# University of Cincinnati

**Date: 8/11/2020**

**I, Anujj Saxena, hereby submit this original work as part of the requirements for the degree of Master of Science in Mechanical Engineering.**

It is entitled:

**Robot Localization Using Artificial Neural Network Under Intermittent Positional Signal**

Student's name: **Anujj Saxena**

This work and its defense approved by:

**Committee chair: Manish Kumar, Ph.D.**

**Committee member: Janet Jiaxiang Dong, Ph.D.**

**Committee member: David Thompson, Ph.D.**

36905

University of Cincinnati

Mechanical Engineering

# Robot Localization Using Artificial Neural Network Under Intermittent Positional Signal

A thesis submitted to the Graduate school of the
University of Cincinnati in partial fulfillment of the requirements for the degree of

*Master of Science*

in the Department of Mechanical Engineering of
the College of Engineering and Applied Science
by

**Anujj Saxena**

M.S. University of Cincinnati

November 16, 2020

Committee Chair: Dr.Manish Kumar, Ph.D.

# Abstract

Unmanned Aerial Vehicles (UAV) are gaining attention in the civilian domain with their numerous potential applications. This has been demonstrated recently in light of developments around the pandemic, where UAVs were used by law enforcement departments of various countries of the world. Multinational Corporations such as Mercedes Benz partnered with Matternet for drone-based deliveries in Switzerland. Ford recently filed a patent for a drone system that can be integrated with a car that could provide emergency services. UAVs rely very much on positional signals for navigation. Positional signals such as a global positioning system (GPS) are susceptible to an outage for periods ranging from one second to a minute. This work provides a novel approach by introducing an Artificial Neural Network (ANN) in the cases where there are long gaps in positional signal received by a UAV. During our prior research, similar problems were manifesting during bridge inspection during flights flown by the drones. Even in our experiments with indoor localization systems using 'Decawave', we faced similar problems. Decawave comprises Ultra-Wide-band modules that use Positioning and Networking Stack (PANS), a software library, that implements the Two-Way-Ranging method for localization. In the proposed work, an ANN is trained on drone dynamics for a pre-traveled path. Then this pre-trained network, during flight, uses back-propagation to update its weights/parameters in an online fashion, where-by it learns to "fill in" the GPS signal gaps by predicting the dynamics. In the event of a GPS Signal loss, this ANN, receiving the current state of the body as input, performs a forward propagation to predict the rigid body dynamics for the next state. The online learning capability ensures that this ANN's weights are updated to reflect changing dynamics arising from changes such as different payloads. The results highlight a comparative study between a drone that implements only Extended Kalman Filter (EKF) and one that uses the suggested new approach with ANNs, and show the advantages of the proposed approach over the traditional EKF based approaches.

# Acknowledgements

विद्‌यां ददाति विनयं, विनयाद् याति पात्‌रताम्।
पात्‌रत्‌वात् धनमाप्‌नोति, धनात् धर्‌मं ततः सुखम्॥

Knowledge gives discipline, from discipline comes worthiness, from worthiness one gets wealth, from wealth one do good deeds, from good deeds comes joy.

Foremost, I would like to respectfully thank Professor Dr. Manish Kumar for his knowledge, guidance and support throughout this work. I would like to express my gratitude for various projects, for the subject 'Decision Engineering' and a lot many things.

I would like to thank my committee members Dr. David F. Thompson and Dr. Janet Jiaxiang Dong.

I am thankful to everyone at Cooperative Distributed Systems Lab Aditya, Rumit, Zach, Oyinda, Ashwin, Rohit, Drew, and many others for their help during experiments. Thanks to Madhvan for those little to elongated discussions on drones and controllers.

A heartfelt thanks to Kevin, Sparsh and many others who were a part of this journey. Lastly but most importantly, I would like to appreciate constant support and encouragement of my parents. My deepest love, admiration to my elder brother Ankit Saxena for believing in me.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Unmanned aerial vehicle, commonly known as a drone, is a flying robot or aircraft without an on-board human pilot. In general, a drone system consists of a UAV platform, a ground control station, and a system that maintains communication between the two. With change in time these drones have increased autonomy. Now a days these drones can be controlled by stationed human pilots or autonomously utilizing on board computers. Drones were developed for tasks identified as ”dull, dirty or dangerous” as discussed in the article [2]. Initially developed for military purposes, in this day and age drones are gaining significant momentum in civilian applications such as agriculture, policing, first responder, product deliveries, aerial photography, infrastructure inspections and many more. FPV drone racing is also an upcoming sport. There is an estimated growth in drone sales from $4.4 billion in 2018 to $63.6 billion by 2025, with an estimated shipment of 29 million by 2021[3].

## 1.1 Motivation and Objectives

In our prior research, one of the requirements was to inspect bridges in the Ohio region using drones.To generate an accurate 3D model for bridge inspection. A 3D rendering software such as PX4D requires an accurate GPS coordinates of the images gathered. While capturing pictures for creating the models, one of the problems was intermittent GPS signal (fig 1.1) particularly when drone went underneath the bridge where GPS

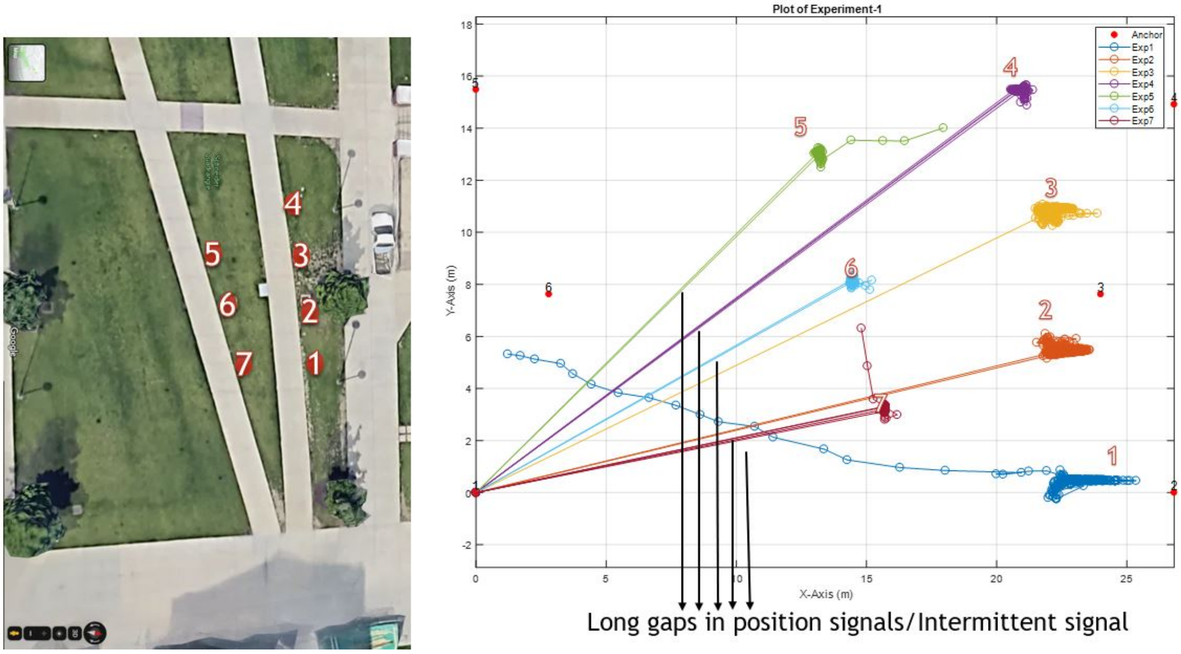


Figure 1.1: Intermittent Signal

signals were usually not available. Therefore an alternative localization tool was required especially in the cases such as under the bridge. This led to our work firstly experimenting in the field of WiFi based triangulation method.

The work involved experimentation with WiFi routers commonly available in the market. The router used during the initial phase of the experiments is 'NETGEAR 5000'(fig 1.3), Where signals were received using a raspberry-pi model 3 (fig 1.2). The work involved evaluating the drop in received signal strength. During experiments, it was found that with standard WiFi devices and receivers such as raspberry pi 3, it was hard

Figure 1.2: Raspberry Pi

https://i1.wp.com/www.extremetech.com/wp-content/uploads/2019/11/NETGEAR-Nighthawk-Pro-Gaming-XR500-640x495.jpg?resize=640%2C495&ssl=1

Figure 1.3: NETGEAR 5000

to find a gradual drop in Received Signal Strength Indicator(R.S.S.I) with distance (ref figure). After a series of experiments with WiFi and a raspberry-pie 3 module discussed in details later in the chapters, it was concluded that though these devices work, the results are not accurate enough to solve the problem. Therefore we started to look into other localization tools available in the market. During our search we came across a device named Decawave, which uses Ultra Wide Band (UWB) technology for localization. Ultra Wide Band is an upcoming technology discussed briefly in [4].

Decawave is built mainly for indoor localization and hence gives good results in an indoor environment. But for Decawave to work in an outdoor environment, firmware changes were made and tested, which are discussed further in the thesis. In one of our last experiments we were able to successfully experiment these devices in an outdoor soccer field. While experimenting with these devices in the outdoor environment, we observed another issue of intermittent signal or a long gap in positional signal. While experimenting with these devices it was found that due to some firmware issues there were few of 5~10 seconds long gaps where no signal was received from these devices(fig).

Small Unmanned Aerial Vehicle (UAV) navigation depends on the combination of global positioning system and Inertial Measurement Unit (IMU) to determine accurate position, velocity and attitude. Even in cases of autonomous flight, reliability lies more on the sensor part of the UAV, as sensors affect the feedback received by a controller. Hence for this thesis, we implemented an artificial neural network, training itself using back propagation in an online fashion. When there is no measurement from GPS for a long time this neural network predicts the next state of the drone.

Artificial Neural Network (ANN) is an upcoming technology requiring good understanding of mathematical principles for its implementation. With advancements in various tools such as Tensor Flow and Pytorch, Artificial Neural Network implementation has eased. However, it still needs an understanding of the mathematics behind neural network and knowledge about Unmanned Aerial Vehicles. Work in this field is related to a variety of methods implementing Artificial Neural Network to improve drone waypoint tracking. PD and PID type controllers are standard in the industrial world. The well-established rules for parameter tuning make them preferable in real-time applications. As discussed in [5], when addressing from a practical viewpoint, the connection between the UAV and the controller is maintained at pwm level. Henceforth, in the work [6], the author talks about utilizing an ANN based nonlinear module to convert the controller output to pwm signal. On the other hand, there are limitations in knowing all the required states for achieving control objectives. In [7] the paper discusses the implementation of Artificial Neural Network to predict states of a helicopter from the feedback control consisting of a kinematic control to generate a desired velocity for the dynamic control, a virtual controller and an optimal controller. Then from the approach discussed in [8] with acceptable results of prediction error less than 2.0 meters, the work [8] assumes that the approach discussed should have limited dependency on the vehicle control system and dynamic models, as getting data is a challenge since manufactures are not willing to share their control system.

There are approaches discussing the implementation of ANN at controller input or output. There is an approach where it is discussed that the controller output can be used as an input for the ANN, and ANN output will be used to determine the pwm signal, thus solving the challenge of underactuated mechanical system[6]. In the second approach, the states of the Unmanned Aerial Vehicle are predicted using the feedback from the controller consisting of positional controller and attitude controller. Then from [8], it can be implied that trajectory modeling can be used in the cases where the vehicle control system and dynamic model is unknown, as it is something not shared by the manufacturers. The system can be accurately trained and tested on the data obtained from the trajectory. Combining the three approaches and answering the challenges that are, first identifying the area where ANN can be implemented, second the input and the output parameters of it, and third the way it can be trained and use, we developed a simulator with scope of implementing an Artificial Neural Network. This ANN is supposed to improve the feedback at the sensor fusion level. A network that is trained on the trajectory model, based on the previous states of UAV, and the overall model predicts the upcoming states, not relying on the type of control system.

## 1.2 Contributions

The work presented here represents an advancement in the field of robot localization. Firstly, we tweaked indoor localization devices and determined standards that can help us in developing a research platform, that enables outdoor localization, where GPS is denied or scanty. The development involved a series of indoor and outdoor experiments along with firmware enhancements. All this equipped the system with better outdoor localization performance.

Secondly, during our experiments we identified another challenge. An intermittent signal or a long gap in the positional signal. This long gap troubles a lot of localization devices and results in an unstable drone flight. First, to solve it, we developed our own simulator. A simulator that is written in python. A scalable module, developed using the concepts of Object-oriented programming giving it a modularity advantage, flexibility for performance improvement and developer-friendliness.

Second, unique to this work, we developed a module for the implementation of artificial neural networks in this simulator. When we started this work, there were no popular simulators available for it. Therefore, we built a simulator that can ease implementation of Artificial Neural Network based model.

Third, this work presents a comparative study between only the Extended Kalman Filter(EKF) and suggested new approach with ANN. This online learning capability tested in the simulator ensures that ANN's weights are updated to reflect changing dynamics arising from changes such as different payloads.

## 1.3 Thesis Organization

The overall goal of this thesis is to highlight the long gaps in position signal, for robot localization, and the study of an Artificial Neural Network approach to address it.

Chapter 1 is a general introduction to drones. This chapter discusses the motivation for the thesis and advancement in UAV technology.

Chapter 2 of this thesis delves into various localization terminologies and technologies, the literature behind these technologies, and some alternative work published in this field.

Chapter 3 discusses in detail the implementation of an indoor localization module ”Decawave”, and it’s implementation in an outdoor environment. Experimental work, results, conclusion and,formulation of the idea from the experiments is discussed in the chapter.

Chapter 4 elaborates the design parameters considered to develop a simulator that can provide an environment to test the idea.

Chapter 5 discusses the results of this comparative study. In this chapter one can find a detailed discussion on the progress of the work and graphs related to the same.

Chapter 6 concludes this work with a short discussion on how the simulator can be used to take this work further and also discusses the direction it take in the future.

# Chapter 2

# Background and Literature Survey

The work progressed in three different phases. In the first phase we researched about commonly used ranging techniques using WiFi network. In the second phase we experimented with indoor localization devices using Ultra Wide Band. And in the last or third phase we developed a UAV simulator and implemented an Artificial Neural Network module in it.

## 2.1 Localization and Ranging techniques

**Localization**[9]: The knowledge of it's own position, by a robot, with respect to it's environment, based on the measurement data for that instance is called localization. It is an important process in terms of navigation and mapping by a robot. Error in localization has a chain effect, causing hindrance in the navigation of the robot.

The purpose of localization is to boost a robot with intelligence. From [10],[11] and [12] it can be concluded that sonar and lidar are very effective sensors in an indoor environment as they can very well identify corridors, flat walls and other structural regularities. But in an outdoor environment these sensors become less effective due to structural irregularities. Similarly, there is non-availability of GPS in partially enclosed environments such as patios, bridges, closed campuses etc. or where there is no satellite visibility [13].

Well known methods for positioning of robots are Kalman filtering, Markov methods, and Monte Carlo localization. Among which one can find a lot of good papers on Monte

Carlo based localization. In these papers various categories of vision-based systems have been discussed. For example, first according to two image view types, where some system use ground-view images [14], [15] and other use omni-directional images [16], [17], [18]. Second based on weather or not the system is provided with a map. As discussed in [19], where a combination of vision and laser has been used to perform SLAM. A 3D laser is used for localization and loop closure through camera which is a unique approach implementing a 3D laser-vision SLAM system with automated loop closure system. Ideas like this provide a good insight into methodologies that can be implemented for outdoor implementation of SLAM. In any case SLAM described well in [20], [21], [22] is still an active branch of research in robotics.

**Ranging**[23]: Common techniques for ranging are as follows:

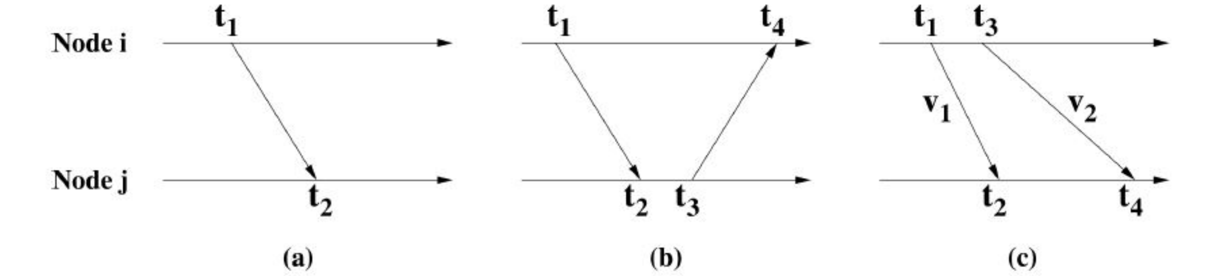
- **Time of Arrival** (ToA, time of flight): In this technique, distance is measured by using known signal propagation time and known signal velocity between sender and receiver.

- **One-way ToA**: In this method, a signal is propagated one way. To implement this method an accurate synchronization between the sender and receiver clock is required.

$$dist_{ij} = (t2 - t1)Xv$$

- **Two-way ToA**: In this method, a signal round trip time is measured at the sender device. A third message is necessary at the receivers end to calculate it's location.

$$dist_{ij} = \frac{(t_4 - t_1) - (t_3 - t_2)}{2} Xv$$



- **Time Difference of Arrival(TDoA)**: In TDoA technique, two radio signals with different velocities (generally because of two different devices) are sent with no clock synchronization, which can measure distance accurately by utilizing additional hardware. Suppose a signal is sent at $t_1$ and received at $t_2$, followed by an acoustic signal (sent at $t_3 = t_1 + t_{wait}$ and received at $t_4$) then -
  $dist = (v_1 - v_2)X(t_4 - t_2 - t_{wait})$

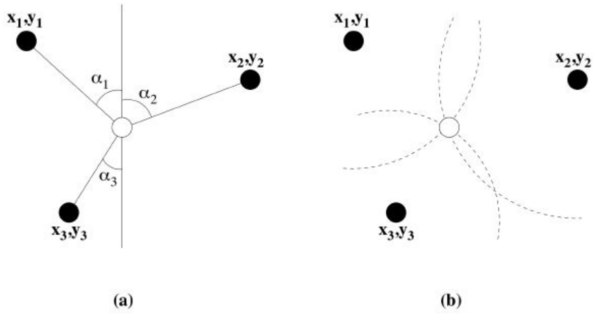
- **Angle of Arrival (AoA)**: This method very much depends upon the direction of signal propagation, achieved using an array of antennas or microphones. This method can give high accuracy depending on orientation, spatial separation of the antennas or microphones leading to differences in the arrival times, amplitudes, and phases.

- **Received Signal Strength(RSS)**: This approach relies on the phenomenon that signal decays with distance.Now a days devices measure signal strength using received signal strength indicator(RSSI). In open spaces RSS degrades with $distance^2$ and is expressed by *Friis transmission equation*:

$$P_r/P_t = G_t G_r \lambda^2/(4\pi)^2 R^2$$

Utilizing above ranging techniques, following methods are utilized to achieve localization:

- **Triangulation**: It is a range based localization employing geometric properties of a triangle to estimate location. It relies on angle and the distance between the anchor nodes in a two dimensional space.



## 2.2 Ultra Wide Band / Decawave

Ultra wide band is a low energy, short range, high band width radio communication technology. It can be used over a large portion of the radio spectrum.

Recently, ultra wide band has gained popularity as a WiFi based positioning system for precise localization. One such example is [24] where, authors are proposing an adaptive Kalman filter to solve positioning problem of intelligent luggage following. Two way ranging approaches face significant drawbacks while scaling, because of exchange of multiple frames during each ranging. In paper [25] the author puts forward and validates a unique approach for a multi-user time-difference of arrival(TDOA) based localization system utilizing wireless clock synchronization. In the experiment designed, accuracy of the system is measured using a robotic movement and an optical reference system (Mocap system) against the Decawave system. The paper puts forth that good accuracies are achievable by the time difference of arrival positioning with *wireless clock synchronization* and also claims that the results obtained are comparable with two way ranging based approaches.

There are papers such as [26] where a low cost time of arrival(TOA) based localization system is discussed. Paper [27] discusses about an experiment, where 90% quantile (which means in normal/Gaussian/bell curve) of the two-dimensional positioning error is in the range of 20 cm. In another reference [28], it is highlighted in the introduction that TWR is not capable of scaling to a significant amount of tags, which was also observed by us during extensive testing of the hardware.

Based on the above survey, we looked into paper [28] where methods for global position estimation using inertial sensors, molecular vision, and ultra wide band(UWB) sensors are discussed. The paper discusses an improved global pose estimation without using a simultaneous localization and mapping framework, supported by a small number of easy to hide (UWB) beacons with known positions. However, this paper uses a constant velocity model and over a ground robot, which is a different case from our interest, where we want to estimate accurate position of the drone without any loss of signals.

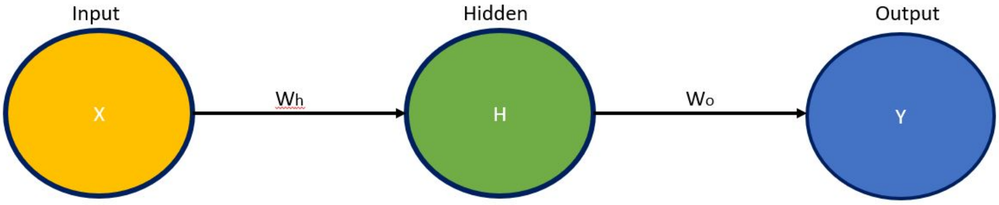


Figure 2.1: One-Dimensional Neural Network

## 2.3 Artificial Neural Networks

An artificial neural network performs tasks on the basis of examples.In general, it learns without being programmed with task-specific rules [29].

**Forward Propagation:**[30] Forward propagation is the way neural network makes predictions. In forward propagation input data is successively propagated through network layer to the final layer which outputs a prediction. For example, in Fig. 2.1 a single pass of forward propagation can be written mathematically as:

$$Prediction = A(A(XW_h)W_o)$$

**Backward Propagation:**[31] Backward propagation is a machine learning algorithm for training feed forward neural network in case of supervised learning. In Back propagation the algorithm computes the gradient of the loss function with respect to the weight of the network for a single input output example.

# Chapter 3

# Localization using Decawave: Experiments

Wireless sensor networks (WSN) based localization technologies are gaining popularity in precise location based applications. A similar experiment, where UWB devices were used to localize drone in an indoor environment, is discussed in [32]. Indoor experimental results show that 20 cm level of precision can be achieved using DecaWave; also verified in [33]. During our experiment, we used a kit named MDEK-1001 (quick detail available at [34]) containing 12 such modules. The overall Decawave antenna works at 6.5 GHz. It also constitutes a 3-axis accelerometer and implements Two-Way-Ranging technique. Other technical details about the hardware are available in [35]. The technique used in our approach is described as DWM1001 Two-Way-Ranging Real Time Location System (DRTLS) in [36]. Network components for the same are shown in Fig. 3.1. In this system a bunch of DWM1001 modules are configured as an anchor(fixed in place), a tag(mobile) or a bridge node on a gateway. A gateway is an optional unit in the system that connects the system to outside network. These gateways provide access to configure the DRTLS, and enable to display tag's location at local or remote application. For our case we used a Raspberry Pi 3 model B as a gateway hardware. To pivot the whole system across Raspberry Pi as an on board computer.

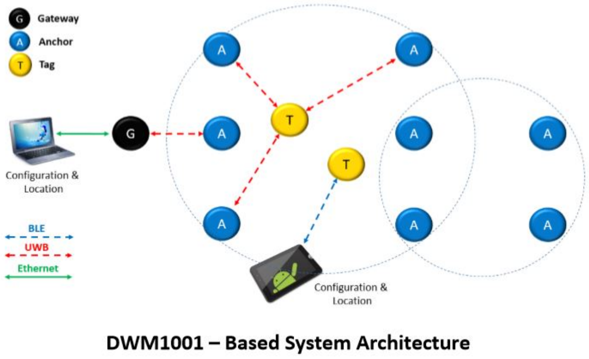


Figure 3.1: Decawave Based System Architecture

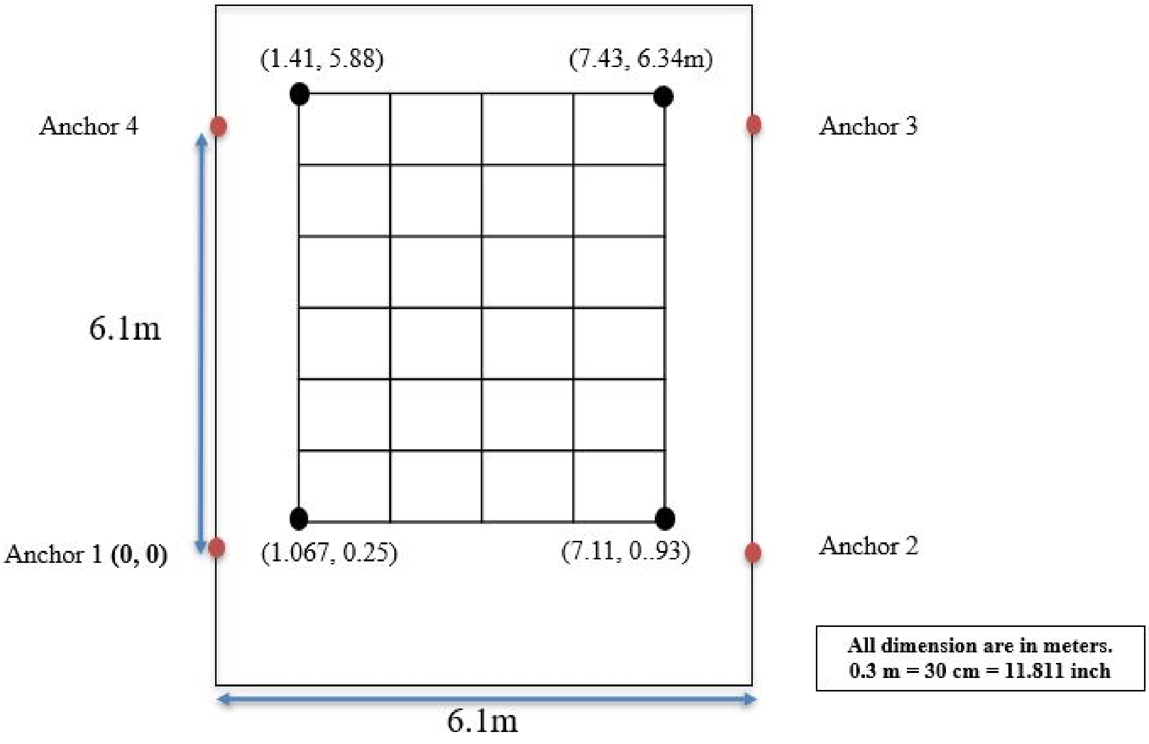


Figure 3.2: Experimental setup for indoor testing

## 3.1 Indoor testing of the hardware:

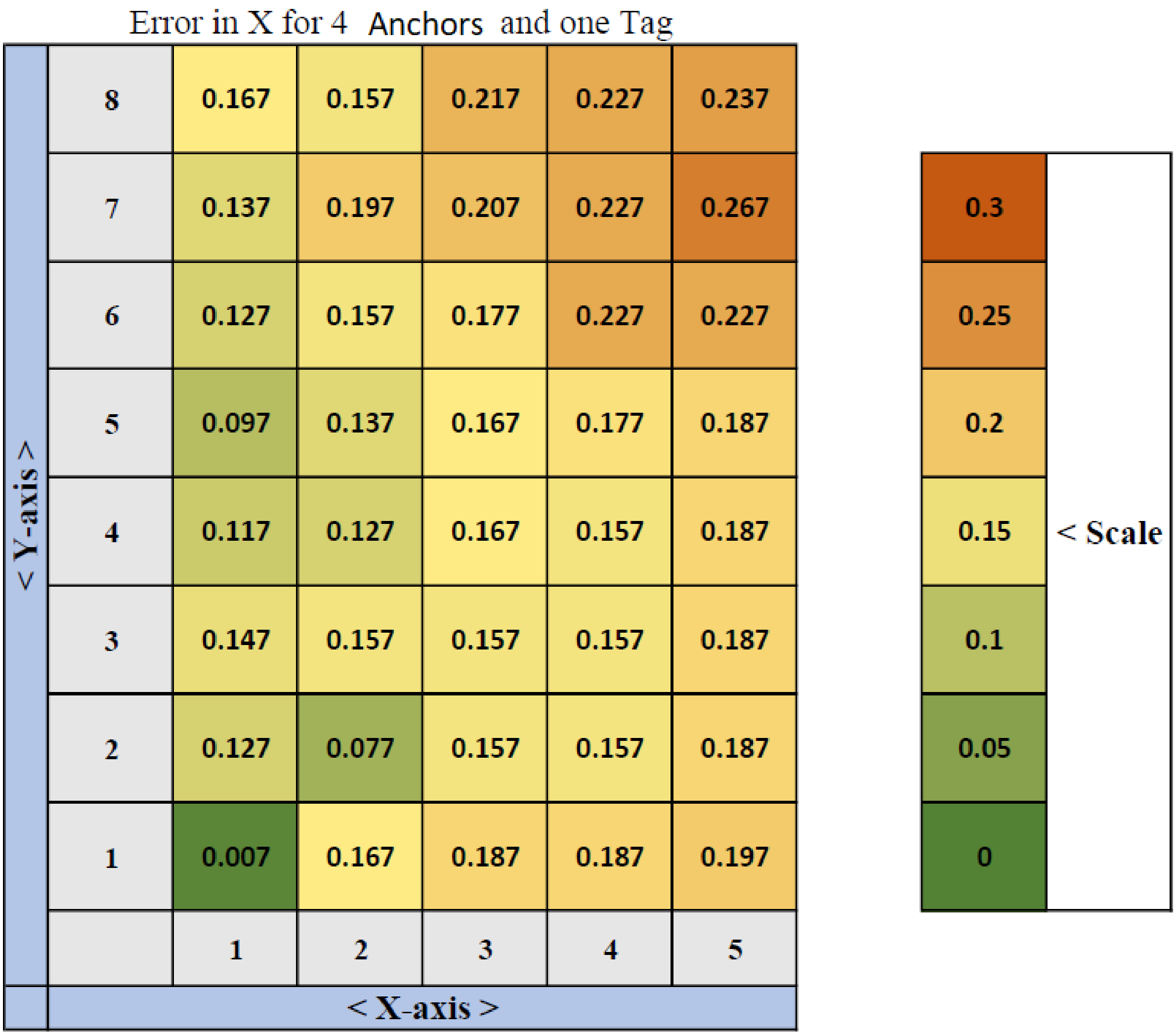
In the paper [37], the author discusses the difficulties in acquisition and tracking of GPS signal in an indoor environment. The author indicates the longer integration time needed in case of successful acquisition and tracking of GPS signal. The paper discusses an improvement in results using cellular networks. In any case, a major contributor that makes Decawave an obvious choice is it's wireless clock synchronization as discussed in [38]. However, to have confidence on these devices, we conducted our own indoor experiments.

### 3.1.1 Experiment 1: Heat map of accuracy

For our first indoor experiment, we used 4 anchors and one tag to measure accuracy at various points in the room, as in Fig. 3.2. All errors and distances are in meters. Based on the experimental work, we plotted the heat map of accuracy as shown in the Fig. 3.3 and 3.4. The room size was 6.1m X 6.1m with no obstacles in the room and all the communication channels are kept in line of sight. The tag is moved in a region enclosed by anchors.

**Experimental Procedure**:

- Calibration of the system (ref Fig. 3.2): In this step, we placed anchors at the required spots in the room, indicated by the black dots in Fig. 3.2 and then let the system generate it's own map first. Once we got an estimate by the decawave PANS, we used to manually measure the distances and enter the correct distances in the mobile application provided along with the hardware. The end of this step is a precisely working anchor system where each anchor knows about its exact location in Cartesian coordinate system.
- Static reading between all anchors with a step size of 4 feet (ref Fig. 3.3 and 3.4): In this step we navigated through the whole room in a manner discussed. Moved 4 feet, stayed for 1 min, made and noted an observation, and moved to the next step. We traversed horizontally and then vertically in a zig-zag pattern.



Error in X for 4 Anchors and one Tag

| < Y-axis > | | | | | |
|---|---|---|---|---|---|
| 8 | 0.167 | 0.157 | 0.217 | 0.227 | 0.237 |
| 7 | 0.137 | 0.197 | 0.207 | 0.227 | 0.267 |
| 6 | 0.127 | 0.157 | 0.177 | 0.227 | 0.227 |
| 5 | 0.097 | 0.137 | 0.167 | 0.177 | 0.187 |
| 4 | 0.117 | 0.127 | 0.167 | 0.157 | 0.187 |
| 3 | 0.147 | 0.157 | 0.157 | 0.157 | 0.187 |
| 2 | 0.127 | 0.077 | 0.157 | 0.157 | 0.187 |
| 1 | 0.007 | 0.167 | 0.187 | 0.187 | 0.197 |
| | 1 | 2 | 3 | 4 | 5 |

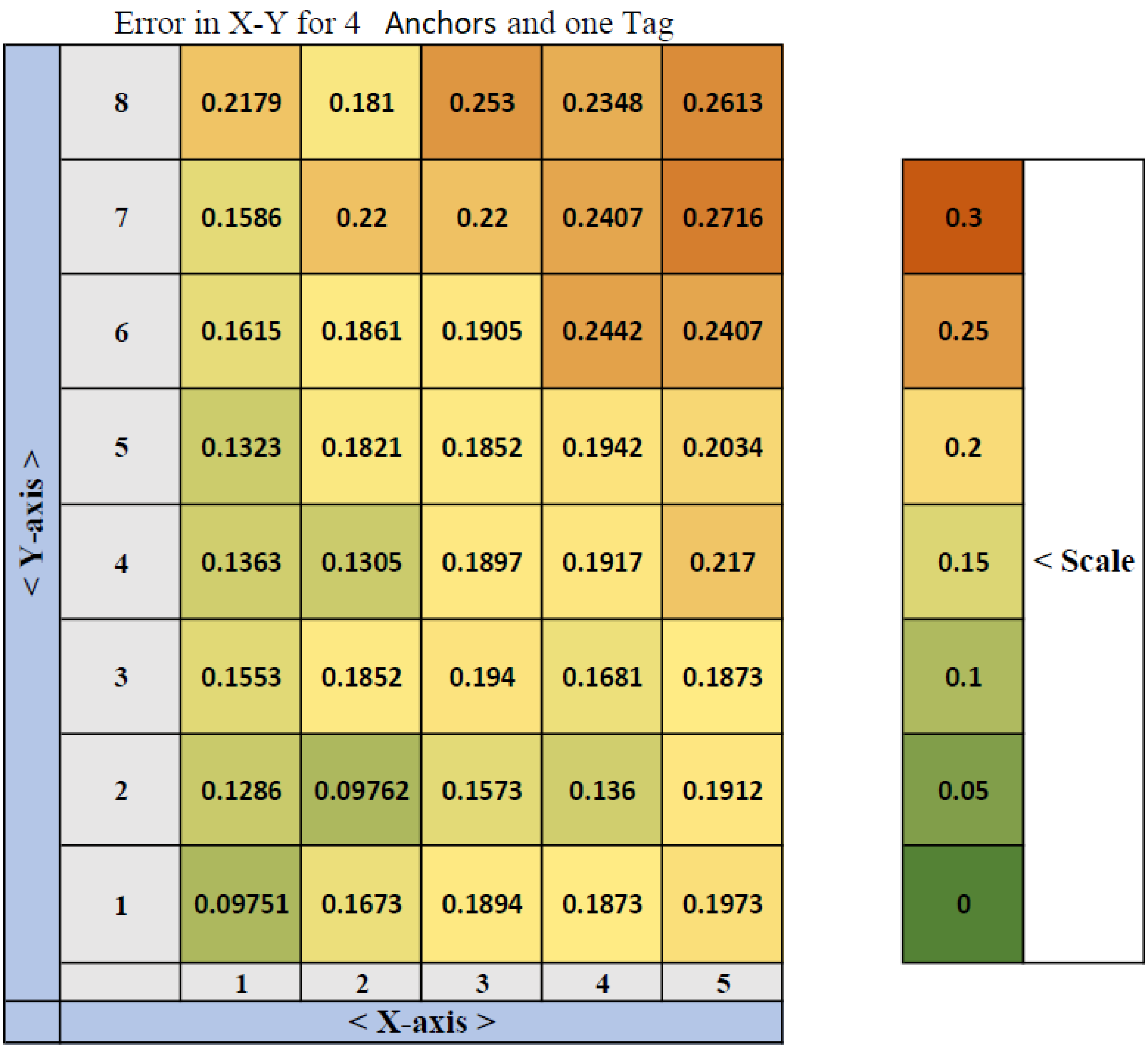
< X-axis >

Figure 3.3: Heat Map from indoor testing only X

### 3.1.2 Experiment 2: Reliability on Decawave Positioning and Networking Stack

In our second experiment instead of calibrating the system we completely used the Decawave Positioning and Networking Stack to identify the location of the a third device based on other two.

**Conclusion:** From this and similar experiments, it was concluded that, Decawave can indeed give an indoor accuracy of <30 cm.



Error in X-Y for 4 Anchors and one Tag

| < Y-axis > | | | | | |
|---|---|---|---|---|---|
| 8 | 0.2179 | 0.181 | 0.253 | 0.2348 | 0.2613 |
| 7 | 0.1586 | 0.22 | 0.22 | 0.2407 | 0.2716 |
| 6 | 0.1615 | 0.1861 | 0.1905 | 0.2442 | 0.2407 |
| 5 | 0.1323 | 0.1821 | 0.1852 | 0.1942 | 0.2034 |
| 4 | 0.1363 | 0.1305 | 0.1897 | 0.1917 | 0.217 |
| 3 | 0.1553 | 0.1852 | 0.194 | 0.1681 | 0.1873 |
| 2 | 0.1286 | 0.09762 | 0.1573 | 0.136 | 0.1912 |
| 1 | 0.09751 | 0.1673 | 0.1894 | 0.1873 | 0.1973 |
| | 1 | 2 | 3 | 4 | 5 |
| | < X-axis > | | | | |

| | |
|---|---|
| 0.3 | < Scale |
| 0.25 | |
| 0.2 | |
| 0.15 | |
| 0.1 | |
| 0.05 | |
| 0 | |

Figure 3.4: Heat Map from indoor testing X-Y both

## 3.2 Outdoor testing of the hardware

### 3.2.1 Viability of System in outdoor environment.

To check the viability of the system in an outdoor environment, we conducted an experiment with three different anchor distance configurations.
Objectives for the experiment:

- Firstly, to verify that these devices can work efficiently up to 60m outdoors.
- Secondly, to check signal continuity in the outdoor environment.
- Thirdly, to check reliability of the signal in outdoor environment.

In this set of experiments, we arranged 4 anchors at three different rectangular configuration:-

- 14.9m X 26.8m (Experiment1.1)
- 30.5m X 26.8m (Experiment1.2)
- 46.65m X 26.8m (Experiment1.3)

**The Following are observations and conclusions from the set of experiments conducted; referring to Fig. 3.5 :**

- Long Gaps in Positional Signal (LGPS) were observed in all three experiments. These LGPS increase as the configuration gets bigger.
- Number of anchors in contact falls as the configuration is made bigger i.e. in experiment 1, we can see that tag was able to connect with 4 anchors more times as compared to experiment 2, and in experiment 3, we observed instances where it was not able to communicate with any of the anchors.

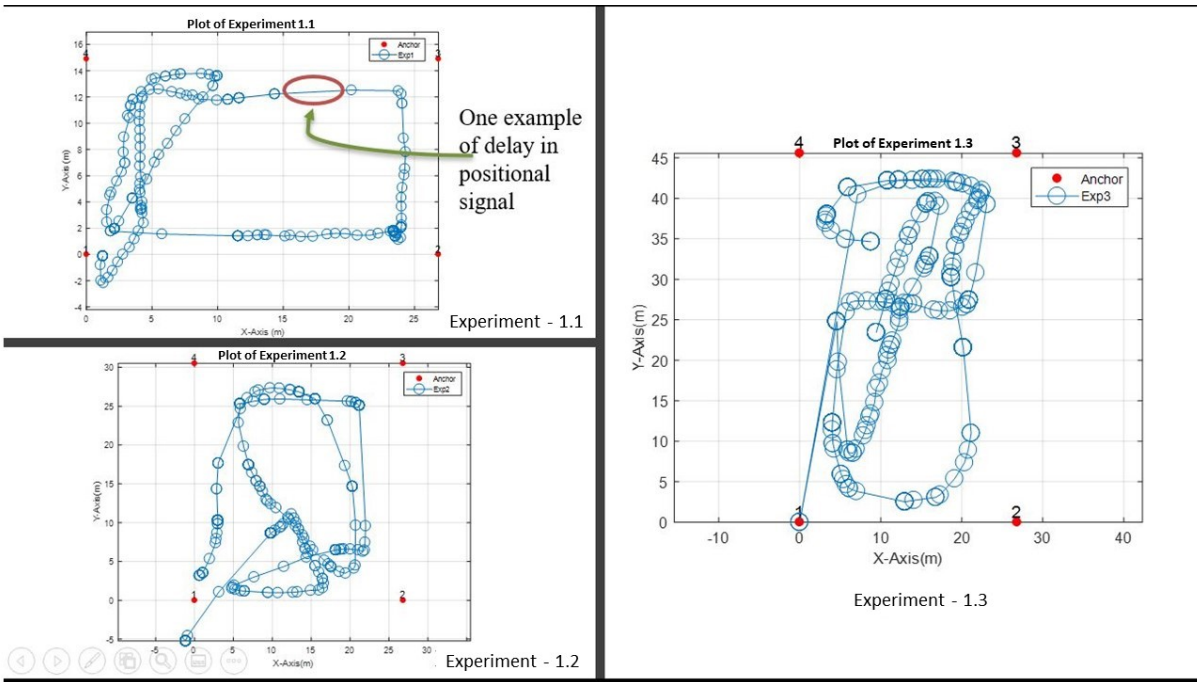


Figure 3.5: Experiment Set 1

### 3.2.2 Best distance/configuration for optimum results.

In our next set of experiments, we aim to identify the best possible distance/configuration where a tag can get maximum number of anchors while moving in an outdoor environment Objectives for the experiment:

- To identify an optimum configuration of anchors that provides best accuracy and coverage.

Note: We have already done an experiment with acceptable results in a 6.1m X 6.1m configuration outdoors.

In the experiment, we set up 2 different anchor configurations with

- 6 anchors, 14.9m X 26.8m (Experiment1), additional anchors added at 7.62m in X direction.

- 8 anchors, 30.5m X 26.8m (Experiment2),

**Conclusion from the experiment referring to Fig.3.7:**

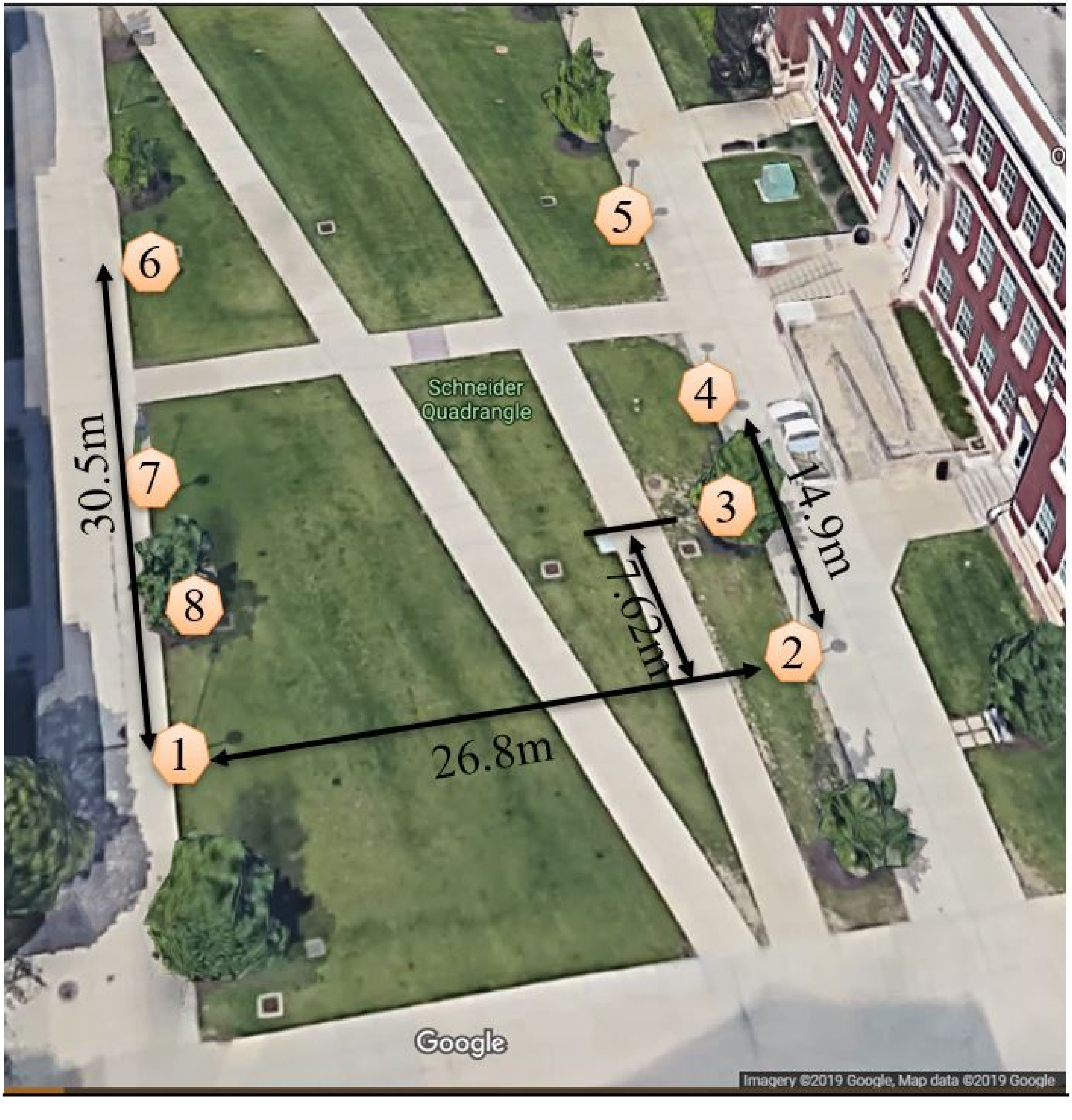


Figure 3.6: Setup for the Experiment 2

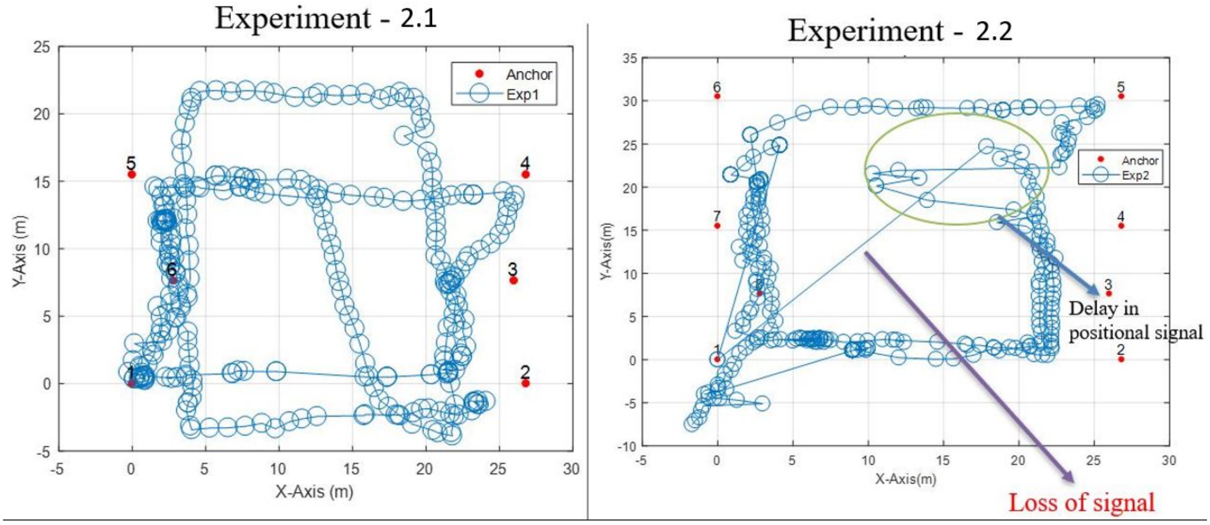


Figure 3.7: Experiment Set 2

- From experiment 1, we can conclude that there was an acceptable reduction in Long Gaps in Positional Signal (LGPS) as we placed more anchors in the same configuration.

- During experiment 2, we observed some LGPS. This observation was noted when raspberry pi is somewhere between anchors 4, 5, 6, and 7. This can be assumed to be happening because of interference of these anchors and system indecisiveness in selecting anchors.

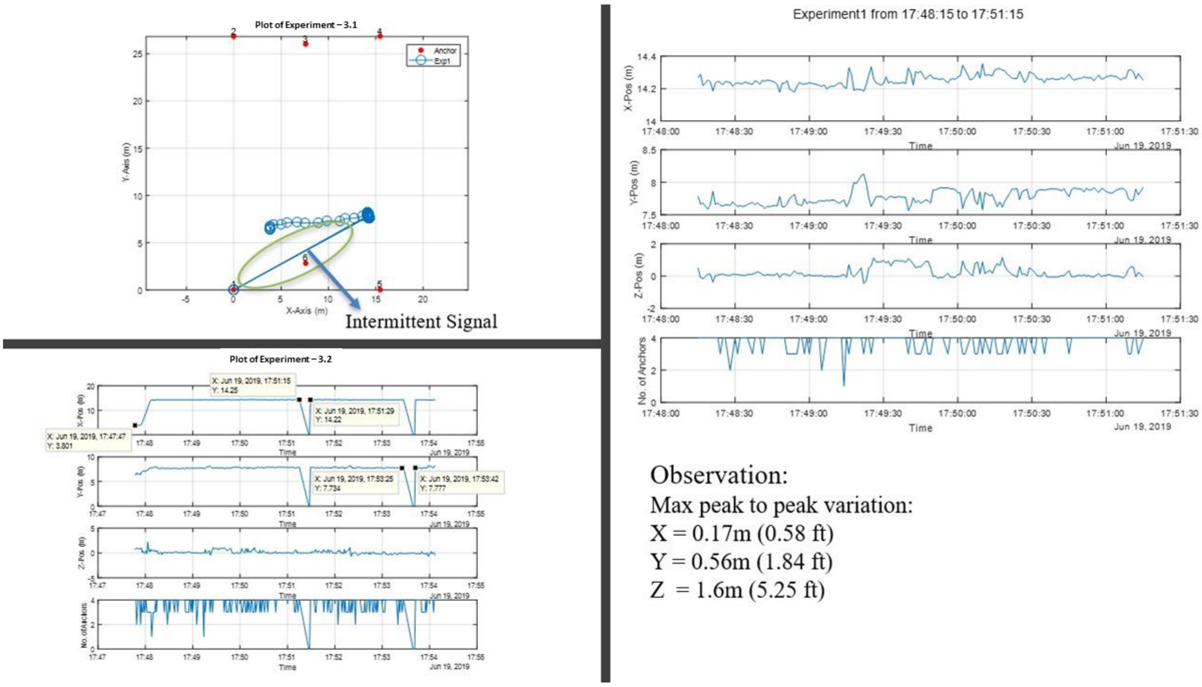
### 3.2.3 Noises and accuracy of the system outdoors.

From previous experiments, we have a promising configuration. Now to be more certain about our claim we designed another experiment.
Objective for the experiment:

- To understand the type of noises this system can generate and what will be it's magnitude?

In the experiment we set up 6 anchors in a rectangular configuration with

- 14.9m X 26.8m (Experiment1) total configuration size, additional anchors added at 7.62m in X direction.



Observation:
Max peak to peak variation:
X = 0.17m (0.58 ft)
Y = 0.56m (1.84 ft)
Z = 1.6m (5.25 ft)

Figure 3.8: Experiment Set 3

Referring to Fig.3.8, following conclusions can be drawn from the experiment,:

- Obtained magnitude of noise was in acceptable range for x and y.
- Max peak to peak variation:
  - X = 0.17m (0.58 ft)
  - Y = 0.56m (1.84 ft)
  - Z = 1.6m (5.25 ft)
- Observed a sudden loss of signal in the system for a duration of about 15 sec.
- System was able to maintain connection with 4 anchors for majority, duration of the experiment.

### 3.2.4 Study of intermittent signal and accuracy of the system.

This set of experiment was performed to further study the LGPS and accuracy of the data from the origin and w.r.t it's previous position.

Objective for the experiment:

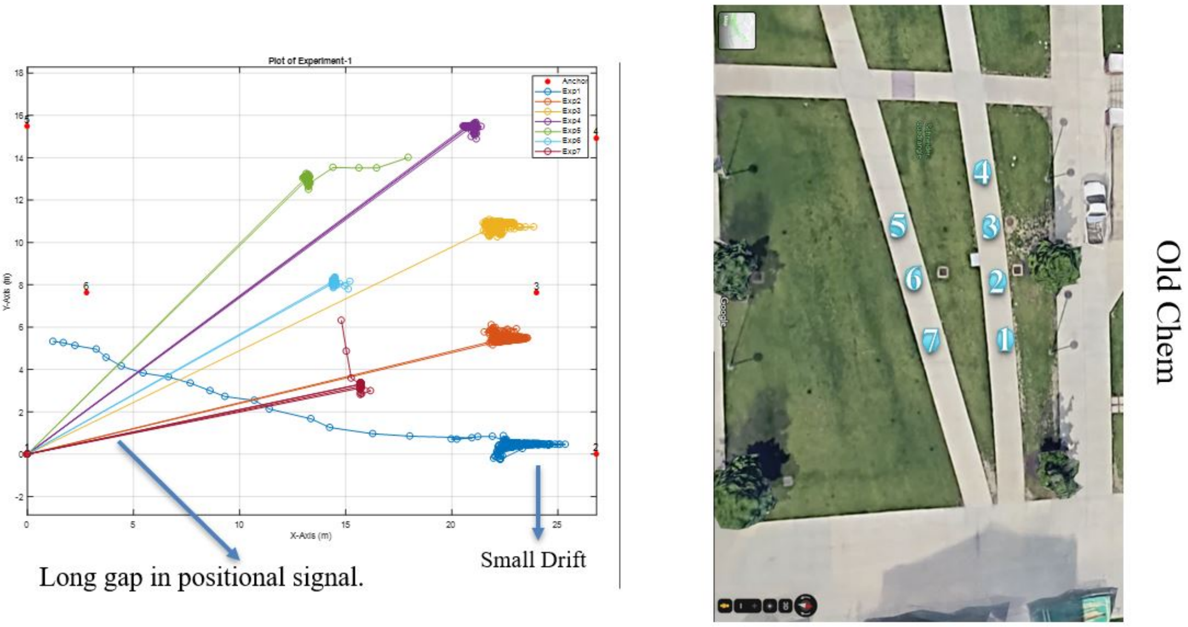


Figure 3.9: Setup for the Experiment 4

- To measure number of Long Gaps in Positional Signal obtained during each experiment.
- Identify the reason for Long Gaps in Positional Signal (e.g. time, temperature etc.).
- Find number of repetition in Long Gaps in Positional Signal.
- Measure the accuracy of the system.

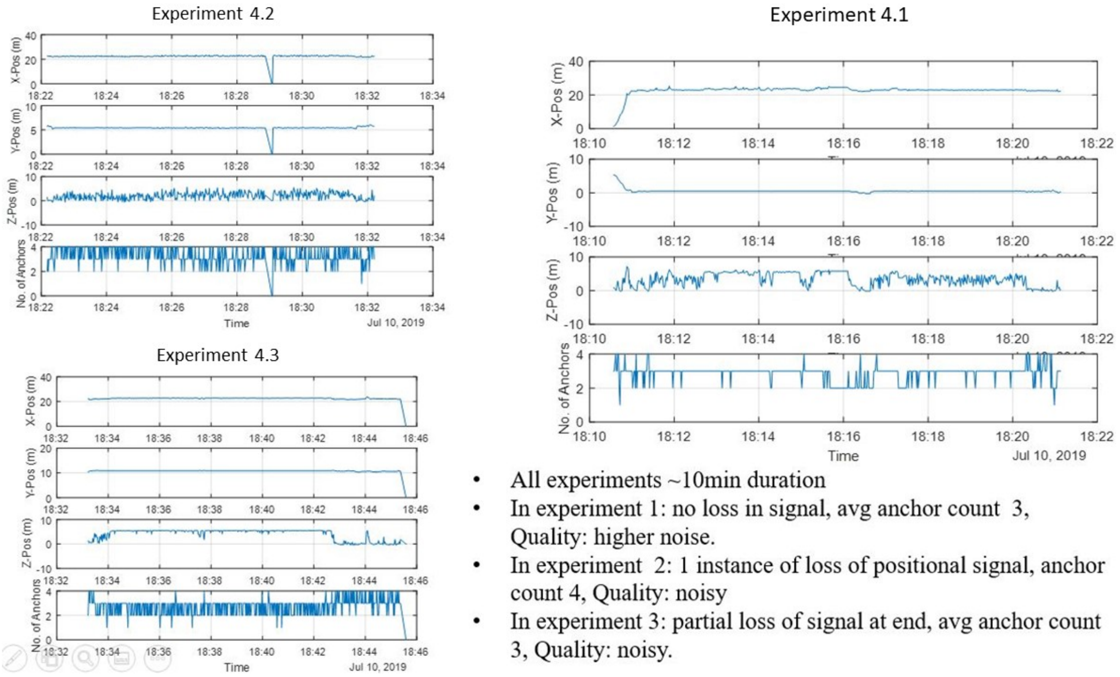
**In the experiment, we set up 6 anchors in a rectangular configuration with**

- 14.9m X 26.8m total configuration size, additional anchors added at 7.62m in Y direction.

**Conclusion from the experiment, referring to Fig. 3.10, 3.11 and 3.12:**

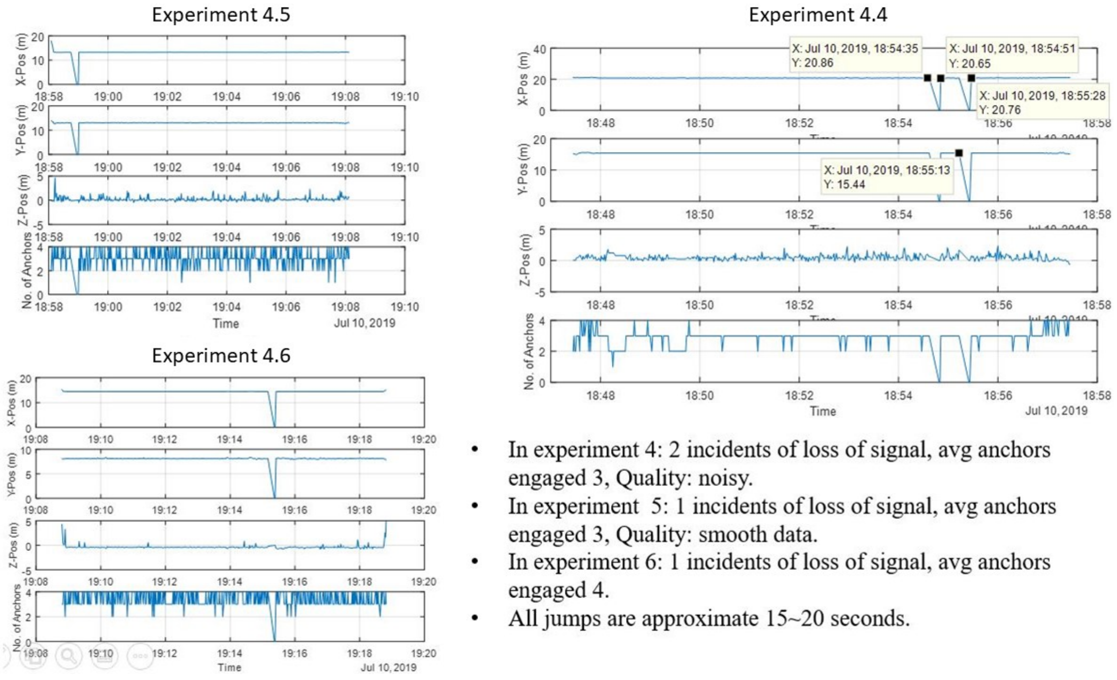
- No Long Gaps in Positional Signal observed in 1st experiment.
- Observed a sudden LGPS in the system. All gaps are about 15 20 sec.

$$\%Error = \frac{SystemMeasured - TapeMeasured}{TapeMeasured} X100$$



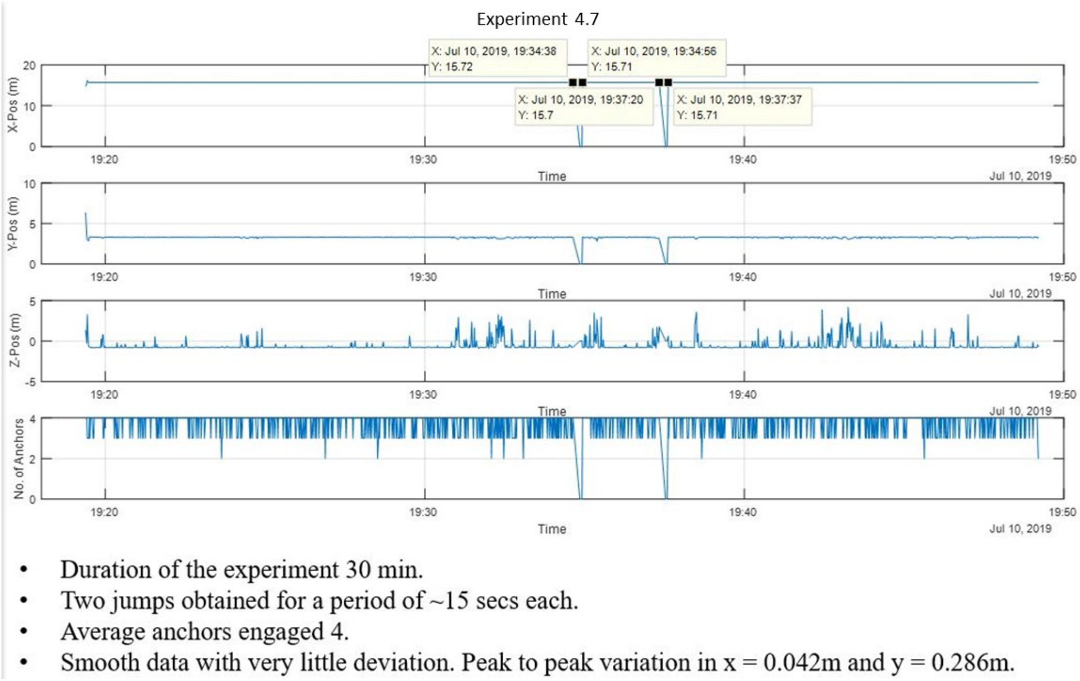
- All experiments ~10min duration
- In experiment 1: no loss in signal, avg anchor count 3, Quality: higher noise.
- In experiment 2: 1 instance of loss of positional signal, anchor count 4, Quality: noisy
- In experiment 3: partial loss of signal at end, avg anchor count 3, Quality: noisy.

Figure 3.10: Experiment Set 4a



- In experiment 4: 2 incidents of loss of signal, avg anchors engaged 3, Quality: noisy.
- In experiment 5: 1 incidents of loss of signal, avg anchors engaged 3, Quality: smooth data.
- In experiment 6: 1 incidents of loss of signal, avg anchors engaged 4.
- All jumps are approximate 15~20 seconds.

Figure 3.11: Experiment Set 4b



- Duration of the experiment 30 min.
- Two jumps obtained for a period of ~15 secs each.
- Average anchors engaged 4.
- Smooth data with very little deviation. Peak to peak variation in x = 0.042m and y = 0.286m.

Figure 3.12: Experiment Set 4c

- System was able to maintain good accuracy in the outside environment. % Error ranges between 4.07 to 0.11. (for Euclidean distances)

| **Experiment** | **Tape Measured (in m)** | **System Measured (in m)** | **Error(in m)** | **Error(in %)** |
|---|---|---|---|---|
| Experiment 1 | 22.27 | 23.1778 | -0.9078 | 4.0762 |
| Experiment 2 | 22.4536 | 23.2008 | -0.7472 | 3.3276 |
| Experiment 3 | 24.282 | 24.9739 | -0.6919 | 2.8496 |
| Experiment 4 | 25.8318 | 25.9354 | -0.1036 | 0.4012 |
| Experiment 5 | 18.34 | 18.5689 | -0.2289 | 1.248 |
| Experiment 6 | 16.5353 | 16.5551 | -0.0197 | 0.1191 |
| Experiment 7 | 15.9512 | 16.0518 | -0.1006 | 0.6306 |

Table 3.1: Error Observed in each experiment

| Conclusion from Outdoor Experiments | | |
|---|---|---|
| Experiment Set | Objectives | Conclusion |
| 1 Set | Viability of System in outdoor environment. | Documentation claims about 60m range. However, system has limitations and cannot work in all distance configurations. |
| 2 Set | To identify best distance/-configuration where we can get optimum results. | Configuration identified. Experimental observation: intermittent signal and disturbance due to irregularity in the configurations. |
| 3 Set | To identify noises and accuracy of the system outdoors. | Acceptable accuracy. Data gaps in positional signal 15 20 sec in length. |
| 4 Set | Study of intermittent signal and accuracy of the system. | Acceptable error. Irregularity in signal needs more study. |

## 3.3 Final conclusion

From Indoor experiment error obtained during the experiments was always < 30 cm. We are able to exploit Decawave utility outdoors. But PANS calibration is a tedious process and is required before any deployment.

It cannot be a ready to fly system, on every occasion, we have to spend time calibrating it and then we will be able to use these devices. Also, DRTLS i.e. Decawave Real-Time localization system limits our interaction with 4 anchors during localization.

However, we did a systemic study of using these devices outdoors and our experience from indoor testing was able to set the stage for outdoor testing. We systematically analysed each parameter of the hardware and made it to the point where it can give it's best performance for our outdoor experiment. The problem of an intermittent signal needs to be addressed.

# Chapter 4

# Design of Simulator

The existing drone simulators don't interface with machine learning models for training or control. After some good research with these simulators the two simulators identified were 'AirSim' and 'Gazebo'. But both of these simulators need higher processing power. Figuring out integration of ML algorithms or rewriting their filter model is another challenge. The purpose of this simulator is to provide quick and easy means to understand platform and to develop the idea of filtering for positional signal using Artificial Neural Network. This chapter discusses the simulator in detail. The simulator is designed in python programming language. The simulator design is based on Object Oriented Programming (OOPS) to keep the code modular, easy to understand and for faster development. The simulator constitutes of a controller module which is further divided into position_control class and attitude_control class, a sensor module and a filter module discussed further (Fig. 4.1).

In the module, the Earth is considered as the inertial reference of frame, which is a requisite so that it can be modeled as a rigid body and required equations of motion can be derived. The inertial reference frame is kept standard, i.e., NED. Which means X-axis pointing in north, Y-axis pointing in east, and the Z-axis is pointing down.[39]

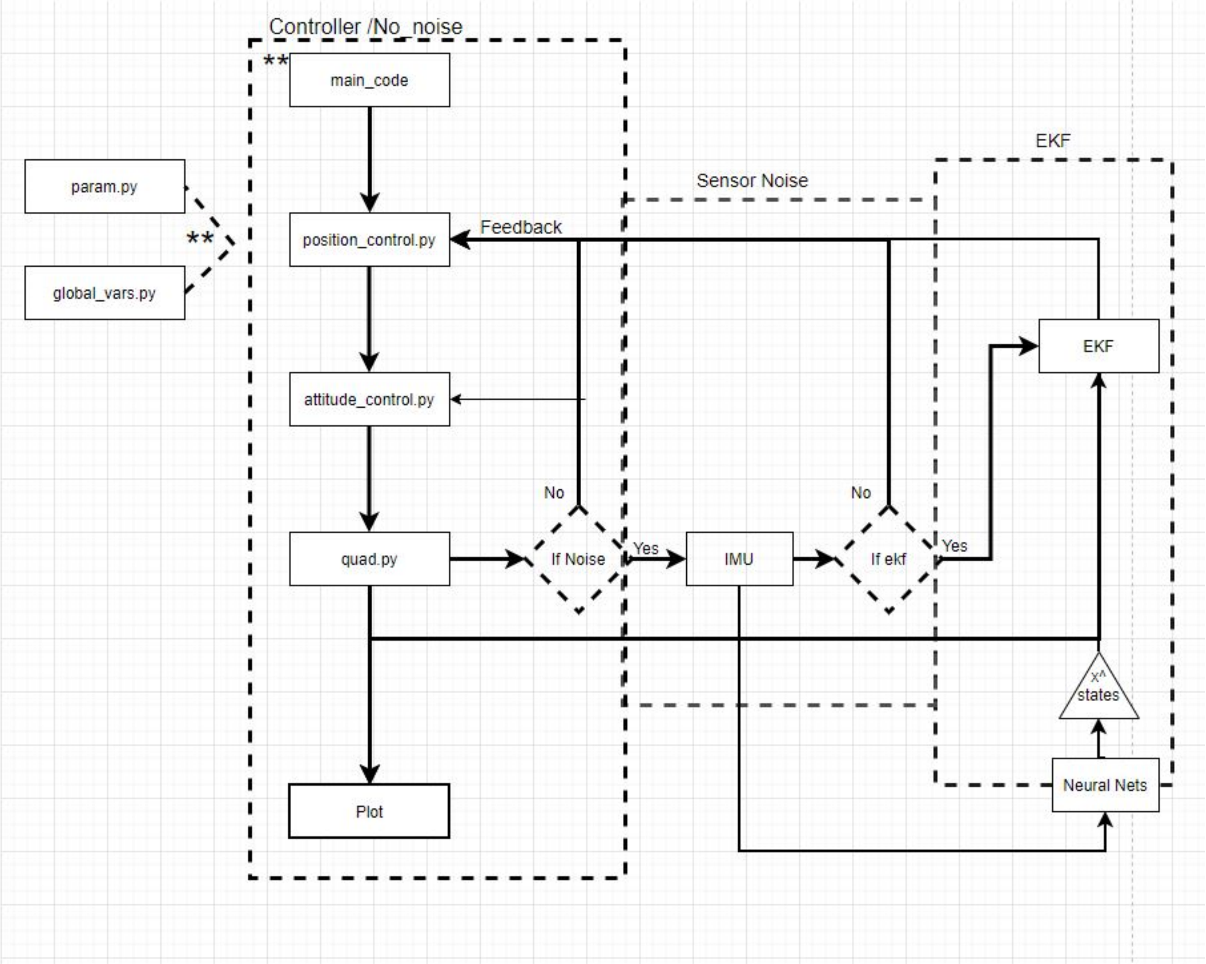


Figure 4.1: Code Architecture

## 4.1 main_code

The main computer program is represented by main_code in the Fig. 4.1 and named as main_code.py in the simulator. This python script is suppose to connect all the different code blocks discussed further and execute them in systematic order. The parameters of the UAV are set to roughly match characteristics of a Hummingbird UAV sold by Ascend technologies, as most of these parameters are referenced from [40]. This method initializes drone parameters such as

- dt - size of each time step in the simulation.
- m - mass of the quad-rotor
- l - length of the arm of the quad-rotor
- g - gravity
- $\omega_h$ - Motor speed
- $K_F = 2.2X10^{-4}$ - constant to relate force and motor speed
- $K_M = 5.4X10^{-6}$ - constant to relate moment and angular speed

It also initializes the way-point for the trajectory to be followed by the drone and then initiates the simulation.

## 4.2 Controller

The purpose of a flight control is to ensure stable behavior of a UAV so that it is able to reach it's waypoints with acceptable accuracy even in the presence of external disturbances. This block constitutes of scripts position_control.py and attitude_control.py discussed in detail below. The design of controller is strongly inspired from [40] "Robot controller" section. It is a linear controller, where the error in position (i.e, Position Desired - Current Position) generates an acceleration set point. This acceleration commands desired thrust. The error in the angles (i.e. $Angle^{Desired} - Angle^{Current}$) generates angular rate set points, which then calculates desired moments. The states can be controlled by using either PD or PID control.

### 4.2.1 Position Control

Based on the desired way point, current position and current velocity, this script calculates the desired acceleration based on difference in the desired position and current position, current velocity and desired velocity . For tuning purposes it has values of Proportional gain (Kp), Derivative gain (Kd) and Integral gain (Ki).
To calculate error in position $e_p$, the following approach is implemented:

$$e_p = p^{des} - p^{cur} \tag{4.1}$$

Similarly, for error in velocity $e_v$,

$$e_v = v^{des} - v^{cur} \tag{4.2}$$

From the equation 4.1 and 4.2 we can calculate desired acceleration, $\ddot{r}_i$ as:

$$\ddot{r}_i = kpXe_p + kdXe_v \tag{4.3}$$

### 4.2.2 Attitude Control

Based on desired acceleration, desired angles ($\phi$, $\Theta$, $\Psi$ ), angular velocities (p, q, r) and $\delta W_f$ (total force required in z direction), desired rotor speed of all independent four rotors is calculated.
Angles desired ($\phi^{des}, \theta^{des}$) are calculated as follows:

$$\phi^{des} = (1/g)X(\ddot{r}_x{}^{des}Xsin(\psi^{des}) - \ddot{r}_y{}^{des}Xcos(\psi^{des})) \tag{4.4}$$

$$\theta^{des} = (1/g)X(\ddot{r}_x{}^{des}Xcos(\psi^{des}) + \ddot{r}_y{}^{des}Xsin(\psi^{des})) \tag{4.5}$$

$$\Delta\omega_F = (m/(8XK_F X\omega_h))X\ddot{r}_z^{des} \tag{4.6}$$

Angular velocities($p^{des}, q^{des}, r^{des}$) desired are taken as 0.

- $p^{des} = 0$
- $q^{des} = 0$
- $r^{des} = 0$

From equation 4.4, 4.5 and 4.6, respective rotor speeds are calculated as follows:

$$\Delta\omega_\phi = k_{p,\phi}(\phi^{des} - \phi) + k_{d,\phi}(p^{des} - p) \tag{4.7}$$

$$\Delta\omega_\theta = k_{p,\theta}(\theta^{des} - \theta) + k_{d,\theta}(q^{des} - q) \tag{4.8}$$

$$\Delta\omega_\psi = k_{p,\psi}(\psi^{des} - \psi) + k_{d,\psi}(r^{des} - r) \tag{4.9}$$

Based on above desired rotor speed equation 4.7, 4.8 and 4.9 corresponding rotor speed by individual rotor ($\omega_1^{des}, \omega_2^{des}, \omega_3^{des}, \omega_4^{des}$) can be calculated as:

$$\omega_1^{des} = (\omega_h + \Delta\omega_F) - \Delta\omega_\theta + \Delta\omega_\psi \tag{4.10}$$

$$\omega_2^{des} = (\omega_h + \Delta\omega_F) + \Delta\omega_\phi - \Delta\omega_\psi \tag{4.11}$$

$$\omega_3^{des} = (\omega_h + \Delta\omega_F) + \Delta\omega_\theta + \Delta\omega_\psi \tag{4.12}$$

$$\omega_4^{des} = (\omega_h + \Delta\omega_F) - \Delta\omega_\phi - \Delta\omega_\psi \tag{4.13}$$

This rotor speed is then used further for calculations of dynamics of the Quad-rotor.

## 4.3 Dynamics of Quadrotor

The design of quadrotor is referenced from [41]. Efforts are taken to keep the notation close to that of typical aeronautics literature. Fig. 4.2 systematically represents following nine state vectors:

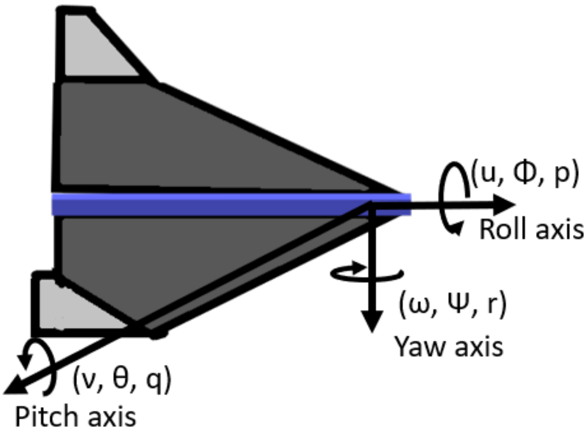


Figure 4.2: Definition of axis

$u$ = body frame velocity in $\hat{i}^b$ in $F^b$,

$v$ = body frame velocity in $\hat{j}^b$ in $F^b$,

$w$ = body frame velocity in $\hat{k}^b$ in $F^b$,

$\phi$ = roll angle respect to $F^{v2}$,

$\theta$ = pitch angle respect to $F^{v1}$,

$\psi$ = yaw angle respect to $F^v$

additional three state variables used are:

$p_n$ = the inertial (north) position of the quadrotor along $\hat{i}^i$ in $F^i$,

$p_e$ = the inertial (east) position of the quadrotor along $\hat{j}^i$ in $F^i$,

$h$ = the altitude of the aircraft measured along $-\hat{k}^i$ in $F^i$

The quadrotor is in the inertial frame with position $(p_n, p_e, h)$, with positive h defined along negative Z-axis in the inertial frame. The velocities linear (u, v, w) and the angular (p, q, r) are with respect to the body frame($F^b$) of the quadrotor. The three Euler angles roll $\phi$, pitch $\theta$ and yaw $\psi$ are given with respect to vehicle 2-frame($F^{v2}$), vehicle 1-frame($F^{v1}$) and the vehicle frame ($F^v$) as described in [41].

Based on obtained $\omega_{des}$ values, position(at t-1) and velocity(at t-1) change in position of the drone are calculated as follows:

First of all forces produced by each motor is calculated using:

$$F = K_F \omega^2$$

for all four rotors. Then added to find the actual force on the quad. As the state variables $p_n$, $p_e$ and $-h$ are in inertial frame and the velocities in body frame. Hence, the relationship between position and velocities can be given by:

$$\begin{vmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{-h} \end{vmatrix} = \begin{vmatrix} c(\theta)Xc(\psi) & c(\theta)Xs(\psi) & s(\theta) \\ s(\phi)Xs(\theta)Xc(\psi) - c(\phi)Xs(\psi) & s(\phi)Xs(\theta)Xs(\psi) + c(\phi)Xc(\psi) & s(\phi)Xc(\theta) \\ c(\phi)Xs(\theta)Xc(\psi) + s(\phi)Xs(\psi) & c(\phi)Xs(\theta)Xs(\psi) - s(\phi)Xc(\psi) & c(\phi)Xc(\theta) \end{vmatrix} \cdot \begin{vmatrix} u \\ v \\ w \end{vmatrix}$$

** where 'c' represents cosine and 's' represents sine angles respectively.

Above represents rotational matrix to convert body frame $F^b$ to world frame

The relationship between the absolute angles $\phi$, $\theta$ and $\psi$ and the angular rates p, q, and r is complicated. The angular rates are defined in the body frame $F^b$, the roll angle $\psi$ is defined in $F^{v2}$, the pitch angle $\theta$ is defined in $F^{v1}$, and the yaw angle $\psi$ is defined in the vehicle frame $F^v$. To relate p, q and r with $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$ we can use following transformation:

$$R_{pqr} = \begin{vmatrix} 1 & sin(\phi)tan(\theta) & cos(\phi)Xtan(\theta) \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi)(cos(\theta)^{-1}) & cos(\phi)(cos(\theta)^{-1}) \end{vmatrix} \tag{4.14}$$

$$\begin{vmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{vmatrix} = \begin{vmatrix} 1 & sin(\phi)tan(\theta) & cos(\phi)Xtan(\theta) \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi)(cos(\theta)^{-1}) & cos(\phi)(cos(\theta)^{-1}) \end{vmatrix} \cdot \begin{vmatrix} p \\ q \\ r \end{vmatrix} \tag{4.15}$$

Based on the total force and the rotational matrix R, acceleration in the three directions is calculated.

$$\ddot{r} = (1/m)X \begin{vmatrix} 0 \\ 0 \\ -mg \end{vmatrix} + R^T. \begin{vmatrix} 0 \\ 0 \\ F_{total} \end{vmatrix} \tag{4.16}$$

The angular velocity of the quadrotor airframe is with respect to the inertial frame. The control force is computed and applied in the body coordinate system. To express acceleration in the body coordinate system we can use acceleration ($\ddot{r}$) obtained consecutively velocity($v$) and position ($p$) as follows:

$$v = v_{t-1} + \ddot{r}Xdt$$

then,

$$p = p_{t-1} + vXdt$$

Therefore, change in the angular velocities ($\dot{p}, \dot{q}, \dot{r}$) can be calculated using equation:

$$\dot{p} = (lXK_FX(\omega_2^2 - \omega_4^2) - qXrX(I_{zz} - I_{yy}))/I_{xx}$$

$$\dot{q} = (lXK_FX(\omega_3^2 - \omega_1^2) - pXrX(I_{xx} - I_{zz}))/I_{yy}$$

$$\dot{r} = K_m((\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) - pXqX(I_{yy} - I_{xx}))/I_{zz}$$

Product of above equations with $\Delta_t$ (time stamp) results in current angular velocities (p,q,r)

From angular velocities respective angles can be calculated as follows.

$$\begin{vmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{vmatrix} = R_{pqr}. \begin{vmatrix} p_{t-1} \\ q_{t-1} \\ r_{t-1} \end{vmatrix} \tag{4.17}$$

Therefore we get:

$$\begin{vmatrix} \phi \\ \theta \\ \psi \end{vmatrix} = \begin{vmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{vmatrix} X\Delta_t \tag{4.18}$$

## 4.4 Sensor Model

The purpose of this block or model is to simulate a sensor's functionality. In the simulator, we have modeled two sensors Inertial Measurement Unit (IMU) and global positioning system (GPS). The IMU constitutes of gyro and accelerometer. A gyro constitutes a small vibrating lever which on angular rotation undergoes a change in the frequency of the vibration and detects the rotation based on the Coriolis effect. On the other hand, an accelerometer contains a small plate or mass attached to the torsion lever or spring. It measures acceleration by the change in capacitance between the fixed plate and the moving mass or lever [42]. The output of an accelerometer is given by:

$$y_{acc} = k_{acc}A + \beta_{acc} + \eta_{acc}$$

where $y_{acc}$ is in Volts, $k_{acc}$ is a gain, A is the acceleration in meters per second squared, $\beta_{acc}$ is a bias term, and $\eta_{acc}$ is zero-mean white noise. The gain $k_{acc}$ is given with the sensor specification and varies with different manufacturers. As offsets like $k_{acc}$ and $\beta_{acc}$ can be eliminated through a calibration before each flight and hence their values are assumed in the simulator. For $\eta_{acc}$ we have added a Gaussian noise of mean$(\mu) = 0$ and variance$(\sigma) = 1$. Accelerometers are analog devices that are sampled by the on-board processor. In the simulator, the sampling rate is set as 100Hz. A quick text for above can be found in [41] and [43] or a detailed intuitive explanation is available in [44]. A GPS gives latitude and longitude, which can be converted to the Cartesian system if the origin and orientation of the drone is known and formulas for the same can be found in [45]. But it is still an active field of research in terms of accuracy and methods[46]. As per [47] GPS signals are accurate within a 4.9m radius and from [48] update rate is around 5 Hz, similar noise and update rate is set in simulator.

Drone is a complex design and to exploit it's ease of use, these drones are equipped with a series of sensors. Aside from mechanical components that give lift and maneuver, the purpose of sensors is to constantly collect information from their surroundings and provide feedback, where feedback constitutes of information such as current position, speed, nearby obstacles etc.

As an IMU experiences force, and based on it calculates acceleration, following equations are used for it's design referred from [49].

$$\begin{vmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{vmatrix} = 1/m \begin{vmatrix} 0 \\ 0 \\ F_{total} \end{vmatrix} - \begin{vmatrix} -gsin(\theta) \\ gcos(\theta)sin(\phi) \\ gcos(\theta)cos(\phi) \end{vmatrix} + \begin{vmatrix} rv - qw \\ pw - ru \\ qu - pv \end{vmatrix} \tag{4.19}$$

Rest of the calculations for transformations, velocities, positions and angular velocities are same as discussed in quadrotor model.

## 4.5 Filter Model

The objective of this section is to describe numerical filter, Dynamic observer theory and implementation of Extended Kalman filter. A numerical filter executes a set of mathematical operations on a discrete-time signal to reduce noise that distorts the process-variable measurement. The introduction of such a digital filter in the feedback loop generally results in a smoother control effect and a stable flight [50][51]. According to statistics and control theory, Kalman Filter is an algorithm that uses a series of measurements observed over time to estimate unknown variables that tend to be more precise than a single measurement, containing statistical noise and other variations [52]. Extended Kalman Filter is a nonlinear variant of the Kalman filter, which linearizes about an estimate of current mean and variance using Taylor Series. In short EKF helps in getting a linear estimate of a nonlinear function [53].

In general an Extended Kalman Filter is a two step process; prediction and update. As per dynamic observer theory a linear time-invariant system can be modeled as follows:

$$\dot{x} = Ax + Bu$$

$$y = Cx$$

A continuous time observer for this system is given by the equation

$$\dot{\hat{x}} \ = A\hat{x} + Bu + L(y - C\hat{x})$$

where

$$L(y - C\hat{x})$$

part is copy of the model and the latter part is the correction due to sensor reading In general, the sensors are usually sampled and give an update at some sample rate $T_s$. To modify observer equation such that it incorporates this behaviour, following equation is implemented to channelize the system model between samples.

$$\dot{\hat{x}} = A\hat{x} + Bu \tag{4.20}$$

can also be written as:

$$\dot{\hat{x}} = f(x, u)$$

and then estimate is updated when a measurement is received using the equation

$$\hat{x}^{+} = \hat{x}^{-} + L(y(t_k) - C\hat{x}^{-} \tag{4.21}$$

, where $t_k$ is the time, measurement is received and $\hat{x}^{-}$ is the state estimate from equation 4.20. An algorithm based on above approach can be found in [1] as Algorithm 1.

### Extended Kalman Filter

- **Predict**
  - Predict state estimate:

    $\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k)$
  - Predict Covariance estimate

$\hat{P}_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k$

- **Update**
  - **Kalman Gain**

    $K_k = P_{k|k-1} C_k^T (C_k P_{k|k-1} C_k + R_k)_k^{-1}$
  - **Updated State estimate**

    $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$
  - **Updated Covariance estimate**

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

Similarly, a derivation of the Continuous-discrete Kalman filter can also be found in [1] at section 8.5. For simulator implementation following equations have been used to define state vector, i.e., $\hat{x}$, quadrotor dynamic equations given by $f(x, u)$ and 'A' represent partial derivative of $f(x, u)$ w.r.t $x$.

### 4.5.1 Prediction

$$\hat{x} = \begin{vmatrix} \hat{p}_x \\ \hat{p}_y \\ \hat{p}_z \\ \dot{\hat{p}}_x \\ \dot{\hat{p}}_y \\ \dot{\hat{p}}_z \\ \hat{\psi} \\ \hat{\theta} \\ \hat{\phi} \end{vmatrix} \tag{4.22}$$

Based on $\hat{x}$ the propagation model is as follows:

$$f(x,u) = \begin{vmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \\ cos(\phi)sin(\theta)\ddot{r}_z \\ -sin(\theta)\ddot{r}_z \\ g + cos(\theta)cos(\phi)\ddot{r}_z \\ p + qsin(\phi)tan(\theta) + rcos(\phi)tan(\theta) \\ qcos(\psi) - rsin(\psi) \\ qsin(\phi)/cos(\theta) + rcos(\psi)/cos(\theta) \end{vmatrix} \tag{4.23}$$

$$A = \delta f/\delta x$$

$$A = \begin{vmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & s(\phi)s(\theta)\ddot{r}_z & c(\phi)c(\theta)\ddot{r}_z & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1c(\phi)\ddot{r}_z & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1s(\phi)c(\theta)\ddot{r}_z & c(\phi)s(\theta)\ddot{r}_z & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & qc(\phi)tan(\theta) - rs(\phi)tan(\theta) & (rc(\phi) + qs(\phi))/c(\theta)2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -qs(\phi) & -rc(\phi) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & (qc(\phi) - rs(\phi))/c(\theta) & -(qs(\phi) + rc(\phi))t(\theta)/c(\theta) & 0 \end{vmatrix} \tag{4.24}$$

### 4.5.2 Update

The algorithm for implementing Extended Kalman filter in the simulator is as follows

```
1. Initialize: $\hat{x} = 0$.
2. At each sample time dt:
3. for i = 1 to N do Prediction: propagate the equations.
4. $\hat{x}- > \hat{x} + (T_{out}/N)(f((\hat{x}, u)))$
5. $A = \delta f/\delta x$
6. $P- > P + (T_{out}/N)(AP + PA^T + GQG^T)$
7. end for
8. for A measurement has been received from sensor i then Correction: Measurement Update
9. $C_i = \delta f/\delta x$
10. $P- > (I - L_i C_i)P$
11. $\hat{x}- > \hat{x} + L_i(y_i - c_i(\hat{x}))$
```

## 4.6 Artificial Neural Network (ANN)

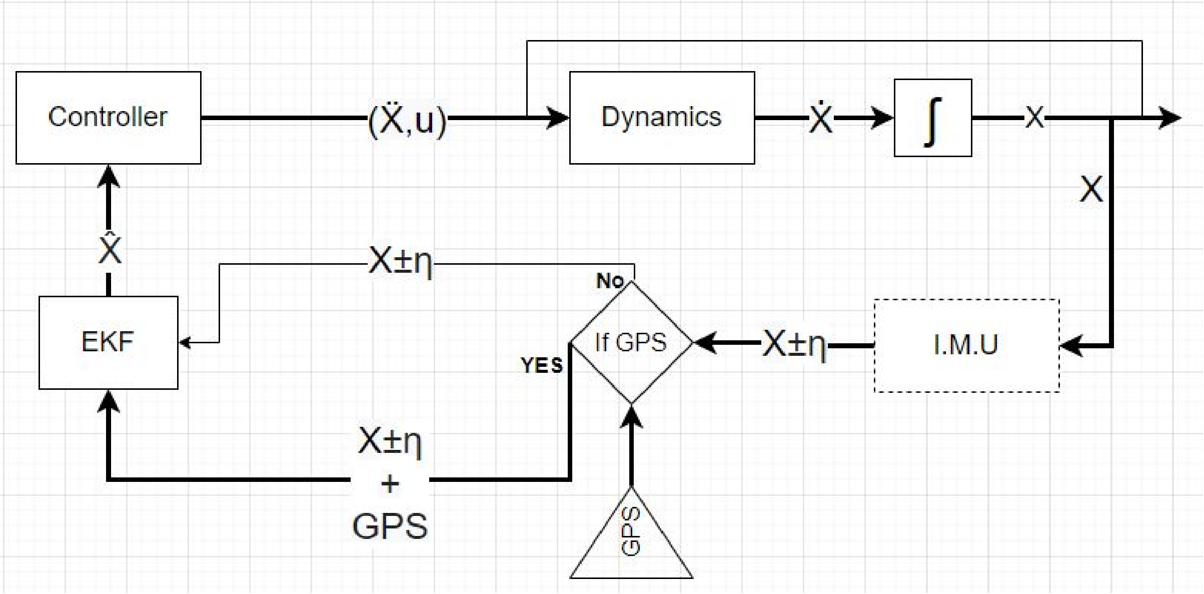


Figure 4.3: Schematics of a controller.

This section presents the solution designed to address the intermittent signal challenge. An intermittent signal is a long gap in the positional signal. In EKF this issue is addressed by setting a sampling rate [54] and depends very much upon the sensor signal. Due to various reasons such as heating and interference in wired/wireless communication, sensors are susceptible to faulty signals. These faults cause delays which results in a sudden abrupt maneuver by a UAV causing damage to drone or giving rise to an abrupt behavior [55]. The novel approach discussed is supposed to handle this abrupt maneuver by giving smoother feedback. As discussed in the introduction of chapter 4, confidence in the accuracy of the results suffers due to complexity involved in available simulators. Hence to be confident about the working of this novel approach we designed our simulator and a module for implementation of Artificial Neural Network based approach.

The basic goal of a autopilot design is to control the inertial position ($p_x$, $p_y$, $p_z$) and attitude ($\phi$, $\theta$ and $\psi$) of the UAV. When designing a PD or PID controller, tuning gains of the controller are an important aspect of control system design. Multiple methodologies are suitable for this purpose such as, Dahlin PID controller, Ziegler-Nichols criterion method and successive loop closure based approach etc. But there are no formal analysis or comparison studies regarding self-tuning controller[56]. Referring to figure 4.3 as discussed in [57], controller output is basically pwm signals to the rotor. Based on the pwm signal there is change in dynamics or state of a quadrotor. These changes in state are sensed by an IMU, which in combination with GPS signal (which has a slower update rate then IMU) is passed through the filter (EKF). Then Extended Kalman Filter calculates a much accurate feedback for the controller which in turn generates better pwm signal and the loop continues.

As discussed above and referring to Fig.4.4, the update received is not at regular intervals. For example from the Fig.4.4 between the time stamp $t_5$ and $t_6$ there is a increase in the deviation which gets corrected at the time stamp $t_6$ due to long gap. This sudden change give rise to a abrupt maneuver, and to control this behaviour we have implemented Artificial Neural Network, which in such situations is suppose to provide feedback based on previous learning. Hence, ANN takes over the system and provides state variables required by EKF in the cases of GPS cutoff. This effect will smoother the feedback and will give linearity to the update rate as now there is update from only one sensor(IMU), which has a higher frequency with the only disadvantage being drifting.

As in Fig.4.5, the loop is similar to that of a controller discussed above, with difference in the feedback that reaches the controller. Feedback consisting of current states represented by X in equation 4.26. X constitutes of inertial position ($p_x$, $p_y$, $p_z$), velocity position ($\dot{p_x}$, $\dot{p_y}$, $\dot{p_z}$) and attitude ($\phi$, $\theta$ and $\psi$, p, q and r) at time t-1, and rotor angular velocities $(\omega_1, \omega_2, \omega_3, \omega_4)$. Mentioned states X are passed through a recursive neural network with two hidden layers to keep training the module using Backward propagation while it is receiving a GPS signal. When GPS signal is not available the system switches from EKF to Neural Network based EKF.

Implementation of Kalman Filter with artificial neural network.

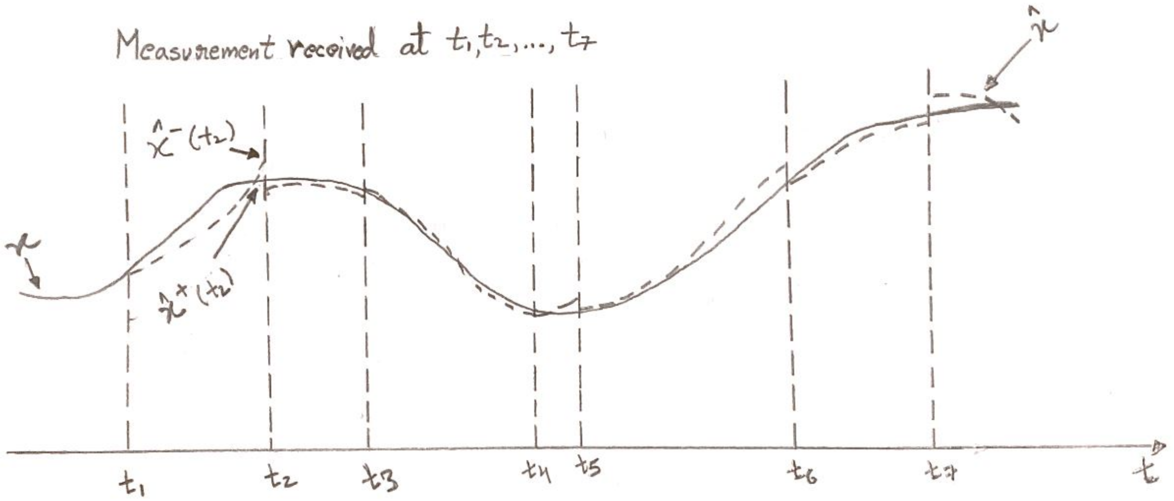


Figure 4.4: continuous-discrete dynamic observer time-line representation. The vertical lines represent sample times at which measurements are received. In between these measurements, the state is propagated using equation (4.20). Once there is a received measurement, the state is updated using equation(4.21) [1]

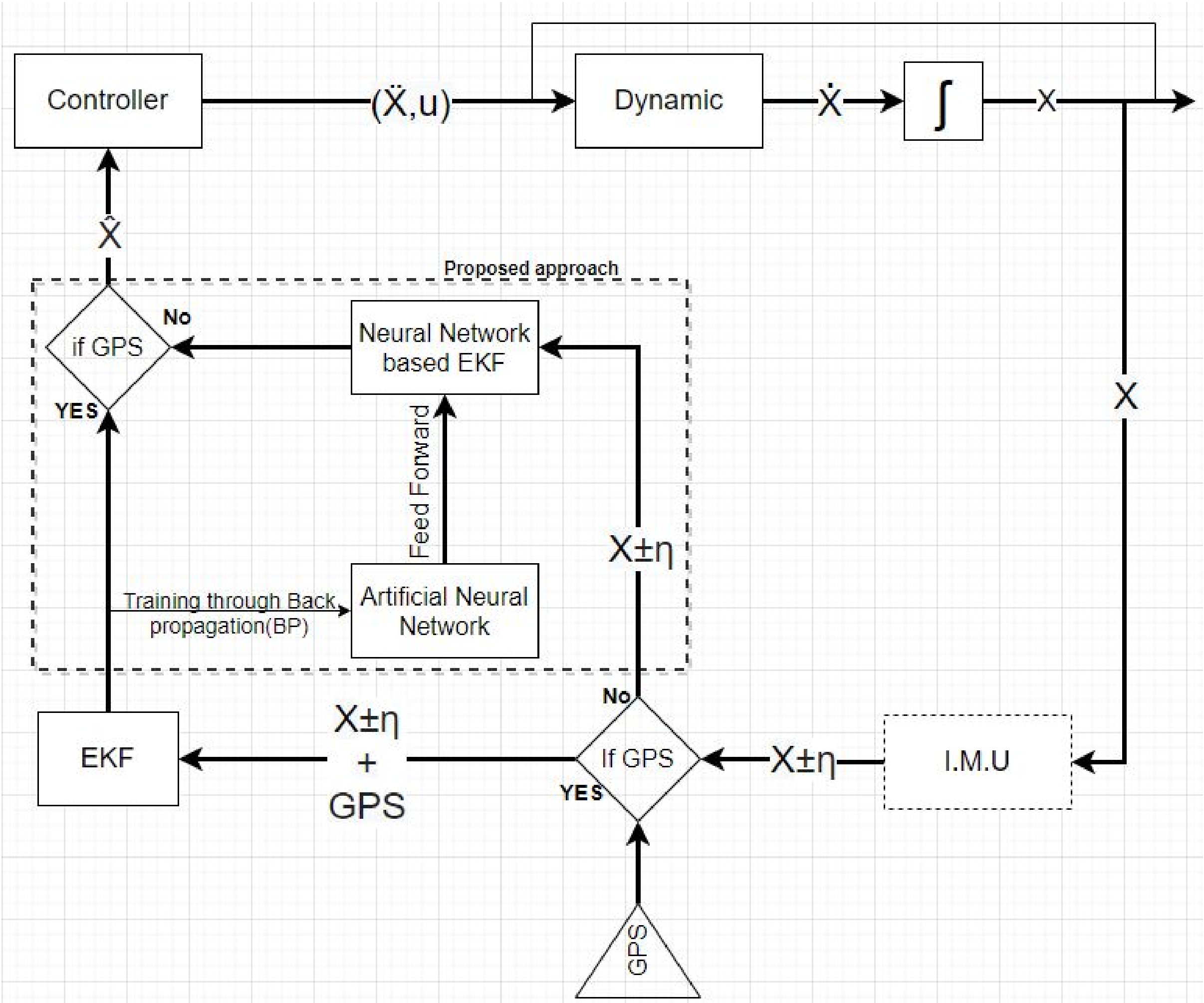


Figure 4.5: Schematics of Neural Network implementation.

### 4.6.1 Algorithm

**Case 1: Predicting States $\hat{x}$ using Neural network**

In this case we predicted $\hat{x}$ using ANN.

Y or $\hat{x}$ or output of ANN

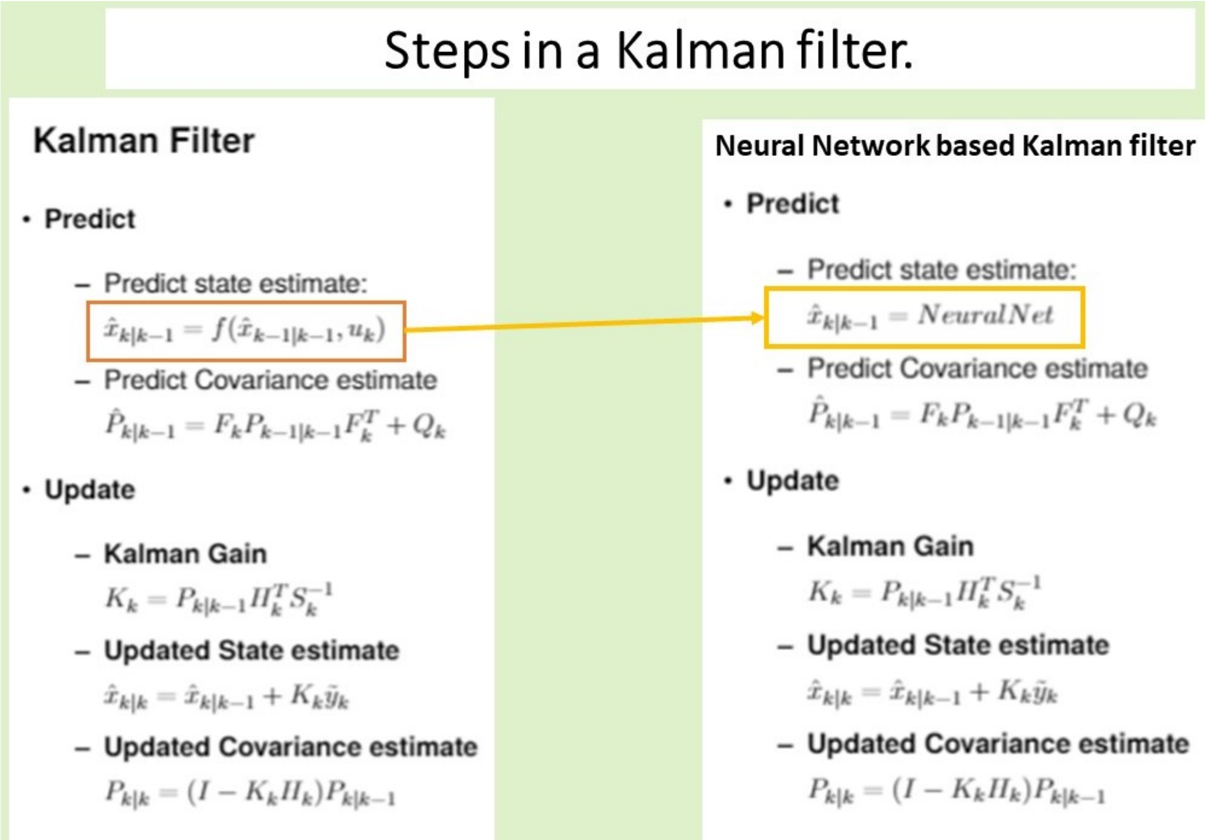
$$Y(\hat{x}) = \left| \hat{p}_x{}^t \quad \hat{p}_y{}^t \quad \hat{p}_z{}^t \quad \dot{\hat{p}}_x^t \quad \dot{\hat{p}}_y^t \quad \dot{\hat{p}}_z^t \quad \hat{\phi}^t \quad \hat{\theta}^t \quad \hat{\psi}^t \quad \right|^T \tag{4.25}$$

Input to ANN

$$X = |\hat{p}_x{}^{t-1} \; \hat{p}_y{}^{t-1} \; \hat{p}_z{}^{t-1} \; \dot{\hat{p}}_x^{t-1} \; \dot{\hat{p}}_y^{t-1} \; \dot{\hat{p}}_z^{t-1} \; \hat{\phi}^{t-1} \; \hat{\theta}^{t-1} \; \hat{\psi}^{t-1} \; \omega_1 \; \omega_2 \; \omega_3 \; \omega_4 \; p^{t-1} \; q^{t-1} \; r^{t-1} \; |^T \tag{4.26}$$

### 4.6.2 Implementation

In the proposed approach Fig.:4.5 and Fig.:4.6



Steps in a Kalman filter.

Kalman Filter

- Predict
  - Predict state estimate:
    $\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k)$
  - Predict Covariance estimate
    $\hat{P}_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k$
- Update
  - Kalman Gain
    $K_k = P_{k|k-1} H_k^T S_k^{-1}$
  - Updated State estimate
    $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$
  - Updated Covariance estimate
    $P_{k|k} = (I - K_k H_k) P_{k|k-1}$

Neural Network based Kalman filter

- Predict
  - Predict state estimate:
    $\hat{x}_{k|k-1} = NeuralNet$
  - Predict Covariance estimate
    $\hat{P}_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k$
- Update
  - Kalman Gain
    $K_k = P_{k|k-1} H_k^T S_k^{-1}$
  - Updated State estimate
    $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$
  - Updated Covariance estimate
    $P_{k|k} = (I - K_k H_k) P_{k|k-1}$

Figure 4.6: Implementation

**Extended Kalman Filter With Neural Network**

- **Predict**
  - Predict state estimate:

    $\hat{x}_{k|k-1} = \hat{N}NX$

  - Predict Covariance estimate

    $\hat{P}_{k|k-1} - > F_k P_{k-1|k-1} F_k^T + Q_k$

- **Update**
  - **Kalman Gain**

    $K_k = P_{k|k-1} C_k^T (C_k P_{k|k-1} C_k + R_k)_k^{-1}$

  - **Updated State estimate**

    $\hat{x}_{k|k} - > \hat{x}_{k|k-1} + K_k \tilde{y}_k$

  - **Updated Covariance estimate**

    $P_{k|k} - > (I - K_k H_k) P_{k|k-1}$

$*\hat{N}N = NeuralNet$

# Chapter 5

# Simulation Results and Discussion

This chapter presents comparative results of Artificial Neural Network (ANN) based filtering. Referring to Fig.(4.3), the goal of the research is to improve the feedback, i.e., $\hat{X}$ of the controller input as good feedback will result in better estimate for a controller. This chapter is divided into two sections. The first section discusses the comparative study of flights with Extended Kalman Filter and Artificial Neural Network and the second section talks about incorporating changes such as a change in the payload. A detailed implementation and working of the simulation is discussed in chapter 4.

General understanding of these graphs [Fig.: 5.1]: The graphs that you will come across further in this thesis will have distance on the y-scale and time on the x-scale. They represent the motion of a drone in the x, y, and, z axis to reach their desired x, y and, z (represented by a red line). The line representing 'true' is the data obtained from the IMU sensor.The line representing ekf/neural is the data obtained after filtering from respective filters EKF or ANN. For all the cases the UAV is considered to be starting from Origin i.e. [0,0,0] and reaches the given way point ([30,30,30]).

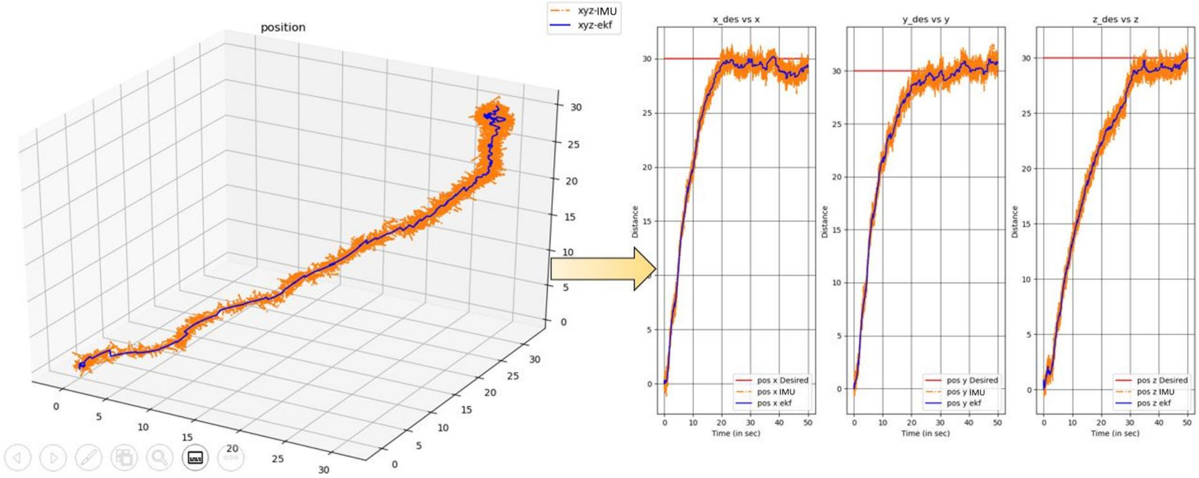


Figure 5.1: The EKF flight transition of 3D to 2D representation.

## 5.1 Comparative Study

The comparative study discusses on following cases:

- Comparison in an Extended Kalman Filter flight and an Artificial Neural Network flight. Where improvement in the feedback is discussed and explained by the means of graphs.
- Comparison in an Extended Kalman Filter flight and an Artificial Neural Network flight with time variation in GPS signal received.

### 5.1.1 Comparison between flight with an EKF and flight with ANN based filter

Figure 5.2 and 5.3 represent flight with EKF and flight with ANN respectively for PD controller. It is simulated that GPS/Positional signal is received every second. In the Fig. 5.2 and 5.3 the UAV is trying to reach desired way point [30, 30, 30] in x, y and z direction respectively. As it can be inferred from the figures the flight with ANN is much smooth compared to the one with EKF.

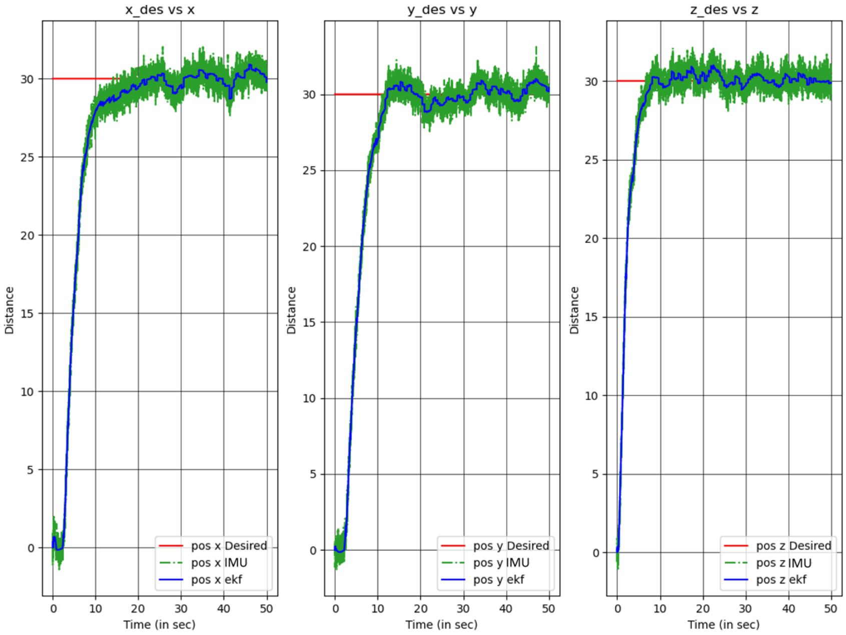


Figure 5.2: EKF

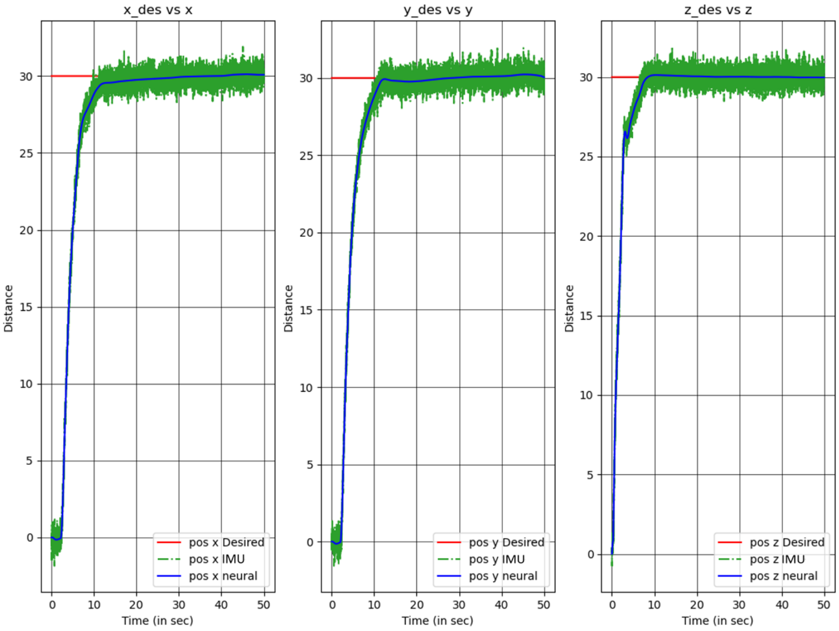


Figure 5.3: Artificial Neural Network

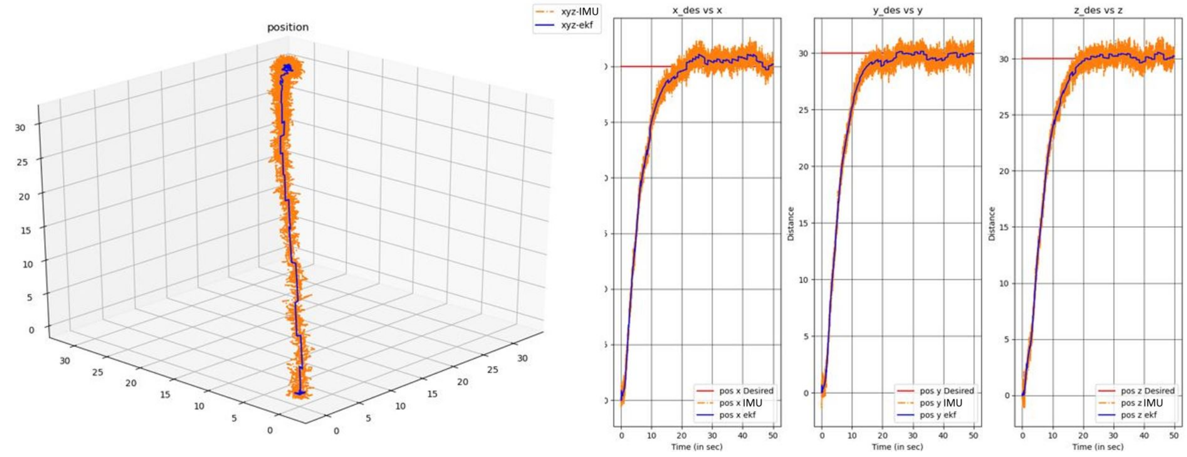


Figure 5.4: EKF Flight with Positional signal every 1 sec

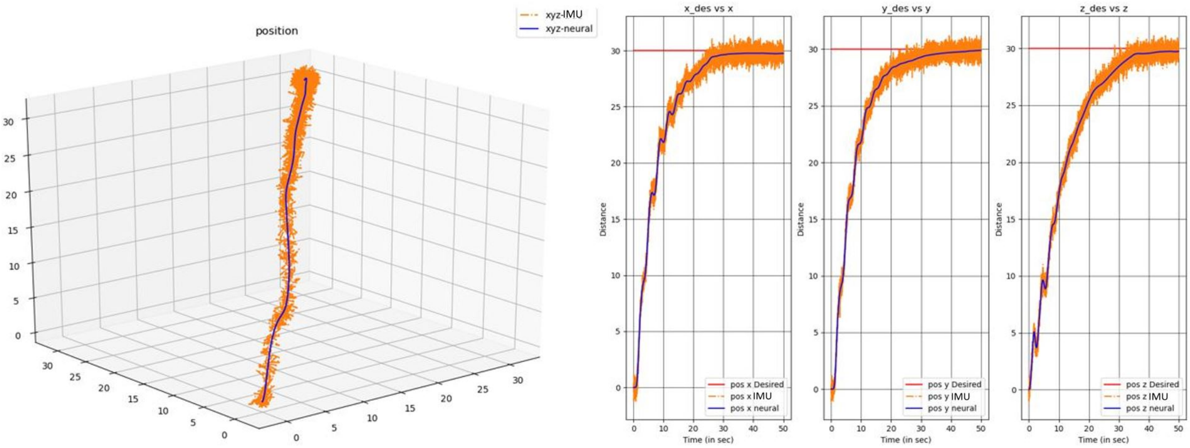


Figure 5.5: ANN Flight with Positional signal every 1 sec

### 5.1.2 Comparison in an EKF flight and an ANN flight with time variation in GPS signal received

As from the graphs Fig. 5.4 and Fig. 5.5 represent flight when UAV receives signal every 1 second. Similarly, Fig. 5.6 and Fig. 5.7 represent flight when UAV receives signal every 6th second. Referring to Fig. 4.4 in chapter 4, it can be inferred that as the frequency of the updates increases there are higher number of corrective steps, which indicates the sensitivity and correctness of the simulator.

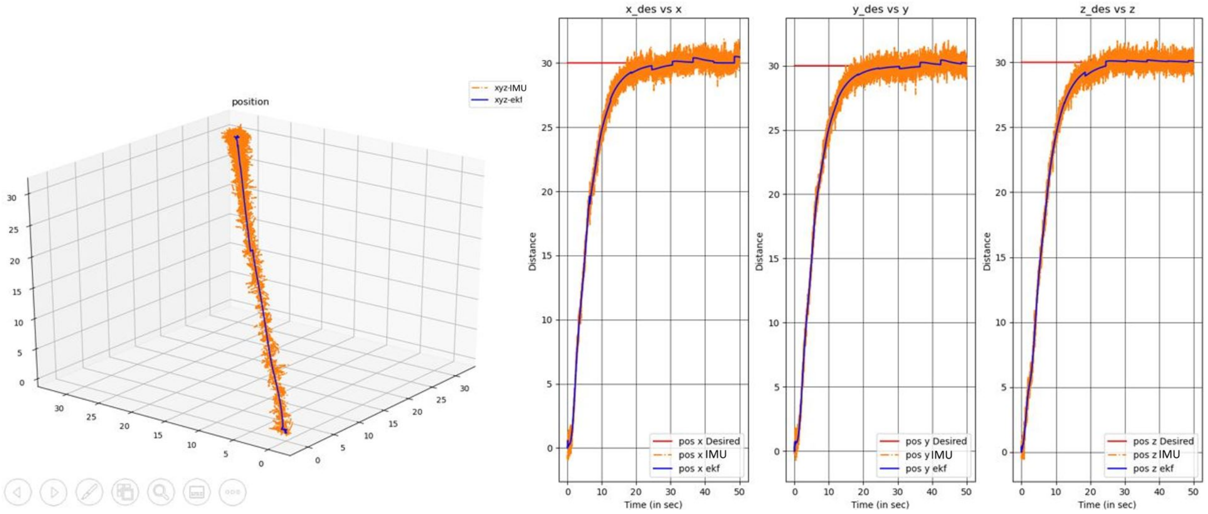


Figure 5.6: EKF Flight with Positional signal every 6 sec

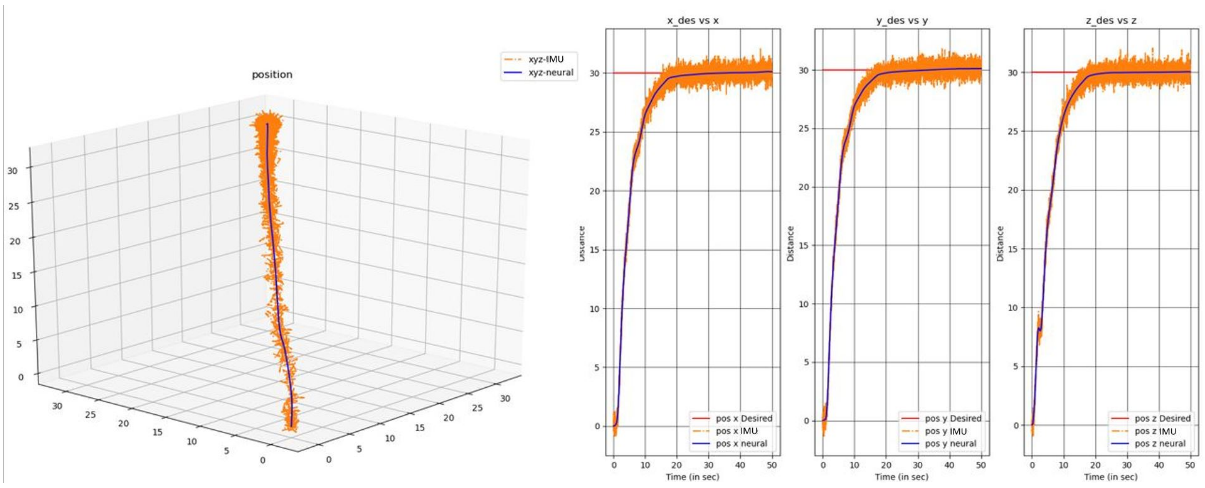


Figure 5.7: ANN Flight with Positional signal every 6 sec

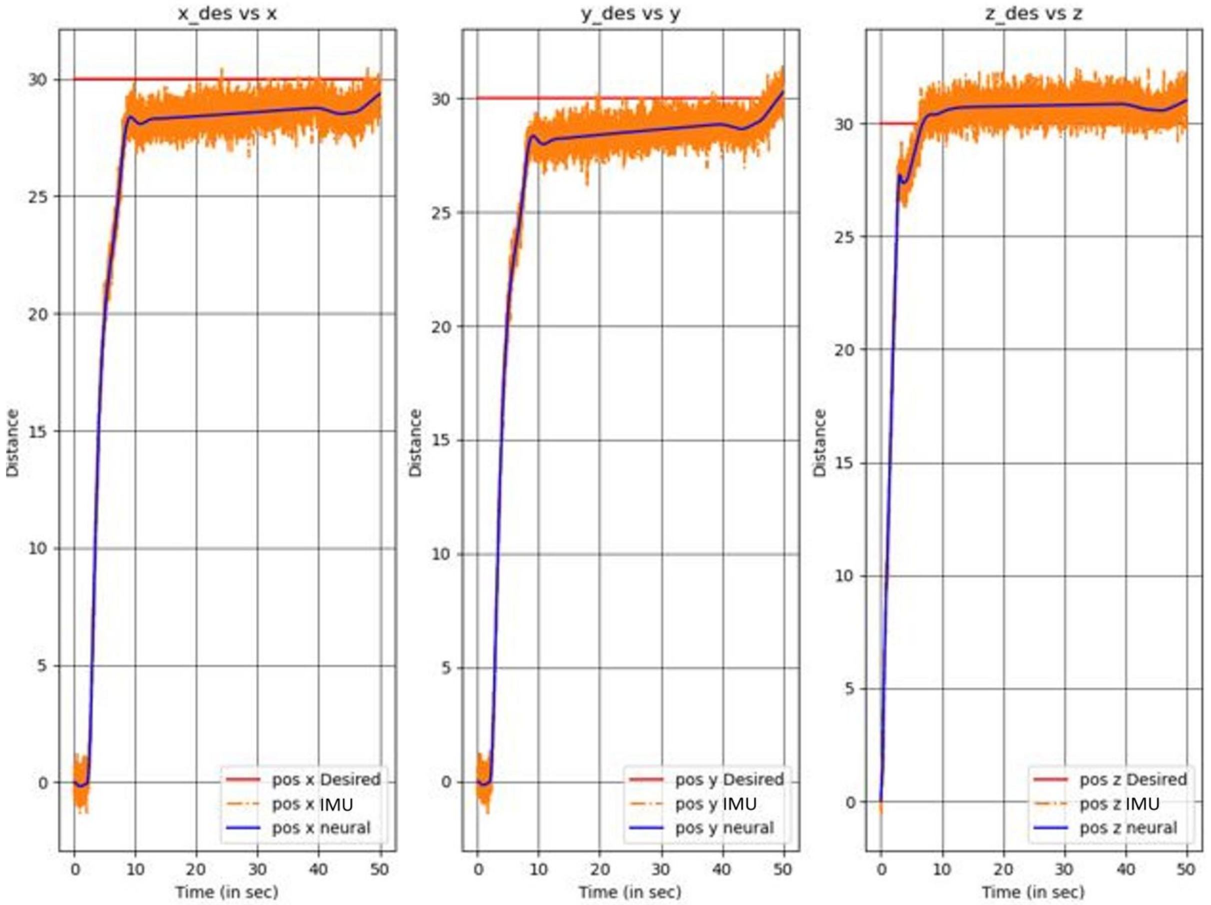


Figure 5.8: ANN based PID controller with drop of payload at 20 sec

## 5.2 Change in Payload

In this part of the work, considering payload of the drone to be 1 Kg which is approximately 15% of the weight of the drone. This payload is dropped during mid flight at around 20 seconds. The ANN based feedback system is able to cope up with the change in payload ref Fig.5.8.

# Chapter 6

# Conclusions and Future Work

## 6.1 Conclusion

UAV technology is an upcoming field. It is just a matter of time that drones will be a part of many enterprises from law enforcement, to construction management to warehouse management to home deliveries, etc. A robust system is the need of the hour. Localization is still a challenge and everywhere researchers are developing a variety of robust mechanisms to tackle it. This work discusses a systematic approach implemented to use indoor localization hardware (UWB modules) in an outdoor environment. Difficulties faced and a step by step approach implemented to tackle the challenge and build up a UWB based system that can be used in bridge inspection, are discussed in detail in Chapter 3.

To develop a further robust solution we dug deeper and implemented a state of art Artificial Neural Network-based algorithm that trains in an online fashion and during the case of an intermittent signal stabilizes the drone maneuver by forward propagation. In the process of addressing the challenge, a python-based simulator was developed bringing in the benefits of modular code, ease of understanding, and faster development. This helped in the comparative study of the novel approach of ANN-based Kalman filtering and Extended Kalman filtering across the cases such as a 1-sec intermittent signal, then a 6-sec intermittent signal, and then system withstand in case of a change in the payload.

Based on the results, it can be concluded that ANN-based filtering gives better control feedback and hence improves the overall flight maneuver.

## 6.2 Future Work

The work shows a possibility of improvement in the feedback of the controller input using Artificial Neural Network. For works such as [58] where they improved performance of their model by choosing appropriate sampling algorithm, implementing ANN based Extended Kalman filter can further improve the performance or could be an interesting approach to observe.

A much trained model or a model with Recurrent neural network (RNN) may provide better results as RNN were designed to work with sequential prediction problems.

Changing the piece of code into Cython can further improve the performance of the simulator for faster and better computation.

# Appendix A

# Python code for Artificial Neural network based EKF implementation

```
# -*- coding: utf-8 -*-
## ###############################################################
# KALMAN FILTER:
# It is an iterative mathematical process that uses a set of equations and consecutive data inputs to quickly
# estimate the true value, position, velocity of the object being measured. When the measured values contain
# unpredicted or random error, uncertainity or variation.
## ###############################################################
import numpy as np
from numpy import sin, cos, tan, pi, sign
from scipy.integrate import ode
from global_vars import global_vars
```

```
import params
import logging
import torch
import sys

class neural_net_based_EKF:

    def __init__(self):
        self.Q = np.identity(9)*0.01
        self.x_hat = np.array([[0], [0], [0], [0], [0], [0],
            [0], [0], [0]])
        self.P = np.identity(9)
        self.R = np.array([ [0.01, 0, 0, 0, 0, 0, 0, 0, 0],
                            [0, 0.01, 0, 0, 0, 0, 0, 0, 0],
                            [0, 0, 0.01, 0, 0, 0, 0, 0, 0],
                            [0, 0, 0, 0.01, 0, 0, 0, 0, 0],
                            [0, 0, 0, 0, 0.01, 0, 0, 0, 0],
                            [0, 0, 0, 0, 0, 0.01, 0, 0, 0],
                            [0, 0, 0, 0, 0, 0, 0.01, 0, 0],
                            [0, 0, 0, 0, 0, 0, 0, 0.01, 0],
                            [0, 0, 0, 0, 0, 0, 0, 0, 0.01]])

    #
        ######################################################################

    # Predict
        ###########################################################
    #
        #####################################################################

    # ------------------------------------------------------------------------
    # Predict state estimate
    # ------------------------------------------------------------------------

    def state_estimate(self, dt, glb):
        # print('t: ',t, 'Ts :' , Ts)
        state = glb.state
        u = state[0]
        v = state[1]
        w = state[2]
        phi = state[3]
        theta = state[4]
        psi = state[5]
        p = state[6]
        q = state[7]
        r = state[8]
        # x_hat = px_hat, py_hat, pz_hat, phi_hat, theta_hat, psi_hat
        x_hat = self.x_hat
        # print(x_hat)
        az = -1*state[15]/params.m
        # print(az)

        DynamicsDot = np.array([
                    [                                                        u],
                    [                                                        v],
                    [                                                        w],
                    [                                        cos(phi)*sin(theta)*az],
                    [                                                 -sin(phi)*az],
                    [                          params.g + cos(theta)*cos(psi)*az],
                    [     p + r*cos(phi)*tan(theta) + q*sin(phi)*tan(theta) ],
                    [                                   q*cos(phi) - r*sin(phi)],
                    [      (r*cos(phi))/cos(theta) + (q*sin(phi))/cos(theta)]])


        x_hat = x_hat + DynamicsDot*dt
        # print('self.x_hat:  ',np.shape(self.x_hat))
        # print('x_hat:  ',np.shape(x_hat))

        return x_hat

```

```
    # ---------------------------------------------------------------------
    # Predict covariance estimate
    # ---------------------------------------------------------------------

    def covariance_estimate(self, dt, glb):

        state = glb.state
        u = state[0]
        v = state[1]
        w = state[2]
        phi = state[3]
        theta = state[4]
        psi = state[5]
        p = state[6]
        q = state[7]
        r = state[8]
        az = -1*state[15]/params.m
        P = self.P
        # P = np.identity(6)
        Q = self.Q


        F_matrix = np.array([
            [0, 0, 0, 1, 0, 0,                           0,
                                                0,   0],
                [0, 0, 0, 0, 1, 0,                    0,                    0,    0],
                [0, 0, 0, 0, 0, 1,                    0,                    0,    0],
                [0, 0, 0, 0, 0, 0,   sin(phi)*sin(theta)*az,   cos(phi)*cos(theta)*az,   0],
                        [0, 0, 0, 0, 0, 0,
                                -1*cos(phi)*az,
                                                    0,
                            0],
                        [0, 0, 0, 0, 0, 0, -1*sin(phi)*cos(theta)*az, cos(phi)*sin(theta)*az,
                          0],
                        [0, 0, 0, 0, 0, 0, q*cos(phi)*tan(theta)-r*sin(phi)*tan(theta), (r*cos(phi)+q*sin(phi))/cos(theta)**2,
                          0],
                [0, 0, 0, 0, 0, 0,
                                    -q*sin(phi),
                        -r*cos(phi),
                    0],
                [0, 0, 0, 0, 0, 0, (q*cos(phi) - r*sin(phi))/cos(theta), -(q*sin(phi)+r*cos(phi))*tan(theta)/cos(theta), 0]])

    # print(np.dot(F_matrix,P*Ts))
    P_k = P + (np.dot(F_matrix,P) + np.dot(P,F_matrix.transpose()) + Q)*dt
    return P_k

    # #########################################################################
    # Update ##############################################################
    # #########################################################################

    # -------------------------------------------------------------------------
    # Kalman Gain
    # ------------------------------------------------------------------------

    def cal_kalman_gain(self, dt, nn_model, ann_input, measure, ymea_nn, signal, t, glb):
        # print('pos: ', quad.pos)
            # x_hat = self.state_estimate(dt,glb)
            # x_hat = nn_model.neural_net_predict(x_new)
        # print('XXHat-shape:  ', x_hat)
            # Pk = self.covariance_estimate(dt,glb)
        # print('Pk: ', Pk)
            if t < 10:
                self.x_hat = self.state_estimate(dt,glb)
                self.Pk = self.covariance_estimate(dt,glb)
        # try:
```

```
        C = np.identity(9)
        R = self.R
        I = np.identity(9)
        # print(f'signal: {signal}')
        if signal:
            x_hat = self.x_hat
            self.Pk = self.Pk
            self.Y = np.array([[measure[0]], [measure[1]], [measure[2]], [measure[3]], [measure[4]], [measure[5]], [measure[6]], [measure[7]], [measure[8]]])
            L = np.nan_to_num(np.dot(np.dot(self.Pk,C.transpose()),np.linalg.inv(R + np.dot(np.dot(C, self.Pk),C.transpose()))))  #kalman_gain
            foo = (x_hat + np.dot(L,(self.Y - np.dot(C, x_hat))))
            # foo  = (x_hat + np.dot(L, np.transpose(self.Y - np.transpose(np.dot(C,x_hat))))) #same as EKF
            self.Pk = np.dot((I - np.dot(L,C)),self.Pk)
        else:
            x_hat = nn_model.neural_net_predict(ann_input).detach().numpy()
            x_hat = np.reshape(x_hat,(len(x_hat),1))
            self.Pk = self.covariance_estimate(dt,glb)
            self.Y = np.array([[measure[0]], [measure[1]], [measure[2]], [measure[3]], [measure[4]], [measure[5]], [measure[6]], [measure[7]], [measure[8]]])
```

```
                L = np.nan_to_num(np.dot(np.dot(self.Pk,C.transpose()),np.linalg.inv(R + np.dot(np.dot(C,self.Pk),C.transpose()))))  #kalman_gain
                # foo = (np.transpose(x_hat) + np.dot(L,(self.Y - np.dot(C, x_hat))))
                foo = (x_hat + np.dot(L,(self.Y - np.dot(C, x_hat))))
                self.Pk = np.dot((I - np.dot(L,C)),self.Pk)
            # print(foo)
            if np.shape(x_hat) ==(9,1) and np.shape(ymea_nn) ==(9,1):
                x_hat = np.reshape(x_hat,(len(x_hat),1))
            else:
                x_hat = np.reshape(x_hat,(len(x_hat),1))
                ymea_nn = np.reshape(ymea_nn,(len(ymea_nn),1))

            ann_input = np.reshape(ann_input,(len(ann_input),1))
            loss_val = nn_model.neural_net_update(torch.from_numpy(np.transpose(ymea_nn)), torch.from_numpy(x_hat), torch.from_numpy(np.transpose(ann_input)), dt)
            # print(foo )
            self.x_hat = foo
            glb.kalman_gain_array.append(L)
            # return np.transpose(foo), self.Pk, loss_val
            return np.transpose(foo), self.Pk, loss_val
    # ----------------------------------------------------------------
```

# Bibliography

[1] McLain Timothy W. Beard Randal W. *8 State Estimation*, pages 143–163. Princeton University Press, 2012.

[2] Captain Brain.P.Tice. UNMANNED AERIAL VEHICLES.- the force multiplier of the 1990s. *Airpower Journal*, DISTRIBUTION A, Published Spring 1991.

[3] Business Insider Intelligence. The Rise of the Drone Industry., Mar 3 2020.

[4] R. S. Kshetrimayum. An introduction to uwb communication systems. *IEEE Potentials*, 28(2):9–13, 2009.

[5] Muhammed T. Köroğlu, Mert önkol, and Mehmet önder Efe. Experimental modelling of propulsion transients of a brushless dc motor and propeller pair under limited power conditions: A neural network based approach. *IFAC Proceedings Volumes*, 42(19):37 – 42, 2009. 2nd IFAC Conference on Intelligent Control Systems and Signal Processing.

[6] M. Ö. Efe. Neural network assisted computationally simple pi$^\lambda$d$^\mu$ control of a quadrotor uav. *IEEE Transactions on Industrial Informatics*, 7(2):354–361, 2011.

[7] D. Nodland, H. Zargarzadeh, and S. Jagannathan. Neural network-based optimal adaptive output feedback control of a helicopter uav. *IEEE Transactions on Neural Networks and Learning Systems*, 24(7):1061–1073, 2013.

[8] Min Xue. Uav trajectory modeling using neural networks, Jun 05 2017. Copyright - Copyright NASA/Langley Research Center Jun 5, 2017; Last updated - 2018-12-13.

[9] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110, 2006.

[10] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: efficient position estimation for mobile robots. pages 343–349, 1999.

[11] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31(1):29–53, 1998.

[12] Feng Lu and Evangelos Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems*, 18(3):249–275, 1997.

[13] K. Lingemann, H. Surmann, A. Nuchter, and J. Hertzberg. Indoor and outdoor localization for fast mobile robots. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2185–2190 vol.3, 2004.

[14] M. Agrawal and K. Konolige. Real-time localization in outdoor environments using stereo vision and inexpensive gps. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 1063–1068, 2006.

[15] J. Park, Soohwan Kim, Nakju lett Doh, and S. Park. Vision-based global localization based on a hybrid map representation. In *2008 International Conference on Control, Automation and Systems*, pages 1104–1108, 2008.

[16] Torralba, Murphy, Freeman, and Rubin. Context-based vision system for place and object recognition. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 273–280 vol.1, 2003.

[17] Iwan Ulrich and Illah Nourbakhsh. Appearance-based place recognition for topological localization. volume 2, pages 1023–1029, 2000.

[18] Paul Blaer and Peter Allen. Topological mobile robot localization using fast vision techniques. volume 1, pages 1031–1036, 2002.

[19] P. Newman, D. Cole, and K. Ho. Outdoor slam using visual appearance and laser ranging. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1180–1187, 2006.

[20] D. Hahnel, W. Burgard, D. Fox, and S. Thrun. An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 1, pages 206–211 vol.1, 2003.

[21] J. Yuan, Y. Huang, F. Sun, S. Huang, and H. Chen. Mobile robot active observation and mapping based on factored method. In *Proceedings of the 10th World Congress on Intelligent Control and Automation*, pages 3577–3582, 2012.

[22] C. Siagian and L. Itti. Biologically-inspired robotics vision monte-carlo localization in the outdoor environment. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1723–1730, 2007.

[23] Waltenegus Dargie and Christian Poellabauer. *Fundamentals of Wireless Sensor Networks: Theory and Practice.* International series of monographs on physics. John Wiley and Sons Ltd., 2010.

[24] K. Liu and Z. Li. Adaptive kalman filtering for uwb positioning in following luggage. In *2019 34rd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pages 574–578, 2019.

[25] J. Tiemann, F. Eckermann, and C. Wietfeld. Atlas - an open-source tdoa-based ultra-wideband localization system. In *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–6, 2016.

[26] J. Tiemann, F. Schweikowski, and C. Wietfeld. Design of an uwb indoor-positioning system for uav navigation in gnss-denied environments. In *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–7, 2015.

[27] G. Vairetti, S. H. Jensen, E. De Sena, M. Moonen, M. Catrysse, and T. van Waterschoot. Multichannel identification of room acoustic systems with adaptive filters based on orthonormal basis functions. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 16–20, 2016.

[28] H. E. Nyqvist, M. A. Skoglund, G. Hendeby, and F. Gustafsson. Pose estimation using monocular vision and inertial sensors aided with ultra wide band. In *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–10, 2015.

[29] Wikipedia contributors. Artificial neural network — Wikipedia, the free encyclopedia, 2020. [Online; accessed 6-May-2020].

[30] Stanford class of Spring 2020. Setting up the data and the model, 2020. [Online; accessed 6-May-2020].

[31] Wikipedia contributors. Backpropagation — Wikipedia, the free encyclopedia, 2020. [Online; accessed 6-May-2020].

[32] Janis Tiemann, Florian Schweikowski, and Christian Wietfeld. Design of an uwb indoor-positioning system for uav navigation in gnss-denied environments. pages 1–7. IEEE, 2015.

[33] J. Wang, A. K. Raja, and Z. Pang. Prototyping and experimental comparison of ir-uwb based high precision localization technologies. In *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, pages 1187–1192, 2015.

[34] Decawave Ltd 2017. Decawave quickstart guide, 2020.

[35] Decawave Ltd 2017. Dwm1001 product brief.pdf, 2020.

[36] Decawave Ltd 2017. Dwm1001 system overview.pdf, 2020.

[37] G. Dedes and A. G. Dempster. Indoor gps positioning - challenges and opportunities. In *VTC-2005-Fall. 2005 IEEE 62nd Vehicular Technology Conference, 2005.*, volume 1, pages 412–415, 2005.

[38] C. McElroy, D. Neirynck, and M. McLaughlin. Comparison of wireless clock synchronization algorithms for indoor location systems. In *2014 IEEE International Conference on Communications Workshops (ICC)*, pages 157–162, 2014.

[39] George J. Vachtsevanos, K. Valavanis, Ohio Library, and Information Network. *Handbook of unmanned aerial vehicles*. SpringerReference, Dordrecht, 2015 edition, 2014;2015;.

[40] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar. The grasp multiple micro-uav testbed. *IEEE Robotics Automation Magazine*, 17(3):56–65, 2010.

[41] Randal Beard. Quadrotor dynamics and control rev 0.1, 2014.

[42] Silicon Sensing. Inertial sensor technology.

[43] RANDAL W. BEARD and TIMOTHY W. McLAIN. *Sensors for MAVs*, pages 120–142. Princeton University Press, 2012.

[44] *The Kinematics and Dynamics of Aircraft Motion*, chapter 1, pages 1–62. John Wiley & Sons, Ltd, 2015.

[45] John W. Hager, James F. Behensky, Brad W. Drew, and DEFENSE MAPPING AGENCY HYDROGRAPHIC/TOPOGRAPHIC CENTER WASHINGTON DC. The universal grids: Universal transverse mercator (utm) and universal polar stereographic (ups). edition 1, 1989.

[46] Wikipedia contributors. Geographic coordinate conversion — Wikipedia, the free encyclopedia, 2020. [Online; accessed 24-July-2020].

[47] Enge Per van Diggelen Frank. The world's first gps mooc and worldwide laboratory using smartphones.

[48] Inertial Sense. Data-sheet.

[49] Randal W. Beard and Timothy W. McLain. *Small unmanned aircraft: Theory and practice*. 2012.

[50] Wikipedia contributors. Digital filter — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Digital_filter&oldid=967510352`, 2020. [Online; accessed 25-July-2020].

[51] VANCE VANDOREN. The basics of numerical filtering. `https://www.controleng.com/articles/the-basics-of-numerical-filtering/#:~:text=A%20numerical%20filter%20inserted%20in,Filtering%20without%20a%20computer`, 2008. [Online; accessed 25-July-2020].

[52] Wikipedia contributors. Kalman filter — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Kalman_filter&oldid=967968225`, 2020. [Online; accessed 26-July-2020].

[53] S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

[54] Reza Hashemi and Sebastian Engell. Effect of sampling rate on the divergence of the extended kalman filter for a continuous polymerization reactor in comparison with particle filtering**the research leading to these results was funded by the erc advanced investigator grant mobocon (fp7/2012-2017) under the grant agreement n° 291458. *IFAC-PapersOnLine*, 49(7):365 – 370, 2016. 11th IFAC Symposium on Dynamics and Control of Process SystemsIncluding Biosystems DYCOPS-CAB 2016.

[55] Siddharth. Sridhar, OhioLINK Electronic Theses, and Dissertations Center. *Non-Linear Control of a Tilt-Rotor Quadcopter using Sliding Mode Technique*. University of Cincinnati, Cincinnati, Ohio, 2020.

[56] Ansu Singh, Deokjin Lee, Dong Hong, and Kil Chong. Successive loop closure based controller design for an autonomous quadrotor vehicle. *Applied Mechanics and Materials*, 483:361–367, 12 2013.

[57] McLain Timothy W. Beard Randal W. *6.1 Successive Loop Closure*. Princeton University Press, 2012.

[58] Ruijie He, S. Prentice, and N. Roy. Planning in information space for a quadrotor helicopter in a gps-denied environment. In *2008 IEEE International Conference on Robotics and Automation*, pages 1814–1820, 2008.