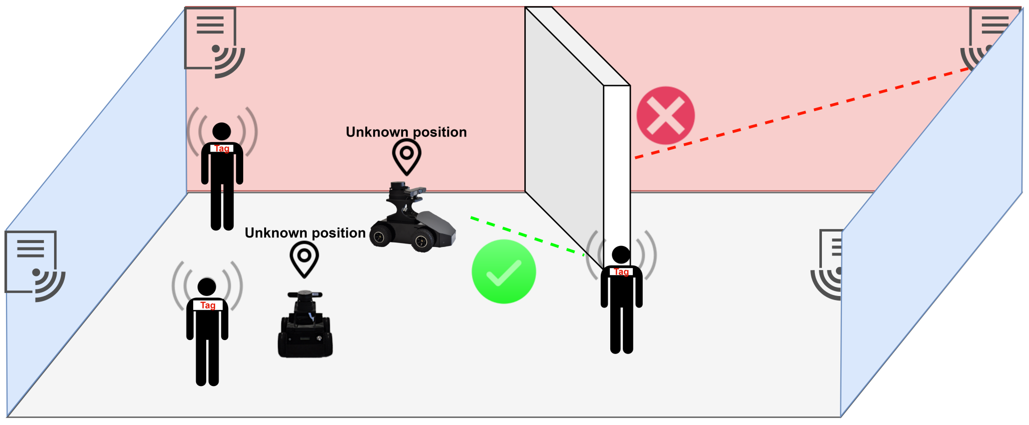
Machine Learning and UKF based Indoor Localisation using Ultra Wide Band (UWB) Sensors



Group 935
Long thesis
Control and Automation
Aalborg University
25-05-2022

AALBORG UNIVERSITET
STUDENTERRAPPORT

Control and Automation

Frederik Bajers vej 7B
9220 Aalborg Øst
https://www.en.aau.dk/education/master/control-automation

Title:

Machine learning and UKF based localisation using Ultra Wide Band(UWB) sensors

Project:

P9-project

Project period:

September 2021 - May 2022

Project group:

CA-9 935

Participants:

Pero Le
Rakesh Sundar
Karanam Venkata Sreeram

Supervisor:

Ming Shen
Henrik Schiøler

Total page number: 104
Appendix: 89 to 104
Date of completion: 25/05-2022

Synopsis:

The position of autonomous robot is of great significance for the purpose of navigation in an indoor environment. The goal of this project is improved localisation in Non Line of sight(NLoS) conditions. The current technologies during Line of sight (LoS) such as GPS, Bluetooth has an uncertainty of greater than 20 cm, whereas Ultra Wide Band sensors are accurate with a lower uncertainty of less than 8 cm. In this project, the tracking of an autonomous robot, assisted by Ultra Wide Band (UWB) sensors, in an indoor environment by deploying Unscented Kalman Filter (UKF) has been explored to mitigate effect NLoS. The autonomous robot in this project is addressed as "Deepcar", provided by a company known as SMPL robotics. A machine learning classification model is employed to detect whether a robot is in NLoS or LoS with the beacon by analysing the power of the impulse responses received at the beacon. This classification model is developed using TREK 1000 evaluation kit.
In the proposed approach, during NLoS conditions, persons in the indoor environment, who also carry UWB tags are used as non stationary beacons and are assigned variable weights in UKF, hence aiding in position estimation of the Deepcar, along with stationary beacons.

Preface

This report has been written in the period from September 1 to May 25, 2022, for the 3rd and 4thsemesters of the Masters in Control and Automation at Aalborg University.

We thank Prof. Ming Shen, Prof. Henrik Schiøler for their guidance and supervision throughout the course of this project. We would also like to thank Frank Rasmussen, Ben Klauman Krøyer, Phd student Qirui Hua, PhD student Zhao Zhou, and Jeppe Bro Christensen for their assistance in this project.

Aalborg University, June 1, 2022

[illegible]

Pero Le

<ple15@student.aau.dk>

Rakesh Sundar

Rakesh Sundar

<rkanth20@student.aau.dk>

K.V.Sreeram

Karanam Venkata Sreeram

<ksreer20@student.aau.dk>

Contents

Introduction 1

As technology continuously progresses, the need of robots becomes more significant and especially autonomous robots. The field of interest is increasingly pointed towards the ability to autonomously navigate in an indoor environment and nonetheless the property of avoiding static and dynamic obstacles.

An another field that is heavily researched is the use of robots in warehouses. For example, this could be to pack or move goods within the warehouse. Big companies like, Amazon, Proshop, Alibaba and BMW are just a few of the companies that utilize robot technology to their advantages [1], [2], [3].

Robot being aware of its own position is an integral part of enabling precise navigation. Complex indoor environment such as coal mines, hospitals, and warehouses have an essential need of more precise localisation[4]. The order of localisation, that utilises GPS and other outdoor satellite systems, is in meters, which is suitable for outdoor purposes[5], but indoor environment necessitates even more precision. Ultra Wide Band (UWB) Technology is a wireless communication technology, which is especially suitable for indoor positioning because of its superior advantages such as low power consumption and good resistance to interference[6]. Based on Time Difference of Arrival (TDOA) approach, Ultra Wide Band has a capacity to localise the robot in order of centimeters. Hence, it is decided to explore the topic of indoor localisation using UWB sensors.

1.1 Project setting

The application of autonomous robots has a great variety of usage. Ranging from

- Eldercare.
- Parcels allocation.
- Cleaning.
- Autonomous car.
- etc.

In all sectors, accuracy, cost-efficiency and reliability plays a significant role in order to avoid damaging or hurting the people and the environment where robot is operated. Especially, in real-time systems created for indoor navigation face the problem of accuracy with technologies such as GPS, bluetooth and Wi-Fi, however a new aspect has made a prominent appearance in the later years, which is UWB. It has the opportunity to fill-in the gap [7]. Hence, an autonomous robot, with Ultra Wide Band sensors mounted, has been considered in this project.

1.2 Initial problem statement

Based on the project setting described in the section 1.1, the problem statement can be reformulated in the following way:

"How an autonomous robot can be localised using Ultra Wide Band Sensors in an indoor environment?"

1.3 State of the art

Robot localization implies the robot's capacity to determine its orientation and position within a defined coordinate system (frame of reference).

The problem of localisation in an indoor environment can be solved in multiple ways. This section deals with exploring the current methodologies employed in NLoS identification and localisation techniques. The NLoS identifications are done based on statistical information, channel diagnostic algorithm, machine learning classifiers and these are discussed in [8] [9] [10]. The localisation is a two step approach where in the first step certain position parameters are obtained and in the second step the position is determined by deploying position estimation algorithms and these are discussed in [11] [12].

For NLoS Identification: In the approach of using relevant statistical tools the characteristics of the received multipath components are used. This is employed by using the kurtosis, the rms delay spread and the mean excess delay to view the delay and amplitude information during NLoS and LoS conditions. If this information is available prior for both LoS and NLoS conditions in a particular environment then likelihood ratio tests can be conducted to determine the hypothesis (LoS or NLoS). In the approach of using channel diagnostic algorithm(CDA), we use two received signal strengths for identifying the LoS or NLoS condition. One is first-path signal power level(FSL) and the other is received signal power level(RSL). The unit of these two power levels is dBm. A CDA uses the difference between RSL and FSL, and if the difference greater than $10dB$ then it is said to be in NLoS condition whereas when the difference is lesser than $6dB$ it is said to be in LoS condition. However the region of difference between $6dB$ and $10dB$ is not defined and hence may cause improper identification. In the approach of using Machine Learning classifier, a set of features are selected for the training data set. The classification of LoS and NLoS can be done using SVM. SVM is a supervised learning classifier and it is one of the most widely used because of its low generalization error, requires only a few user defined variables[10].

For Localisation: The position parameters are determined initially and the next step is estimation of the position. The positioning algorithms are classified by the signal properties they utilise. The position parameters that can be used for positioning are Channel Impulse response (CIR), multipath power delay profile (PDP), Angle of arrival (AOA), Time of arrival (TOA), Time Difference of Arrival ($TDOA$) and Received Signal Strength Indication($RSSI$)[11]. The signal properties such as AOA, TOA, $TDOA$ and $RSSI$ are used for geometric methods of position estimation and the signal properties such as CIR and PDP is used by fingerprinting. The positioning algorithms convert the recorded signal properties into distances and angles after which the position of the actual object to be tracked is computed. Hence both the signal property and the positioning algorithm work cohesively to estimate the position of the target node. The accuracy of the estimated position is determined by the positioning algorithm deployed and it is also determined by the accuracy of the signal information collected[12]. The prominent positioning algorithms are :

- Fingerprinitng or Scene Analysis
- Trilateration
- Triangulation
- Proximity

1.4 Scope and limitations

This project employs the use of an autonomous robot named "Deepcar". It is equipped with camera, LIDAR sensor and UWB decawave module. Deepcar is designed for many functionalities such as self driving using camera, mapping of environment using LIDAR, indoor localization. One of the functionalities explored in this project is indoor localization using UWB decawave module.

Due to the insufficient documentation of deepcar, the state model and the controller for the deepcar are not designed in this project. Position localization of deepcar must be accurate for better navigation in an indoor environment. UWB positioning is accurate when the tag on deepcar is in line of sight (LoS) with all the anchors. But UWB is prone to many problems such as clock synchronization between two transceivers, Non Line of sight (NLoS) interference, interference due to environment, and so forth[13]. In case of Non Line of sight (NLoS) with one of the anchors, the positioning is greatly affected by the reflection of waves from the obstacles, that are causing NLoS. The positioning of deepcar can be made better by utilising persons.

This project builds on the work presented in article [14]. Identification of LoS/NLoS has been given more significance, and hence different metrics for the classifier are considered to deploy it in real-time. The environment in which the deepcar is operated is considered to be of much importance because it should be mapped before implementation. The environment that does not vary much with time is preferably considered such as warehouse in an industry, but if the environment is unpredictable such as home, the deepcar might not be the best option. This report is aimed to analyse the effect of interference due to NLoS conditions on localisation of robot using Ultra Wide Band technology and exploring approaches to improve accuracy of localisation [15].

1.5 Use case for indoor navigation of Deepcar

Deepcar is navigated in a closed environment (such as room, inventory house) which is mapped into a 2D grid, including all the stationary obstacles (such as walls, racks, pillars). A fixed number of anchors, which are decawave UWB modules configured as anchors, are positioned at the corners of the environment. The number of anchors should be chosen on the basis of location of stationary obstacles in the considered setting.

In this project(figure 1.1), any person entering the environment is considered as a non stationary anchor by the UWB tag he/she wears. These non stationary anchors are considered as obstacles by the deepcar. During NLoS conditions with the fixed anchors, a person, whose position can be estimated accurately using indoor localization, is considered as an anchor and the deepcar estimates its position.

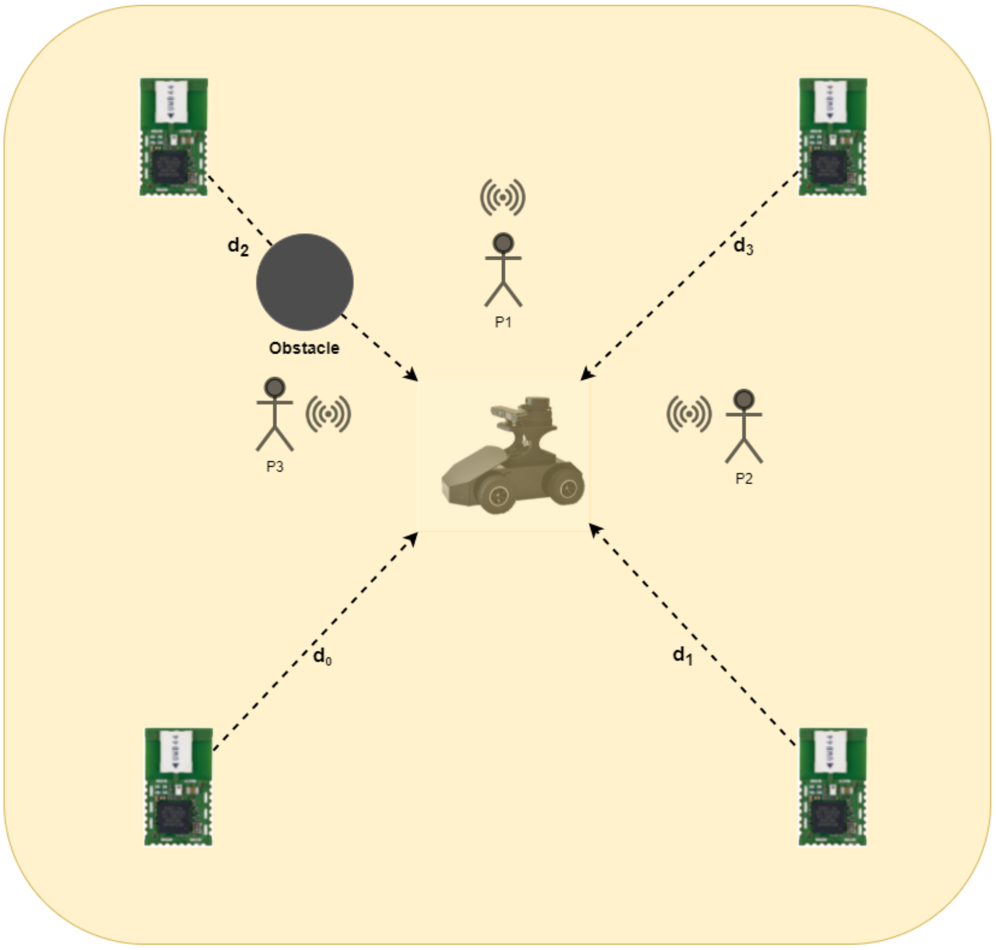


Figure 1.1: Demonstration of Use Case

1.6 Final problem statement

Based on the limitations presented in the previous section, the problem statement can be reformulated in the following way:

"How to improve localisation performance in an indoor environment using Ultra Wide Band, in Non Line of Sight (NLoS) conditions?"

1.7 Solution overview

As described in the use case, the position of deepcar is calculated by using the anchors in the room. The line of sight or non line of sight condition of deepcar with an anchor is identified by employing a trained machine learning model, which takes Channel Impulse Response(CIR) from the tag connected to deepcar. The machine learning model is developed for the environment where deepcar is operated. An evaluation kit is used to obtain CIR's for various settings of LoS and NLoS conditions, which are used to train the model. An unscented kalman filter has been designed to track deepcar in both LoS and NLoS conditions. In LoS conditions, UKF utilizes the stationary anchors to localise the deepcar. In NLoS conditions, UKF utlilizes the persons as non-stationary anchors to estimate deepcar position accurately.

Line of sight and Non-Line of sight 2

This chapter will deal with the identification of the two scenarios, "line of sight" (LoS) and "non-line of sight" (NLoS) by using channel impulse response (CIR). All the provided information is from the following source [16].

2.0.1 Line of sight

In case of line of sight two scenarios could emerge.

1. **Optically LoS:** Specify the arrangement of no obstruction of physical objects in-between the transmitter and receiver antenna. This is considered in this chapter.
2. **Fresnel Zone LoS:** This refers to the area around optically LoS into which electromagnetic waves spread out, this type of LoS is known as Fresnel Zone and has its application in field such as RF.

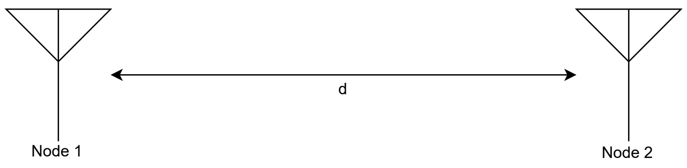


Figure 2.1: Example of Line of sight

Communication range

First approach is to deal with the power signal in order to know the range of the signal, then follow up by an accuracy technique, which is Time-of-Flight (ToF) in order to estimate the distance.

The communication range is defined upon two criteria, "the level of arrived signal" and "the sensitivity of the receiver antenna". As long as the signal level surpasses a certain threshold, the signal will be acknowledged.

The following equation will describe the received power signal:

$$P_R = P_T + G - L - 20\log_{10}(\frac{4\pi f_c d}{c}) \tag{2.1}$$

where
$\mathrm{P_R}$ = Received Signal level.
$\mathrm{P_T}$ = Transmitted power,

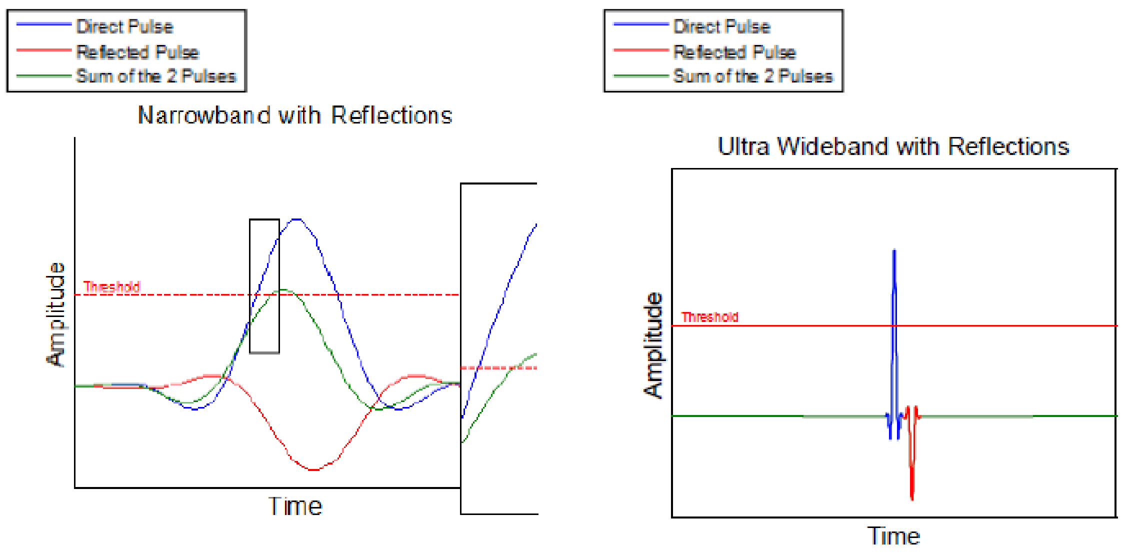
G = The gain from the transmitting, received antennas and other amplifiers.
L = Losses in the system like PCB, cable, connector etc.
c = Speed of light (299792458).
f_c = Center frequency of the channel used (in Hertz).
d = Distance in meters between transmitter and receiver.

The received signal P_R will indicate the robustness of the communications by considering the Link Margin, which is the difference between the value of P_R and the threshold of the receiver. A large Link Margin represents a robust communication and can be attenuated if wanted, whereas the opposite indicates a bad signal.

Distance Determination

The following objective is to determine the distance between two antennas in LoS. In order to determine the distance, time-of-flight is a necessary tool. It is with great importance to accurately measure the time-of-flight, as 0.1 ns provides a measurement precision of 3 cm, whereas 1 ns gives a deviation of 30 cm.

The ideal case of LoS occurs when there is no multipath propagation, however this is not possible in real life scenario even when no object is obstructing. The effect of these multipath(s) could change the direct pulse in normal circumstances, however this is not a severe problem in case of UWB since it is narrow pulses.



The most common reflector is the ground which could occur in both the outdoor and indoor environment. This leads to an infinite number of multipaths among the antennas.

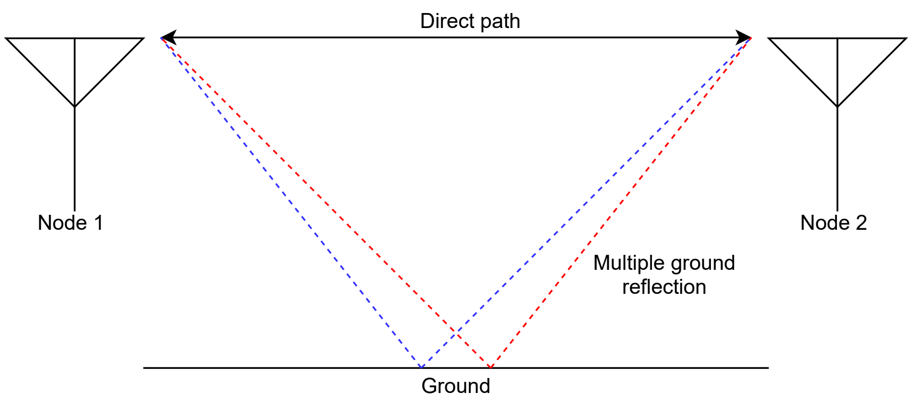


Figure 2.2: Illustration of ground as a reflector

2.0.2 Non-line of sight

NLoS differs from the LoS as the peripheral field contains obstruction and eventually this will delay, reflect or refract the signal.

When occupying NLoS, three main categories should be examined.

1. Time of flight error due to the refractive index, which depends on the hindering object.
2. Attenuation, in the signal power, leads to a reduction in the communication range.
3. Attenuation of the direct path signal as interference, of multipath, occurs as a result of the reflection and refraction.

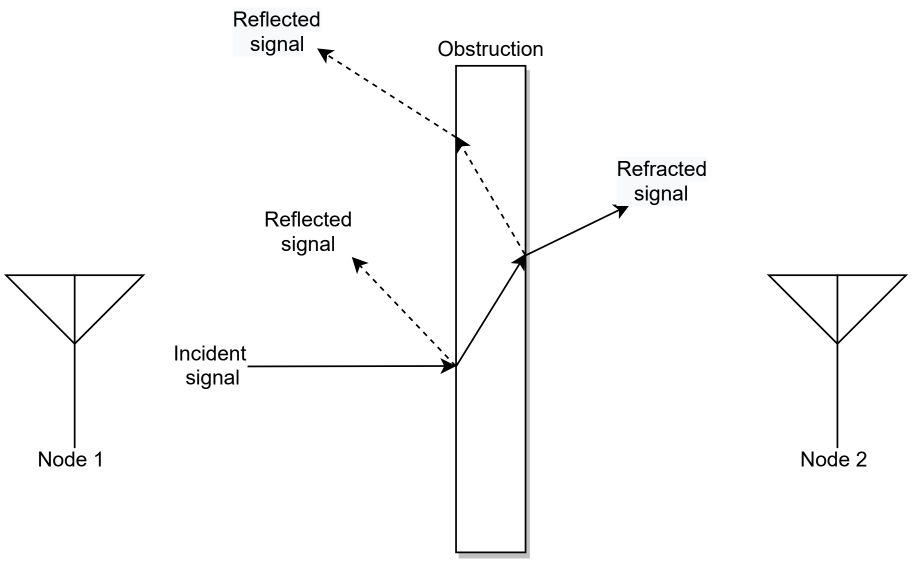


Figure 2.3: NLoS with reflection and refraction of the incident signal

Communication range

The effect of signal attenuation, by obstacle, on communication range depends on the material, thickness and the frequency of the incident signal. So the obstructions attribute to Friis equation likewise:

$$P_R = P_T + G - L - 20\log_{10}(\frac{4\pi f_c d_1 + d_2}{c}) - L_{MATERIAL} \tag{2.2}$$

where
P_R = Received signal level.
P_T = Transmitted power.
G = The gain from the transmitting, received antennas and other amplifiers. The DWM1000 emits sufficient power so that -41.3 dBm/MHz is radiated from the transmitting antenna.
L = Any losses in the system like PCB, cable, connector etc.
c = Speed of light (299792458 m/s).
f_c = Center frequency of the channel used (in Hertz).
d_1 = Distance in free space from transmitter to attenuating material.
d_2 = Distance in free space from attenuating material to receiver.
$L_{MATERIAL}$ = Describe the loss in material.

Effect of refraction

Effect of refraction contributes to a delayed signal, which affects the calculation of the position as mentioned before. The following equation will describe effect of NLoS:

$$ToF = \frac{d-w}{c} + \frac{w \cdot R}{c} ToF = \frac{d + w \cdot (R-1)}{c} d' = ToF \cdot c = d + w \cdot (R-1) \tag{2.3}$$

where
d = Physical distance between two nodes.
d' = Calculated distance between two nodes.
c = Speed of light.
w = Width of the object between the two nodes.
R = Refractive index.

NLOS and multipath operation

Third case describes the NLOS scenario with multipath.

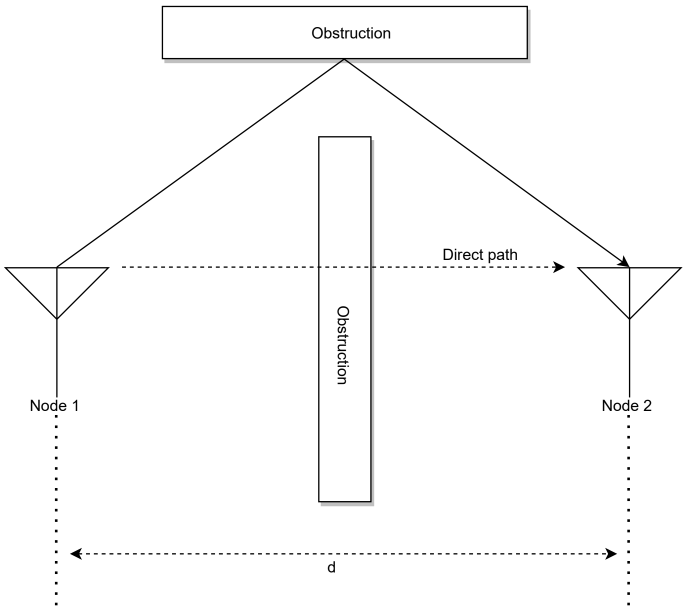


Figure 2.4: NLoS alter the signal and delays the original signal

When the scenario is in NLoS as in figure 2.4 the effect is a misinterpretation of the distance, because the amplitude of the direct path is attenuated and has become lower than the multipath's amplitude, hence misinterpreting multipath signal as a direct path signal.

This could also be illustrated as in figure 2.5

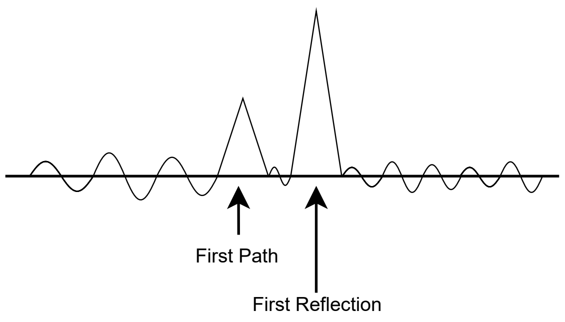


Figure 2.5: Reflection signal power interfere the direct path signal power

UWB Positioning 3

This chapter gives a brief description about the estimation of distance between two Ultra Wide Band (UWB) modules using a Two way ranging(TWR) protocol. A mathematical approach has also been discussed to estimate the coordinates of deepcar using anchors.

3.1 Ultra Wide Band (UWB) Sensors

Information is transmitted in UWB by radio energy at specific time intervals and occupying a large bandwidth (typically more than 500 MHz). The key difference between radio systems and UWB is that a small duration pulse is directly propagated, without any Radio Frequency (RF) mixer in UWB systems. Detection of UWB detection is difficult because of the low Power Spectral Density (PSD) and pseudo random characteristic of the signal. The advantages of having high bandwidth are as follows [17]:

- High Multipath resolution
- Resistant to Multipath propagation and jamming/interference
- Good Low Probability of Interception and Detection (LPI & LPD) properties

In the deep car, UWB sensors are used primarily for following purposes:

- Positioning of the deep car using tags
- Obstacle Detection

3.1.1 UWB Positioning Algorithm

UWB positioning is nearly accurate because of the short duration pulses. If the arrival time of one of these pulses is measured, it is possible to determine the distance between the receiver and the transmitter with a small uncertainty. If the distances from multiple receivers and one transmitter are measured, then the position of the transmitter can be determined with more precision using various position estimation techniques.

3.1.2 Distance Determination between Anchor & Deepcar

Distance of anchor from tag is obtained using time of flight(ToF) of the radio signal between them. ToF measurements are obtained using a ranging protocol Two Way Ranging (TWR) technique.

Two Way Ranging(TWR)

This is the simplest protocol, which uses only three messages. Tag sends out a start message at instant t_1. Tag receives two messages consisting an acknowledge request in one message and time stamps in the other (Figure 3.1). From the timestamps received, the ToF [18] is measured as follows:

$$ToF = \frac{1}{2}(t_4 - t_1 - (t_3 - t_2)) \tag{3.1}$$

It is important to note that the timestamps t_2 and t_3 are computed by the same crystal oscillator as t_1 and t_4 are computed. In real systems, it is always not ensured that there will be no clock drifts between the tag and the anchor. In such cases, a co-efficient(k) is defined to account for this drift. Consider the frequency of the tag as f_T and the frequency of the anchor as f_A. Then $k = \frac{f_T}{f_A}$ [18]. The equation (3.1) can be modified as

$$ToF_{modified} = \frac{1}{2}(t_4 - t_1 - k(t_3 - t_2)) \tag{3.2}$$

This measurement is used in calculating distance(d_{TA}) between the anchor and the tag in Line of Sight conditions as

$$d_{TA} = c \cdot ToF_{modified} \tag{3.3}$$

where c is defined as the speed of electromagnetic wave.

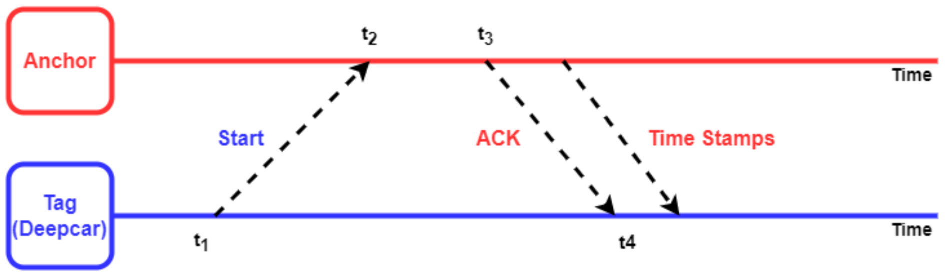


Figure 3.1: TWR Ranging

3.1.3 Mathematical Approach for determination of Deepcar Position

When a tag is connected to the deepcar, distances of each anchor from the tag are used to determine the position of deepcar mathematically. It is assumed that the distance(d_i) of tag from the anchor is interpreted as tag could lie anywhere on the sphere of radius(d_i) with anchor at the centre of sphere. Hence, for 3D position estimation, at least four anchors should be mounted. As the visualisation of spheres is difficult, 2D position calculation(figure 3.2) is represented using three anchors [19].

Assume anchors are denoted by A_i and deepcar is denoted as DC. Prior before computation, position of anchors and distances are obtained.

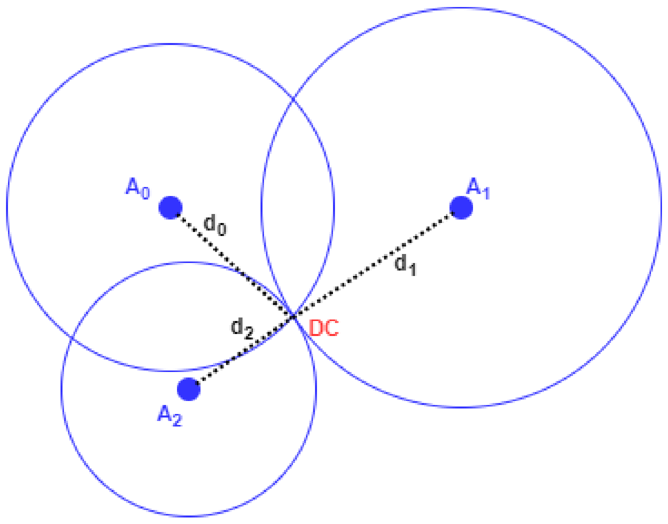


Figure 3.2: 2 dimensional Position estimation of Deepcar

$$\begin{aligned}(x_{DC} - x_{A_0})^2 + (y_{DC} - y_{A_0})^2 &= d_0^2\\(x_{DC} - x_{A_1})^2 + (y_{DC} - y_{A_1})^2 &= d_1^2\\(x_{DC} - x_{A_2})^2 + (y_{DC} - y_{A_2})^2 &= d_2^2\end{aligned} \tag{3.4}$$

Equations in (3.4) are rearranged into a set of two equations by subtraction

$$\begin{aligned}2(x_{A_1} - x_{A_0})x_{DC} + 2(y_{A_1} - y_{A_0})y_{DC} &= d_0^2 - d_1^2 - \left(x_{A_0}^2 + y_{A_0}^2\right) + \left(x_{A_1}^2 + y_{A_1}^2\right)\\2(x_{A_2} - x_{A_1})x_{DC} + 2(y_{A_2} - y_{A_1})y_{DC} &= d_1^2 - d_2^2 - \left(x_{A_1}^2 + y_{A_1}^2\right) + \left(x_{A_2}^2 + y_{A_2}^2\right)\end{aligned} \tag{3.5}$$

Equation (3.5) expressed in matrix form (3.6)

$$AX_{DC} = B \tag{3.6}$$

where

$$A = 2\begin{bmatrix}x_{A_1} - x_{A_0} & y_{A_1} - y_{A_0}\\x_{A_2} - x_{A_1} & y_{A_2} - y_{A_1}\end{bmatrix} X_{DC} = \begin{bmatrix}x_{DC}\\y_{DC}\end{bmatrix}$$

$$B = \begin{bmatrix} d_0^2 - d_1^2 - (x_{A_0}^2 + y_{A_0}^2) + (x_{A_1}^2 + y_{A_1}^2) \\ d_1^2 - d_2^2 - (x_{A_1}^2 + y_{A_1}^2) + (x_{A_2}^2 + y_{A_2}^2) \end{bmatrix}$$

X_{DC} gives the position of deepcar, whereas A and B are obtained from position of anchors and distance measurements on the tag. Least squares technique is applied to solve matrix equation (3.6) for position of deepcar. This approach can be expanded for n anchors and 3 dimensional position estimation.

$$X_{DC} = \left(A^T A\right)^{-1} A^T B \qquad (3.7)$$

Machine Learning 4

This chapter deals with different machine learning models and the topics correlated, such as classification, model selection and feature extraction. An intuitive understanding on design of classifier and the concerned mathematical aspects have also been discussed.

4.1 Introduction

The fundamental idea of machine learning concerns design and optimization of an algorithm that will improve the performances of the algorithm by examples or experience (training).

There are different types of machine learning and it is mainly divided into three categories as following [20]:

- Supervised Learning: Supervised learning uses labelled data, which means by providing the inputs along with the outputs, the algorithm will be trained to correctly map the input with the corresponding output. Furthermore it can be divided into two subsections, which are Classification and Regression.
  - Classification: Classification considers the discrete variable case (pattern recognition).
  - Regression: Regression appertain to real-valued output (prediction), which are continuous variables.
- Unsupervised Learning: Unsupervised learning uses unlabelled data, which differs from the labelled one by not having the output given in prior, and thereby only tries to find various patterns in the data set by the following approaches:
  - Density estimation.
  - Clustering.
  - Dimensionality reduction.
- Reinforcement learning: As this project does not concerns reinforcement learning, it is not described.

4.1.1 Classification

Classification is the prediction of a random variable Y from the a variable X in discrete state.

Definition 4.1.1. Consider following data set $(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_n, \boldsymbol{y}_n)$ where $\boldsymbol{x}_i = (x_{i1}, ..., x_{id})^T \in \mathcal{X} \subset \mathbb{R}^d$ could be described as a d-dimensional vector and $\boldsymbol{y}_i \in \{1, ..., k\}$ retrieve values in some finite set $\mathcal{Y}$. Classification rule is a function $h : \mathcal{X} \to \mathcal{Y}$. Each of the discrete variable represent data or observation and has a respective label corresponding to it. Thus, the function $h(x)$ will predict Y based on the X [21].

4.1.2 Deep neural network (DNN)

Deep neural network consists of several segments such as input layer, hidden layers and output layer. The first layer is the input layer. This layer takes in multiple inputs at a time as illustrated in figure 4.1.

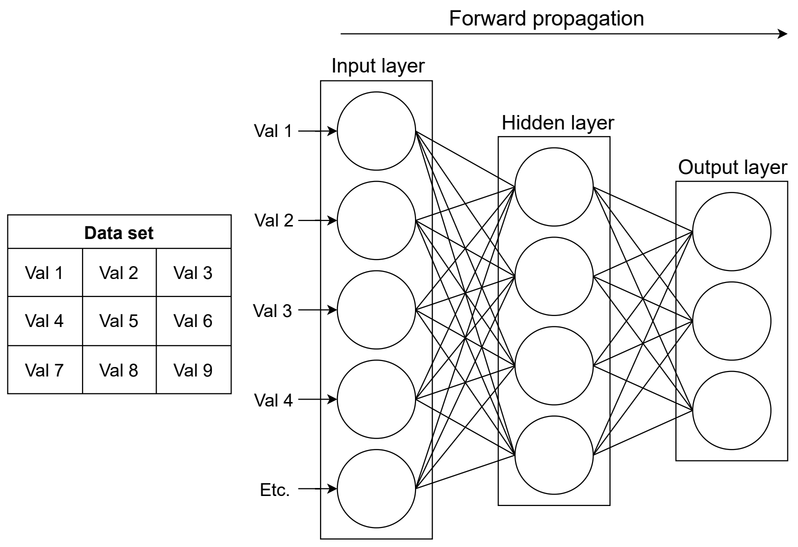


Figure 4.1: Principle of DNN classifier

These inputs are assigned weights individually and forwarded into the hidden layer by channels(indicated by lines). Inside the hidden layer, there is a bias, which is added to the previous input and is given to a activation function (Sigmoid, Relu etc). Further, this output is compared to a threshold, whether to propagate the weighted input or not, and this process is repeated until this propagation reaches output layer as in figure 4.2.

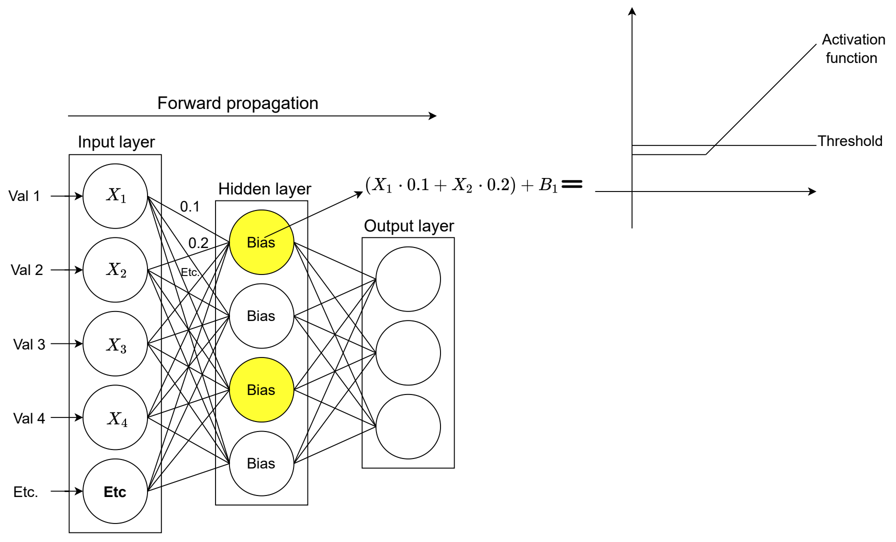


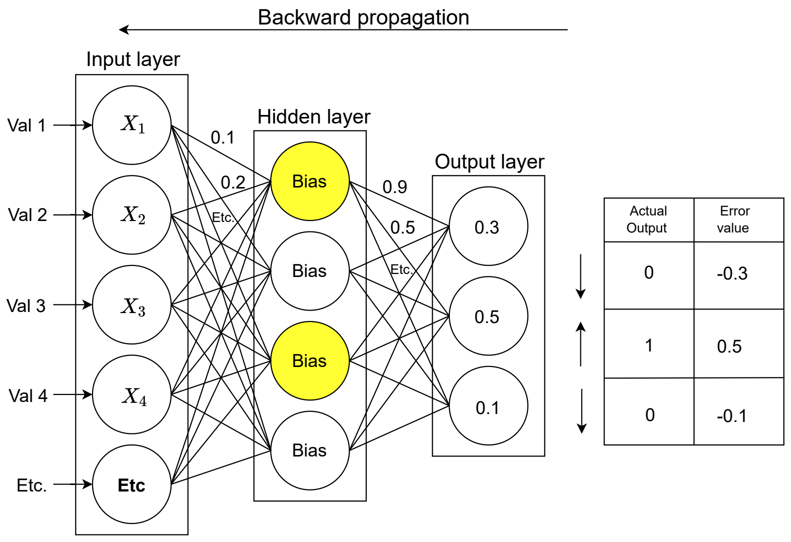
Figure 4.2: DNN threshold reached

During training phase, the predicted output, at the output layer, is compared to the actual value, thus computing the error, which is propagated back (only if the error is a non-zero value) during backward cycle to update the weights, assigned in the hidden layers, so as to lower the deviation between the actual and predicted values as shown in figure 4.3,[22].

The error formula inside of the output layer is given by:

$$\text{Error} = \sum_{i=1}^{n} (y_n - f(x_n))^2 \tag{4.1}$$

where
y$_n$ = actual value
$f(x_n)$ = predicted value



Actual Output	Error value
0	-0.3
1	0.5
0	-0.1

Figure 4.3: DNN Evaluation

4.1.3 Support Vector Machines (SVM)

The objective of support vector machine is to obtain a hyperplane(s) that segregates classes. In order to explain the concept of SVM, an example of a two-class classification will be considered (see figure 4.4), and a linear model is developed for this purpose [23][24].

$$y(x) = w^T \phi(x) + b \tag{4.2}$$

where
$\phi(x)$ = feature space transformation.
b = bias parameter.
w = distance, which is normal to the hyperplane.

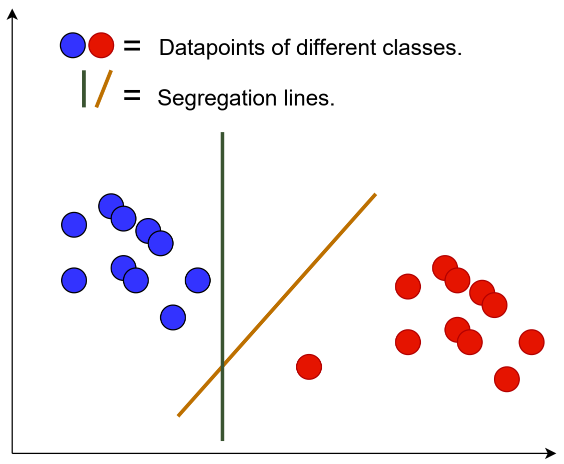


Figure 4.4: SVM classifier with two classes

To achieve the most optimal hyperplane, it is important to notice that the training data x_1,...x_n, also comprises a target value t_1,...t_n where $t_n \ \varepsilon \ \{-1, 1\}$. In case of linear distinction, the variables w and b from equation 4.2 will be optimized by satisfying the following conditions[24]:

$$y(x) = w^T\phi(x) + b > 0 \quad \text{for } t_n = 1 \tag{4.3}$$

$$y(x) = w^T\phi(x) + b < 0 \quad \text{for } t_n = \text{-1} \tag{4.4}$$

These lines are parallel to the segregation line and is displaced by a distance, called margin. The margin (between parallel lines on support vectors) is maximised, thus aiding us in future inferences of datapoints. This could be express as following:

$$\arg\max_{w,b}\{\frac{1}{\parallel w \parallel}\min_{s}[t_n(w^T\phi(x_n) + b)]\} \tag{4.5}$$

The following illustration (figure 4.5) demonstrates the concept in more detail, and it is implied that the support vectors affects the position and the distance of the segregation line(s).

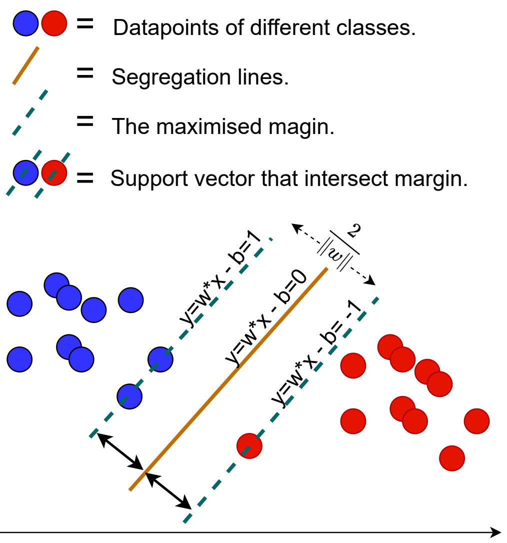


Figure 4.5: Detailed concepts of SVM.

The figure 4.5 exemplify the principle of a hard margin, which indicates that the data is restricted to be exactly on the lines, however it will affect the performance of the classifier and therefore a soft margin is often considered, as shown in figure 4.6. Soft margin allows the datapoints to exceed the segregation lines, which pertain to the bias and variance trade off, which will be described later in the report[25].

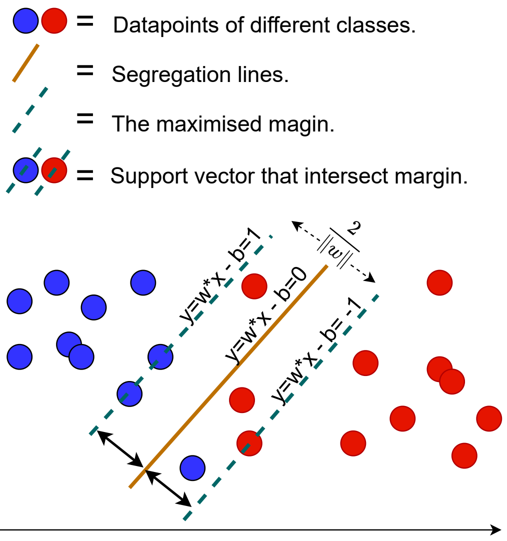


Figure 4.6: Illustration of SVM in case of soft margin.

4.1.4 K-nearest neighbor (KNN)

K-nearest neighbor is a classification method that classifies data without any assumptions regarding the distribution, it considers the label for the nearest neighbors to the point x. To select the closest neighbor(s) the Euclidean distance is one of the commonly used:

$$\sqrt{\sum_{r=1}^{d}(\boldsymbol{x}_{ri} - \boldsymbol{x}_{rj})^2} \tag{4.6}$$

The value of k (number of neighbors) is inversely proportional to the complexity of the classification model. [26],[27].

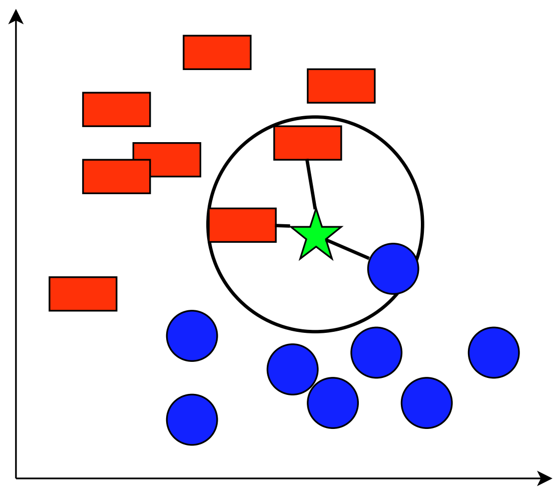


Figure 4.7: Knn with 3 nearest neighbor.

4.1.5 Principal Component Analysis (PCA)

Conventionally, all of the dimensions of the dataset produce some variation in datapoints. The dimensions that contain lower variance (dimensions that will not affect the separation boundary to a higher degree) can be disregarded, prior to training the classification model. Hence the dimensions of the dataset are reduced. This task is fulfilled by adopting one of the dimensionality reduction techniques termed as "Principal Component Analysis".

Principal Component Analysis is projection of datapoints onto a lower dimensional subspace, known as "principal subspace", such that variance of projected data is maximized[28].

Consider a dataset $x = x_1, x_2, x_3...x_N$, where N is the number of datapoints. Assume there are M dimensions in the dataset considered. The variance and mean of x is calculated from equation (4.7) and equation (4.8).

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_n \tag{4.7}$$

$$S = \frac{1}{N}\sum_{i=1}^{N}(x_n - \overline{x})(x_n - \overline{x})^T \tag{4.8}$$

Let the data be projected onto a lower dimensional vector u, chosen arbitrarily. The variance of the projected data is given by the equation (4.9)[28]. Each vector, along which there is a variation in x, is obtained by maximizing the variance of projected data with respect to vector u, as shown in equation (4.10)[27].

$$\frac{1}{N}\sum_{i=1}^{N}\left\{u^T x_n - \overline{x}\right\}^2 = u^T S u \tag{4.9}$$

$$\begin{aligned} \max_{\boldsymbol{u}} \quad & \boldsymbol{u}^T \boldsymbol{S}\, \boldsymbol{u} \\ \text{subject to} \quad & \boldsymbol{u}^T \boldsymbol{u} = 1 \\ & \boldsymbol{S} = \boldsymbol{S}^T \geq 0 \end{aligned} \tag{4.10}$$

Due to the constraint of unitary vector, the optimization problem is constrained. It is transformed into an unconstrained optimization by employing Lagrange multipliers (λ)[28]. Since, co-variance matrix is symmetric, another constraint can be removed,

$$\max_{\boldsymbol{u}} \quad u^T S\, u + \lambda(1 - u^T u) \tag{4.11}$$

By differentiating and equating the (4.11) to zero, the vectors, along which there is data variance, are the eigen vectors of covariance matrix (S)(shown in equation (4.12))[28]. The eigen vector, which has maximum value, is taken as the first principal component. Thus, principal components vector (u) is a decreasing magnitude eigen vector matrix.

$$Su = \lambda u \tag{4.12}$$

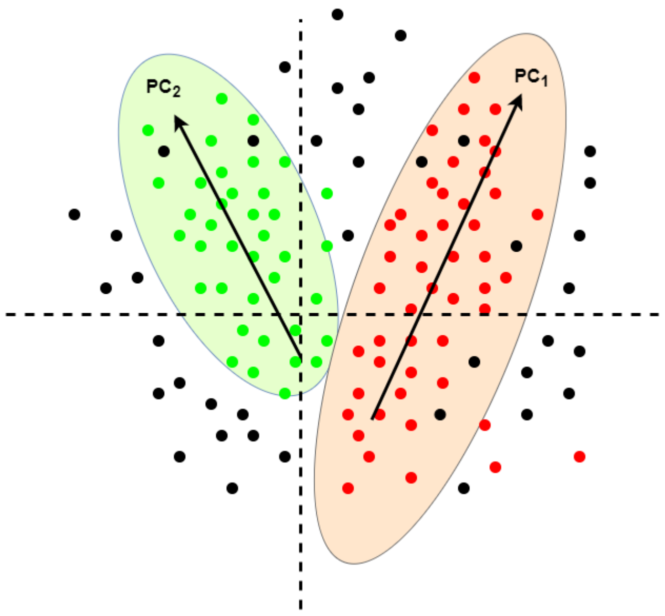


Figure 4.8: Demonstration of PCA

The data considered in figure 4.8 is of higher dimensions, represented in two dimensional space. The first principal component(PC_1) represents the direction of maximum variation in data (region in red) and the second principal component (PC_2) represents the direction in which there is second maximum variation in data (region in green). The direction of the variation in data points, colored black, might be represented by rest of the principal components. An appropriate number of principal components are chosen based on the amount of data variation, needed for the classifier.

4.1.6 Model selection

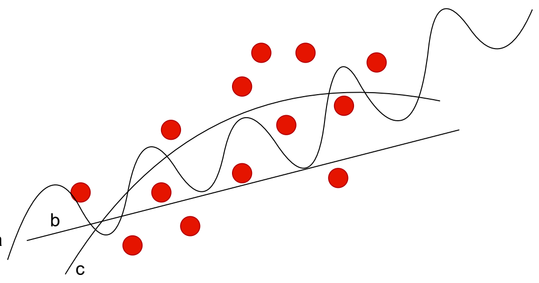


Figure 4.9: Model selection depicts 3 polynomials.

Overfitting signifies a model (a) capturing true relationship between datapoints in the training set, resulting in low bias and high variance, as depicted in figure 4.9. This model can't be generalized appropriately, when new datapoints are tested using this kind of model. The overfitting model tends to provide a poor performance, since the model is trained to fit a specific pattern of the data. In contrast, a model that underfits consist of high bias and low variance, this type of model is illustrated as polynomial (b) in figure 4.9. Model b does not replicate the model well, however it has a consistent generalisation for new datasets. The objective of machine learning is to find a model that generalizes well even in the case of new datapoints, which is (in this case) represented as polynomial (c) as it contains the flexibility to replicate the datapoints better than (b), but not as sensitive as polynomial (a) [29].

In order to estimate a well defined classifier, the needs of true error and sample error is essential. The ideal case is to obtain a low true error, which means to achieve a low probability of misclassification, considering the following equation [30]:

$$L(h) = P(f(\boldsymbol{X}) \neq h(\boldsymbol{X})) \tag{4.13}$$

where
P = probability.
h(x) = hypothesis function.
f(x) = true/target function.

True error represent the entire population, which makes it difficult to compute, hence a representative amount of the population is obtained and used, giving the sample error function.

$$\text{Sample error} = \frac{1}{n}\sum\nolimits_{x\in S} \delta(f(\boldsymbol{X}) \neq h(\boldsymbol{X})) \tag{4.14}$$

The sample error is minimized by the objective function (also known as Mean Squared Error (MSE)) in order to acquire the generalised true error.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2 = Var(\hat{Y}_i) + Bias^2(\hat{Y}_i) \tag{4.15}$$

where
$\hat{Y}_\text{i}$ = estimate value.
Y_i = true value.

$$Bias(\hat{f}) = E[\hat{Y}_i] - Y_i \tag{4.16}$$
$$Var(\hat{f}) = E[(\hat{Y}_i - E[\hat{Y}_i])^2] \tag{4.17}$$

Since variance and bias are used to compute MSE (equation (4.15)), it is possible to justify for the optimal complexity point for the machine learning algorithm as shown in the figure 4.10. However to ensure that the complexity of the model selected is well designed, a technique called cross-validation should be utilized [21].

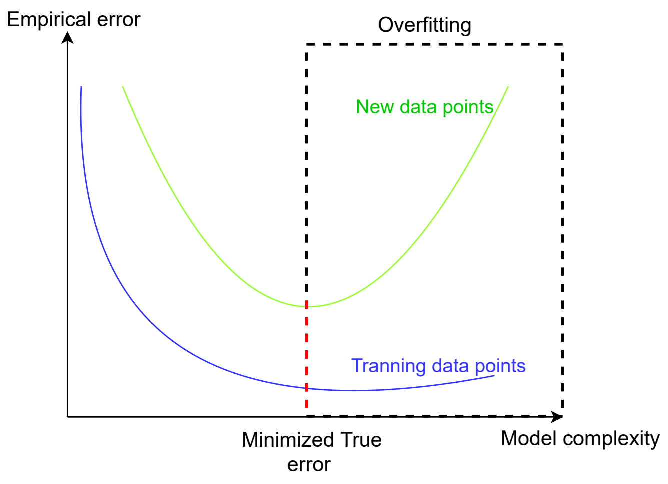


Figure 4.10: Desired true error point indicated by the red dotted line.

4.1.7 Cross validation

The main justification of cross-validation is to assure the complexity of the model holds true. This is done by dividing the total amount of sample into two parts. These parts are randomly split into training set and validation set. However this comes with a disadvantage, of using decreased amount of data for training and is computationally heavy. Usually, when validating the generalised model without cross validation, the predictive performance will be noisy, but K-fold cross-validation compensates this problem by repeatedly evaluating the data K times. This means that if K=4 the data set is divided into 4 parts, approximately at same size, and evaluated 4 times, like depicted in figure 4.11,[21],[24]

$$\text{Amount used for training data} = \frac{K-1}{K} \tag{4.18}$$

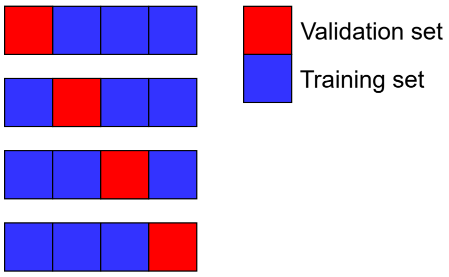


Figure 4.11: Illustration of the cross validation technique.

Kalman Filter 5

This chapter is dedicated to provide an intuitive understanding on the working of Kalman filter, instead of being mathematically focused, and is extensively derived from the "Kalman Filtering:Theory and Practice using MATLAB"[31] by Mohinder.S.Grewal.

5.1 Introduction

Kalman filter is a statistical algorithm, also termed as Linear Quadratic Estimation(LQE), to obtain a close approximate in dynamic systems when the measurement device is erroneous. The Kalman filter algorithm uses a string of measurements detected over a period, together with statistical white noise and other uncertainties, and also it has a relatively simple structure and requires small computational power. This is a straightforward approach that is employed in almost all radar systems.

5.2 Working

Kalman filters are used to estimate states based on linear dynamical systems modelled in state space. The process model is defined for change of the state from time instance k–1 to time instance k as:

$$x_k = Fx_{k-1} + Bu_{k-1} + w_{k-1} \tag{5.1}$$

where
F = state transition matrix applied to the previous state vector x_{k-1}
B = control-input matrix applied to the control vector u_{k-1}
w_{k-1} = the process noise vector that is assumed to be zero-mean Gaussian with the covariance Q.

The measurement model and the process model together explain the relation between the measurement and the states at that current time instance k as:

$$z_k = Hx_k + v_k \tag{5.2}$$

where
z_k = measurement vector

H = measurement matrix
v_k = the measurement noise vector that is assumed to be zero-mean Gaussian with the covariance R.

The purpose of the Kalman filter is to produce estimate of x_k at time k , with the given initial estimate of x_0, the system details are given by F , B , H , Q , and R, and by utilising the string of $z_1, z_2, z_3 \ldots, z_k$ which are measurements. The matrices are taken to be invariant over time so the subscripts of these are ignored because of this assumption. The statistical properties of the noise is reflected through the covariance matrices, but however in many practical applications the actual statistical properties of the noises are not known. Hence, to obtain the desired performance generally the Q and R are treated as tuning parameters[32].

5.2.1 Algorithm

Kalman filter algorithm consists of two stages: prediction and update [1]. The Kalman filter algorithm is summarized as follows:

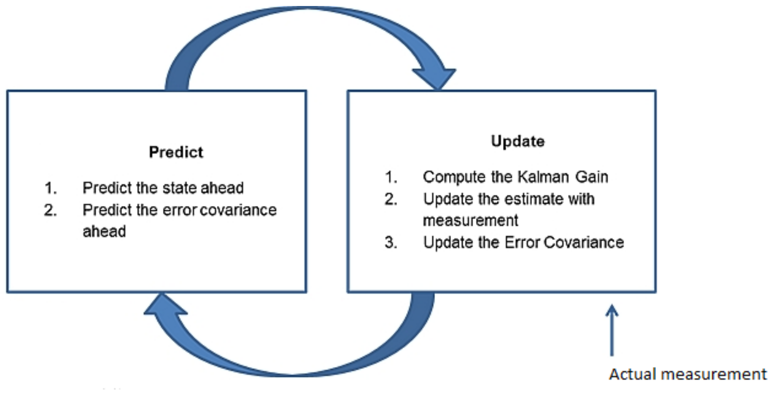

Predict
1. Predict the state ahead
2. Predict the error covariance ahead
Update
1. Compute the Kalman Gain
2. Update the estimate with measurement
3. Update the Error Covariance
Actual measurement

Figure 5.1: Steps in KF[33]

[1]The equations presented in the algorithm are from the "Kalman Filtering:Theory and Practice using MATLAB" book [31]

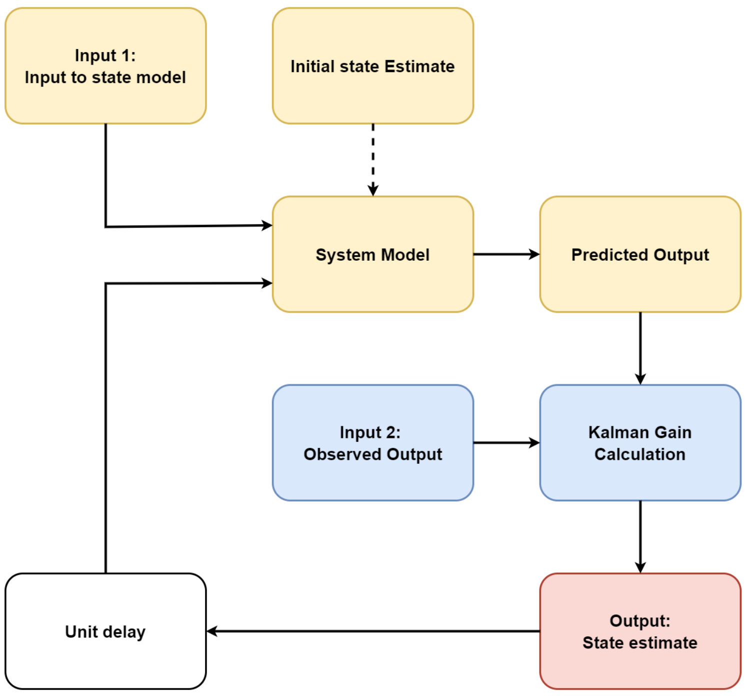


Figure 5.2: Flowchart of KF

Predict Step

From the input to the system model, previous error co-variance and state predictions, the new state and error co-variance estimates are determined from the system model (5.1) as in equation (5.3). At the first iteration, kalman filter is initialized with initial estimate of state and error co-variance.

$$\begin{aligned}\hat{x}_{k|k-1} &= F\hat{x}_{k-1|k-1} + Bu_k \\ P_{k|k-1} &= FP_{k-1|k-1}F^T + Q\end{aligned} \tag{5.3}$$

Kalman Gain

This is computed from the new error covariance estimate from the predict step as in equation (5.4). The Kalman gain can be intuitively understood to be a metric, which decides the degree of weight assigned to the observed output and the estimated output.

$$K = \frac{\left[FP_{k|k-1}F^T + Q\right]H^T}{\left[H\left[FP_{k|k-1}F^T + Q\right]H^T + R\right]} \tag{5.4}$$

Update Step

The difference between the output computed from the new state and co-variance predictions and observed output is utilised to compute a state and co-variance estimate as in equation (5.5). The new predictions are fed as input to the predict step in the next iteration and the loop repeats.

$$\begin{aligned} \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K(y_k - \hat{y}_{k|k-1}) \\ P_{k|k} &= (I - KH)P_{k|k-1}(I - KH)^T + KRK^T \end{aligned} \tag{5.5}$$

5.3 Unscented Kalman Filter

The Kalman Filter is the optimal algorithm for its state estimation if

- System model is linear
- Process and Measurement noises are additive and Gaussian in nature.

When the system is non-linear, the Kalman Filter cannot be applied, and in practical cases, most system dynamics are non-linear. Therefore, non-linear filtration algorithms such as an Extended Kalman Filter (EKF), and an Unscented Kalman Filter (UKF) have been developed.

In EKF, non-linear functions are not approximated globally and done locally with the linear equations derived from Taylor expansion. In highly dynamic systems these approximations can introduce significant errors, because only the first part of the Taylor series expansion is utilised by the linearization. During such instances, it may be advantageous to make do of the UKF algorithm, as a substitute to linearizing non-linear equations. THe UKF algorithm performs non-linear transformations on a set of deterministically selected sigma points and this transformation is called an unscented transformation (UT) and it is more beneficial since the mean and the covariance matrix of the transformed points are precise in the Taylor series expansion upto the second order(5.3) after the non-linear transformations. In algorithms that utilise linearization of the non-linear equations, this accuracy is obtained only upto the first part of the Taylor expansion. A supplementary benefit of the UKF algorithms is the implementation can be easily done, because the Jacobian matrix elements need not be obtained. The sigma point transformation is performed instead of these matrix calculations. points[34][35][36].

The designed system is depicted by a non-linear observation model which shows the relationship between the object's position and measured distances and that is why one of possible non-linear filters should be applied. Hence in our situation the Unscented Kalman Filter has been opted.

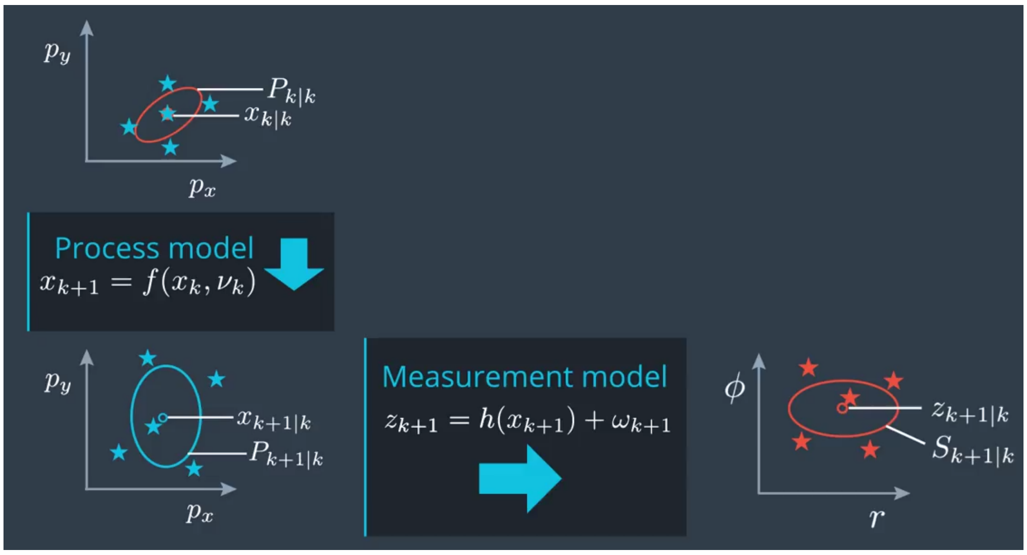


Figure 5.3: UKF prediction process[37]

Scaled Unscented Transform

Before implementation of UKF, sigma points(χ_i) are generated based on certain sampling strategies and the weights (W_i) are also chosen based on these strategies. Depending on our application, different strategies are considered. In this project, scaled Unscented Transform (UT) is chosen for selection of weights and samples.

Three scaling parameters (α, β, κ) are selected based on application. From the mean and covariance of the input state distribution, samples are chosen according to equation 5.6 and weights are determined from equation 5.7.[2]

$$\begin{aligned} \chi_0 &= \hat{x} \\ \chi_i &= \hat{x} + \sqrt{n + \lambda}\, c_i \\ \chi_{n+i} &= \hat{x} - \sqrt{n + \lambda}\, c_i \end{aligned} \tag{5.6}$$

$$\begin{aligned} W_0^{[\hat{y}]} &= \lambda/(n + \lambda) \\ W_0^{[P_{yy}]} &= \lambda/(n + \lambda) + 1 - \alpha^2 + \beta \\ W_i &= 1/[2(n+\lambda)] \\ W_{n+i} &= 1/[2(n+\lambda)] \end{aligned} \tag{5.7}$$

where
$\hat{x}$ = mean of the input states (x)
n = number of states
c_i = i^{th} column of a cholesky factor C_{xx} of covariance matrix
$\lambda = \alpha^2\kappa + (1 - \alpha^2)n$
W_0 = weight associated with first sample

[2]Sampling and Weighting strategies described here are considered from the "Kalman Filtering:Theory and Practice using MATLAB" [31]

$W_0^{[P_{yy}]}$ = weight associated with first value of output covariance matrix
W_i = weight associated with i^{th} sample

$$\begin{aligned} \hat{x} &= \sum_i W_i \, \chi_i \\ P_{xx} &= \sum_i W_i(\chi_i - \hat{x})(\chi_i - \hat{x})^T \end{aligned} \tag{5.8}$$

The mean and covariance from the generated sigma points are computed (equation (5.8)), so as to utilise them in the calculation of the sigma points in the next iteration.

The generated sigma points are propagated through a non-linear function f as in equation (5.9). Further mean and covariance of the output are calculated as in equation (5.10).

$$\xi_i = f(\chi_i) \tag{5.9}$$

$$\begin{aligned} \hat{y} &\approx \sum_i W_i \, \xi_i \\ P_{yy} &\approx \sum_i W_i(\xi_i - \hat{y})(\xi_i - \hat{y})^T \end{aligned} \tag{5.10}$$

Unscented Kalman Filter Algorithm

Equations, describing how scaled Unscented transform is utilised for tracking of state, have been presented [3].

[3]Equations provided here are taken from the book:"Kalman Filtering:Theory and Practice using MATLAB"

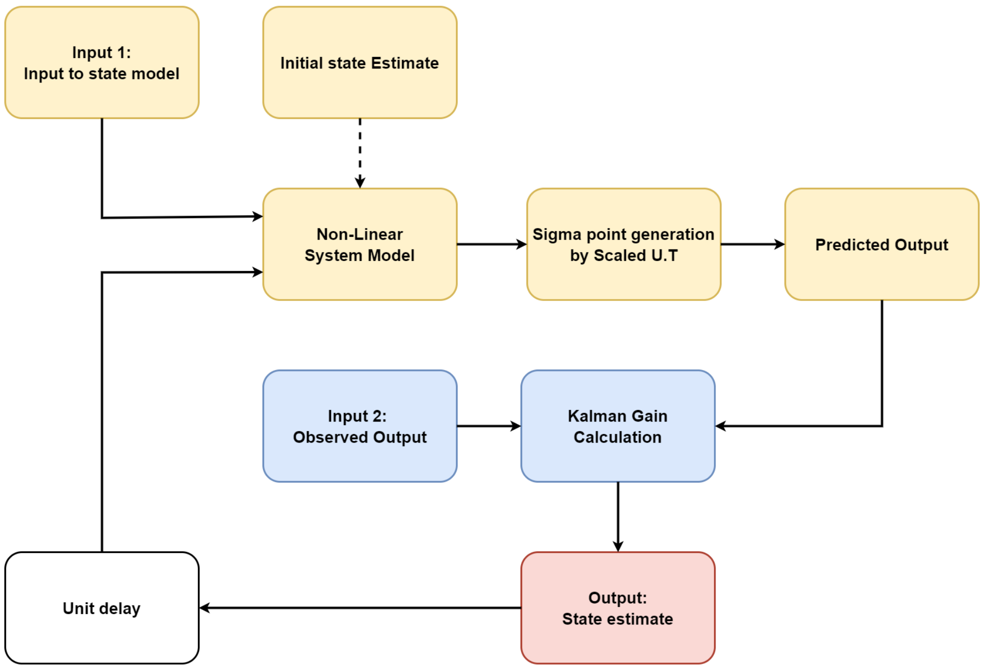


Figure 5.4: UKF Algorithm

Predict Step

Scaled UT is deployed for non linear state transition function (F, B) as in equation (5.11).

$$U[F, \hat{x}_{k-1|k-1}, P_{k-1|k-1}] = [\hat{x}_{k|k-1}, P_{k|k-1}] \tag{5.11}$$

Kalman Gain

Kalman gain is computed from comparison of the predicted output with the measured output as in equation (5.12).

$$K = \frac{[\sum_i W_i(\chi_i - \hat{x}_{k|k-1})(\xi_i - \hat{y}_{k|k-1})^T]}{[\sum_i W_i(\xi_i - \hat{y}_{k|k-1})(\xi_i - \hat{y}_{k|k-1})^T + R]} \tag{5.12}$$

Update Step

From the kalman gain (K), state and covariance predictions in current iteration are computed as in equation (5.13)

$$\begin{aligned} \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K(y_k - \hat{y}_{k|k-1}) \\ P_{k|k} &= P_{k|k-1} - K \left[\sum_i W_i (\xi_i - \hat{y}_{k|k-1})(\chi_i - \hat{x}_{k|k-1})^T \right] \end{aligned} \tag{5.13}$$

Onboard Components 6

This chapter briefly discusses the structure of UWB module and the necessary components embedded on it. Ultra Wide Band transceiver (known as DW1000) considered in this project is manufactured by Decawave company. This transceiver is interfaced with STM micro-controller for power management, clock control and for integration with wide range of micro-controller units (MCU's).

6.1 Antennas

Antennas embedded on this module are Dielectric Chip antennas, which are usually used for microwave frequencies. These antennas are preferred as they have lower energy losses at microwave and millimeter wave frequencies.

When this module is connected to power source such as battery, the display on the module shows the information regarding the configuration (tag/anchor), name, distances of anchors on the tag (when the tag is also switched on). A module can be configured as an anchor or tag based on certain configuration (figure 6.1)of Dual In-line Package (DIP) switches.

6.1.1 Anchor/Tag Configuration

The state of switch 4 aids us to identify a module as an anchor or tag. The unit of anchor is provided by binary configuration of switches $(5, 6, 7)$. For example, if the switches configuration are On(4), Off(5), Off(6), On(7), then the unit is anchor and it's number is 1.

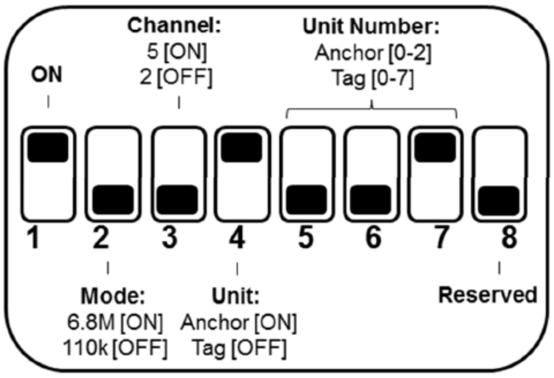


Figure 6.1: DIP switch configuration for both tag and anchor [38]

When this module is plugged into deepcar, the distances of anchors, from the deepcar, are obtained through a serial port. Through TDOA based localization algorithm (section 3.1.3) in deepcar, which takes distances as input, position of deepcar is determined.

6.2 TREK1000

- TREK1000 is an evaluation kit that is used to evaluate external module called DW1000 Ultra-Wideband transceiver IC (IEEE 802.15.4-2011 standard).
- Accuracy of TREK in the X and Y plane that is less than 20 cm and supports a data rates of 110 kbps, 850 kbps and 6 8Mbps.
- Furthermore, DW1000 has a span of RF ranging from 3.5 GHz to 6.5 GHz .

The power consumption of TREK1000 is defined to be low (lower than Zigbee, Wi-Fi and BLE). Usually TREK 1000 is deployed to evaluate performance of DW1000 IC in the following use cases:

1. Tracking: Determination of tag location using stationary beacons.
2. Geo-fencing: Entrance or exit of a tag in zones around anchor.
3. Navigation: Position of tag determined by stationary anchors at every step in navigation [38].

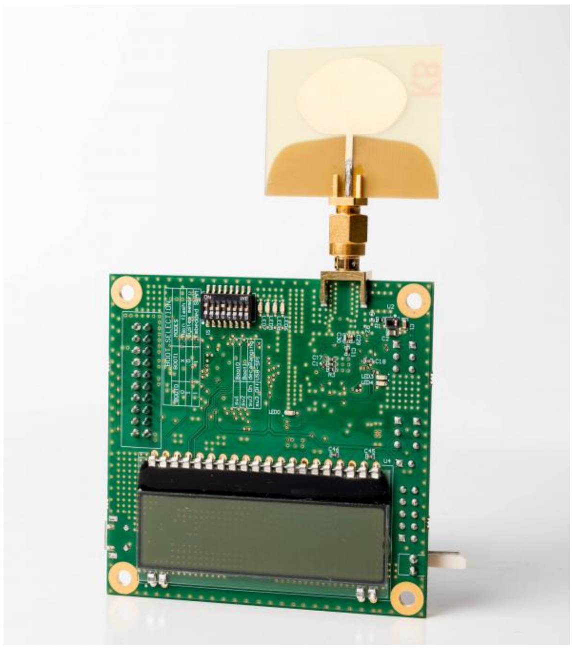


Figure 6.2: TREK1000 module [39]

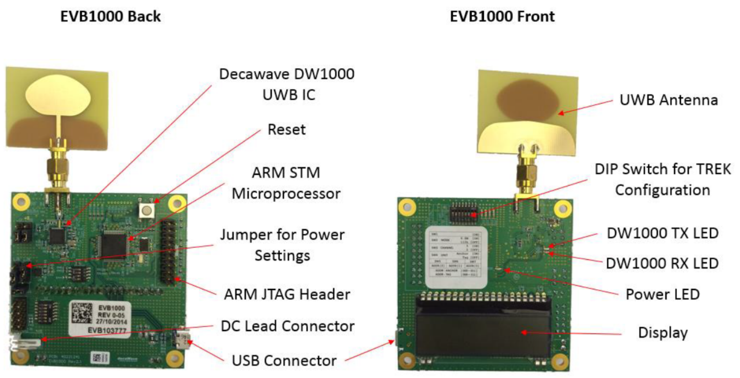


Figure 6.3: Components of TREK[39]

6.3 DWM 1000 Chip

DWM1000 chip is a micro-controller unit board carrying DW1000 Ultra-Wideband transceiver IC (IEEE 802.15.4-2011 standard), that is mounted on deepcar. This chip is relatively identical to TREK 1000 board. However, firmware flashed on this chip is solely convenient to display and send the distances, from beacons to tag. Antenna mounted is dielectric chip antenna, different from the antenna in TREK 1000. From the display on this chip, the id of the anchor or tag is observed, and if it is tag, then the distances, of different anchors, are also observed in a cyclic order [40].

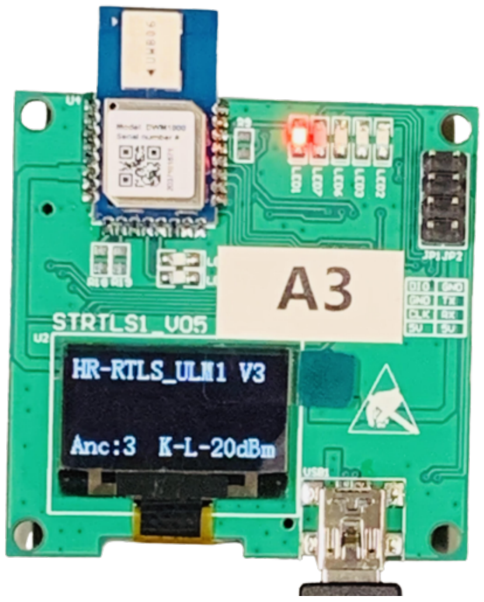


Figure 6.4: DWM1000

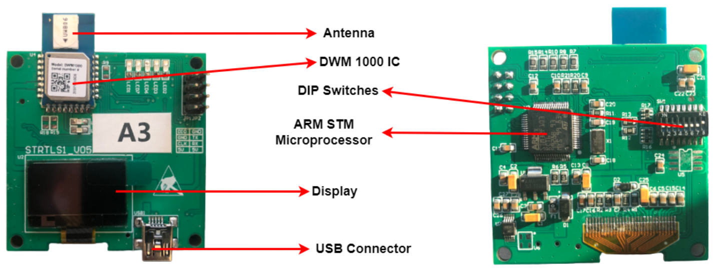


Figure 6.5: Components of DWM1000

NLoS Identification 7

This chapter is dedicated to training and testing various machine learning classifiers after which it is deployed for the identification of LoS and NLoS conditions when the deepcar is navigated through the environment.

7.1 CIR Response

The CIR response obtained from TREK1000 is depicted in figure 7.1. Impulse response consists of 1000 samples, among which 342 samples are considered based on the first detected path.

Metrics in CIR

- Amplitude of the peak path
- Index of the peak ray in CIR
- Position of the first detected path in CIR and the amplitudes of the first, second and third peaks around it.
- Standard noise deviation

Saturation in CIR response is given by a metric "mc", which is the ratio of the maximum amplitude among the first, second, and third peak amplitudes, to the amplitude of the peak path [41]. If the CIR is saturated, a LoS path between an anchor and a tag is ensured.

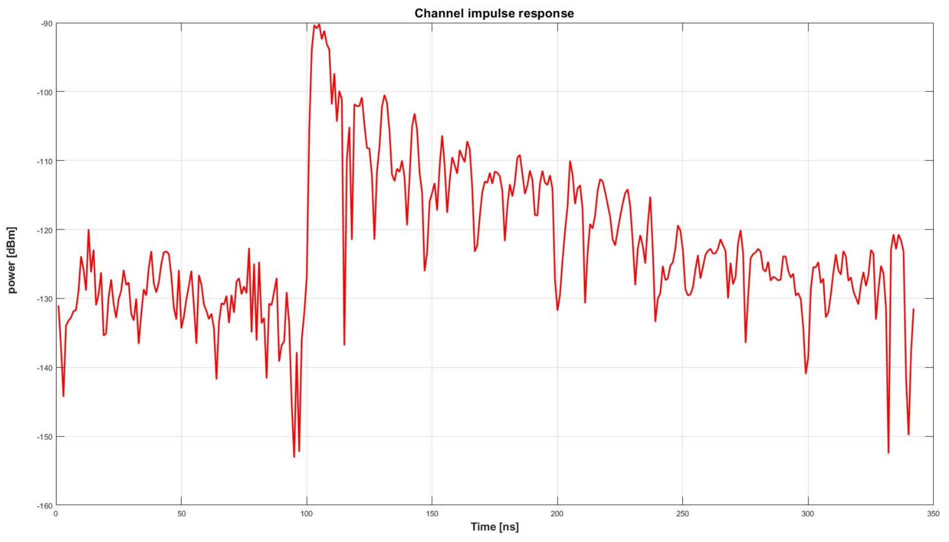


Figure 7.1: CIR Response in typical LOS scenario

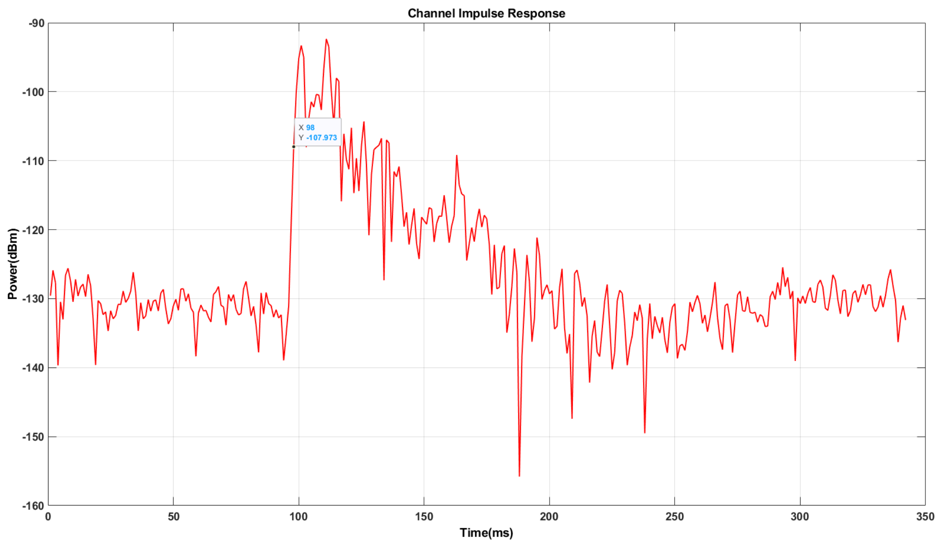


Figure 7.2: CIR Response in typical NLOS scenario

7.1.1 Acquisition of CIR

CIR are collected for various LOS and NLOS conditions for training a machine learning model. TREK evaluation kit is used to obtain CIR's (TREK setup is described in appendix A).

Training Data collection

To develop a machine learning classifier that classifies whether a tag is in line of sight or non line of sight with anchor, it is necessary to obtain training data in an environment that suits the environment where deepcar is operated.

Figure 7.3 shows the 16 probable positions of tag in a room while an anchor is positioned at the corner of the room at a height of 60cm above the tag. These positions are selected on the basis of navigation of deepcar. For each position, 10 impulse responses are recorded, aggregating to a total of 160 responses. Impulse responses have been recorded when the anchor and the tag are separated by $10m, 15m, 20m, 30m$ in a corridor.

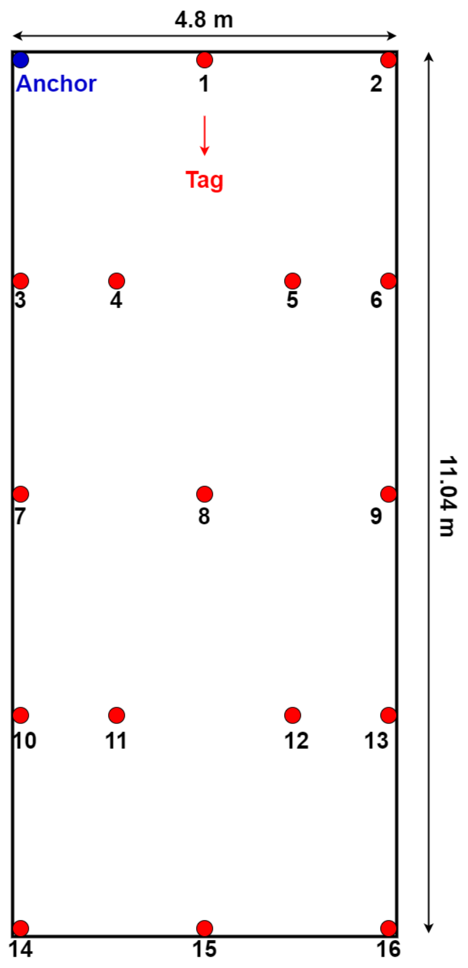


Figure 7.3: Setting of the room in which CIR's are recorded for possible Line of Sight(LoS) scenarios.

Figure 7.4 shows the 2 probable positions of the anchor and 4 probable positions of the corridor. Anchor can also be positioned at the other end of the corridor, but it is not considered because of the two way communication between anchor and tag. For each possible position, 10 impulse responses are recorded, aggregating to a total of 80 responses.

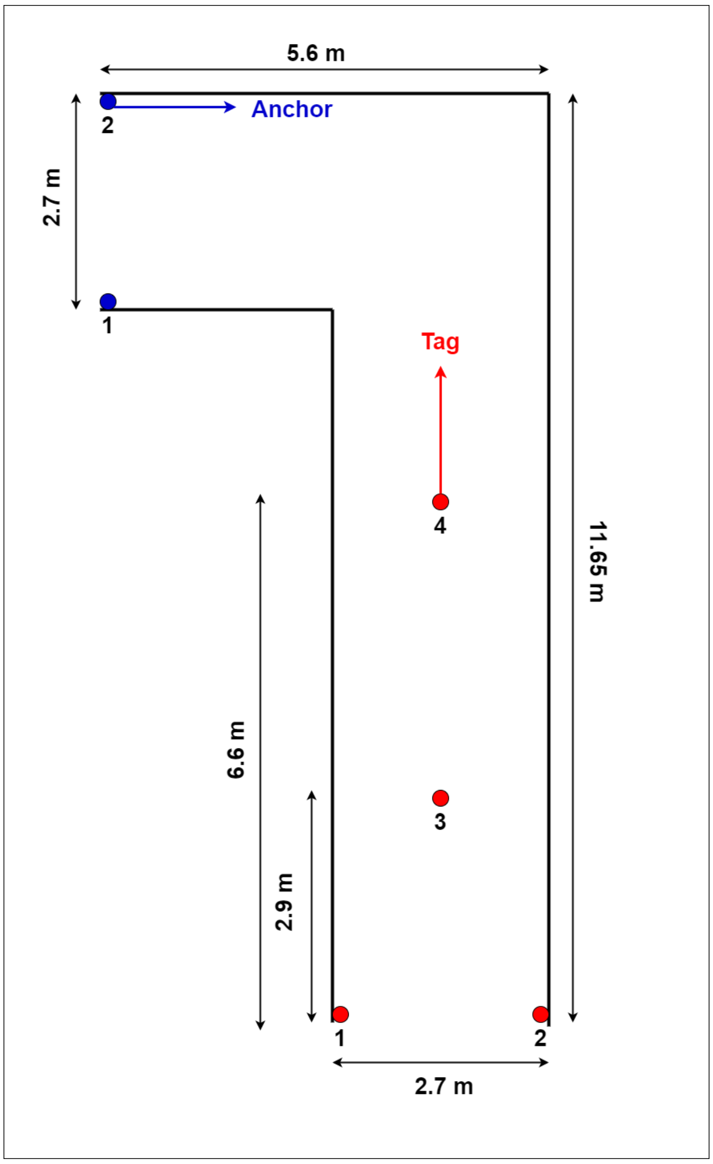


Figure 7.4: Setting of the Corridor in which CIR's are recorded for possible Non Line of Sight(NLoS) scenarios.

7.2 Experimental Setup for Classifier

The software tool used for this section is MATLAB. Each classifier with its particular parameters will be explained and various evaluation metrics are calculated for comparative analysis.

The experiment is conducted in the 'Motion Tracking' Lab at Aalborg University with a volume space of 3.4m x 3m. The anchors are represented by A_0, A_1, A_2 and are placed in the volume as depicted in the figure 7.5 and the tag is represented by a solid green circle on the deepcar (Deepcar is setup as described in appendix A). The deepcar (denoted as *DC*) is navigated from start point 'A' to end point 'B' . A metallic plate with dimensions 0.8m x 0.75m x 5mm is kept parallel to the path taken by the deepcar.

Due to this metal plate, the path of the deepcar is divided into 3 regions, these 3

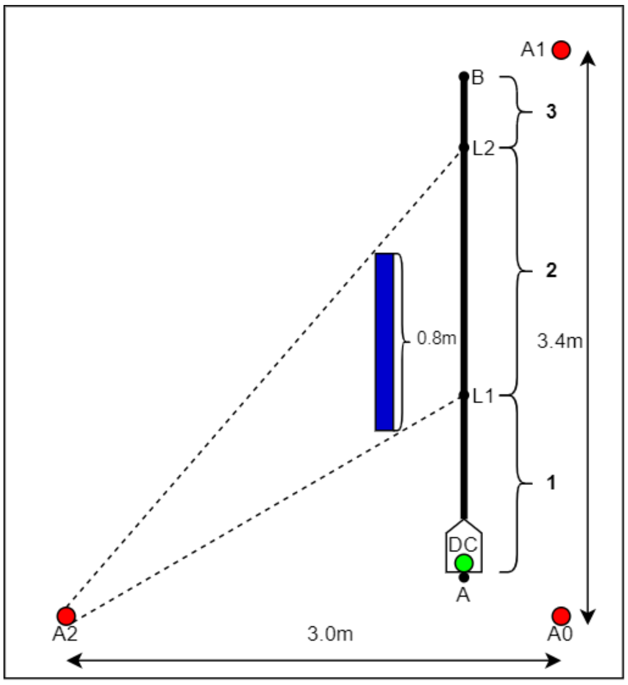


Figure 7.5: Top view of setup in Motion Tracking Lab

regions are constructed based on the LoS and NLoS condition created by the metal plate with respect to tag of *DC* and anchor A_2. These regions are:

1. Region 1: This represents a region of LoS from point A to point L1.
2. Region 2: This represents a region of NLoS from point L1 to point L2.
3. Region 3: This represents a region of LoS from point L2 to point B.

The goal of the Machine Learning classifier is to identify the above mentioned 3 regions when the deepcar is navigated from point A to point B in real-time. The classifier is implemented by using CIR's obtained by the TREK1000's anchor placed at A_2 and tag placed at *DC*.

7.3 Data-set Creation

The training data has been obtained by recording CIR's using a pair of TREK1000, where one functions as an anchor and the other functions as a tag. The anchor is tethered to a PC through the USB-serial port and the CIR's are recorded using MATLAB. The CIR obtained is a vector of size 1x342, where the values represent the magnitude of dBm between the anchor and tag, at that instance and a total of 342 such instances are captured and clipped within a timeframe of 1000 nanoseconds for each sample run. The number of sample runs can be both increased and decreased in MATLAB based on the requirements of the experiment conducted. Hence the final CIR vector output from the MATLAB can be depicted as Nx342 where 'N' represents the number of sample runs selected.

This is repeated for various experiments conducted which implies the dataset of CIR's is created for different conditions which involves varying between LoS and NLoS states as described below:

- Different settings, which involves varying the distance between the tag and the anchor.
- Different environmental conditions, which involves recording in rooms or spaces with various building materials and objects in the space.

All the experiments conducted and its parameters are explained in section 7.1.1.

The resulting CIR vector output from this approach is a data-matrix of size 1409x342 where 1409 implies the number of samples collected at all the different settings. However after initial analysis through a 'probe and study' method it was concluded that the entire 342 instances of the CIR were not mandatory for the identification of NLoS and LoS conditions.

The reduction of size of the data-matrix across the column(X-axis) would result in requiring a lower computational power which would thereby result in lesser training time. In addition to this it decreases the complexity of the trained model and can also prevent overfitting of the training data. Therefore it can be concluded that if reducing the size of the data-matrix still yields in a highly accurate identification of LoS and NLoS then that would be the most optimal and efficient way to train the model.

7.4 Training data

The data-matrix is clipped across the column after observing the 'first path index' for that specific sample run from the TREK1000. After obtaining the index, 'n' data points are opted after this index for a specific sample run so the data-matrix can be represented as 1409 x n. The value of 'n' is opted to be 101 since through experimentation it was found that 'n' greater than 140 did not yield in better accuracy. Hence for every sample run 101 data points were selected after their respective first path indices, which resulted in a 1409x101 training data-matrix.

In addition to clipping 101 data points after first path index, 7 supplemental characteristic features of the CIR are selected. These features are unique to the CIR of a specific sample run and hence will assist the model training in a constructive way. These features are namely: first peak, second peak, third peak, rxpreamble count, maximum growth of CIR, first path index and distance between anchor and tag which is calculated by the TREK1000. Therefore the final data-matrix after appending the above said features is a vector of dimensions 1409 x 108.

The classification is performed for 2 classes namely LoS and NLoS, where LoS is represented by class '0' and NLoS is represented by class '1'. The matrix 1409 x 108 can be classified into two classes 0 and 1. Among the 1409 samples, samples 1 to 416 are CIR's pertaining to LoS conditions hence class 0, and samples 417 to 1409 are CIR's pertaining to NLoS conditions hence class 1. A class label column is created containing 416 0's and 992 1's representing LoS and NLoS is appended to the data-matrix to obtain class labels corresponding to CIR's during training. Hence the training data-set is created with appropriate class labels, 108 features and 1409 data samples. The dimensions of the labelled training set is 1409 x 109.

7.5 Test data

The test data set is created in real-time in MATLAB from the anchor tethered to the PC through the USB serial data port. For every iteration, a CIR vector of size 1 x 342 is obtained and this vector has to match the feature size of the trained model which is 108. Hence similar to the training set this vector is also clipped for 101 data features from the first path index and appended with the same 7 supplemental features to obtain the 1 x 108 test data. This test data is fed to the trained model and it predicts the class label 0 or 1. This procedure is repeated for every iteration.

7.6 Classifier Training and Validation

The classifier is trained with the 1409 x 109 data-set with the corresponding class label 0 and 1. The classifier is trained and is validated using 5-fold cross validation. K-fold cross validation where 'k' represents the number of folds is performed to prevent over-fitting while training the model.

Prior to deployment in the environment the performance metrics such as Confusion matrix, ROC curve are calculated. Additionally for Neural networks the performance plot, training state plot and error histogram are obtained.

1. Confusion Matrix : This is a matrix table to evaluate the classification accuracy of the classifier. The columns of the matrix depict the predicted class of the data and the rows of the matrix depict the true class of the data. Therefore the diagonal elements represent the data vectors that has been predicted accurately while the non diagonal elements represent the data vectors that has been predicted inaccurately. Hence the classification accuracy is calculated as the sum of the elements in the diagonal divided by sum of all elements. Also other terminologies such as True Positive Rate (TPR) or sensitivity, False Negative Rate (FNR) or miss-rate, Positive Predictive Value (PPV) or precision and False Discovery Rate (FDR) or fall-out are calculated to assess the performance the classifier[42].
2. ROC Curve : Receiver Operating Characteristic curve is a graph depicting performance of a classifier model at variable thresholds. The curve is plotted between 2 parameters namely : True Positive Rate (TPR) and False Positive Rate (FPR) where they are obtained from True Positive(TP), True Negative(TN), False Positive(FP) and False Negative(FN). TPR and FPR are calculated as:

$$TPR = \frac{TP}{TP + FN} \tag{7.1}$$

$$FPR = \frac{FP}{FP + TN} \tag{7.2}$$

'Area Under the ROC Curve' also called as AUC is an algorithm to calculate the threshold points in an ROC curve. AUC determines the two-dimensional area under the ROC and gives a compiled measure of performance with the various classification thresholds. AUC ranges in value from 0 to 1 and hence it can be said that if AUC is

1 if the model predictions are 100% accurate and if AUC is 0 if the model predictions are 100% inaccurate[42].

3. Performance plot : The performance plot is charted between Cross-Entropies and Epochs for the training, validation and test data-set. Cross-entropy is commonly used as a loss function when optimizing classification models and it is favourable to minimize cross-entropy so that the classifier yields a higher accuracy. One Epoch is if the complete training data-set is cycled in forward and backward through the neural network exactly once. The plot is drawn to select the best epoch based on the value of cross entropy differences between the training, validation and testing. This is necessary to choose the neural network model that is neither underfitted nor overfitted[43].
4. Error histogram : It is a histogram of error calculated between target values and predicted values after training a neural network. Bins in the histogram, represent the number of vertical bars on the graph. The number of bins is 20 by default in MATLAB. The Y-axis denotes the number of samples from the data-set which lie in a particular bin and it is plotted for training, validation and test. The zero error line in the histogram represents the zero error value on the X-axis which is the error axis[44].
5. Parallel coordinate plot : This plot enables in visualising higher dimensional data and aids in comparing the features of the observations individually. The plot consists of vertical columns that are spaced equally and the number of parallel vertical columns equals the number of dimensions. A point from the higher dimensional data is represented as a line passing through all the vertical columns where the intersection points represent the value of the observation in that specific column. In this report the plot is drawn for 5 features which are columns 101,102,103,104 and 106 these are first path index, first peak, second peak, third peak and distance between tag and anchor(in mm)[45].

The classifiers implemented in the experiments are K-Nearest Neighbours, Support Vector Machine and Neural Network. All the above mentioned metrics are obtained individually for each classifier.

7.6.1 Support Vector Machine

A SVM model with kernel function as Linear and Box constraint level as 1 is opted. The training data-set matrix containing 1409 observations, 108 predictors and 2 response classes without dimensionality reduction is given to the SVM. It takes 0.58655 seconds to train and has a prediction speed of approximately 64,000 observations per second. The training data is validated by performing 5-fold cross-validation to avoid overfitting. The SVM results in a classification accuracy of 100%.

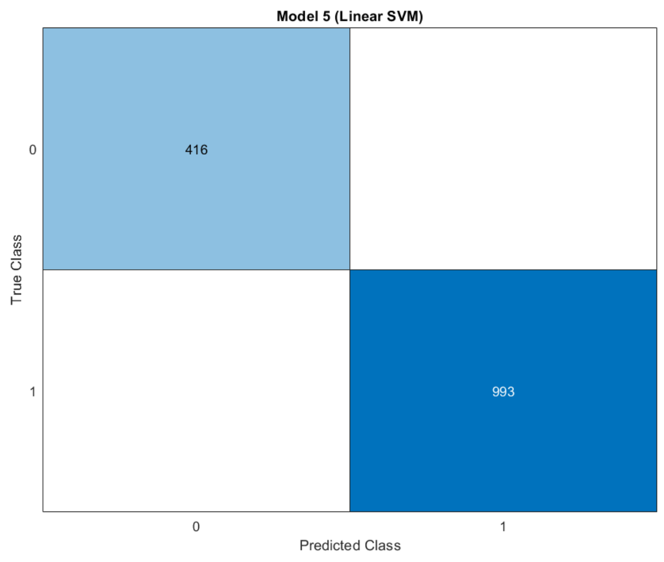


Figure 7.6: Confusion matrix of the trained SVM model where the diagonal elements represent the true classes, and in this model the SVM classified all the data points in their respective classes. The 416 data points in the light blue box represent LoS condition and the 993 data points in the dark blue box represent NLoS condition

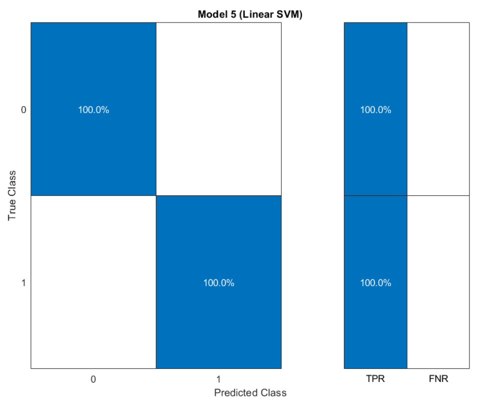


Figure 7.7: SVM - TPR & FNR

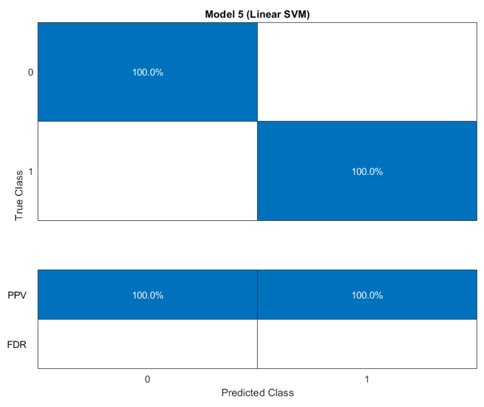


Figure 7.8: SVM - PPV & FDR

In figure 7.7 confusion matrix of the SVM depicting individual class classification accuracy on the left and on the right each class's TPR and FNR are calculated. For this model the LoS and NLoS class accuracies are 100% hence the TPR for LoS and NLoS is also 100% while the FNR is absent or 0%. In figure 7.8 confusion matrix of the SVM depicting individual class classification accuracy on the top and on the bottom each class's PPV and FDR are calculated. For this model the LoS and NLoS class accuracies are 100% hence the PPV for LoS and NLoS is also 100% while the FDR is absent or 0%

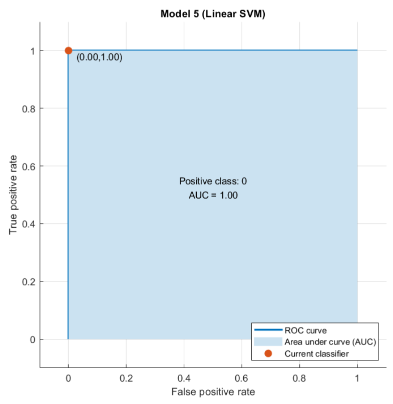


Figure 7.9: SVM - ROC for class 0

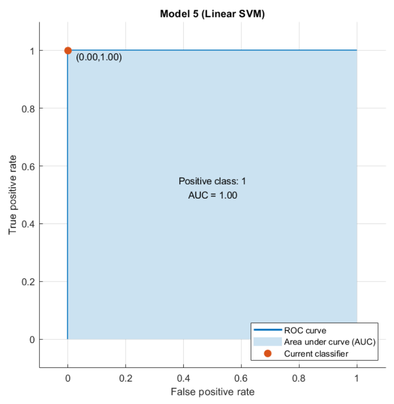


Figure 7.10: SVM - ROC for class 1

The figure 7.9 represents the ROC curve for the LoS classification and the AUC under the curve is 1 which implies a 100% accurate classification and the figure 7.10 represents the ROC curve for the NLoS classification and the AUC under the curve is 1 which implies a 100% accurate classification.

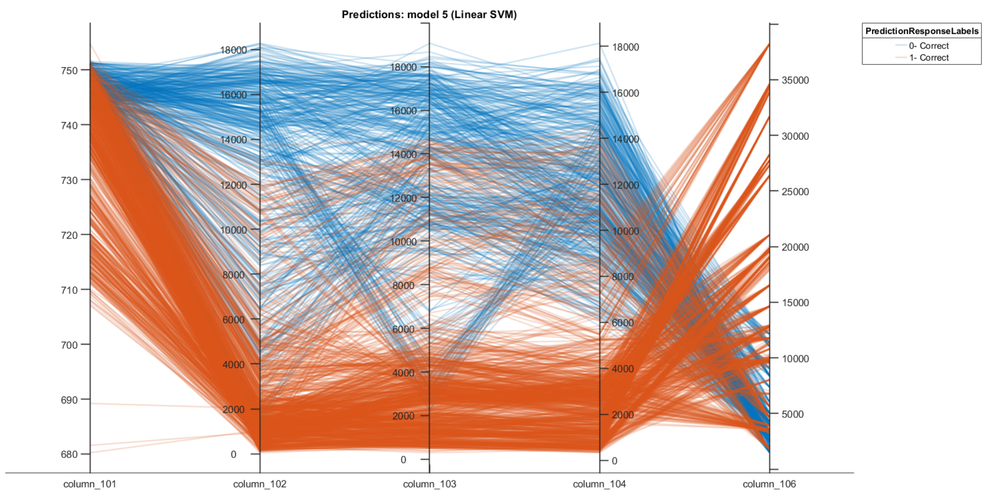


Figure 7.11: is a parallel coordinate plot for the SVM for the feature set 101,102,103,104 and 106 to visualise the data in LoS and NLoS conditions. The dark orange lines depict the data during NLoS and blue lines depict the data during LoS. If the lines are in a continuous form the classification of that data vector is accurate if the lines are in a dashed format the classification of that data vector is inaccurate.

7.6.2 K-Nearest Neighbour

A K-NN model with distance metric as Euclidean, number of neighbours as 1 and the distance weight as Equal is opted. The training data-set matrix containing 1409 observations, 108 predictors and 2 response classes without dimensionality reduction is given to the K-NN. It takes 2.6856 seconds as training time and has a prediction speed of approximately 13,000 observations per second. The training data is validated by performing 5-fold cross-validation to avoid overfitting. The K-NN results in a classification accuracy of 100%.

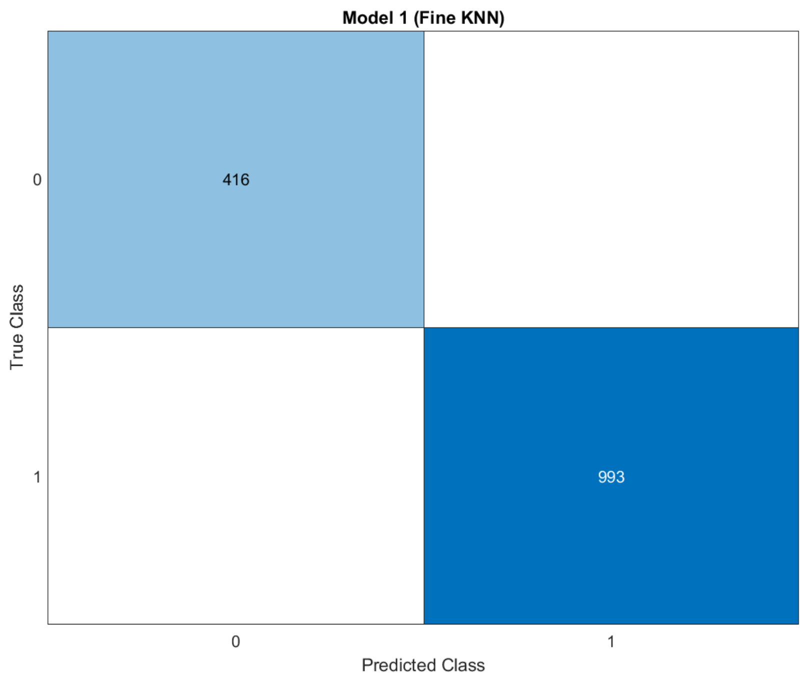


Figure 7.12: Confusion matrix of the trained K-NN model where the diagonal elements represent the true classes, and in this model the K-NN classified all the data points in their respective classes. The 416 data points in the light blue box represent LoS condition and the 993 data points in the dark blue box represent NLoS condition

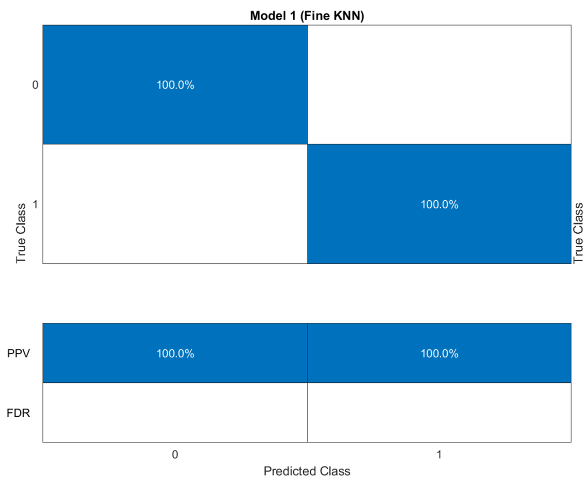


Figure 7.13: K-NN : PPV & FDR

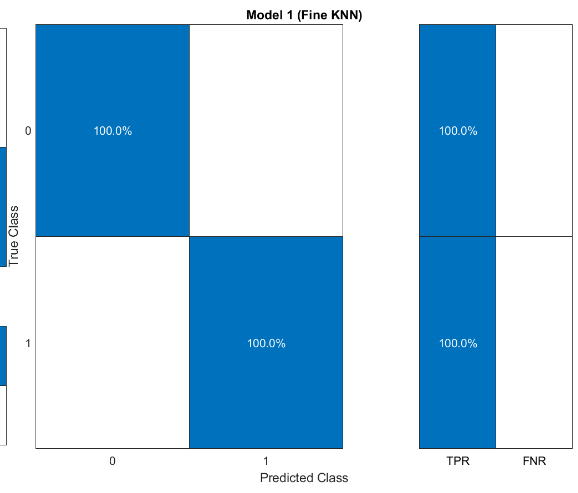


Figure 7.14: K-NN : TPR & FNR

In figure 7.14 confusion matrix of the K-NN depicting individual classification accuracy on the left and on the right each class's TPR and FNR are calculated. For this model the LoS and NLoS class accuracies are 100% hence the TPR for LoS and NLoS is also 100% while the FNR is absent or 0%. In figure 7.13 confusion matrix of the K-NN depicting individual classification accuracy on the top and on the bottom each class's PPV and

FDR are calculated. For this model the LoS and NLoS class accuracies are 100% hence the PPV for LoS and NLoS is also 100% while the FDR is absent or 0%

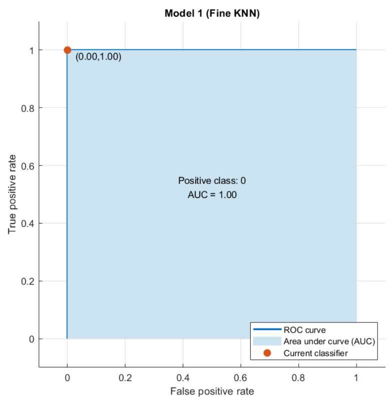


Figure 7.15: K-NN : ROC for class 0

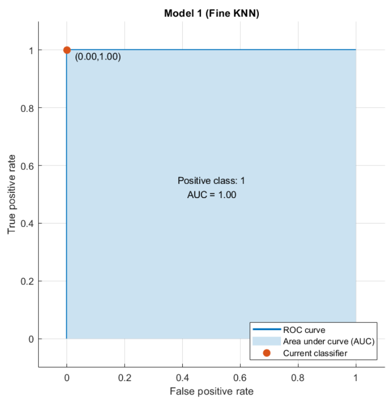


Figure 7.16: K-NN : ROC for class 1

The figure 7.15 represents the ROC curve for the LoS classification and the AUC under the curve is 1 which implies a 100% accurate class classification and the figure 7.16 represents the ROC curve for the NLoS classification and the AUC under the curve is 1 which implies a 100% accurate class classification.

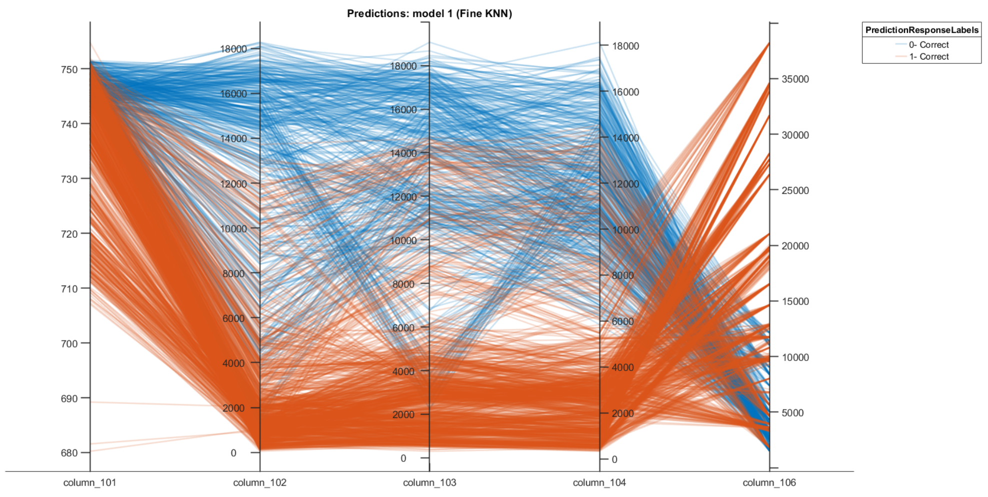


Figure 7.17: is a parallel coordinate plot for the K-NN for the feature set 101,102,103,104 and 106 to visualise the data in LoS and NLoS conditions. The dark orange lines depict the data during NLoS and blue lines depict the data during LoS. If the lines are in a continuous form the classification of that data vector is accurate if the lines are in a dashed format the classification of that data vector is inaccurate.

7.6.3 Neural Network

A two layer feed-forward neural network where one layer is a hidden layer and the other is a output layer is used. The hidden layer contains a hyperbolic tangent sigmoid transfer function and the output layer contains a soft max transfer function. The soft max function is selected to standardize the output so that it can be understood as a probabilistic distribution.

The training algorithm opted for the neural network is 'scaled conjugate gradient backpropagation' and the model is chosen to have 25 neurons in the hidden layer. The number of inputs in the neural network correspond to the number of predictors in the dataset, which is 108. The output of the neural network has 2 possible class labels which are 0 and 1 corresponding to LoS and NLoS conditions respectively.

The training data-matrix is split randomly into 3 kinds of samples namely: training, validation and testing in the following percentages: 70%, 15%, 15%. The training sample is presented to the network during training period and, the weights and biases are adjusted according to its error. The validation sample is used to calculate network generalization and to stop training when generalization does not continue to improve, this is performed when there is an increase in cross-entropy error of validation sample. The testing sample has zero effect on training the neural network hence it provides an independent assessment of the the network's performance during and after training.

It takes less than 1 second to train and validate the network's performance. The number of iterations run to reach minimum gradient is 36. The neural network yields in a final accuracy of 99.8% for 25 neurons in the hidden layer, also the performance of the network was assessed for various number of neurons in the hidden layer and it was concluded that increasing the number of neurons more than 40 yielded in results with higher irregularity and variability in the evaluation metrics. In between 25 to 40 neurons the accuracy did not improve, hence 25 neurons is opted since it would require lesser computational power and hence lesser training times.

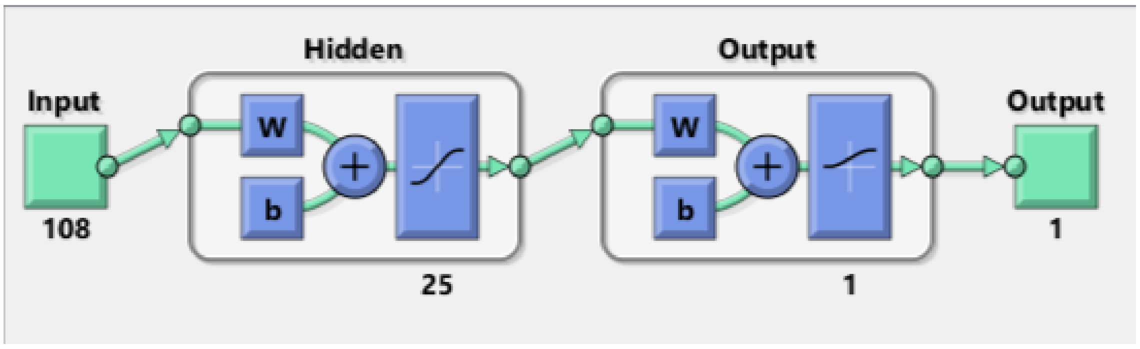
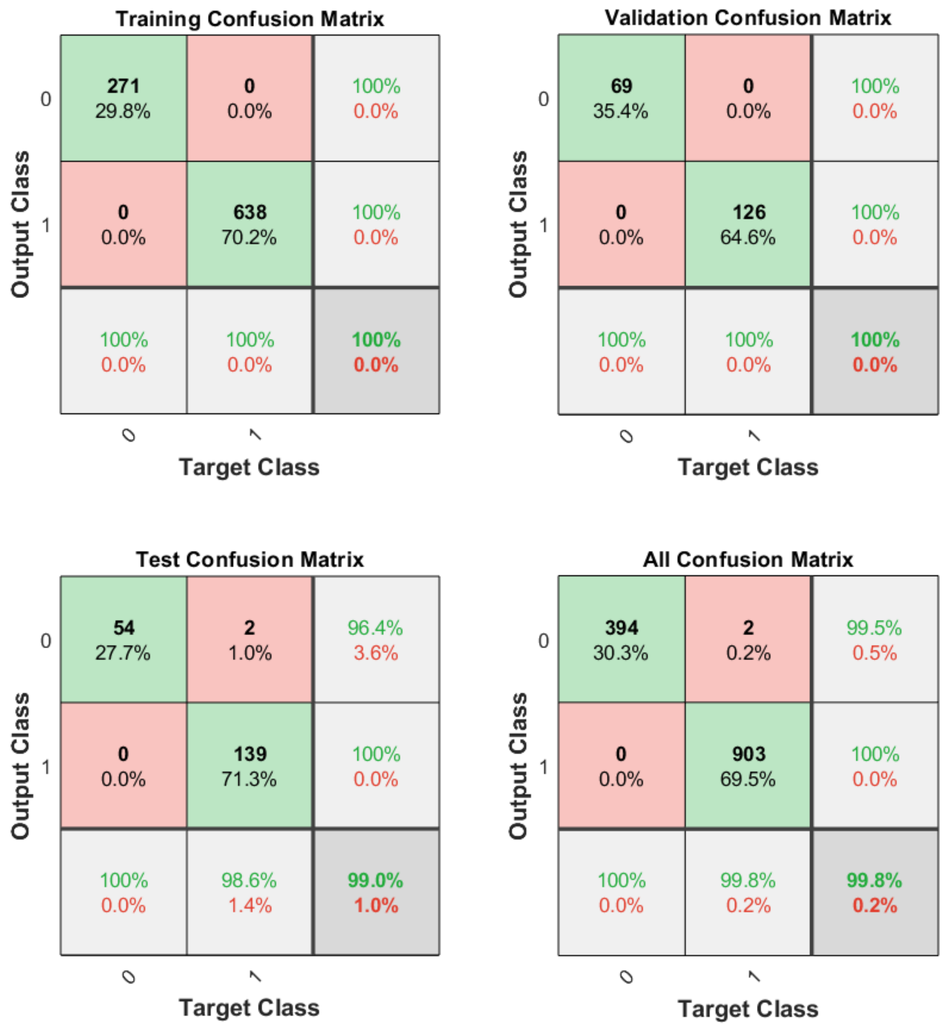


Figure 7.18: Structure of the neural network implemented



Training Confusion Matrix

Output Class \ Target Class	0	1	
0	**271** 29.8%	**0** 0.0%	100% 0.0%
1	**0** 0.0%	**638** 70.2%	100% 0.0%
	100% 0.0%	100% 0.0%	**100%** **0.0%**

Validation Confusion Matrix

Output Class \ Target Class	0	1	
0	**69** 35.4%	**0** 0.0%	100% 0.0%
1	**0** 0.0%	**126** 64.6%	100% 0.0%
	100% 0.0%	100% 0.0%	**100%** **0.0%**

Test Confusion Matrix

Output Class \ Target Class	0	1	
0	**54** 27.7%	**2** 1.0%	96.4% 3.6%
1	**0** 0.0%	**139** 71.3%	100% 0.0%
	100% 0.0%	98.6% 1.4%	**99.0%** **1.0%**

All Confusion Matrix

Output Class \ Target Class	0	1	
0	**394** 30.3%	**2** 0.2%	99.5% 0.5%
1	**0** 0.0%	**903** 69.5%	100% 0.0%
	100% 0.0%	99.8% 0.2%	**99.8%** **0.2%**

Figure 7.19: represents the confusion matrices of the neural network during the training, validation, test and the overall confusion matrix. The training and validation confusion matrices have a classification accuracy of 100%, the test confusion matrix has an overall classification accuracy of 99% since 2 data vectors belonging to LoS is inaccurately misclassified as NLoS. Therefore the overall classification accuracy of the neural network is calculated to be 99.8%.

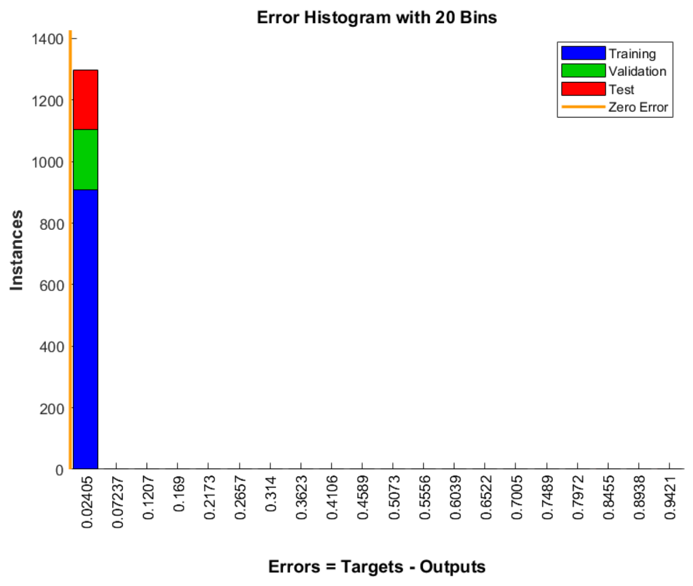


Figure 7.20: is an error histogram with 20 bins. The Y-axis contains the observations and the X-axis contains the errors. The zero error line lies depicted by the orange line is on the left side of the bins.

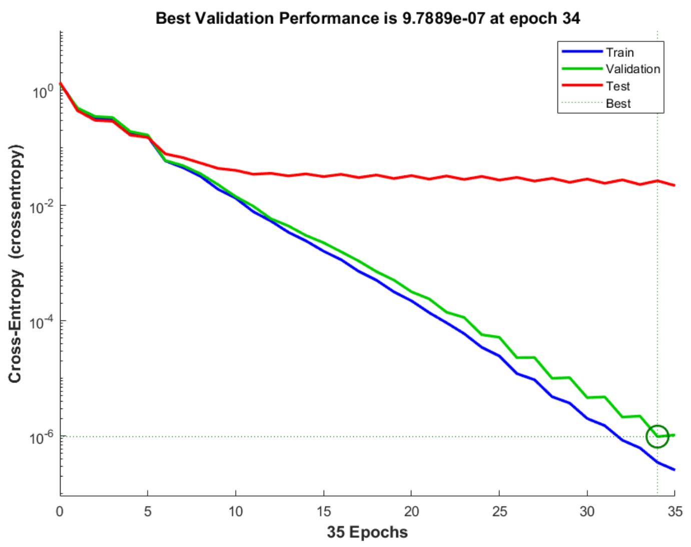


Figure 7.21: is the performance plot of the neural network, the neural network epoch with the best validation performance is opted as the final trained model.

7.7 Dimensionality reduction

The training data-set has 108 features/predictors, by performing dimensionality reduction the number of features needed for training the model is reduced. A commonly used dimensionality reduction technique is principal component analysis. The training data is normalized before applying PCA so that the magnitude of the variables do not affect it. The PCA is performed and is keeping enough components to cover 95% variance in the data. The variance per component in decreasing order are: 55.7%, 7.7%, 4.7%, 3.5%, 3.1%, 2.5%, 2.2% and so on. In this manner the first 24 principal components are kept to achieve the set 95% variance.

7.7.1 PCA and Support Vector Machine

The SVM has the same model type as explained in the previous subsection for Support Vector Machine. PCA is performed on the training data-set matrix containing 1409 observations, 108 predictors and 2 response classes. After reduction and selection of the first 24 principal components the dataset is given to the SVM for training and 5-fold cross-validation. The combined time for PCA and training SVM is 0.81137 seconds and has a prediction speed of 19,000 observations per second. The SVM results in a classification accuracy of 99.2%.

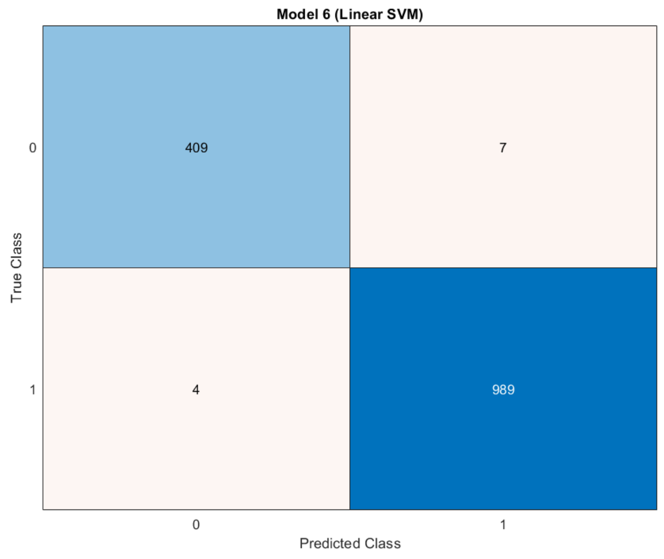


Figure 7.22: Confusion matrix of the trained PCA-SVM model where the diagonal elements represent the true classes, and in this model the PCA-SVM classification had some inaccuracies. The light blue box represents LoS condition and the dark blue box represent NLoS condition. The total LoS data vectors are 416 and total NLoS data vectors are 993. Among the 416 LoS vectors, 409 have been classified correctly as LoS and the remaining 7 have been classified as NLoS. Among the 993 NLoS vectors, 989 have been classified correctly as NLoS and the remaining 4 have been incorrectly classified as LoS

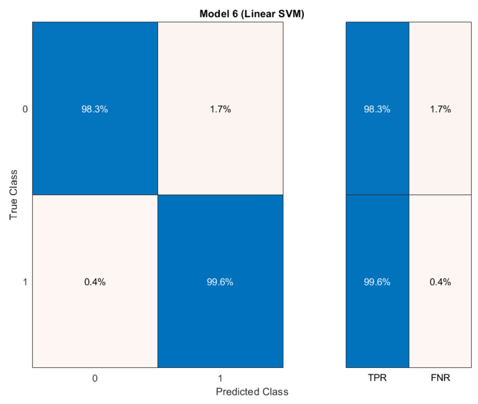


Figure 7.23: PCA-SVM : TPR & FNR

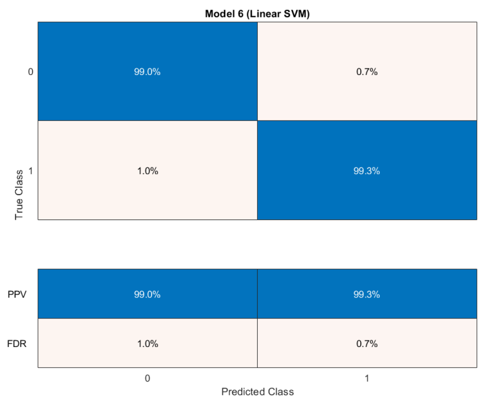


Figure 7.24: PCA-SVM : PPV & FDR

In figure 7.23 confusion matrix of the SVM depicting individual classification accuracy on the left and on the right each class's TPR and FNR are calculated. TPR is a ratio between true positives and total positives, hence for class 0 classification we obtain a TPR of 98.3% and for class 1 classification we obtain a TPR of 99.6%. FNR can be obtained as a value subtracted from 1 and TPR, hence for class 0 classification we obtain FNR of 1.7% and for class 1 classification we obtain FNR of 0.4%. In figure 7.24 confusion matrix of the SVM depicting individual classification accuracy on the top and on the bottom each class's PPV and FDR are calculated. PPV is a ratio between true positive and sum of true and false positives, hence for class 0 classification we obtain a PPV of 99% and for class 1 classification we obtain a PPV of 99.3%. FDR can be obtained as a value subtracted from 1 and PPV, hence for class 0 misclassification we obtain FDR of 1% and for class 1 misclassification we obtain FDR of 0.7%.

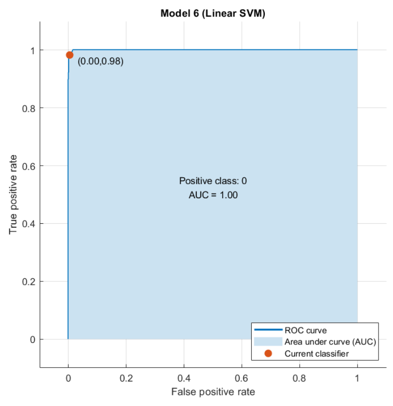


Figure 7.25: PCA-SVM : ROC class 0

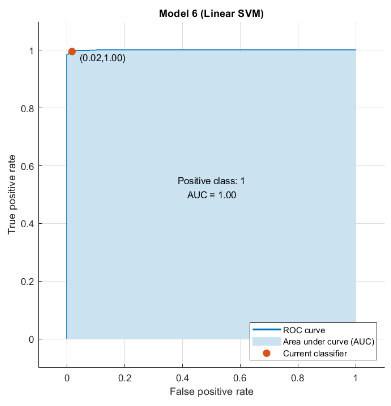


Figure 7.26: PCA-SVM : ROC class 1

The figure 7.25 represents the ROC curve for the LoS classification and the figure 7.26 represents the ROC curve for the NLoS classification and the AUC under the curve is 1. The current classifier needs to be at an optimal point such that the classifier has a high true positive rate and a low false positive rate.

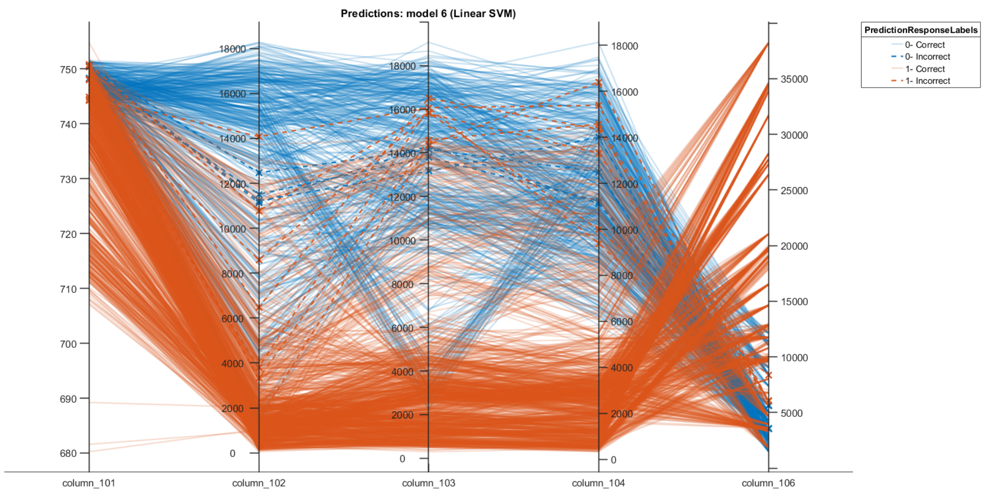


Figure 7.27: parallel coordinate plot for the PCA-SVM for the feature set 101, 102, 103, 104 and 106 to visualise the data in LoS and NLoS conditions. The dark orange lines depict the data during NLoS and blue lines depict the data during LoS. If the lines are in a continuous form the classification of that data vector is accurate if the lines are in a dashed format the classification of that data vector is inaccurate. In this PCA-SVM model there are some misclassifications of LoS and NLoS which can be seen from the dashed lines in the plot.

7.7.2 PCA and K-Nearest Neighbour

The K-NN has the same model type as explained in the previous subsection for K-Nearest Neighbor. PCA is performed on the training data-set matrix containing 1409 observations, 108 predictors and 2 response classes. After reduction and selection of the first 24 principal components the dataset is given to the K-NN for training and validation. The combined time for PCA and training K-NN is 1.3566 seconds and has a prediction speed of 11,000 observations per second. The PCA-KNN results in a classification accuracy of 100%.

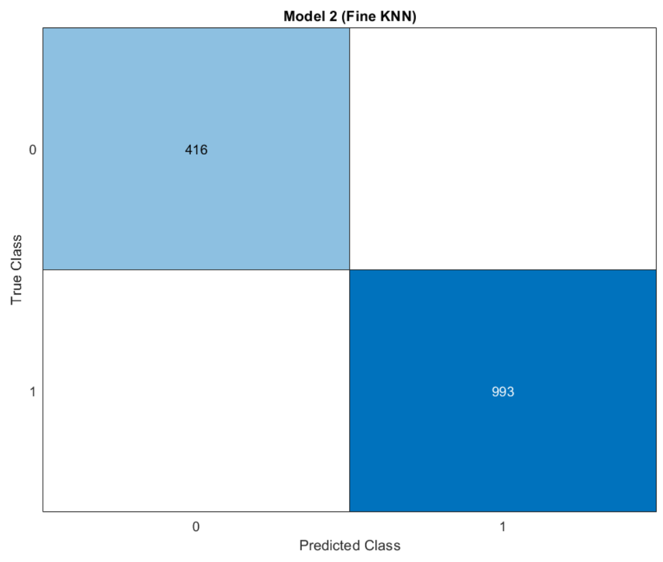


Figure 7.28: Confusion matrix of the trained PCA-KNN model where the diagonal elements represent the true classes, and in this model the PCA-KNN classified all the data points in their respective classes. The 416 data points in the light blue box represent LoS condition and the 993 data points in the dark blue box represent NLoS condition

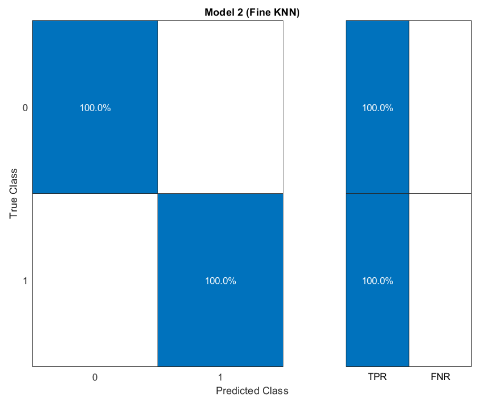


Figure 7.29: PCA-KNN : TPR & FNR

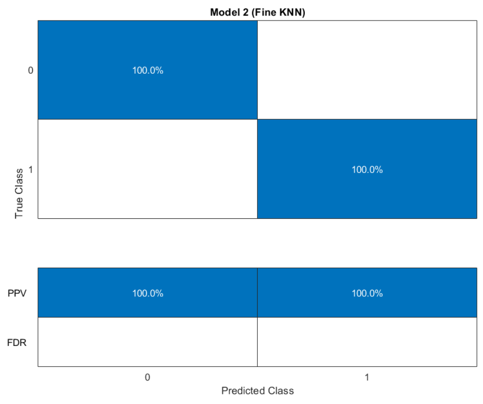


Figure 7.30: PCA-KNN : PPV & FDR

The figure 7.14 Confusion matrix of the PCA-KNN depicting individual class classification accuracy on the left and on the right each class's TPR and FNR are calculated. For this model the LoS and NLoS class accuracies are 100% hence the TPR for LoS and NLoS is also 100% while the FNR is absent or 0%. The figure 7.13 Confusion matrix of the PCA-KNN depicting individual class classification accuracy on the top and on the bottom each class's PPV and FDR are calculated. For this model the LoS and NLoS class accuracies are 100% hence the PPV for LoS and NLoS is also 100% while the FDR is absent or 0%

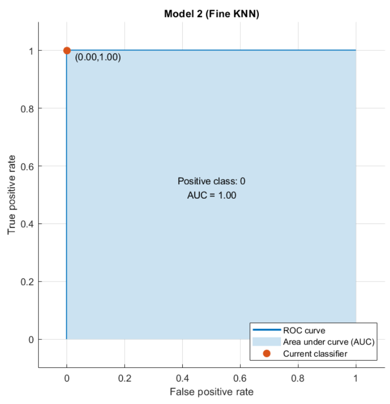


Figure 7.31: PCA-KNN : ROC class 0

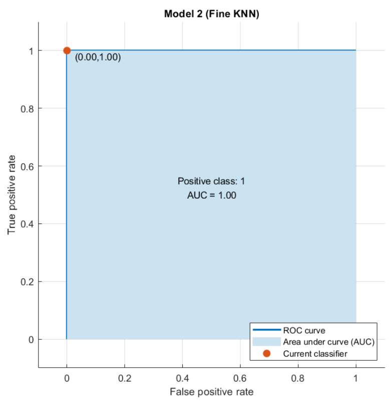


Figure 7.32: PCA-KNN : ROC class 1

The figure 7.15 represents the ROC curve for the LoS classification and the AUC under the curve is 1 which implies a 100% accurate class classification and the figure 7.16 represents the ROC curve for the NLoS classification and the AUC under the curve is 1 which implies a 100% accurate class classification.

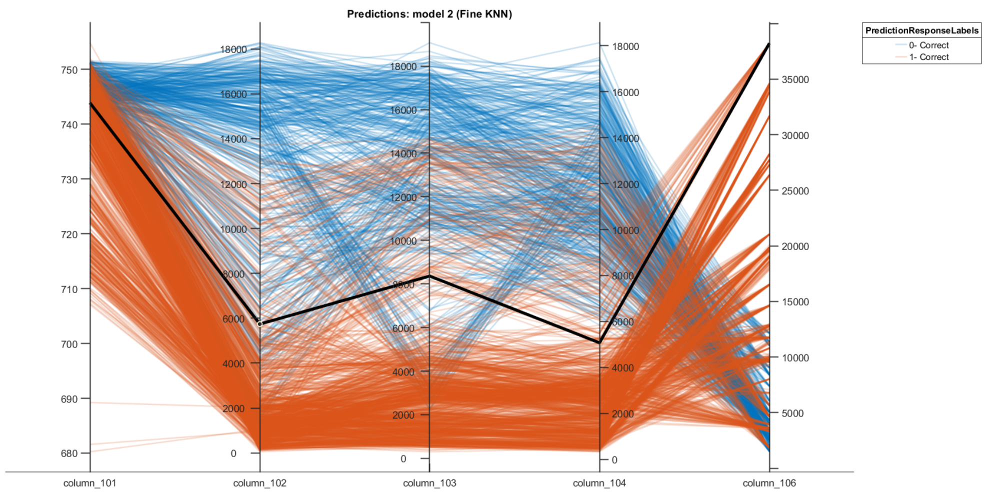


Figure 7.33: Parallel coordinate plot for the PCA-KNN for the feature set 101, 102, 103, 104 and 106 to visualise the data in LoS and NLoS conditions. The dark orange lines depict the data during NLoS and blue lines depict the data during LoS. If the lines are in a continuous form the classification of that data vector is accurate if the lines are in a dashed format the classification of that data vector is inaccurate. The black line is an example of a NLoS data vector and its values across the 5 previously mentioned features

7.8 Classifier Deployment

The 3 classifiers namely: Neural Network, Support Vector Machine and K-Nearest Neighbours have been trained. The experiment is conducted according to the section 7.2 and the test data is collected in real-time during the experiment according to the section 7.5. The experiment is conducted 10 times for each classifier to ensure repeatability of the real-time classification.

After the model is trained and validated, SVM and K-NN are exported as a MATLAB struct variable whereas the neural network is exported as MATLAB function.

A 2 subplot figure is generated in real-time for every iteration/sample run, where the first subplot contains the CIR response and the second subplot displays whether deepcar is in LoS/NLoS.

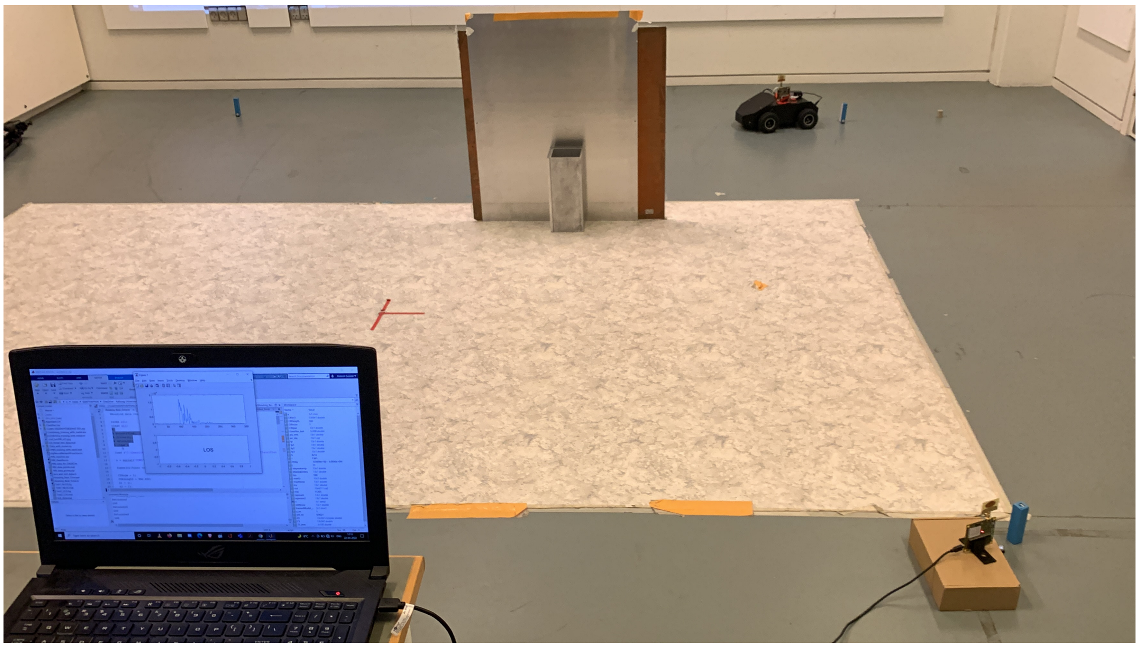


Figure 7.34: Represents Region 1 from the experimental setup. The deepcar is in LoS with the TREK1000 and the trained ML classifier deployed from MATLAB also classifies the deepcar to be in LoS.

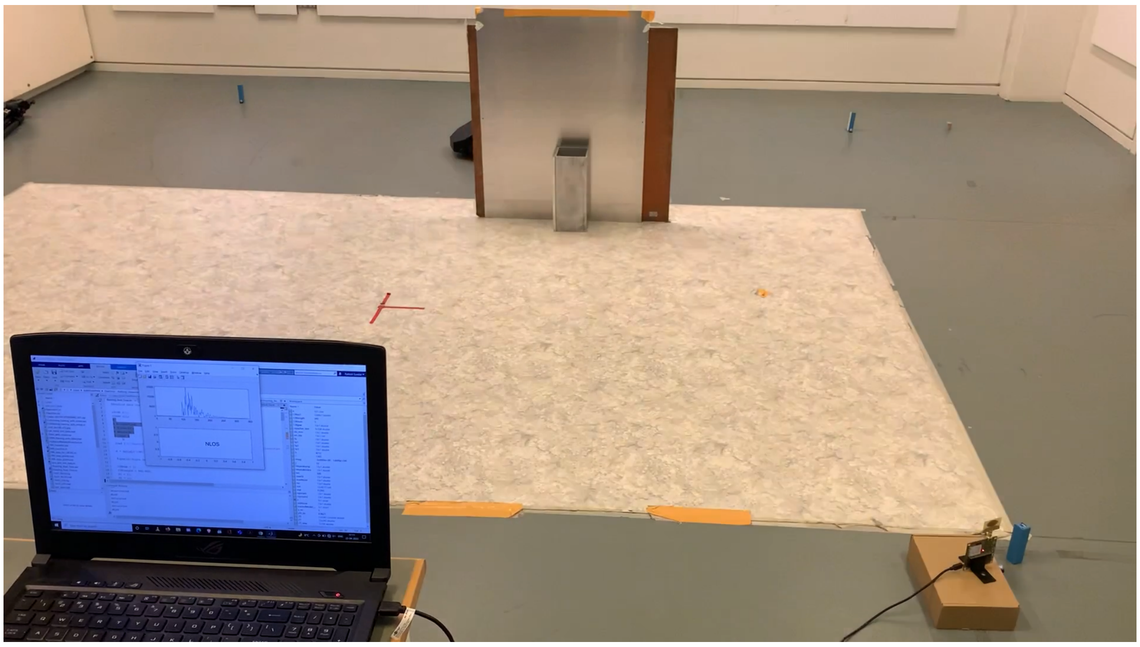


Figure 7.35: Represents Region 2 from the experimental setup. The deepcar is navigated from Region 1 to Region 2 and now the deepcar moves behind the metal plate, hence it is in NLoS to the TREK1000. The trained ML classifier deployed from MATLAB also classifies the deepcar to be in NLoS.

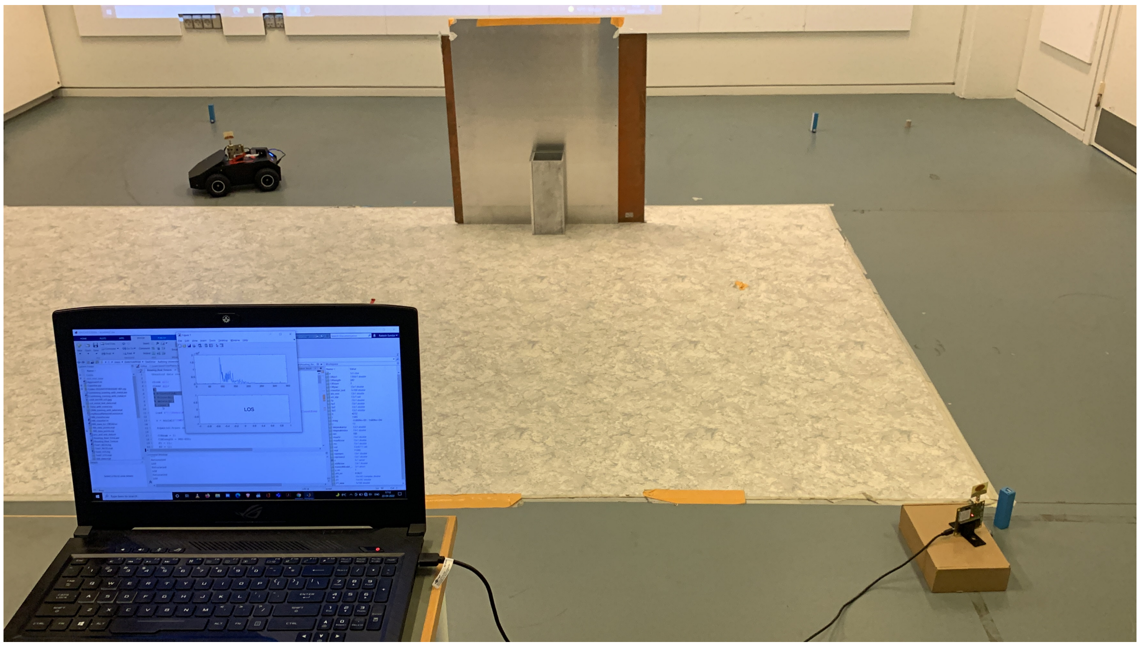


Figure 7.36: Represents Region 3 from the experimental setup, The deepcar is navigated from Region 2 to Region 3 and now the deepcar moves away from the metal plate, hence it transitions back in LoS to the TREK1000. The trained ML classifier deployed from MATLAB also classifies the deepcar to be in LoS

NLoS Mitigation 8

This chapter aims to improve localisation in NLoS conditions using UKF in the simulated environment and the software tool used for this purpose is MATLAB. The deepcar motion is simulated by a state space model. The measurements obtained by UWB anchors are incorporated in the measurement model.

8.1 Modelling of Deepcar

8.1.1 State System Model

Since dynamic model for deepcar cannot be obtained from documentation, it was decided to simulate deepcar's motion by a state space model developed from kinematic equations(8.2). For constraining the motion of deepcar in the room and also to emulate a reasonable trajectory, an appropriate non linear function of velocity has been chosen. State vector(equation (8.1)) includes position in x and y axis, and the orientation of the deepcar. Control input for this state system model is velocity v and steering angle ϕ. Equations 8.2 are represented in state system model(8.3). This state model is only developed so as to display trajectory the deepcar should ideally follow, which is unknown. Process noise is not considered because there is a need to compare the actual trajectory followed by the simulated deepcar with the trajectory predicted by UKF [46].

$$X \;=\; [x\ y\ \theta]^T \tag{8.1}$$

where
x = position of deepcar along x-axis
y = position of deepcar along y-axis
θ = orientation of deepcar measured from x-axis.

$$\begin{aligned} x(n) &= x(n-1) \;+\; v\cos(\theta)\Delta t \\ y(n) &= y(n-1) \;+\; v\sin(\theta)\Delta t \\ \theta(n) &= \theta(n-1) \;+\; \phi\Delta t \end{aligned} \tag{8.2}$$

$$\begin{bmatrix} x(n) \\ y(n) \\ \theta(n) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x(n-1) \\ y(n-1) \\ \theta(n-1) \end{bmatrix} + \begin{bmatrix} \cos(\theta)\Delta t & 0 \\ \sin(\theta)\Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} v \\ \phi \end{bmatrix} \tag{8.3}$$

$$X(n) \;=\; F\ X(n-1) \;+\; B(X(n-1))\ u$$

where
F = State Transition Matrix
B = Input Matrix
u = Control Input
Δt = Time step

8.1.2 Measurement Model

In real system, the distances have been obtained from UWB beacons. This is simulated as determination of euclidean distances between position of deepcar, obtained from the state model, and position of the beacon. Three measurements are obtained from three stationary beacons, whose positions are known in prior. The sensor noise is assumed as a white gaussian noise (R) with certain co-variance.

$$z(i) = d_i(n) = \sqrt{(x_{p_i} - x(n))^2 + (y_{p_i} - y(n))^2} + N(0, R) \tag{8.4}$$

where
x_{p_i} = co-ordinate of i^{th}beacon along x-axis
y_{p_i} = co-ordinate of i^{th}beacon along y-axis
d_i = distance between i^{th}beacon and position
z = measurement variable

8.2 Mathematical Approach for LoS and NLoS Classification

A rectangular obstacle is considered for creating Non-Line of sight conditions. Three Stationary beacons are positioned at corners of the box.

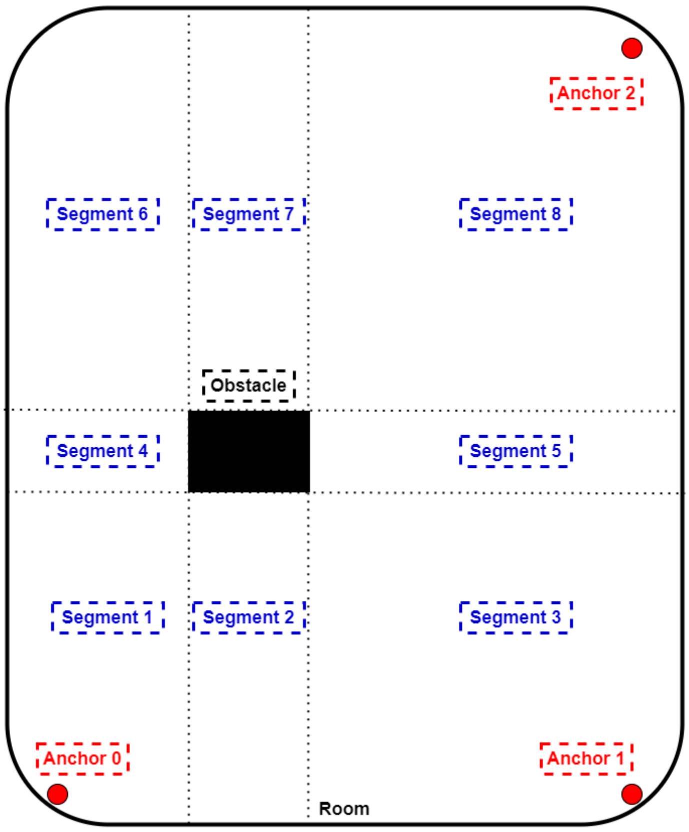


Figure 8.1: LoS and NLoS of a position with beacons in different segments

Different conditions are implemented for checking whether the position is in LoS or NLoS with a beacon at an instant of time. The whole room is segmented into 8 parts, based on the rectangular obstacle as shown in figure 8.1.

- **Condition 1**: If the position of deepcar falls in any one of the segments $(3, 5, 8)$, then deepcar is in LoS with anchor 1 and anchor 2.
- **Condition 2**: If the position of deepcar falls in any one of the segments $(6, 7, 8)$, then deepcar is in LoS with anchor 2.
- **Condition 3**: If the position of deepcar falls in any one of the segments $(1, 2, 3)$, then deepcar is in LoS with anchor 0 and anchor 1.
- **Condition 4**: If the position of deepcar falls in any one of the segments $(1, 4, 6)$, then deepcar is in LoS with anchor 0.

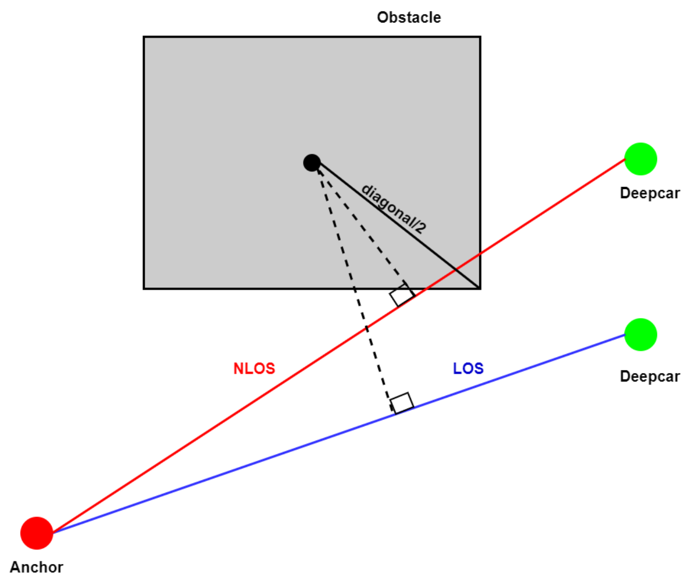


Figure 8.2: LOS and NLOS of a position with beacon

Figure 8.2 depicts the classification of LoS and NLoS of a position based on diagonal length of the obstacle. If the perpendicular distance from the center of the obstacle to the line connecting anchor and deepcar is less than half of the diagonal length, then the position of deepcar is in NLoS with the anchor, or else deepcar is in LoS with the anchor.

The trajectory generated by deploying the state system model and every position is classified as LoS or NLoS based on the above conditions as shown in figure 8.3.

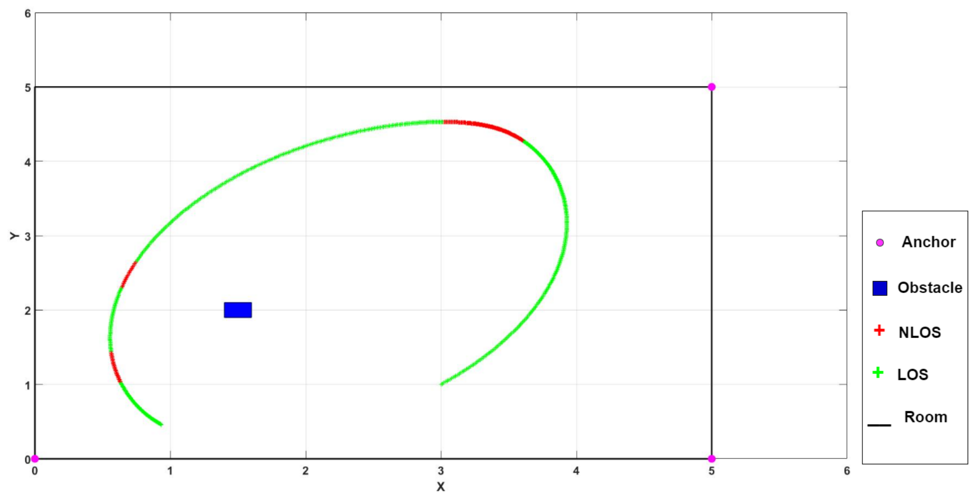


Figure 8.3: LoS and NLoS positions in the Deepcar trajectory

8.3 Conventional Approach

In the conventional approach, the deepcar is localised by trilateration from the measurements obtained from beacons, irrespective of deepcar is in LoS or NLoS with the stationary beacon.

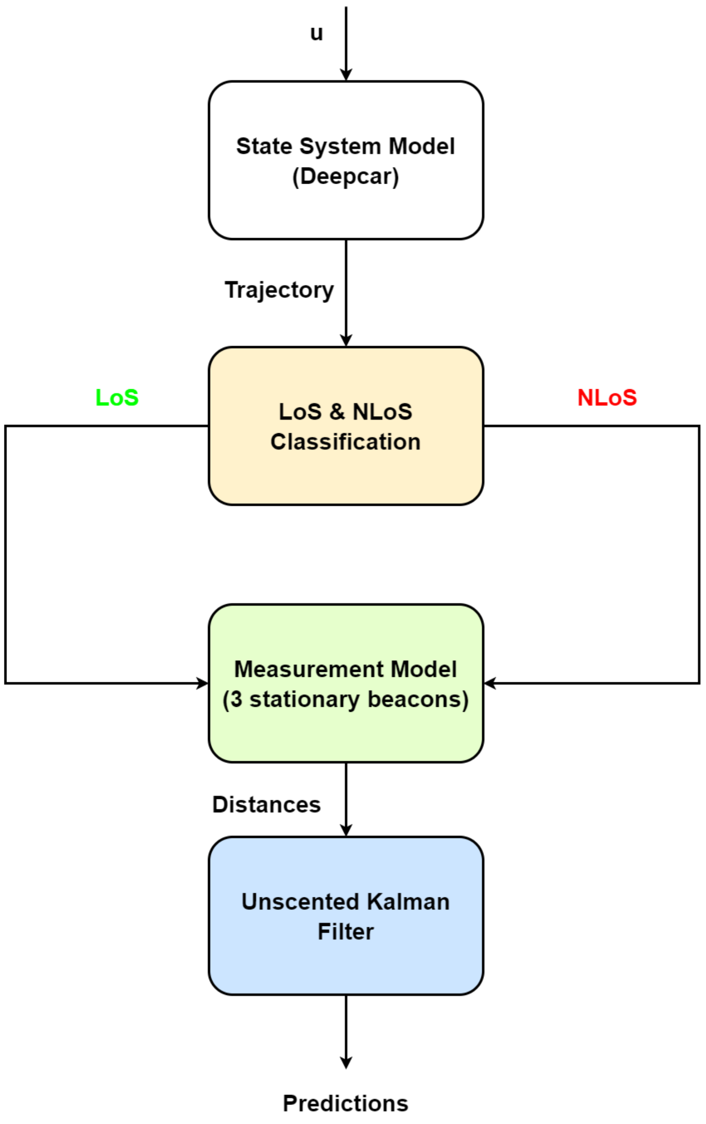

u
State System Model (Deepcar)
Trajectory
LoS
LoS & NLoS Classification
NLoS
Measurement Model (3 stationary beacons)
Distances
Unscented Kalman Filter
Predictions

Figure 8.4: Conventional Approach

Figure 8.4 can be interpreted in real life as: Measurements obtained from UWB stationary beacons are simulated as distances from measurement model. If there is no Non-line of sight condition, then the predictions provided by UKF are sufficiently accurate to ideal trajectory generated by state system model.

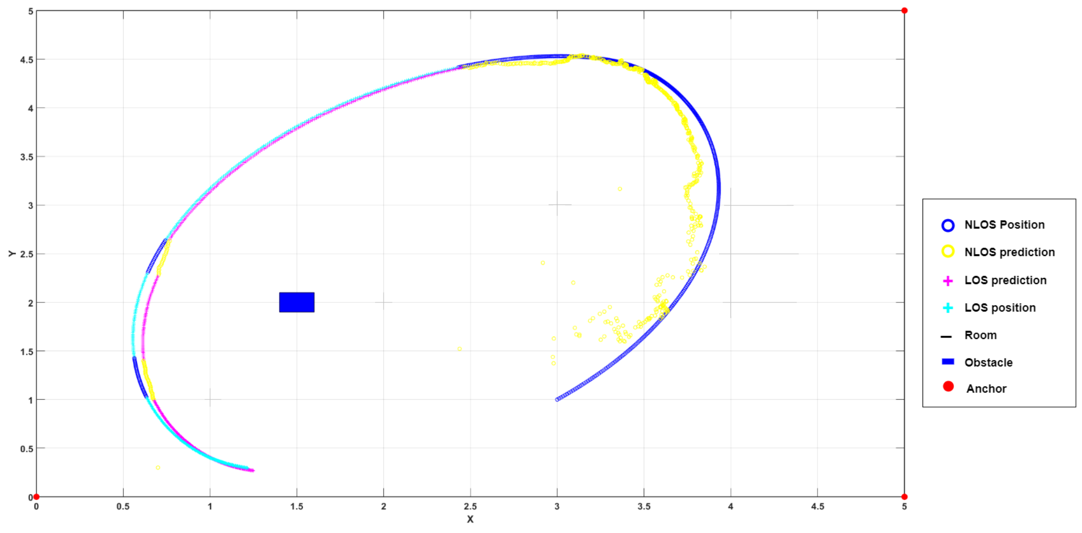


Figure 8.5: Trajectory Prediction by UKF

In figure 8.5, the yellow and pink markers represent the trajectory predicted by UKF, and it is compared to the ideal trajectory (cyan and dark blue markers). This figure is generated to observe the deviation in both LoS and NLoS conditions, but also to compare with the approaches described in section 8.5 in this report. Figure 8.6 shows the percentage of error variation along both dimensions in both LoS and NLoS.

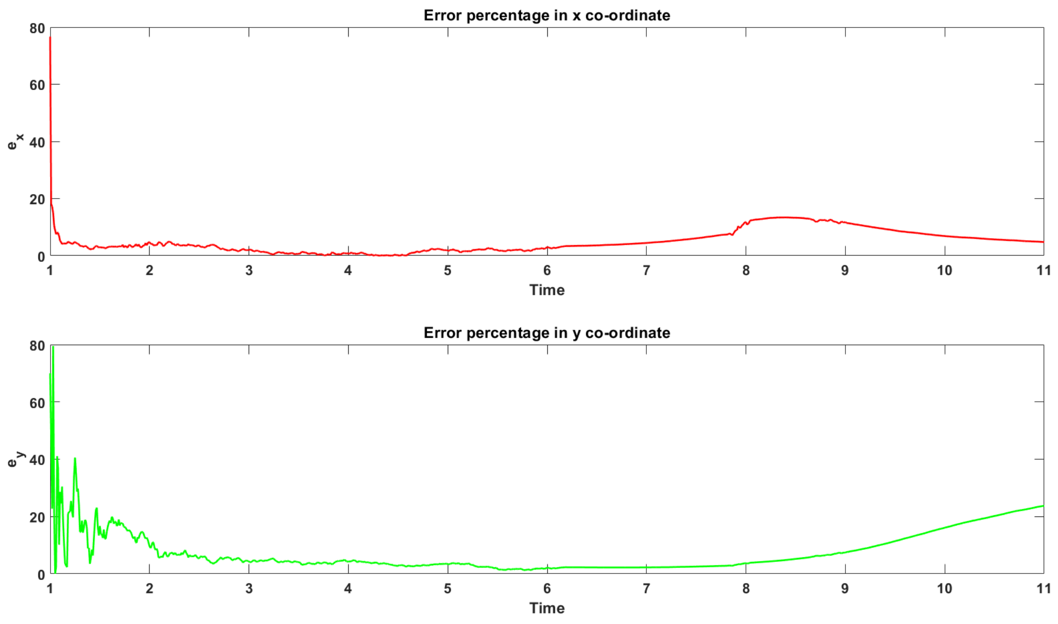


Figure 8.6: Error percentages

8.4 Motion of persons

The motion of a person in a warehouse or any other indoor environment is arbitrary. By taking this fact into consideration, two kinds of random motion have been studied [47].

8.4.1 Geometric Brownian Motion

This is a random statistical motion governed by the differential equations (8.5). Closed form solution of these differential equations is given by equation (8.6) [47].

$$\begin{aligned} dV &= \mu V_t dt + \sigma dW_t \\ X_t &= V_t dt + X_{t-1} \end{aligned} \tag{8.5}$$

$$V_t = V_0 e^{\left(\sigma W_t + \left(\mu - \frac{1}{2}\sigma^2\right)t\right)} \tag{8.6}$$

where
V = Velocity of particle
X = position of particle
dW_t = Random gaussian noise (Differential of Wiener process is equivalent to gaussian noise variable)
σ = volatility
μ = drift
dt = Time step

8.4.2 Ornstein Uhlenbeck Motion

This is a different category of random continuous statistical motion governed by the differential equations (8.7). Closed form solution of these differential equations is given by equation (8.8). Two dimensional ornstein uhlenbeck motion is generated(figure 8.7) so as to visualise the random motion of persons [47].

$$\begin{aligned} dV &= \theta(\mu - V_t)dt + \sigma dW_t \\ X_t &= V_t dt + X_{t-1} \end{aligned} \tag{8.7}$$

$$V_t = V_0 e^{-\theta t} + \mu\left(1 - e^{-\theta t}\right) + \sigma \int_0^t e^{-\theta(t-s)} dW_s \tag{8.8}$$

where
V = Velocity of particle
X = position of particle
dW_t = Random gaussian noise (Differential of Wiener process is equivalent to gaussian noise variable)
θ = tuning parameter 1

σ = tuning parameter 2
μ = drift
dt = Time step

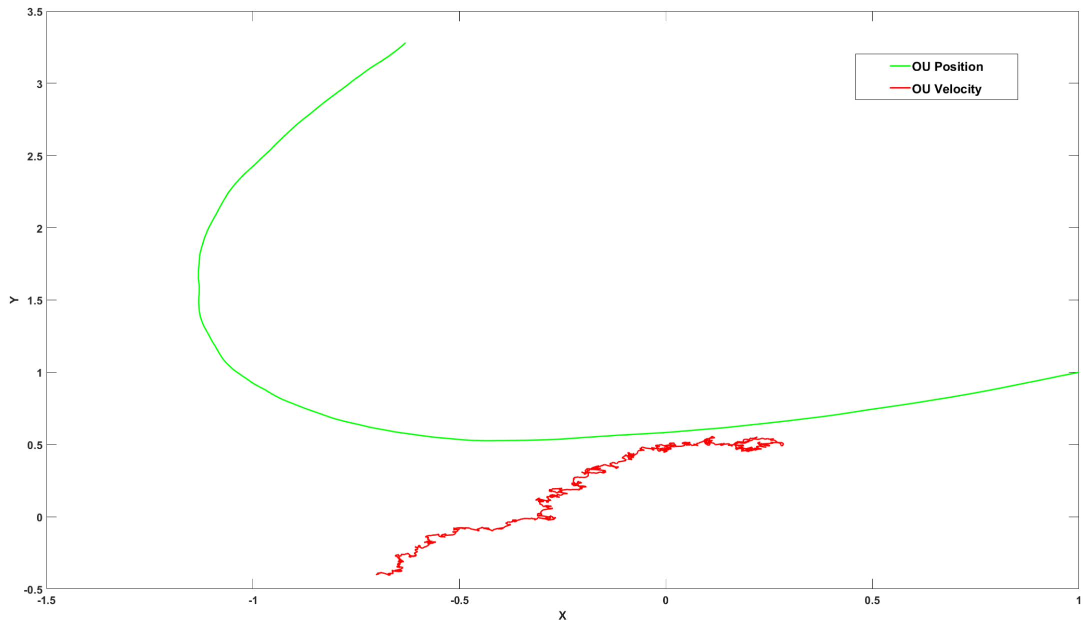


Figure 8.7: The red trajectory depicts the OU velocity across 2 dimensional. The green trajectory depicts the position of the OU in 2 dimension.

However it is complex to constrain these motions within the fixed volume space, hence the persons are kept stationary and the measurements observed from these beacons have a higher uncertainty than the stationary anchors.

8.5 Proposed Approaches

Three approaches have been explored to mitigate the deviation observed in conventional approach.

From the mathematical approach explained in section 8.2, a visibility array has been generated for each position along the trajectory, which is a matrix of 0's and 1's, where 0 indicates LoS and 1 indicates NLoS. For each sample time, a nx1 column matrix is generated (in this case $n = 3$). All of these column matrices are appended at the end of the simulation.

In the following approaches, measurements observed from UWB beacons are simulated as distances from measurement model. These distances are fed into UKF to get the observed deepcar trajectory simulating the process of multilateration at each instant. An ideal trajectory is simulated by state system model to assess the resemblance with observed trajectory.

8.5.1 Approach 1

In this approach, one unscented kalman filter is deployed with a measurement model, which works on the simulated measurements captured from 3 stationary anchors and 3 persons in both LoS and NLoS scenarios, to predict the position of deepcar(figure 8.8).

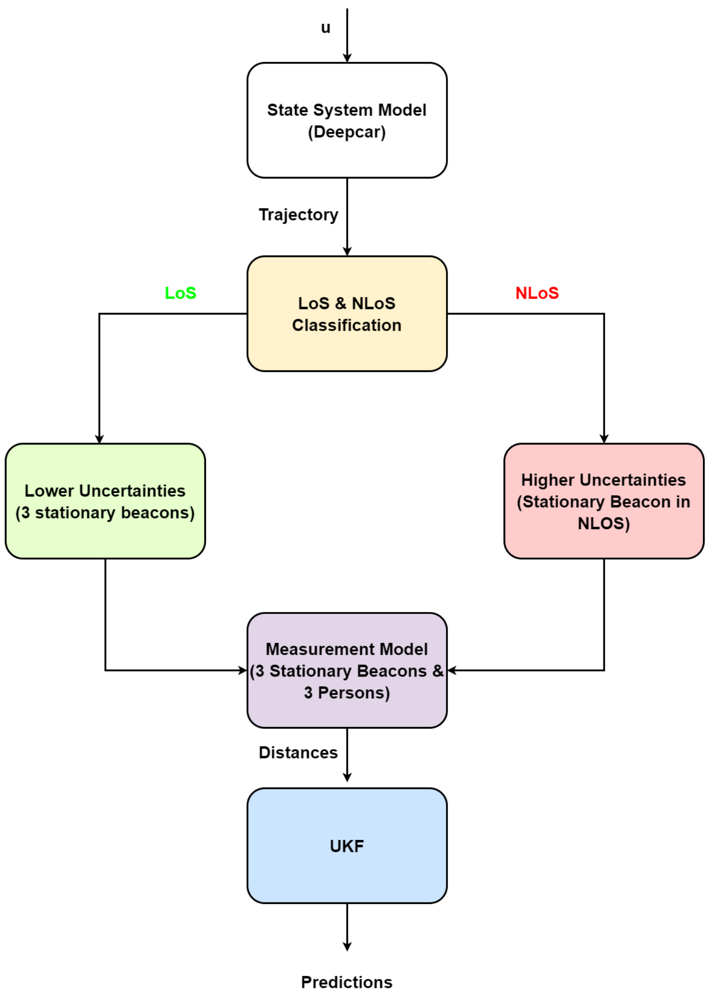

u
State System Model (Deepcar)
Trajectory
LoS
LoS & NLoS Classification
NLoS
Lower Uncertainties (3 stationary beacons)
Higher Uncertainties (Stationary Beacon in NLOS)
Measurement Model (3 Stationary Beacons & 3 Persons)
Distances
UKF
Predictions

Figure 8.8: Approach 1

Size of the uncertainty co-variance matrix in UKF is a factor, which should be taken into the consideration for design process. From figure 8.9, it can be implied that more observations in measurement model does not translate into more accurate UKF predictions.

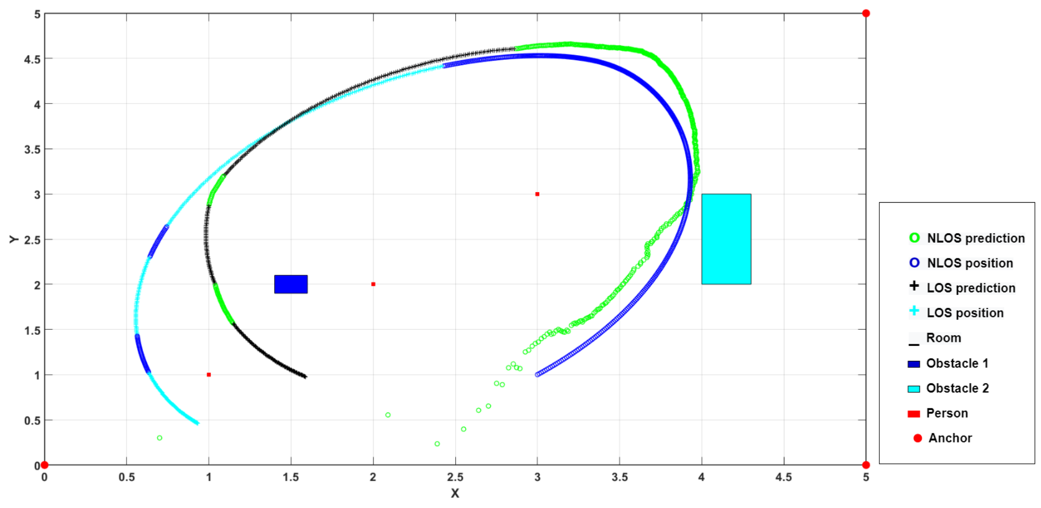


Figure 8.9: Trajectory Prediction by UKF

Figure 8.9 depicts the performance of the UKF when compared to ideal trajectory. From figure 8.10, the error percentage in y-direction is interpreted as UKF prediction in y-direction is similar to that of the conventional UKF, and the magnitude of uncertainty in measurements have been reduced to concretely say that higher dimensional covariance matrix yields a highly inaccurate predictions.

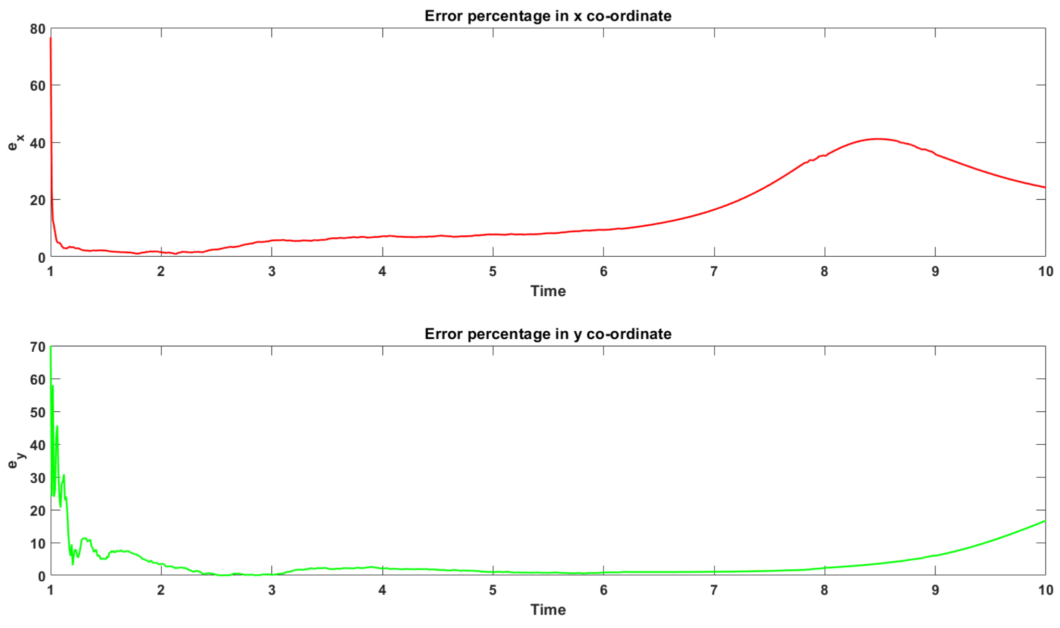


Figure 8.10: Error Percentages

8.5.2 Approach 2

In this approach, two Unscented kalman filters are simultaneously deployed with their respective measurement models to predict the position of deepcar(figure 8.11). Measurement model, in LoS scenario, works on the simulated measurements captured from only 3 stationary anchors. Measurement model, in NLoS, works on the simulated measurements captured from both stationary anchors and persons. It must be noted that the predictions are provided by only one of the UKF's, based on the LoS and NLoS condition of the deepcar.

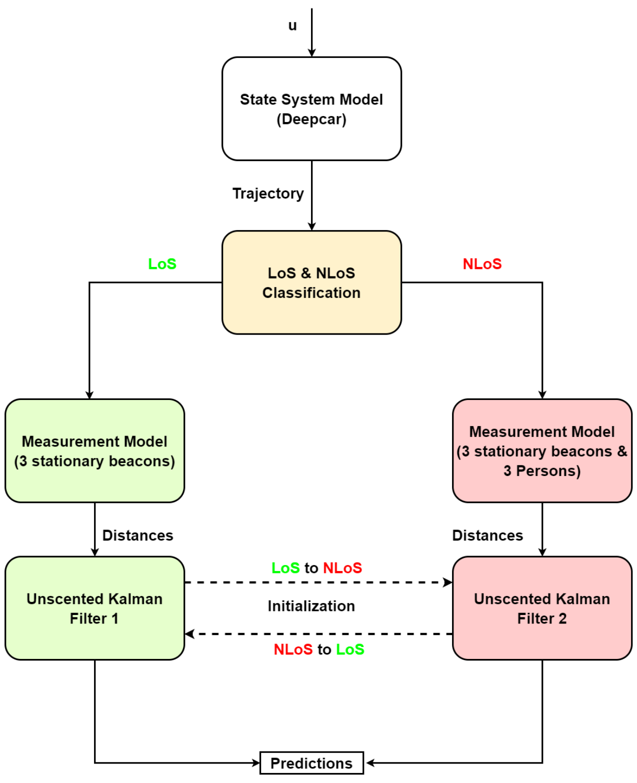

u
State System Model (Deepcar)
Trajectory
LoS
LoS & NLoS Classification
NLoS
Measurement Model (3 stationary beacons)
Measurement Model (3 stationary beacons & 3 Persons)
Distances
Distances
LoS to NLoS
Initialization
NLoS to LoS
Unscented Kalman Filter 1
Unscented Kalman Filter 2
Predictions

Figure 8.11: Approach 2

When the deepcar moves from LoS to NLoS, there is a need to use UKF 2 instead of UKF 1. Since, UKF 2 needs to be initialized with a state estimate, and it must be ensured that this estimate is taken as the last prediction from UKF 1 for generation of sigma points, so that the predictions are continuous. When the deepcar moves from NLoS to LoS, this cycle repeats itself instead initial state estimate for UKF 1 is provided by last prediction of UKF 2.

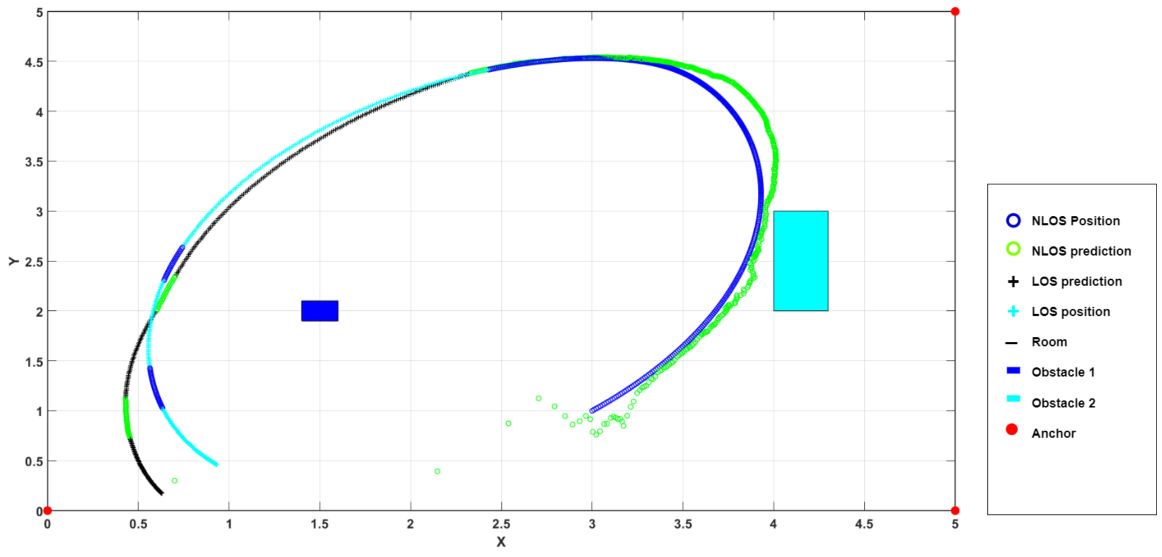


Figure 8.12: Trajectory Prediction by UKF

Figure 8.12 depicts the trajectory predicted using this approach (Green and Black markers). It can be implied that this result is better than the result from approach 1 in NLoS conditions. From the figure 8.13, it can be interpreted that regardless of the shifting between two UKF's, the error percentages are low along both directions.

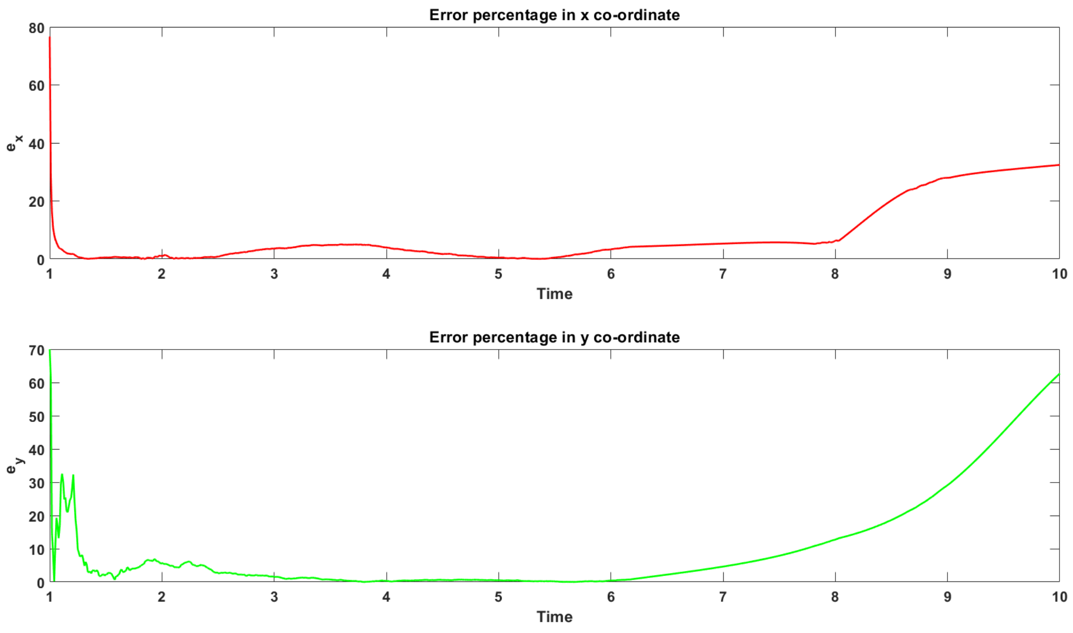


Figure 8.13: Error Percentages

8.5.3 Approach 3

In this approach, when the deepcar is in LoS, the UKF predicts the position of deepcar using measurements from 3 stationary anchors. When the deepcar is in NLoS, the UKF predicts deepcar's position using measurements from stationary anchors, which are in LoS with deepcar, and the persons, who are closest to deepcar, as illustrated in figure 8.14.

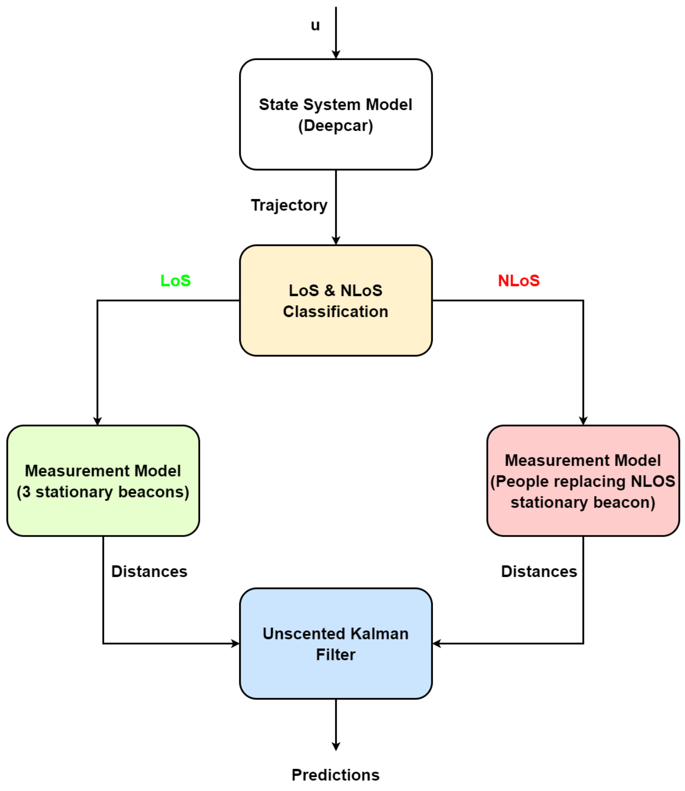

u
State System Model (Deepcar)
Trajectory
LoS & NLoS Classification
LoS
NLoS
Measurement Model (3 stationary beacons)
Measurement Model (People replacing NLOS stationary beacon)
Distances
Distances
Unscented Kalman Filter
Predictions

Figure 8.14: Approach 3

Hypothetically, the measurement from the anchor in NLoS should be replaced by measurement from person (non-stationary anchor) according to following decreasing order in priority:

1. Person who is visible to all stationary anchors.
2. Person who is closer to deepcar, when compared to others.
3. Relative motion of person should be comparably low when compared to others.

Since measurements are obtained from the beacons on people, Unscented Kalman filter should consider a different measurement model. For this purpose, visibility array has been again used to inform the filter to change the uncertainties.

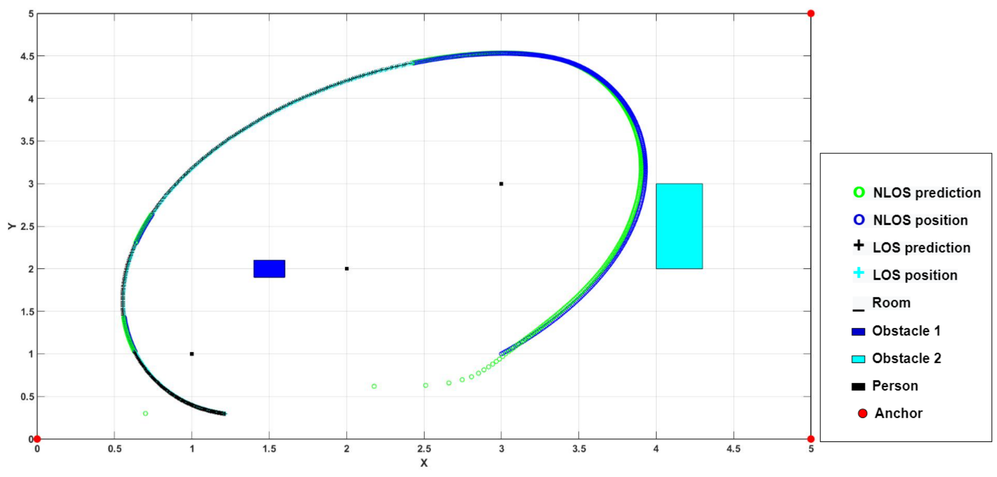


Figure 8.15: Trajectory Prediction by UKF

From the error percentages in figure 8.16 and the observed trajectory in figure 8.15, the result from this approach is nearly accurate to the ideal trajectory, and hence it is concluded that this is the most suited approach to mitigate NLoS effects.

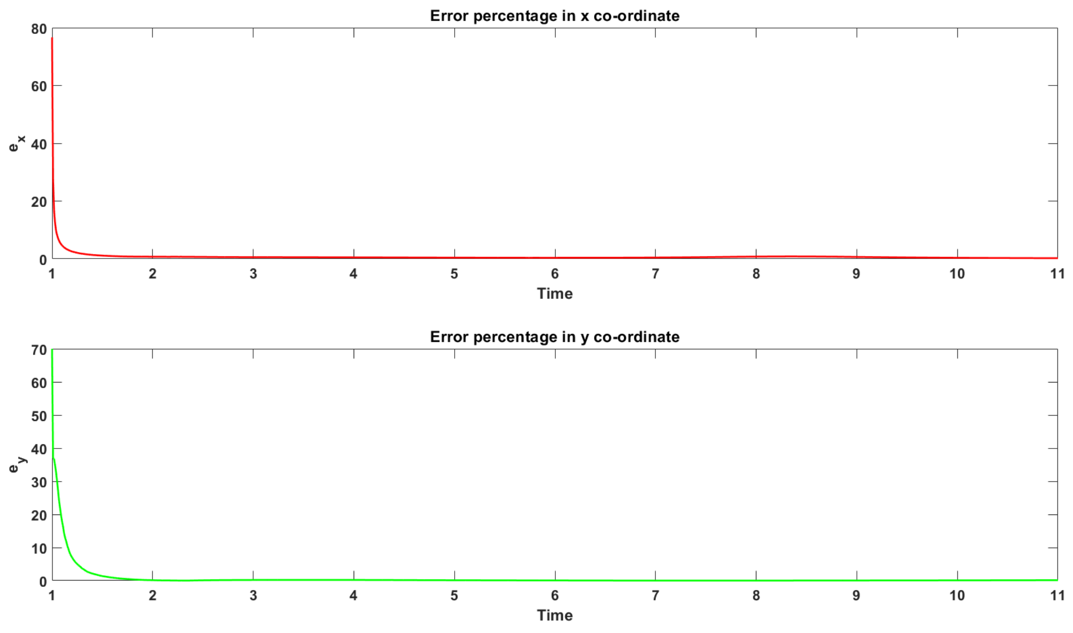


Figure 8.16: Error Percentages

Implementation 9

This chapter presents the collection of Channel Impulse Responses (CIR) deploying TREK1000 evaluation kit, classification models for LoS/NLoS identification, the launch of deepcar, and its localisation using three beacons.

9.1 Real time deployment

This section renders the execution of real time deployment of deepcar localisation. It includes the setup and the results achieved from the experiments, however the procedure and materials list will be attached in appendix.

9.1.1 Vicon lab Experiment

This experiment is conducted in Motion Tracking lab at AAU and the purpose of this experiment was to obtain a ground truth of the trajectory using Vicon Tracker software, so as to compare with the trajectory obtained by deepcar localisation.

Test setup

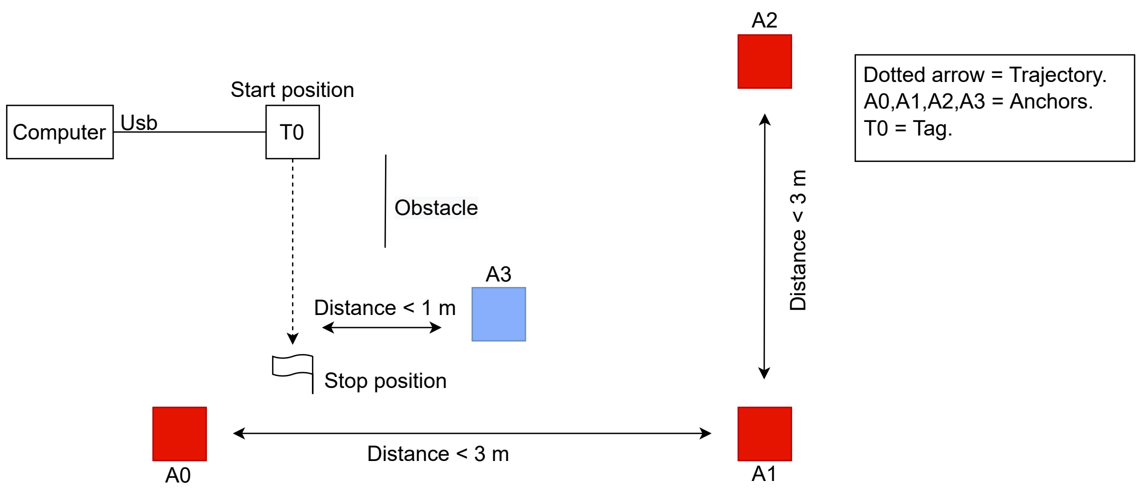


Figure 9.1: Experiment in Vicon lab (AAU's lab)

The setup illustrated in figure 9.1 exemplify the idea of replacing an anchor with a person carrying a tag. The red squares are the anchors and are used to localise deepcar by trilateration algorithm. In case of A_2 being in NLoS, the trilateration algorithm will be executed using A_0, A_1, and A_3, thus switching from anchor A_2 to the person(A_3 - blue square).

Results

The expected results of this experiment was to localise deepcar by using distances obtained from UWB anchors with a minimal offset. When this trajectory was compared with the ground truth generated by Vicon tracker software, the offset in the localised position is found to be greater than 90 cm in x-axis and around 25 cm in y-axis. It was concluded that the volume space, required to conduct this experiment was low and hence experiments were all performed in room A4-106 (AAU's lecture room).

9.1.2 LoS implementation

Due to higher offset in tracking from the previous experiment(9.1.1), the experiment was moved to another room with more volume space to check whether this problem can be solved.

Test setup

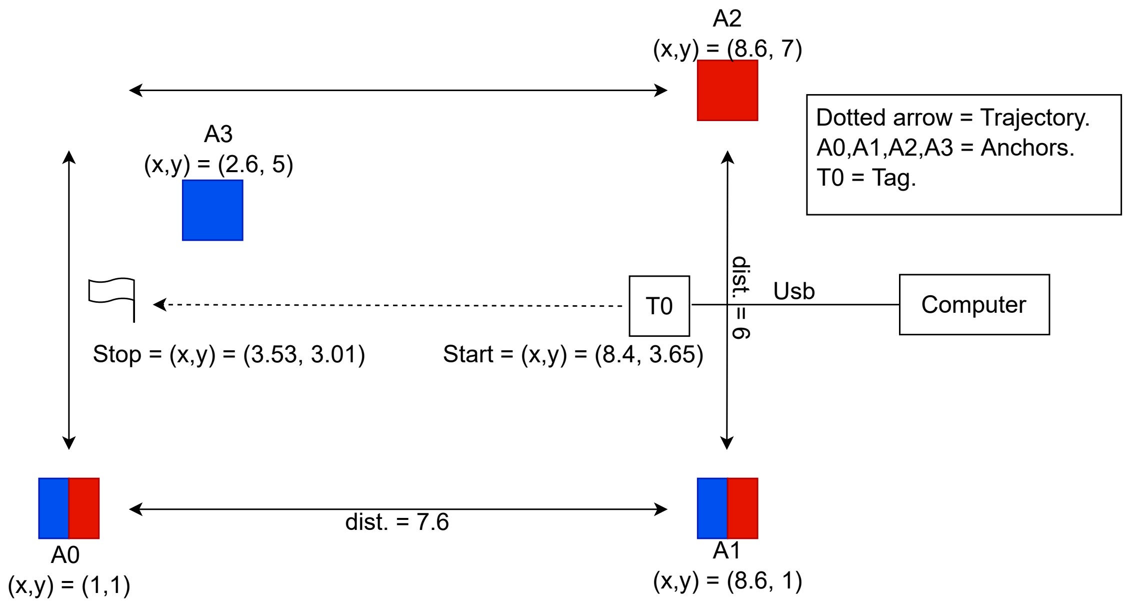


Figure 9.2: Experiment in case of LoS

The test setup that is depicted represent two sets of trilateration performed simultaneously. One consists of the anchors A_0, A_1 and A_2 (coloured in red), the second contains the anchors A_0, A_1 and A_3 (coloured in blue). When deepcar is navigated as shown in figure 9.2, two trajectories appeared by running real-time trilateration in computer.

Results

Moving to bigger volume space reduced the offset value to an extent. As seen in figure 9.3[1] the graphs emulated each other, however the trajectory, colored in red, has noticeable peaks than the trajectory, colored in blue, but this is considered rather insignificant in this case.

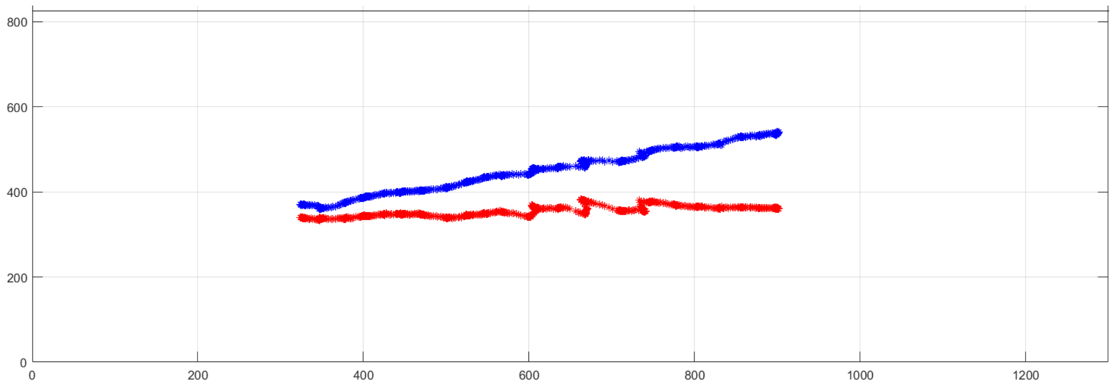


Figure 9.3: Result of LoS for the two trilaterations

9.1.3 Weak NLoS

The purpose of this experiment is to observe the effect of Non Line of Sight (NLoS) on the trajectories (from two trilaterations) of deepcar, created by a metal sheet.

Test setup

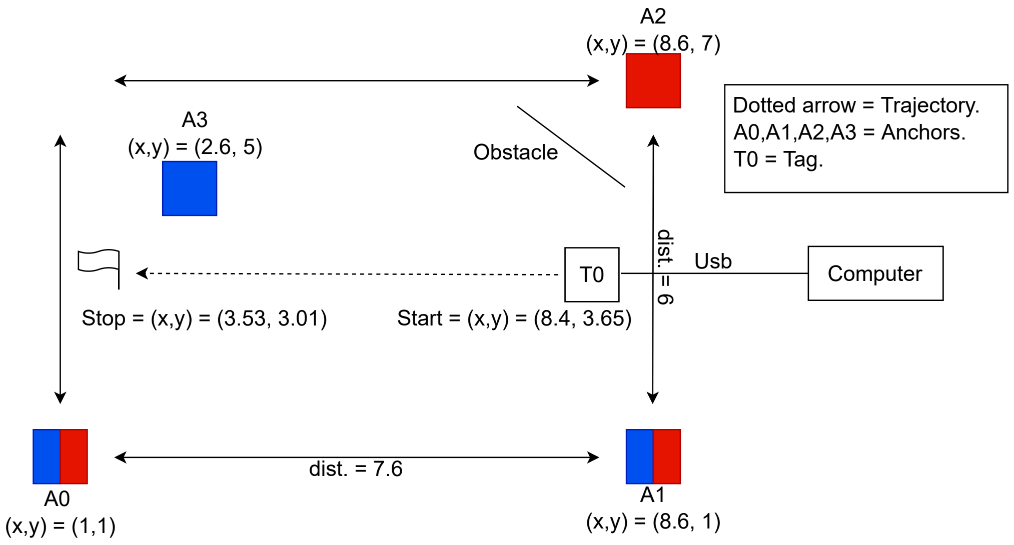


Figure 9.4: Illustration of test setup where A_2 is in NLoS all the time

[1]Note: The axis on the graph is the position in x and y coordinates and the location of the Deepcar represented in the graph is moving from right to left and the likewise is applied for all the following graphs in this chapter.

The test setup in this experiment is similar to the one in section 9.1.2. The only difference is anchor A_2 is in NLoS with deepcar, due to the metal sheet.

Results

This experiment was executed a number of times. The big ordeal was repeatedly finding difficulties due to:

1. No noticeable deviation in the trajectory even though one of the anchors is in NLoS with deepcar.
2. Multipath propagation.
3. Possible synchronization error.

The outcome was not as expected since there are more similarities in the trajectories, except for one noticeable dip/decrease in the red trajectory (around point 740 mm in the x and 390 mm in the y), but theoretically they should be more different because of NLoS situation. Observing figure 9.5[2] the blue trajectory is designed to replicate an LOS situation (created by A_0,A_1 and A_3), whereas the red trajectory imitate an NLoS situation (created by A_0,A_1 and A_2).

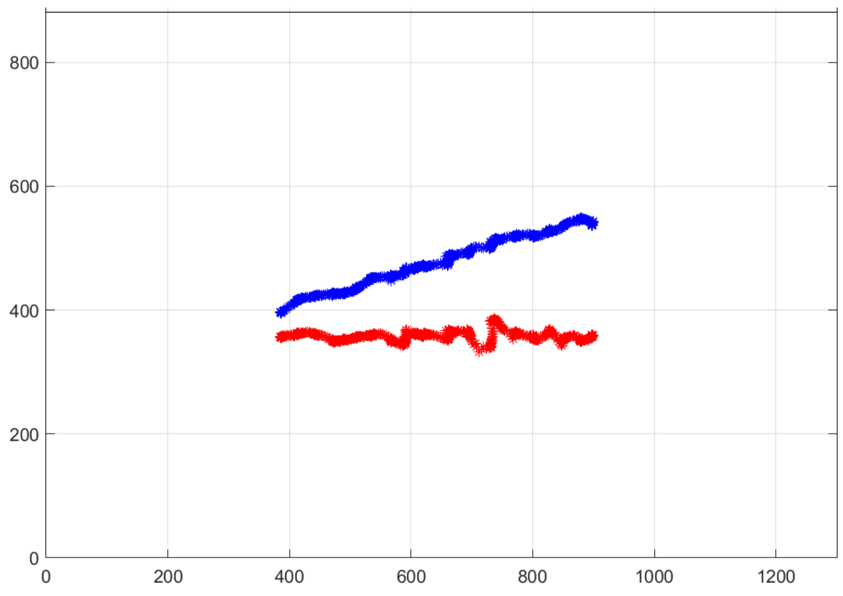


Figure 9.5: Result of weak NLOS created by placing the anchor A_2 behind two metal plates.

9.1.4 Multi-lateration

The purpose of this experiment is to check whether the NLoS situation, created by metal sheet, produces a noticeable deviation in the trajectory, generated by using four anchors, instead of three anchors.

[2]Note: The axis on the graph is the position in x and y coordinates and the location of the Deepcar represented in the graph is moving from right to left and the likewise is applied for all the following graphs in this chapter.

Test setup

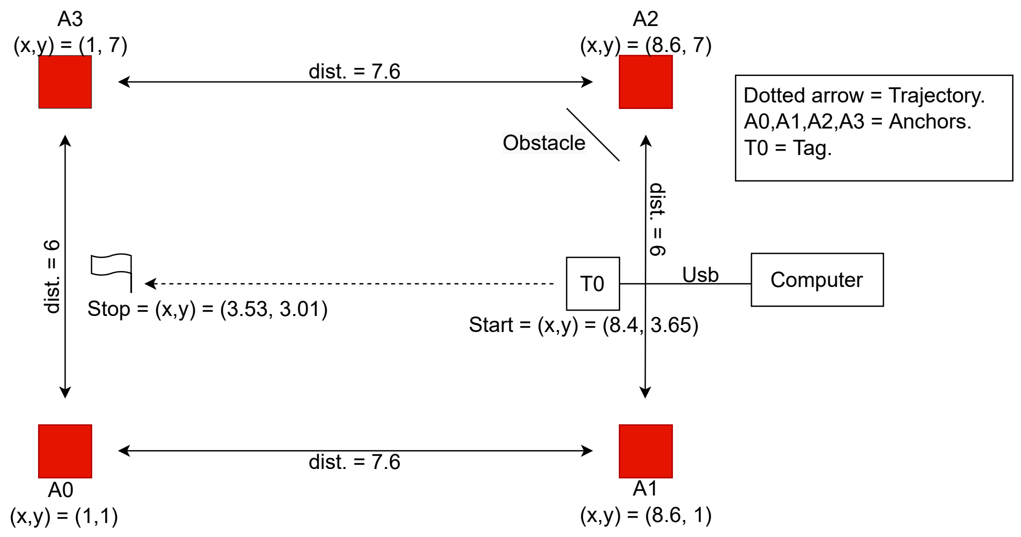


Figure 9.6: Experiment in multi-lateration

Figure 9.6 illustrate the multi-lateration with NLoS situation created by metal sheet. The anchor, that is hindered, is anchor A_2, which is seen at position (8.6,7), top right corner.

Results

The outcome (deviation in NLoS situation) was better than expected, because just by adding one additional beacon to the system, detection of NLOS was visible 9.8[3].

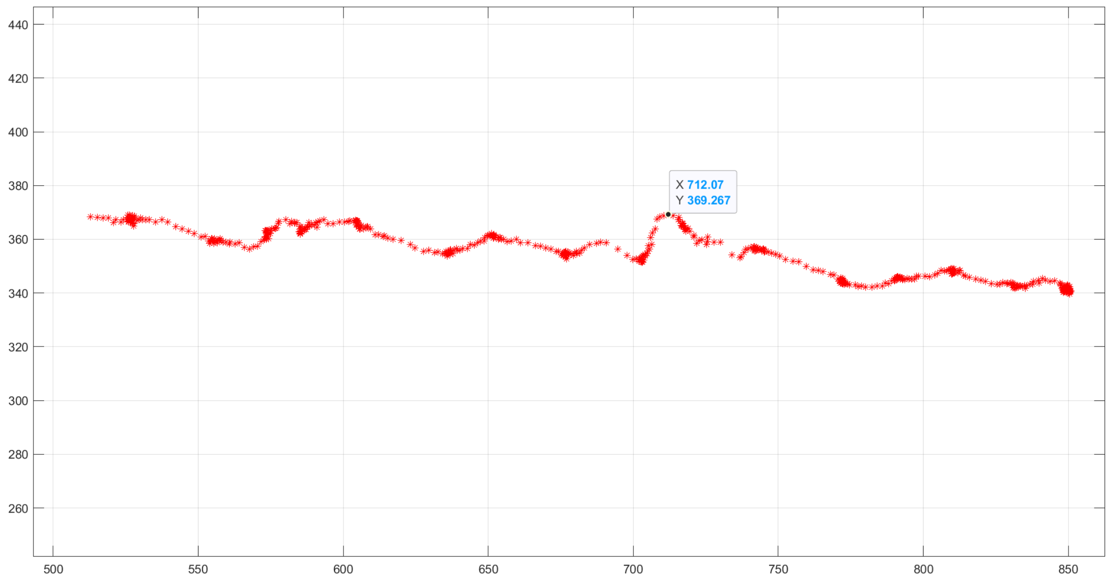


Figure 9.7: Result of multi-lateration in LoS

[3]Note: The axis on the graph is the position in x and y coordinates and the location of the Deepcar represented in the graph is moving from right to left and the likewise is applied for all the following graphs in this chapter.

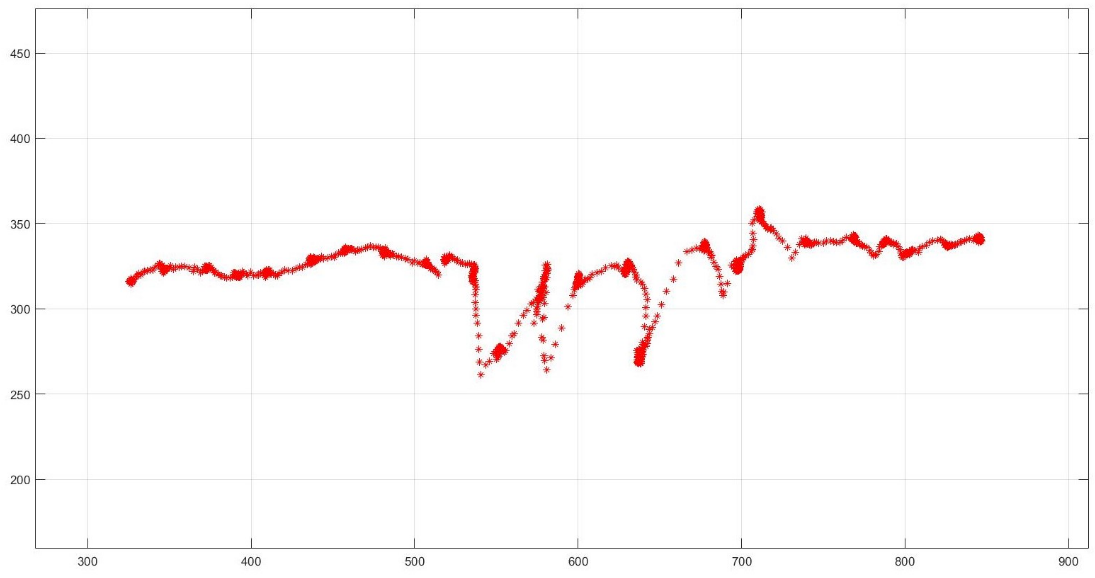


Figure 9.8: Result of multi-lateration in NLoS

9.1.5 Strong NLoS

The purpose of this experiment is also to observe the effect of NLoS, created by a L-shaped obstacle, on the red trajectory, as there is no noticeable deviation from the experiment in section 9.1.3.

Test Setup

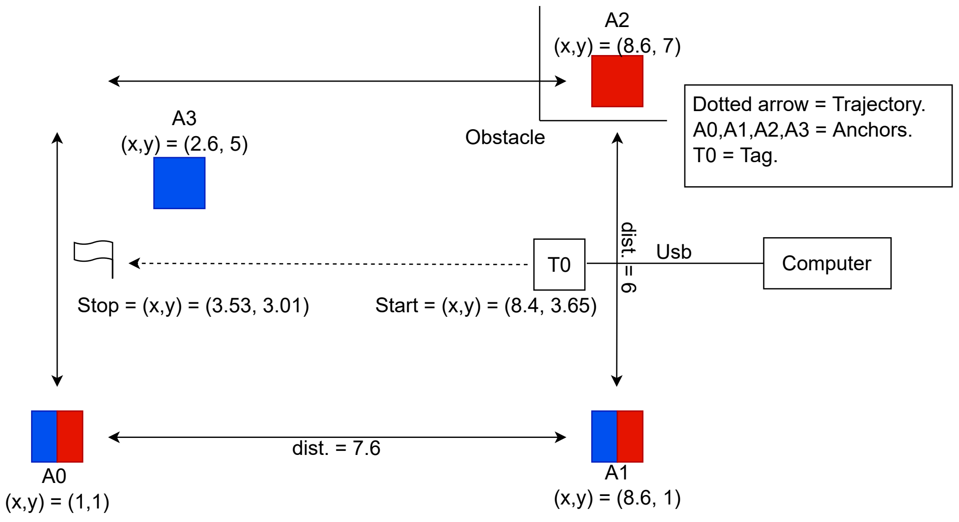


Figure 9.9: Illustration of test setup experiment 2 where A_2 is in NLoS all the time

The test setup in this experiment is similar to the one in section 9.1.2. The only difference is anchor A_2 is in NLoS with deepcar, due to the L-shaped obstacle.

Results

A stronger NLoS scenario has been created, because of more reflections, of the propagated signal, caused by the L-shaped obstacle. This result, shown in figure 9.10[4], can be compared to the 9.1.3 experiment, since it shows how the L-shaped obstacle affects the trajectory.

[4]Note: The axis on the graph is the position in x and y coordinates and the location of the Deepcar represented in the graph is moving from right to left and the likewise is applied for all the following graphs in this chapter.

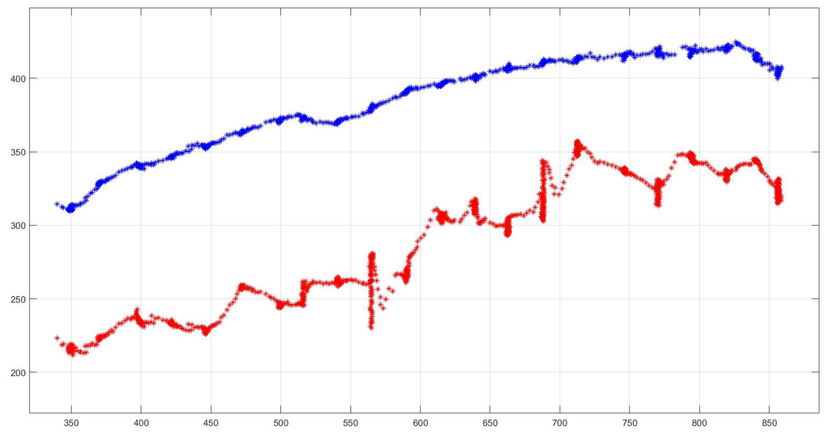


Figure 9.10: Result of stronger NLOS created by covering the sensor in an L-shape.

In figure 9.10 two different trajectories (red and blue) are observed. The red one illustrates A_2 in constant NLOS, whereas the blue trajectory illustrates the trajectory in LOS condition. It is clear to see a bigger fluctuation in the red trajectory.

9.1.6 Transition from LoS to NLoS

The final experiment was to incorporate both LoS and NLoS as this reflects real-life incident, when the deepcar is moving from one place to another.

Test setup

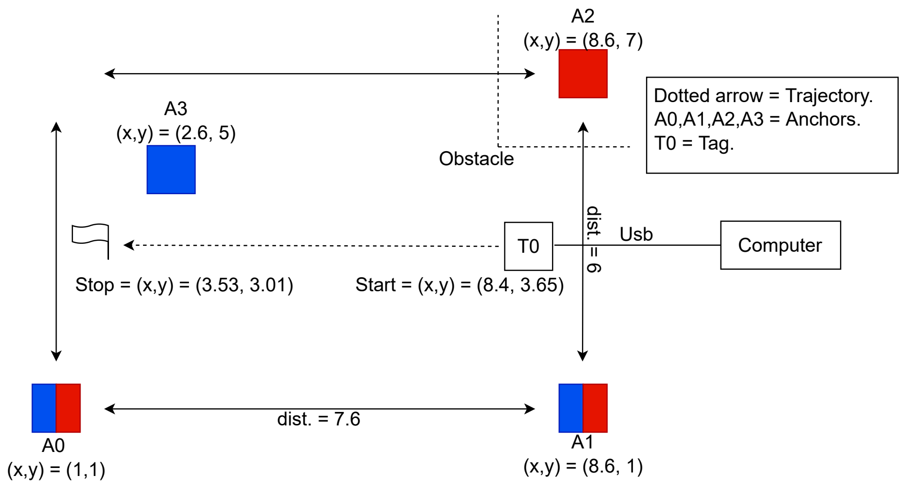


Figure 9.11: Experiment in case of LOS to NLOS

This setup is exactly same as figure 9.9. The only difference is the transition, which is manipulated by placing the metal plates in a L-shape after a certain amount of time passed by, illustrated in dotted line (obstacle).

Results

When implementing the transition from LoS to NLoS, the behaviour of the trajectory evidently shows a clear change, shown in the red vertical line around position 640 in the x axis and 350 (to 250) at the y position. The trajectory keeps fluctuating after that position, which fits the real life experiment, since the metal plates were not removed, hence the deepcar was in consistent NLoS, see figure 9.12[5].

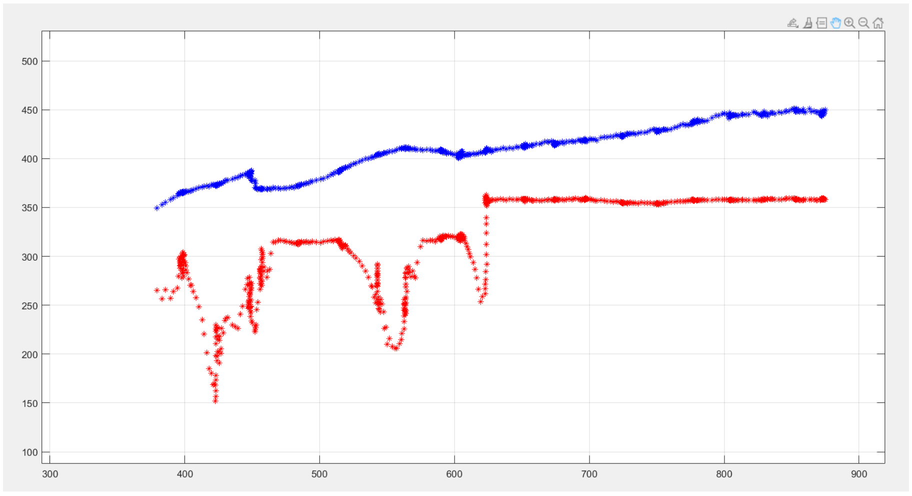


Figure 9.12: Result of transition created

9.2 Inferences

The following points list brief inferences from the experiments:

- From the experiments conducted in Vicon Lab as described in section 9.1.1, the offset in the localised position of the deepcar was due to smaller volume space ($< 10m^2$). Hence, the experiments have been conducted in a larger volume space ($45.6m^2$).
- Practically, there is an offset between trajectories triangulated using three different anchors, as observed from experiment in section 9.1.2. Theoretically, they should coincide with each other.
- NLoS created by the metal plate does not deviate the trajectory, so that the difference is noticeable, as observed from experiment in section 9.1.3.
- Multilateration using all anchors produces a noticeable difference in trajectories, even though NLoS created by metal plate is weak, as observed from experiment in section 9.1.4.
- NLoS created by two metal plates shaped in the form of "L" around anchor A_2 produced a noticeable difference in trajectories, as observed from experiment in section 9.1.5. This experiment is further extended to check the trajectory deviation from LoS to NLoS in experiment described in section 9.1.6.

[5]Note: The axis on the graph is the position in x and y coordinates and the location of the Deepcar represented in the graph is moving from right to left and the likewise is applied for all the following graphs in this chapter.

The problems that hindered us from deploying the idea to a full extent.

1. DWM 1000 chips are not configured to capture CIR response, making it complicated to implement classifier without deploying TREK evaluation kit.
2. The measurements read on UWB tag, should match with the input given to the deepcar, at that instant, that leads the deepcar to a position that we observe in real time.
3. UKF is difficult to implement on the deepcar because of its compact operating system, hence necessitating for a centralised server for tracking of deepcar. This server should keep track of persons, anchors in NLoS and then triangulate the position of deepcar at each instant.

9.2.1 Ideal Physical Deployment

Figure 9.13 provides an overview of deployment. Each workstation sends the status of LoS or NLoS, of the concerned beacon with deepcar, to main workstation for each sample time wirelessly. Persons, acting as tags, will also send their observed distances, from beacons, to the main workstation wirelessly. The deepcar sends the distance measurements to the central workstation over an already established Wi-Fi connection. The UKF in central workstation utilises all the distance measurements from people, deepcar, and statuses, to predict the position of deepcar.

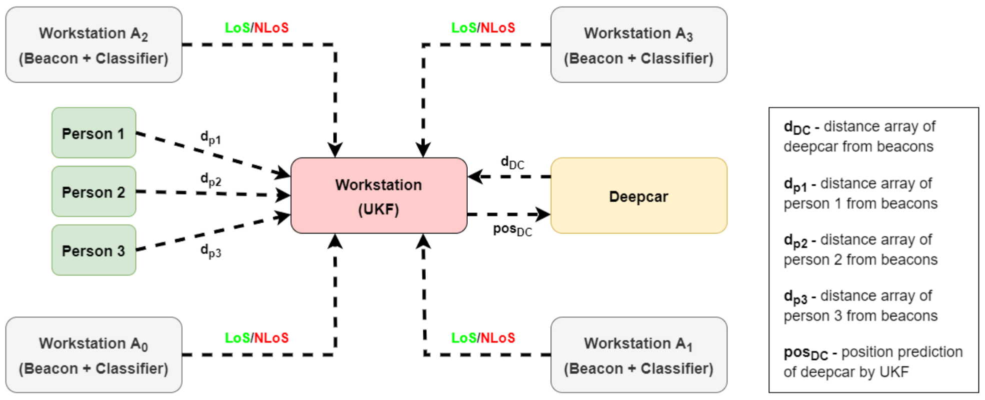

Workstation A2 (Beacon + Classifier)
LoS/NLoS
LoS/NLoS
Workstation A3 (Beacon + Classifier)
Person 1
dp1
Person 2
dp2
Person 3
dp3
Workstation (UKF)
dDC
posDC
Deepcar
Workstation A0 (Beacon + Classifier)
LoS/NLoS
LoS/NLoS
Workstation A1 (Beacon + Classifier)
dDC - distance array of deepcar from beacons
dp1 - distance array of person 1 from beacons
dp2 - distance array of person 2 from beacons
dp3 - distance array of person 3 from beacons
posDC - position prediction of deepcar by UKF

Figure 9.13: Deployment of Kalman Filter

Discussion and Future Work 10

This chapter discusses various problems involved in this project and feasible approaches to solve these problems. Also, this chapter briefly presents the prospects of this project.

10.1 Discussion

This section involves the description of issues (overview is provided below) faced with hardware, software, and how they have been dealt according to the use case.

10.1.1 Overview of Technical Challenges:

- Flashing DWM 1000
- Tuning Parameters of UKF
- Offset in trilateration(A_0, A_1, A_2) and trilateration(A_0, A_1, A_3)
- Low deviation in NLoS
- Non-stationary anchors in Multi-lateration
- State-space Model

10.1.2 Flashing DWM1000

The necessity of flashing the DWM1000 was to capture CIR as it only reads distances. The flashing was successfully done, however the outcome was not as expected because of the following:

1. Unable to configure into reading CIR
2. The DWM1000 did not display whether the signal is transmitted or received.

In order to rectify the problem, the DWM1000 was once again flashed and restored with the original firmware, however the display still remained non-functional. The DWM1000 was interfaced with an API program called "Putty" which indicated that the sensors were able to read and transmit data (distance), however the data structure was corrupted. Hence TREK1000 was again used to capture CIR's and to deploy the classifier.

10.1.3 Tuning Parameters of UKF

For the UKF in our project, the main parameters that need to be chosen carefully are the uncertainty of the measurements provided to UKF and uncertainty in measurement from NLoS anchor and non-stationary anchor. To measure the performance of UKF, error mean along both x and y axes are considered.

Measurement uncertainty covariance(P_{zz}) is a tuning parameter because it can be interpreted as training UKF to deal with the deviation in observations. If we set the P_{zz} high, then UKF is designed to track the state precisely, even with deviations in measurements (whose covariance falls below P_{zz}).

In simulation, it should be noted that the uncertainty provided to both NLoS and non-stationary anchor in the measurement model should match with the deviation it produces in simulation. The uncertainty given to non-stationary anchor should be lower than the uncertainty provided to anchor (which is in NLoS), yet higher when compared with the uncertainty of stationary anchor. The deviation produced in measurement from anchor, which is in NLoS, should be lower such that trajectory does not cross the room.

10.1.4 Cause of Offset in trilateration(A_0, A_1, A_2) and trilateration(A_0, A_1, A_3)

Theoretically, the position of deepcar determined by trilateration from any set of anchors should coincide. The offset in figure 9.3 is due to compounding of following errors [48]:

- Gaussian Errors
- Multi-path Errors, typically due to environment
- Errors due to detected measurement noise by UWB transceiver

Multi-path errors are positive (always higher distances than actual distances because of reflections) and relatively constant in an environment. Gaussian errors are due to blocking potential paths from the first path detected, distance between two transceivers, and also might be due to calibration differences between DWM1000 hardware [48]. From experimental observations, it was also observed that the distance between the tag and an anchor affects the error in distance reading.

10.1.5 Deviation in trajectory due to NLoS

From the results(figure 9.5) of experiment (figure 9.4), there is no noticeable deviation in trajectory because of two reasons:

1. The metal sheet is creating NLoS, but not sufficient enough to cause much deviation in trajectory.
2. From the measurements of A_0 and A_1, mathematically two solutions exist as shown in figure 10.1. The measurement from A_2 is utilised to determine which solution should be considered. Due to weaker NLoS by metal sheet, least squares optimization method will give a solution, that resembles LoS solution.

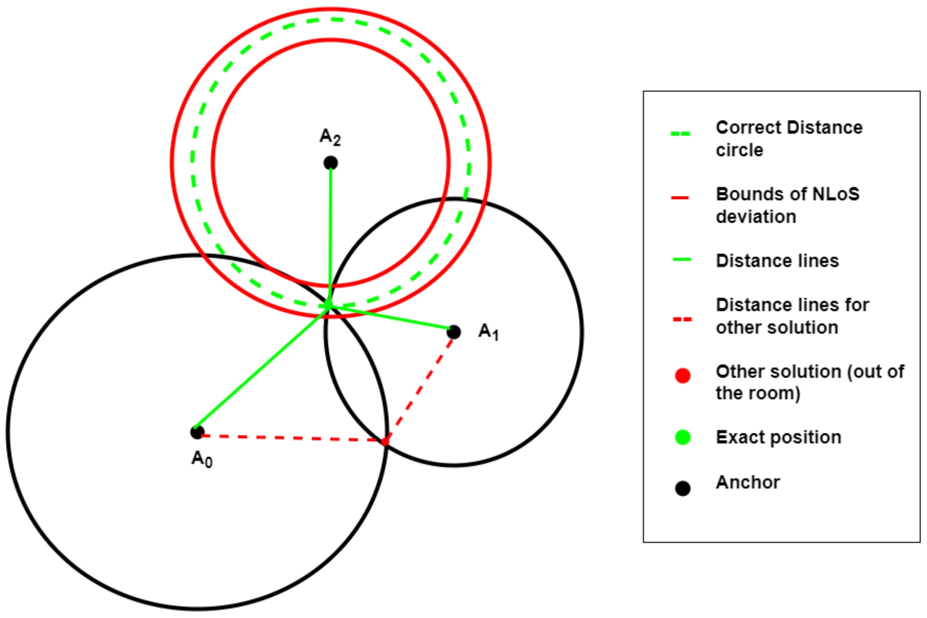


Figure 10.1: Interpretation of no noticeable deviation in trajectory

10.1.6 Multilateration

Since the setup, in figure 9.6, was so conventional, it was further developed to use a fifth beacon for the person. This was not a possibility as the component(DWM1000) was not configured to read distance from a fifth beacon. An alternative has been tried using an additional DWM1000 tag and an anchor (person), configured to send distance reading to this extra tag. Both DWM1000 tags are connected directly to the computer, with fifth anchor at a certain position. These tags should be read simultaneously, but did not, because of MATLAB configuration, and using a Parallel Computing toolbox, in MATLAB, did not solve this problem, thus discarding this idea.

10.1.7 Problems with state space model

The input provided to the state space model is considered as continuous in time for kalman filter, but in implementation (described in detail in appendix J.1), input (velocity given by clicks to deepcar) is in the shape of pulses as illustrated in the figure 10.2. The measurements (distances of anchors observed on tag) are read non-uniformly in the experiments. During the time, when there is no input, the measurement gets clustered for that specific position. During the time, when the input is continuous, fewer measurements are read, when compared to the previous. Hence, it becomes difficult to provide the measurement observed at that specific input to the unscented kalman filter.

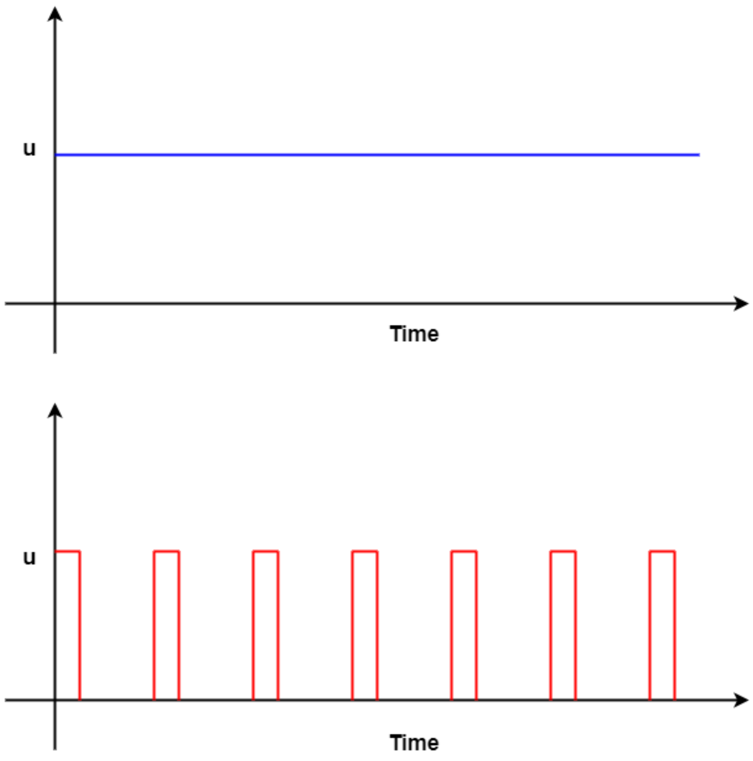


Figure 10.2: Input in simulation and Input in implementation

10.2 Future Work

Software, in which deepcar position is triangulated by UWB, can be upgraded so that either a plan or a scan of an indoor setting can be uploaded, thus aiding us in identification of deepcar location. A Real time robust classifier can be developed so that classification is accurate in various indoor settings.

In the development of state space model, the mean and co-variance of the errors of gaussian nature can be estimated by using Maximum Likelihood(ML) estimation for the DWM1000 hardware [48].

As described in section 9.2.1, future work encompasses the setting up of central server, recording the positions of each moving entity (deepcar and persons) in the warehouse. After obtaining improved localisation of deepcar using people as non stationary anchors during NLoS, a swarm based approach for localisation can be proposed wherein people who are in NLoS with respect to stationary anchors are also localised accurately during NLoS condition by the use of other LoS non stationary anchors, which include people and the autonomous robots. Therefore every object or person with a UWB is interconnected since they may assist one another during NLoS conditions. Design of the state model for the deepcar, that appropriately fits for this use case. An optimal path prediction algorithms could be implemented than PID controller for deepcar.

Based on the Received Signal Strength Index(RSSI) values received by the DWM1000 chips, which are configured as anchors, there might be a possibility to determine whether the respective anchor is either in LoS or in NLoS. If this result, combined with distance, communicated with tag, then workstations at four anchors can be removed, thus reducing monetary costs. Yet, there is a need for centralised server for tracking of persons and utilising them for localisation of deepcar in case of NLoS.

Appendix A

A.1 Deepcar Setup

When deepcar is switched on, an "Xshell" named ssh(secure socket shell) terminal is launched, where commands are provided to the deepcar through a wifi, to which a computer is connected. There are various commands for the execution of different functions (object detection, driving, indoor localization). Indoor localization function is considered for the navigation of deepcar using UWB sensors. Indoor localization is defined as the process for estimating the position of deepcar at any instant by using the anchors. This function is executed in Xshell and an application "main" is launched, where a 2 dimensional map shows the position of anchors and tag. If we select a position on the map, the deepcar moves to this position in the map.

Deepcar is programmed in Robot Operating System (ROS). Different functionalities have been implemented by calling ROS nodes. Each ROS node corresponds to a specific functionality.

1. Connect to the Wifi named "Deepcar" when the deepcar is turned on.
2. Launch "Xshell.exe", which is an SSH terminal.
3. Enter the host address as "192.168.1.102" and enter the username as "Deepcar" and password as "1234"
4. Launch the deepcar by providing command "cd /deepcar_launch"
5. Indoor localization is implemented by calling ROS node by "roslaunch start_uwb.launch"
6. The application "main.exe" should be opened, where the anchor co-ordinates are provided and it is sent to deepcar by clicking "send UWB configuration".

Positions of deepcar can be observed on Xshell terminal and also it can be observed in two different maps in the "main" application.

1. A two dimensional picture (figure A.1) with respect to anchor positions given.
2. A configured map (figure A.2) where only deepcar position is observed by a red dot and the destination is represented by a blue square.

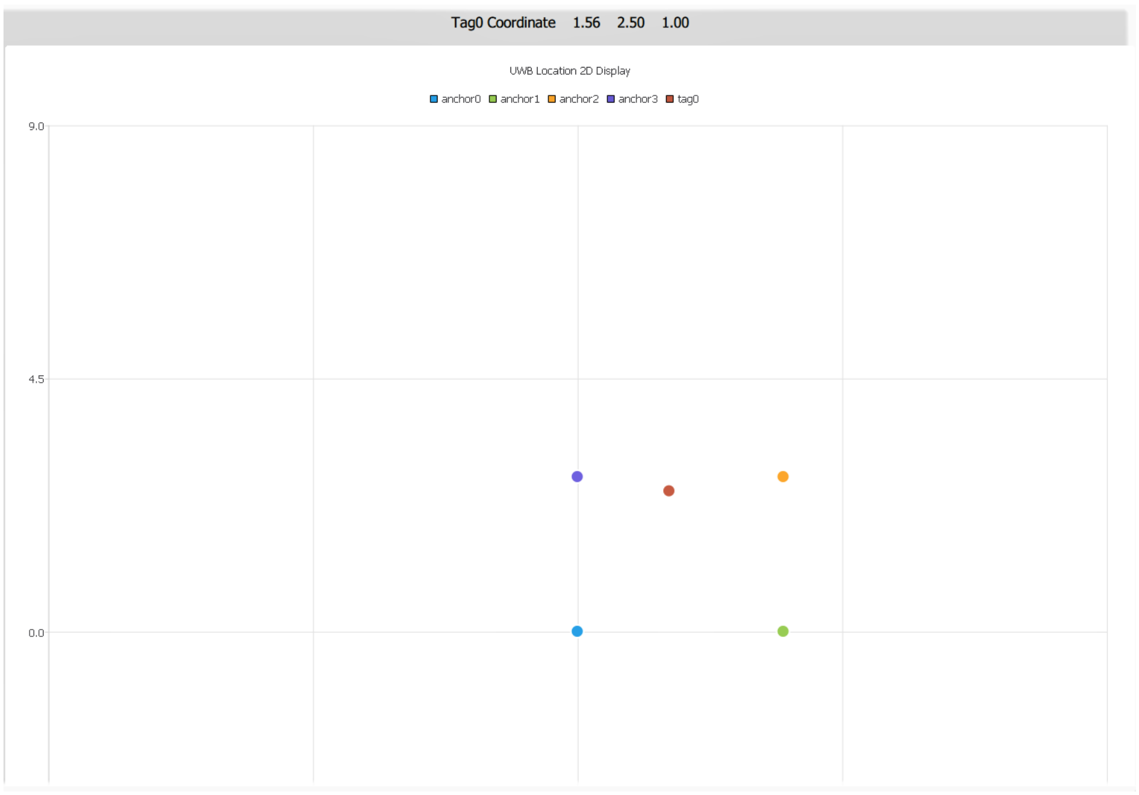


Figure A.1: Position of deepcar with respect to four anchors

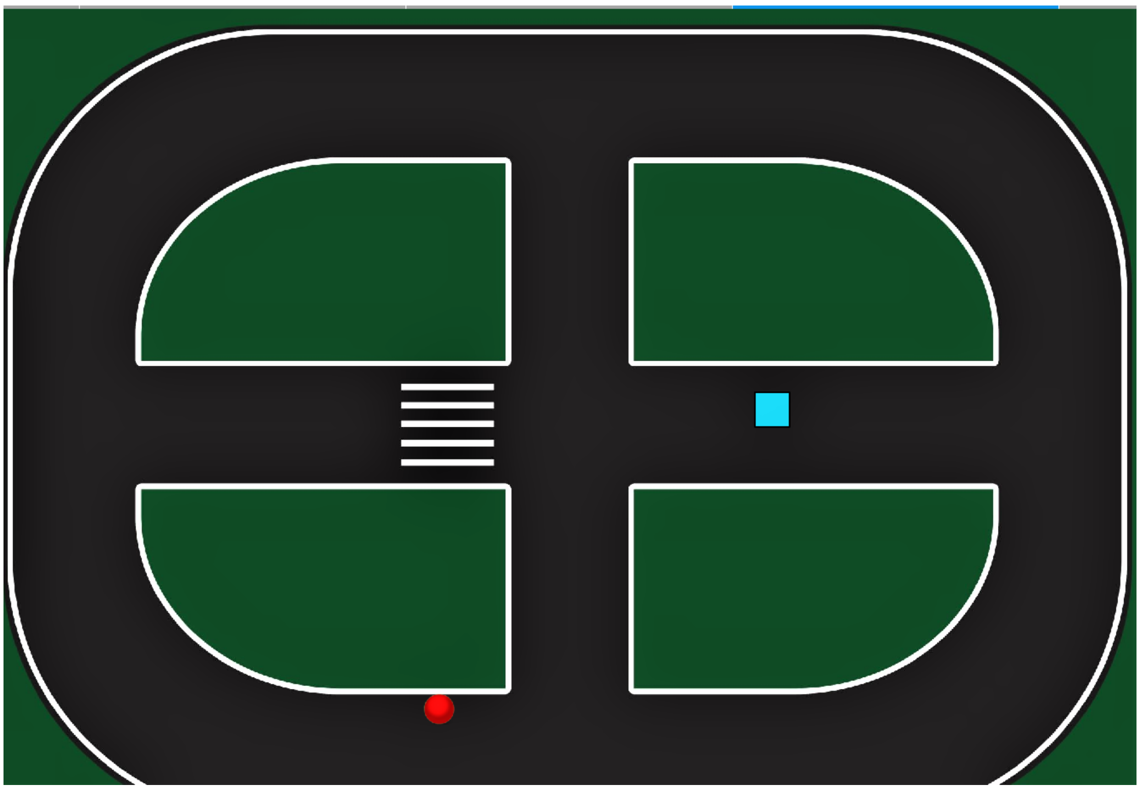


Figure A.2: Position of deepcar in the environment configured

A.2 TREK Setup

1. Connect the TREK module configured as anchor to a computer. Ports in device manager should show the name of port to which TREK is connected.
2. Connect the TREK module configured as Tag to a power source.
3. Ensure proper communication between the anchor and the tag. Check the frequency of blinking in both the anchor and the tag. LED on the tag blinks with higher frequency than the LED's on the anchor.
4. If not proper communication, reset the tag.
5. Run the 'ChannelImpulseSerial.m' to obtain CIR responses.

Appendix B

B.1 Mounting Board for TREK1000

Purpose

The purpose of this experiment is to 3D print two plastic boards to hold TREK modules on a tripod.

Design

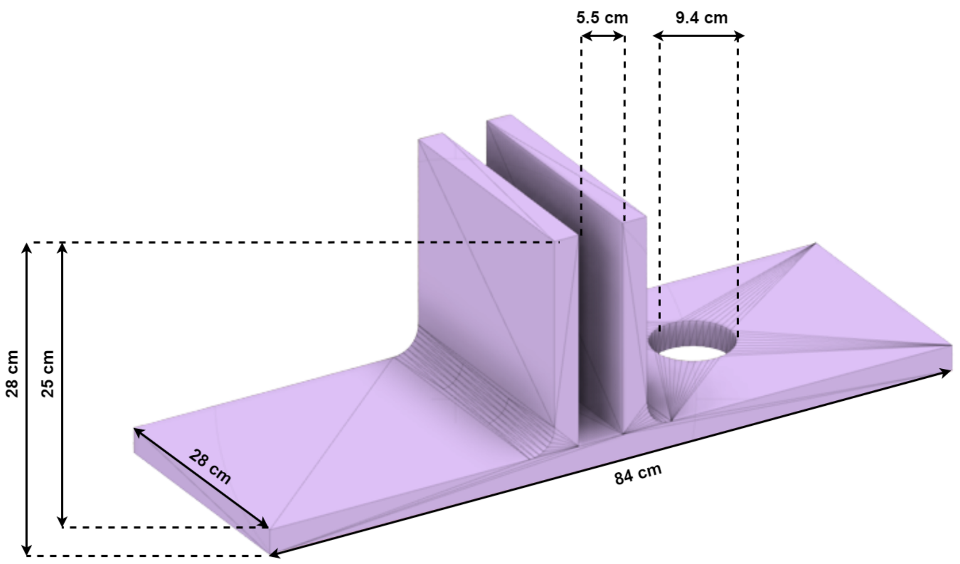


Figure B.1: 3D Design of mounting board

The board is designed in Fusion 360 software. The height of two extrusions (from the base) on the board, and the gap between them, is determined by the dimensions of the TREK board (figure 6.2). The diameter and threading of the hole in the mounting board is determined from the parameters of the fastener on the tripod.

Result

After designing, the stl file is exported to the 3D printing machine to print the mounting boards.

Appendix C

C.1 DWM1000 flashing

Purpose

The intention of this flashing the firmware of DWM1000 was to enable the caption of Channel impulse responses since the DWM1000 was only able to receive distances. This will lessen the complexity of getting acquaintances with an other equipment namely TREK1000 and this component also has some disadvantages such as:

1. Unstable usb port connection.
2. Needs to ensure establishment before operating, which could take a decent amount of time.
3. Sometimes cutting of the connection because of the saturation in the received signal power.

Materials

- 1 UWB Tag
- 4 UWB Anchors
- 1 ST-Link V2 USB Programmer and Debugger
- 1 Computer

Procedure

1. Download and install all necessary software, STM32CubeIDE, STM32 ST-LINK Utility and ST-link driver.
2. After installation open STM32CubeIDE and find "STM32 Project From Existing STM32CubeMX Configuration File (.ioc)".
3. Input a path to the unzipped firmware directory, where the .ioc file is. (In case of the program asking for downloads of additional files, do that)
4. After finish installing the additional files press "Project" in the top bar and select "build project".
5. If everything went right a pop up menu will be shown with zero error and the flashing can be started.
6. Plug the ST-link into the computer.
7. Now open STM32 ST-LINK Utility and setup the connection.
8. Press "file" open the binary files that needs to be loaded onto the DWM1000, then a pop up will appear.
9. click "ok" and the flashing it done.

Test setup

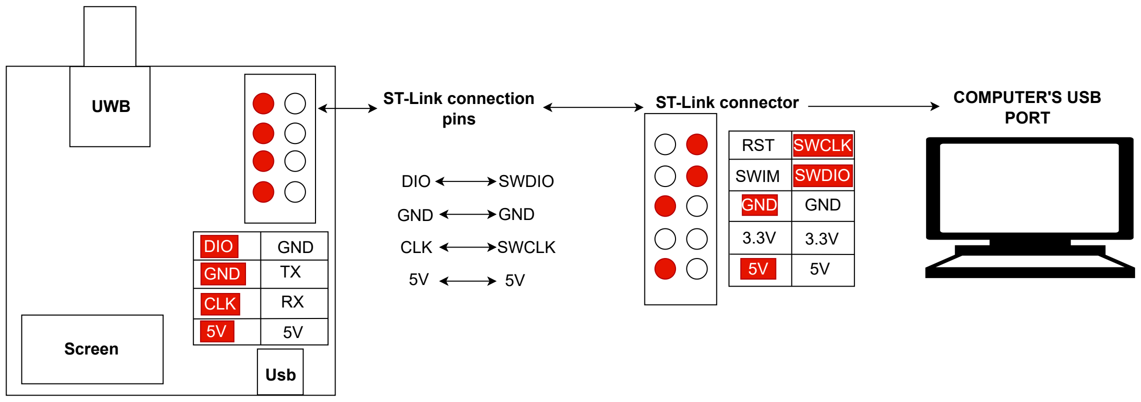


Figure C.1: The setup to enable flashing

Results

After flashing the software the unwanted happen, the new program neither work nor would the display that initially worked show anything. The incident made the project-group unable to do continue work, before gathering a new set of DWM1000, which was provided by the university later on.

Appendix D

D.1 Vicon lab Experiment

Materials

- Deepcar
- 1 UWB Tag
- 4 UWB Anchors
- 4 Powerbanks
- 1 USB cable
- 1 Laptop
- 4 sponges (elevation purpose)
- 1 metal plate of size 80.5 cm (L) x 63 cm (W) x 3 mm (T).
- 1 metal plate of size 77 cm (L) x 74 cm (W) x 2 mm (T).

Procedure

1. The trilateration program should be configured as to calculated the position of the Deepcar.
2. All anchors are connected to powerbanks and placed corresponding to a respective position.
3. The tag is placed on the Deepcar and is connected with a USB cable, this cable is extented with an active USB cable and attached to the computer.
4. Place the Deepcar at a starting position before operating.
5. The Deepcar should be turn on and the wifi-connection to the Deepcar should be established.
6. Configure Deepcar with an application called "Xshell", then setup the link between "Xshell" and "main.exe".
7. Open "main.exe". In "main.exe" maneuver the car in a straight line and gather the data.
8. When reaching the final position, stop the trilateration program.

Appendix E

E.1 LOS Experiment

Materials

- Deepcar
- 1 UWB Tag
- 4 UWB Anchors
- 4 Powerbanks
- 1 USB cable
- 1 Laptop
- 1 Active USB extension cable
- 4 sponges (elevation purpose)
- 1 metal plate of size 80.5 cm (L) x 63 cm (W) x 3mm (T).
- 1 metal plate of size 77 cm (L) x 74 cm (W) x 2 mm (T).

Procedure

1. The trilateration program should be configured as to calculated the position of the Deepcar.
2. All anchors are connected to powerbanks and placed corresponding to a respective position.
3. The tag is placed on the Deepcar and is connected with a USB cable, this cable is extented with an active USB cable and attached to the computer.
4. Place the Deepcar at a starting position before operating.
5. The Deepcar should be turn on and the wifi-connection to the Deepcar should be established.
6. Configure Deepcar with an application called "Xshell", then setup the link between "Xshell" and "main.exe".
7. Open "main.exe". In "main.exe" maneuver the car in a straight line and gather the data.
8. When reaching the final position, stop the trilateration program.

Appendix F

F.1 Weak NLOS Experiment 1

Materials

- Deepcar
- 1 UWB Tag
- 4 UWB Anchors
- 4 Powerbanks
- 1 USB cable
- 1 Laptop
- 1 Active USB extension cable
- 4 sponges (elevation purpose)
- 1 metal plate of size 80.5 cm (L) x 63 cm (W) x 3mm (T).
- 1 metal plate of size 77 cm (L) x 74 cm (W) x 2 mm (T).

Procedure

1. The trilateration program should be configured as to calculated the position of the Deepcar.
2. All anchors are connected to powerbanks and placed corresponding to a respective position.
3. Place the metal plates in front of anchor A_2.
4. The tag is placed on the Deepcar and is connected with a USB cable, this cable is extented with an active USB cable and attached to the computer.
5. Place the Deepcar at a starting position before operating.
6. The Deepcar should be turn on and the wifi-connection to the Deepcar should be established.
7. Configure Deepcar with an application called "Xshell", then setup the link between "Xshell" and "main.exe".
8. Open "main.exe". In "main.exe" maneuver the car in a straight line and gather the data.
9. When reaching the final position, stop the trilateration program.

Appendix G

G.1 Multi-lateration Experiment

Materials

- Deepcar
- 2 UWB Tag
- 5 UWB Anchors
- 5 Powerbanks
- 2 USB cable
- 1 Laptop
- 2 Active USB extension cable
- 5 sponges (elevation purpose)
- 1 metal plate of size 80.5 cm (L) x 63 cm (W) x 3mm (T).
- 1 metal plate of size 77 cm (L) x 74 cm (W) x 2 mm (T).

Procedure

1. The multi-trilateration program should be configured as to calculated the position of the Deepcar.
2. All anchors are connected to powerbanks and placed corresponding to a respective position.
3. Place the metal plates in front of anchor A_2.
4. The tags are placed on the Deepcar and connected with USB cables, these cables are extented with an active USB cable and attached to the computer.
5. Place the Deepcar at a starting position before operating.
6. The Deepcar should be turn on and the wifi-connection to the Deepcar should be established.
7. Configure Deepcar with an application called "Xshell", then setup the link between "Xshell" and "main.exe".
8. Open "main.exe". In "main.exe" maneuver the car in a straight line and gather the data.
9. When reaching the final position, stop the multi-trilateration program.

Appendix H

H.1 Strong NLOS Experiment 2

Materials

- Deepcar
- 1 UWB Tag
- 4 UWB Anchors
- 4 Powerbanks
- 1 USB cable
- 1 Laptop
- 1 Active USB extension cable
- 4 sponges (elevation purpose)
- 1 metal plate of size 80.5 cm (L) x 63 cm (W) x 3mm (T).
- 1 metal plate of size 77 cm (L) x 74 cm (W) x 2 mm (T).

Procedure

1. The trilateration program should be configured as to calculated the position of the Deepcar.
2. All anchors are connected to powerbanks and placed corresponding to a respective position.
3. Place the metal plates in an L shape to cover most possible angles of the sensor A_2.
4. The tag is placed on the Deepcar and is connected with a USB cable, this cable is extented with an active USB cable and attached to the computer.
5. Place the Deepcar at a starting position before operating.
6. The Deepcar should be turn on and the wifi-connection to the Deepcar should be established.
7. Configure Deepcar with an application called "Xshell", then setup the link between "Xshell" and "main.exe".
8. Open "main.exe". In "main.exe" maneuver the car in a straight line and gather the data.
9. When reaching the final position, stop the trilateration program.

Appendix I

I.1 Transition from LoS to NLoS Experiment

Materials

- Deepcar
- 1 UWB Tag
- 4 UWB Anchors
- 4 Powerbanks
- 1 USB cable
- 1 Laptop
- 1 Active USB extension cable
- 4 sponges (elevation purpose)
- 1 metal plate of size 80.5 cm (L) x 63 cm (W) x 3mm (T).
- 1 metal plate of size 77 cm (L) x 74 cm (W) x 2 mm (T).

Procedure

1. The trilateration program should be configured as to calculated the position of the Deepcar.
2. All anchors are connected to powerbanks and placed corresponding to a respective position.
3. The tag is placed on the Deepcar and is connected with a USB cable, this cable is extented with an active USB cable and attached to the computer.
4. Place the Deepcar at a starting position before operating.
5. The Deepcar should be turn on and the wifi-connection to the Deepcar should be established.
6. Configure Deepcar with an application called "Xshell", then setup the link between "Xshell" and "main.exe".
7. Open "main.exe". In "main.exe" maneuver the car in a straight line and gather the data.
8. When reaching a specified distance cover one anchor simulating the scenario of NLOS, this transition should be done continuously.
9. When reaching the final position, stop the trilateration program.

Appendix J

J.1 State space model

Purpose

As the project concerns the localization of the Deepcar, there is utilize an Unscented kalman filter for prediction of the location, and for this purpose the state space model is derived from the following experiment.

Materials

- 1 Deepcar.
- 1 Ruler.
- 1 Markers.
- 1 Stopwatch.
- 1 Computer.

Procedure

1. Mark a start position and place the Deepcar.
2. Turn on the Deepcar and connect to the respective Wifi.
3. Open the application "Xshell" and configure it to the program "main.exe".
4. Open "main.exe" and control the trajectory, here a straight path is choosen.
5. As Deepcar takes clicks as input from "main.exe", the clicks is choosen to have an interval of 5 seconds between every click, and having in total 20 iteration.
6. When reaching the last iteration corresponding to the final/stop position the car is turn off and the stopwatch is also stop.
7. Mark the final destination.

Test setup

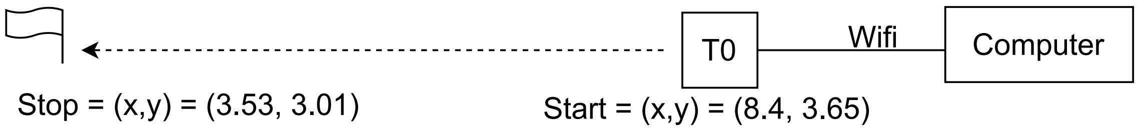


Figure J.1: Experiment in order to obtain the state space model

Results

The result obtain for the state space is use to calculate like following as this could be use in the case of UKF.

Distance calculation:

$\Delta x = 8.41\text{m} - 3.53\text{m} = 4.88\text{m}$
$\Delta y = 3.65\text{m} - 3.01\text{m} = 0.64\text{m}$

where
Initial position in x = 8.41 m.
Initial position in y = 3.65 m.
Final position in x = 3.53 m.
Final position in y = 3.01 m.

Velocity calculation:

$V_{\text{x}} = \frac{4.88m}{105sec} = 4.88\text{m/s}$
$V_{\text{y}} = \frac{0.64m}{105sec} = 4.88\text{m/s}$

where
Total travel time = 105 seconds.
Iterations (clicks) = 20.

Heading angle:

$\theta = \tan^{-1}(\frac{\Delta y}{\Delta x}) = \tan^{-1}(\frac{-0.64\text{m}}{-4.88\text{m}}) = 7.4715°$

Bibliography

[1] Element logic står bag danmarks største robotlager. Proshop. [Online]. Available: https://www.elementlogic.dk/element-logic-star-bag-danmarks-storste-robotlager/

[2] Amazon details new warehouse robots, 'ernie' and 'bert'. Amazon. [Online]. Available: https://www.press.bmwgroup.com/global/article/detail/T0308393EN/bmw-group-is-making-logistics-robots-faster-and-smarter?language=en

[3] Bmw group is making logistics robots faster and smarter. BWM. [Online]. Available: https://www.press.bmwgroup.com/global/article/detail/T0308393EN/bmw-group-is-making-logistics-robots-faster-and-smarter?language=en

[4] X. Cai, P. Wang, L. Du, Z. Cui, W. Zhang, and J. Chen, "Multi-objective three-dimensional dv-hop localization algorithm with nsga-ii," IEEE Sensors Journal, vol. 19, no. 21, pp. 10 003–10 015, 2019.

[5] P. Wang, J. Huang, Z. Cui, L. Xie, and J. Chen, "A gaussian error correction multi-objective positioning model with nsga-ii," Concurrency and Computation: Practice and Experience, vol. 32, no. 5, p. e5464, 2020.

[6] Z. Cui, X. Xu, X. Fei, X. Cai, Y. Cao, W. Zhang, and J. Chen, "Personalized recommendation system based on collaborative filtering for iot scenarios," IEEE Transactions on Services Computing, vol. 13, no. 4, pp. 685–695, 2020.

[7] Sewio. Rtls technology comparison indoor tracking technologies comparison. Sewio. [Online]. Available: https://www.sewio.net/uwb-technology/rtls-technology-comparison/

[8] I. Guvenc, C.-C. Chong, and F. Watanabe, "Nlos identification and mitigation for uwb localization systems," in 2007 IEEE Wireless Communications and Networking Conference, 2007, pp. 1571–1576.

[9] D.-H. Kim, G.-R. Kwon, J.-Y. Pyun, and J.-W. Kim, "Nlos identification in uwb channel for indoor positioning," in 2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC), 2018, pp. 1–4.

[10] M. Liu, X. Lou, X. Jin, R. Jiang, K. Ye, and S. Wang, "Nlos identification for localization based on the application of uwb," Wireless Personal Communications, vol. 119, no. 4, pp. 3651–3670, 2021.

[11] F. Mazhar, M. G. Khan, and B. Sällberg, "Precise indoor positioning using uwb: A review of methods, algorithms and implementations," Wireless Personal Communications, vol. 97, no. 3, pp. 4467–4491, 2017.

[12] W. Sakpere, M. Adeyeye-Oshin, and N. B. Mlitwa, "A state-of-the-art survey of indoor positioning and navigation systems and technologies," South African Computer Journal, vol. 29, no. 3, pp. 145–197, 2017.

[13] L. Zhang, T. Zhou, and B. Lian, "Integrated imu with faster r-cnn aided visual measurements from ip cameras for indoor positioning," Sensors, vol. 18, no. 9, p. 3134, 2018.

[14] J. B. Kristensen, M. M. Ginard, O. K. Jensen, and M. Shen, "Non-line-of-sight identification for uwb indoor positioning systems using support vector machines," in 2019 IEEE MTT-S International Wireless Symposium (IWS). IEEE, 2019, pp. 1–3.

[15] Y. Y. Y. TaoyunZhou, MinleiXiao, "Research on indoor uwb positioning based on expectation maximization in nlos environment," Wiley, 18 January 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/epdf/10.1002/cpe.6278

[16] Decawave. Channel effects on communications range and time stamp accuracy in dw1000 based systems. Decawave. [Online]. Available: https://www.decawave.com/wp-content/uploads/2018/10/APS006_Part-1-Channel-Effects-on-Range-Accuracy_v1.03.pdf

[17] M. H. Ian Oppermann and J. Iinatti, uwb theory and applications. Wiley, 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.466.292&rep=rep1&type=pdf

[18] N. I. Fofana, A. Van den Bossche, R. Dalcé, and T. Val, "An original correction method for indoor ultra wide band ranging-based localisation system," in International Conference on Ad-Hoc Networks and Wireless. Springer, 2016, pp. 79–92.

[19] K. J.-R. T. V. Francois Despaux, Adrien van den Bossche, "N-twr: An accurate time-of-flight-based n-ary ranging protocol for ultra-wide band," 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02316841/document

[20] Z. Hua, , University of Waterloo. [Online]. Available: https://www.youtube.com/watch?v=fyAZszlPphs&list=PLehuLRPyt1Hyi78UOkMPWCGRxGcA9NVOE&ab_channel=DataScienceCourses.

[23] TREK 1000 User Manual, Towards Data Science Ltd. [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

[24] C. M. Bishop, Pattern recongition and Machine learning. Springer Science+Business Media.

[25] The A-Z guide to Support Vector Machine, Analyticsvidhya. [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

[26] K-nearest neighbour: The distance-based machine learning algorithm. Shivam Sharma. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/05/knn-the-distance-based-machine-learning-algorithm/

[27] Group 833 Aalborg University 2021, Fault Detection in a Supermarket Refrigeration System using Machine Learning. [Online]. Available: https://drive.google.com/file/d/1XdnRsg30DX5sWfK0r7DRNhQ1vQoSGxYV/view?usp=sharing,

[28] C. M. Bishop, Pattern recognition and machine learning. springer, 2006.

[29] Machine Learning Fundamentals: Bias and Variance, youtube. [Online]. Available: https://www.youtube.com/watch?v=EuBBz3bI-aA&t=3s&ab_channel=StatQuestwithJoshStarmer.

[30] True Error vs Sample Error, GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/true-error-vs-sample-error/.

[31] M. S. Grewal and A. P. Andrews, Kalman filtering: Theory and Practice with MATLAB. John Wiley & Sons, 2014.

[32] Kf explaination. intechopen. [Online]. Available: https://www.intechopen.com/chapters/63164

[33] Kf steps. towardsdatascience. [Online]. Available: https://towardsdatascience.com/an-intro-to-kalman-filters-for-autonomous-vehicles-f43dd2e2004b

[34] L. D'Alfonso, W. Lucia, P. Muraca, and P. Pugliese, "Mobile robot localization via ekf and ukf: A comparison based on real data," Robotics and Autonomous Systems, vol. 74, pp. 122–127, 2015.

[35] H. Jiezhong, "What is the difference between ekf and ukf?" 12 2019.

[36] P. Pasek and P. Kaniewski, "Unscented kalman filter application in personal navigation," in Radioelectronic Systems Conference 2019, vol. 11442. International Society for Optics and Photonics, 2020, p. 114421C.

[37] Ukf steps. Udacity. [Online]. Available: https://classroom.udacity.com/nanodegrees/nd013/parts/40f38239-66b6-46ec-ae68-03afd8a601c8/modules/0949fca6-b379-42af-a919-ee50aa304e6a/lessons/daf3dee8-7117-48e8-a27a-fc4769d2b954/concepts/3f67dc00-84f0-442b-88bc-1edf7b69caa7

[38] TREK 1000 User Manual, Decawave Ltd. [Online]. Available: https://www.decawave.com/wp-content/uploads/2018/09/trek1000_user_manual.pdf

[39] Trek picture. Decawave Ltd. [Online]. Available: https://www.eenewseurope.com/news/indoor-ultra-wideband-positioninglocating-kits-distribution

[40] Dwm1000. Decawave Ltd. [Online]. Available: https://www.decawave.com/sites/default/files/resources/DWM1000-Datasheet-V1.6.pdf

[41] DW 1000 Metrics for Estimation of Non Line of Sight Operating Conditions, Decawave Ltd. [Online]. Available: https://www.decawave.com/wp-content/uploads/2018/10/APS006_Part-3-DW1000-Diagnostics-for-NLOS-Channels_v1.1.pdf

[42] T. Fawcett, "An introduction to roc analysis," Pattern recognition letters, vol. 27, no. 8, pp. 861–874, 2006.

[43] performance plot. Mathworks. [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/plotperform.html;jsessionid=07754f43ab2fc3d0c48f551b2325

[44] error histogram. Mathworks. [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/ploterrhist.html

[45] parallel plot. Mathworks. [Online]. Available: https://www.mathworks.com/help/stats/parallelcoords.html

[46] F. Kurniawan, "Mobile robot localization via unscented kalman filter," Researchgate, December 2019. [Online]. Available: https://www.researchgate.net/publication/349045014_Mobile_Robot_Localization_via_Unscented_Kalman_Filter

[47] O. A. Bridget Nkemnole1, "Estimation of geometric brownian motion model with a t-distribution–based particle filter," Researchgate, December 2019. [Online]. Available: https://www.researchgate.net/publication/331267832_Estimation_of_geometric_Brownian_motion_model_with_a_t-distribution-based_particle_filter

[48] P. Schwarzbach, R. Weber, and O. Michler, "Statistical evaluation and synthetic generation of ultra-wideband distance measurements for indoor positioning systems," IEEE Sensors Journal, 2021.