \frac{\Pi}{8.33 \times 10^{-3}} \quad ---(9)$$

From the IR receiver circuit output, we can measure the time offset value (time for the laser beam to hit the sensor). Now we know the angular velocity, we got the time the laser beam travelled to hit the sensor, so that we can calculate the angle of arrival (Q) by using equation 10. As shown in the image (figure 4.9), now it is a simple trigonometry problem. By using the tangent formula, we can solve the distance value from origin (equation 11), and this is applicable for both vertical and horizontal laser sweeps. In this way, we do not need to know the base station translation matrix or rotation matrix, just the height of the lighthouse.

$$Angle\,of\,arrival\,(Q) = Time\,of\,arrival\,\times\,\omega \quad ---(10)$$

$$Distance\,from\,origin = \frac{Light\,house\,height}{\tan(Q)} \quad if\;\; Q < 90° \quad ---(11)$$

Chapter 5

Results

The primary measurement we need for the pose estimation is the time offset values, also known as the time laser beam takes to hit the robot with the sensor.

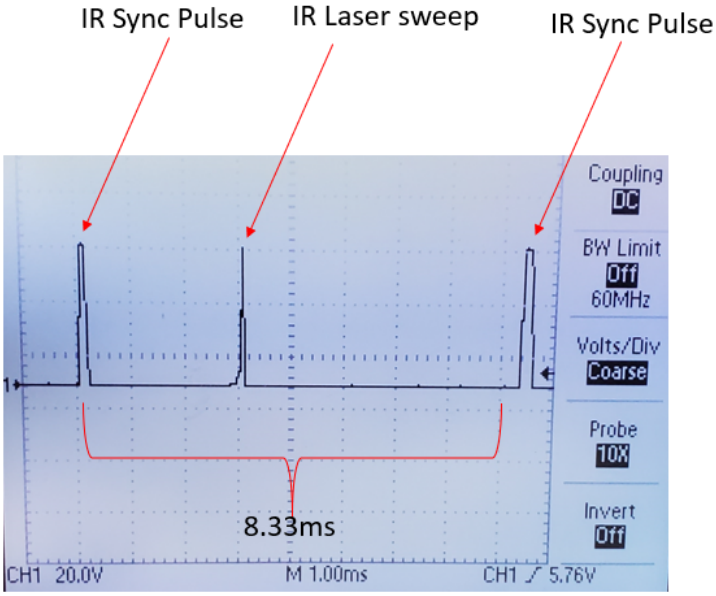

Figure 5.1: IR Circuit Output from Oscilloscope

We connected the IR receiver circuit output to the oscilloscope to observe the detected signal pattern to ensure our circuit is functioning correctly. The above figure 5.1 shows the IR circuit out for detected sync pulses and laser beam with a single base station. The length of the pulses helps us to distinguish the laser sweep and sync pulse clearly.

The laser sweep offset time values were fed into the pose estimation model and visualized the estimated position of the robot in a static state. The robot field was divided into four sections, placed the robot in each box, and plotted the estimated positions. The pose estimation results are shown below with respective time offset values for both vertical and horizontal sweeps.

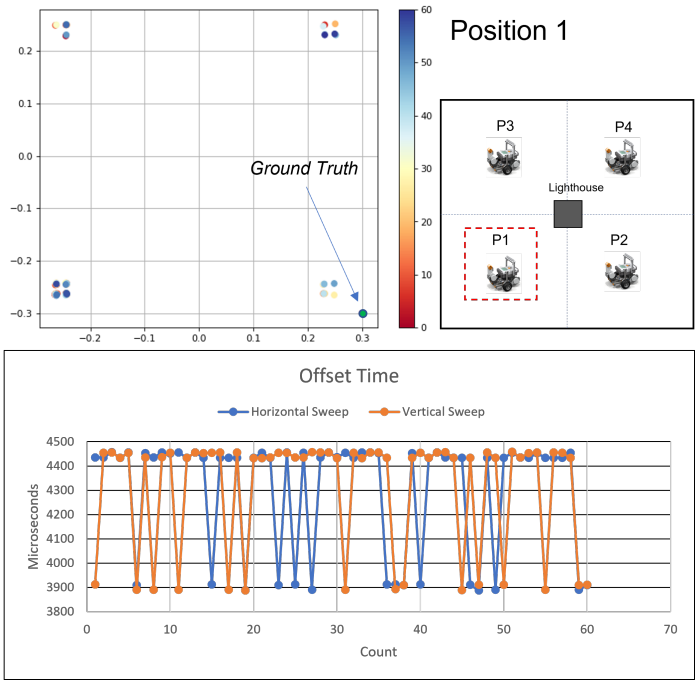

Figure 5.2: Pose Estimation at Position 1

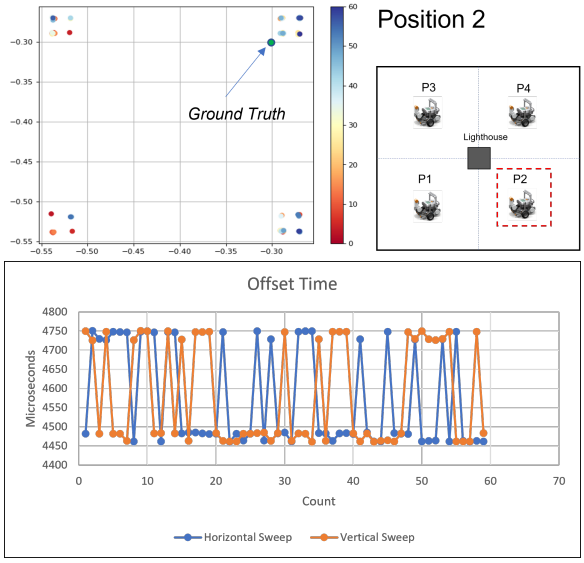

Figure 5.3: Pose Estimation at Position 2

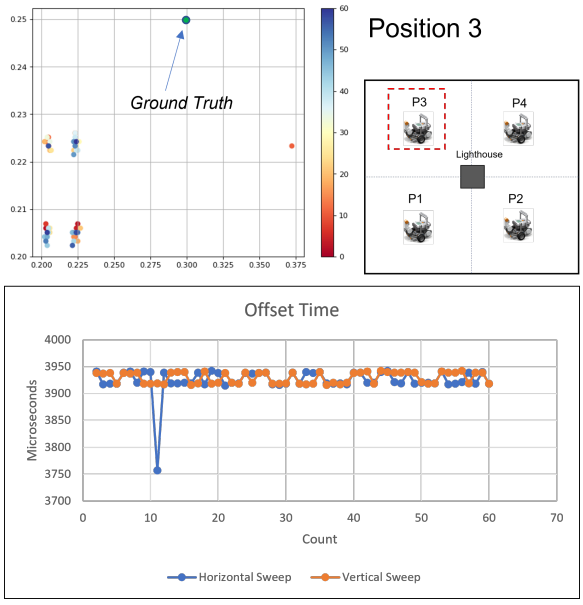

Figure 5.4: Pose Estimation at Position 3

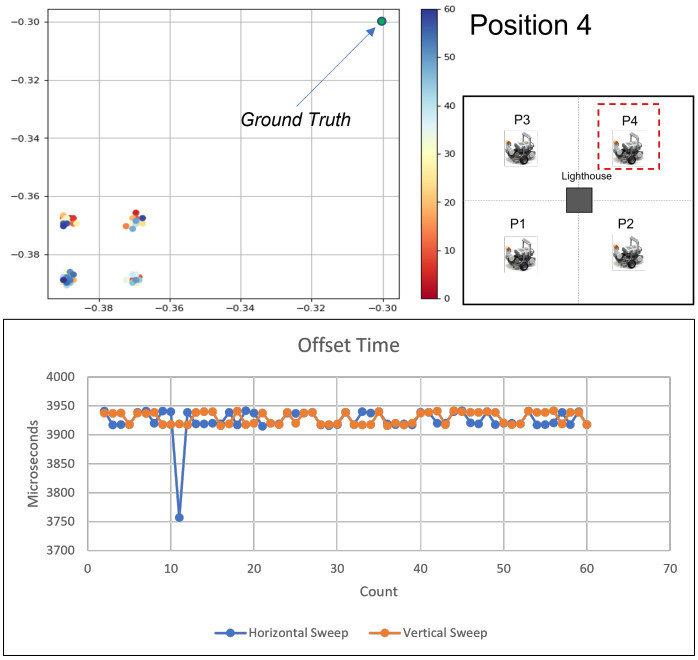

Figure 5.5: Pose Estimation at Position 4

The above visualizations were generated with sixty sample offset measurements so that it gives sixty respective estimated positions. The position plots are with color coordinates as each color implies a respective offset in the count axis. We measured the lighthouse height in meters; therefore, the estimated position graph also represents coordinates in meters from origin.

Our test case shows that the pose estimation model is so stable with the simulated time offset values. But the above images shows the estimated positions are not so accurate, and scattered around. We assume the main cause for this behaviour is the fluctuation in time offset values, as seen in each offset plot. Probably the offset time measurement should be filtered to increase the pose estimation accuracy.

Chapter 6

Discussion

HTC Vive is an affordable solution for indoor localization with great accuracy. Even though it was initially developed for VR games, researchers have lately seen its great scope in robotics localization[28]. This research project discusses the feasibility of using the HTC Vive's lighthouse positioning system for indoor localization purposes. Even though we already have many approaches for localization, indoor localization has always been a challenging task in robotics. Image-based localization may be an acceptable solution for indoor localization, but it holds a space for further development when considering latency in processing. Theoretically, using the HTC Vive lighthouse could help to overcome the time latency concern. Also, various past studies show that it can achieve millimeter precision. The affordable price of the lighthouse system and its fully distributed operation makes it particularly interesting for classroom use and distributed swarm research.

The main goal of this project was to implement a solution for improving the indoor localization position estimation by using the HTC Vive lighthouse positioning system. We already learned from previous studies, with the image-based or feature-based approach, the position update frequency would be around 17Hz. When using the lighthouse positioning system for localization, it gives position updates at a rate of 60Hz with the single base station and 30Hz with two base stations in the ideal scenario. Unless the localization is going to be applied for some high precision requirement drone application, 60Hz is good enough for mobile robot indoor localization application[29].

The pose estimation calculation and the robot control code were executed in the same onboard controller. But as already said, using a dedicated microcontroller like Teensy will help us to improve the processing performance and latency in IR detection. The proposed circuit is an acceptable solution for the IR detection from the HTC Vive lighthouse. Even though we are using a highpass filter in the circuit, we observed some noise at the circuit output. So we recommend applying a noise filtering algorithm after the laser sweep offset time measurement and pre-process it before feeding it into the model. An alternative solution for the proposed circuit was the TS3633-CM1 module. Due to the unavailability of this product in the market, we could not test it with this module.

We need to get the base station's translation and the rotation matrix to implement the pose estimation calculation with two or more lighthouse units. To do that,

we need to calibrate with the steam VR application running on a system and a VR tracker connected to it. The calibration process has been challenging as there could be a version mismatch with VR tracker and running software. Suppose we are implementing the localization solution with a single base station with the proposed model; in that case, we do not need any calibration part; the base station can be considered the origin. But it is always best to set up the steamVR gaming environment with a VR tracker and headset before starting to work with the base station as that is the only way to know how it works in the deployed environment.

With the optimized version of the position estimation algorithm, we can achieve the update rate near theoretical values even in a practical scenario[28][30]. With further development and optimization, the laser-based lighthouse positioning method can be much better than the feature-based method for indoor localization. This project implementation was not fully completed, and it still has space for further development. We mainly need to focuse on implementing a signal processing module that can perfectly detect and measure the laser sweep time-offset values without the fluctuation that we are experiencing right now. With that, the position estimation algorithm could be further improved and achieve precision with up to 3 - 10mm error.

Chapter 7

Conclusion

In this project, we studied the operation of the HTC Vive lighthouse and the feasibility of implementing a solution for indoor localization for robots with a lighthouse positioning system. One of the challenges we took in this project was implementing the localization model with a single lighthouse. Our proposed model for localization with a single lighthouse worked well with simulated measurement values. But, with lighthouse laser sweeps, we suspect there could be a defect in IR receiver circuit or processing in laser sweep offset values. By fixing the measurements of laser sweep offset, it could be further improved for localization with a single lighthouse.

The determination of the number of base stations in a solution is about a trade-off between speed and precision. When using a single lighthouse, we should compromise the pose estimation accuracy, but the pose estimation update rate would be high. The Bitcraze team's research paper[26] result shows that the precision is not so convincing

when using a single lighthouse model compared with two lighthouses. Therefore, unless the application is for lab experiments, localization with a single lighthouse is not recommended for industry-level applications where precision matters the most. This research could be carried forward with the newer version of devices for further development. Undoubtedly, investing more time and effort in localization research with the HTC Vive lighthouse positioning system would lead to a new cutting-edge approach in indoor localization for robots.

7.1 Future Work

This project centralizes the reverse engineering of HTC Vive lighthouse VR technology. But during the implementation, we did not have a few required resources (Eg: VR headset) for testing and calibration. So in the future, we recommend to initially set up a complete VR environment with base stations and make sure all are functioning and calibrated. Various studies show that version 2 lighthouses are more accurate. Therefore we should explore more the possibility of improving the localization accuracy with latest HTC Vive devices in future research works. We also recommend to development of an improved version of a software module to process the lighthouse laser sweeps and produce stable time offset values. That will mitigate the errors in pose estimation.

Bibliography

[1] Miguel Borges, Andrew Symington, Brian Coltin, Trey Smith, and Rodrigo Ventura. Htc vive: Analysis and accuracy improvement. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2610–2615, 2018.

[2] Alexander Shtuchkin. Vive-diy-position-sensor. `https://github.com/ashtuchkin/vive-diy-position-sensor`, 2017.

[3] Diederick C. Niehorster, Li Li, and Markus Lappe. The accuracy and precision of position and orientation tracking in the htc vive virtual reality system for scientific research. *i-Perception*, 8(3):2041669517708205, 2017. `https://doi.org/10.1177/2041669517708205`.

[4] Rui Wu, Jeevitha Pandurangaiah, Grayson Morgan Blankenship, Christopher Xavier Castro, Shanyue Guan, Andrew Ju, and Zhen Zhu. Evaluation of virtual reality tracking performance for indoor navigation. In *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 1311–1316, 2020.

[5] Pawel Kulakowski, Javier Vales-Alonso, Esteban Egea-López, Wieslaw Ludwin, and Joan García-Haro. Angle-of-arrival localization based on antenna arrays for wireless sensor networks. *Computers and Electrical Engineering*, 36:1181–1186, 11 2010.

[6] Jonathan Lwowski, Abhijit Majumdat, Patrick Benavidez, John J. Prevost, and Mo Jamshidi. Htc vive tracker: Accuracy for indoor localization. *IEEE Systems, Man, and Cybernetics Magazine*, 6(4):15–22, 2020.

[7] Hsin-Min Cheng and Dezhen Song. Graph-based proprioceptive localization using a discrete heading-length feature sequence matching approach. *CoRR*, abs/2005.13704, 2020. `https://arxiv.org/abs/2005.13704`.

[8] M. Aqel, M. Marhaban, M. I. Saripan, and N. Ismail. Review of visual odometry: types, approaches, challenges, and applications. *SpringerPlus*, 5, 2016.

[9] Stefan Edelkamp and Stefan Schrödl. Chapter 17 - vehicle navigation. In Stefan Edelkamp and Stefan Schrödl, editors, *Heuristic Search*, pages 737–757. Morgan Kaufmann, San Francisco, 2012. `https://www.sciencedirect.com/science/article/pii/B9780123725127000171`.

[10] Khaoula Lassoued, Oana Stanoi, Philippe Bonnifait, and Isabelle Fantoni. Mobile robots cooperation with biased exteroceptive measurements. In *2014 13th International Conference on Control Automation Robotics Vision (ICARCV)*, pages 1835–1840, 2014.

[11] Mahmoud Mosleh, Mohammed Joudah Zaiter, and Ali Hashim. Position estimation using trilateration based on toa/rss and aoa measurement. *Journal of Physics: Conference Series*, 1773:012002, 02 2021.

[12] Xingxing Li, Xiaohong Zhang, Xiaodong Ren, Mathias Fritsche, Jens Wickert, and H. Schuh. Precise positioning with current multi-constellation global navigation satellite systems: Gps, glonass, galileo and beidou. *Scientific reports*, 5:8328, 02 2015.

[13] José Darrozes, Nicolas Roussel, and Mehrez Zribi. 7 - the reflected global navigation satellite system (gnss-r): from theory to practice. In Nicolas Baghdadi and Mehrez Zribi, editors, *Microwave Remote Sensing of Land Surface*, pages 303–355. Elsevier, 2016. https://www.sciencedirect.com/science/article/pii/B9781785481598500074.

[14] E.C.L. Chan, G. (2012). Differential GPS Baciu, Assisted GPS. In Introduction to Wireless Localization (eds E.C.L. Chan, and G. Baciu). https://doi.org/10.1002/9781118298534.ch9,.

[15] Yongguang Chen and H. Kobayashi. Signal strength based indoor geolocation. In *2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No.02CH37333)*, volume 1, pages 436–439 vol.1, 2002.

[16] F. Cozman and E. Krotkov. Robot localization using a computer vision sextant. In *Proceedings of 1995 IEEE International Conference on Robotics and*

Automation, volume 1, pages 106–111 vol.1, 1995.

[17] Wei Wei, Chunna Tian, and Yanning Zhang. A two-stage facial landmark localization method. In *2015 International Conference on Orange Technologies (ICOT)*, pages 157–160, 2015.

[18] Xuyou Li, Shitong Du, Guangchun Li, and Haoyu Li. Integrate point-cloud segmentation with 3d lidar scan-matching for mobile robot localization and mapping. *Sensors*, 20(1), 2020. https://www.mdpi.com/1424-8220/20/1/237.

[19] Prabin Kumar Panigrahi and Sukant Kishoro Bisoy. Localization strategies for autonomous mobile robots: A review. *Journal of King Saud University - Computer and Information Sciences*, 2021. https://www.sciencedirect.com/science/article/pii/S1319157821000550.

[20] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. Proceedings of sixteenth National Conference on Artificial Intelligence, pp. 343 - 349, July 1999. https://dl.acm.org/doi/10.5555/315149.315322.

[21] Negenborn R. (2003). Robot localization and kalman filters—on finding your position in a noisy world. MS Thesis, Utrecht University, Utrecht. http://www8.cs.umu.se/research/ifor/dl/LOCALIZATION-NAVIGATION/Robot%20Localization%20and%20Kalman%20Filters.pdf.

[22] Hamzah Ahmad and Toru Namerikawa. Extended kalman filter-based mobile robot localization with intermittent measurements. *Systems Science & Control Engineering*, 1(1):113–126, 2013. https://doi.org/10.1080/21642583.2013.864249.

[23] Heber Sobreira, Miguel Pinto, A. Moreira, Paulo Costa, and José Lima. Robust robot localization based on the perfect match algorithm. *Lecture Notes in Electrical Engineering*, 321:607–616, 01 2015.

[24] Xingdong Liu, Xiao Luo, and Xinliang Zhong. Research on simultaneous localization and mapping of indoor mobile robot. *Journal of Physics: Conference Series*, 1074:012099, sep 2018. https://doi.org/10.1088/1742-6596/1074/1/012099.

[25] Alif Ridzuan Khairuddin, Mohamad Shukor Talib, and Habibollah Haron. Review on simultaneous localization and mapping (slam). In *2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, pages 85–90, 2015.

[26] Arnaud Taffanel, Barbara Rousselot, Jonas Danielsson, K. McGuire, Kristoffer Richardsson, Marcus Eliasson, Tobias Antonsson, and Wolfgang Hönig. Lighthouse positioning system: Dataset, accuracy, and precision for uav research. *ArXiv*, abs/2104.11523, 2021.

[27] Mark W. Mueller, Markus Hehn, and Raffaello D'Andrea. Covariance correction step for kalman filtering with an attitude. *Journal of Guidance, Control, and Dynamics*, 40(9):2301–2306, 2017. https://doi.org/10.2514/1.G000848.

[28] Marcus Greiff, Anders Robertsson, and Karl Berntorp. Performance bounds in positioning with the vive lighthouse system. In *2019 22th International Conference on Information Fusion (FUSION)*, pages 1–8, 2019.

[29] Kristian Sletten. Automated testing of industrial robots using htc vive for motion tracking. *MS. thesis, University of Stavanger, Norway*, 2017. http://hdl.handle.net/11250/2455902.

[30] Mohamed Sadiq Ikbal, Vishal Ramadoss, and Matteo Zoppi. Dynamic pose tracking performance evaluation of htc vive virtual reality system. *IEEE Access*, 9:3798–3815, 2021.