Kungliga Tekniska Högskolan

Project in Wireless Communication

Final Report

Wireless Based Positioning

Authors:
Adrien Anxionnat
Baptiste Cavarec
Irlon Santos
Navneet Agrawal
Raees Kizhakkumkara Muhamad
Yuqi Zhang

Mentor:
Satyam Dwivedi

May 26, 2016

Contents

1 Acknowledgment **4**

2 Introduction **4**
2.1 Background 4

3 Problem Specification **6**
3.1 Objective 6
3.2 Overview 6
3.3 Hardware and Software specifications 7

4 Project Management **9**
4.1 Team 9
4.2 Initiation and planning 9
4.3 Execution 11
4.4 Monitoring and Control 11
4.5 Ending phase 13

5 Theory **14**
5.1 Ranging 14
5.1.1 Two-way Ranging 15
5.1.2 Symmetric Two-way Ranging 17
5.2 Positioning 18
5.2.1 Static Positioning 19
5.2.2 Dynamic Positioning 22

6 Development **26**
6.1 C Code 26
6.1.1 Two Way Ranging 26
6.2 MATLAB scripts 33
6.3 Prototype 37

7 Results **39**
7.1 Positioning 39
7.1.1 Simulation results 39
7.1.2 Experiment setup 45
7.1.3 Initialization 46
7.1.4 Static positioning results 48

7.1.5 Dynamic positioning results 52
7.2 Sources of error 53
7.2.1 Dependency on the clock system of each anchor 55
7.2.2 Dependency on the environment and the range itself 60

8 Conclusion **65**

List of Figures

1 System design used on the project. 7
2 EVB1000 from decaWare's EVK1000. 7
3 Project Gantt chart time plan 12
4 Scheme of two-way ranging. 16
5 Scheme of symmetric two-way ranging. 17
6 (a)Trilateration from exact measurements (b)Trilateration from noisy measurements 19
7 Two-way ranging state machine 28
8 Matlab function flow for Static positioning 35
9 Matlab function flow for Dynamic positioning 36
10 Start screen of the user interface on final prototype. 37
11 User interface after running the dynamic algorithm for symmetric two-way ranging with three anchors. 38
12 Effect of condition number on convergence 39
13 Condition number of operation area 40
14 Parameter Initialization Error 41
15 Dynamic Positioning Simulation for straight line path 42
16 Dynamic Positioning Simulation for circular path 43
17 Room B230 setup for experimentation 44
18 Measurement points and path used for experiments 45
19 Performance of TWR for different configuration values 47
20 Performance of SDS for entire configuration values 47
21 Performance of SDS for low step size 48
22 Performance of SDS for optimal configurations 49
23 Convergence analysis for TWR 49
24 Convergence for SDS 50

25 Convergence SDS vs TWR in time 50
26 Heatmap for TWR without any correction being performed . . 51
27 Heatmap for TWR with correction 52
28 Heatmap for TWR with correction 53
29 Heatmap for TWR with correction for each coordinate 54
30 Heatmap for STWR without any correction being performed . 55
31 Heatmap for STWR with correction 57
32 Heatmap for STWR with correction for each coordinate 58
33 Symmetric Two way ranging with Dynamic positioning. 59
34 Two way ranging with Dynamic positioning. 59
35 Ranging error in two different environment 60
36 Ranging bias in function of t_d for anchor 2 61
37 Studying of the crystal frequency in function of time 61
38 Heatmap of TWR range bias for anchor 1 62
39 Heatmap of TWR range bias for anchor 2 62
40 Heatmap of TWR range bias for anchor 3 63
41 Heatmap of range bias for all anchors 63

List of Tables

1 List of properties configured 34
2 Studying of the range bias for each anchor 55
3 Comparison between the relative bias measured and the clock related bias 57

1 Acknowledgment

We would like express our deepest gratitude towards all those who helped in realizing the goals of this project. We really appreciate the help by our mentor Satyam Dwivedi. His guidance helped us learn and implement ideas in our project. A special thanks goes to all members from other project teams, who were so kind to help us in our elaborate experiments.

2 Introduction

This document present the details of the project, entitled Wireless Based Positioning. Mentors for this project are Research scholar Satyam Dwivedi and Prof. Magnus Jansson from Electrical Engineering school at KTH. We are a team of six members.

The report is divided in following way. In the next section (Sec: 3), we will describe the problem tackled in this project in greater details. Sec: 4 deals with the management of the project. Further, in Sec: 5, we explain the theoretical background for this project and discuss various algorithms implemented. Sec: 6 deals with the implementation and development work related to this project. Next, Sec: 7 describes different results and analysis, and sources of error in all implemented algorithms and methods. In the end, we conclude our project with Sec: 8.

2.1 Background

Location awareness is of great benefit to a rich set of applications both in indoor and outdoor environments. In such techniques determining the location of a device or a person is a fundamental problem, and its importance has led to the development of positioning systems. However, due to the poor signal strength of the widely used Global Positioning System (GPS) in the indoor environment or other special environment settings, GPS systems can not always work and many other positioning techniques are pursued to broaden the application of location based services.

Since last decade, interests in the research of localization systems has skyrocketed. Localization systems such as Cricket and Whistle systems are

quite famous. Although these systems have a fantastic market potential, there are still some problems with the existing solutions as follows,

- Location precision is not good enough.
- Multi-path interference and environment noises make some systems become heavily environment and hardware dependent.
- High power consumption and Price.

However, since the ruling of Federal Communications Commission(FCC) in the United States to open up the spectrum from 3.1-10.6 GHz for ultra wideband (UWB) applications in 2002, a quality of UWB localization applications emerge in industry and academic community. These eclectic mix contains localization based tracking services for factory automation, healthcare, safety/security and warehouse & logistics. There are two significant advantages of UWB. One that it allows very high precision measurement of the time of flight of the signal from transmitter to receiver. This can lead to a very accurate distance measurement thus an accurate positioning system. The other advantage is that UWB has the ability to allow direct path to be identified even if it is severely attenuated.

In this project, we will be using Decawave's DW1000 radio capable of working in UWB to give a high precision (centimeter scale) measurements. This radio is mounted on an evaluation board EVK1000 (ARM based). DW1000 is one of the latest devices with good community support and documentation.

3 Problem Specification

This project deals with the specific problem of wireless positioning, positioning in an indoor environment with an accuracy within decimeters from the actual point. Detailed specification of the problem are discussed in sections following.

3.1 Objective

The main objective of this project is to design a wireless positioning system capable of calculating the position to an accuracy within decimeter precision. The objectives can be further divided as follows:

- Positioning with accuracy within decimeters from actual point.
- Positioning relative to three or more than three fixed anchor nodes.
- Implement algorithms for static and dynamic positioning scenarios.
- Design and develop a positioning system using Decawave DW1000 UWB radio and EVB1000 evaluation board.

3.2 Overview

The basic design for a positioning system consists of using the location of three base stations (or anchors) to calculate the position of a mobile station (named as tag on this project). Further details about positioning and ranging can be found in Sec: 5.

Devices (EVB1000 evaluation board + DW1000 radio) consists of a radio system and a microprocessor (Sec: 3.3). In our system (Fig: 1), the mobile device will communicate with each of the anchor devices to calculate the range measurement from respective anchor nodes. The mobile device will be connected to a computer running MATLAB and will feed the ranging data into MATLAB constantly. The final position of the mobile device will be calculated by an algorithm that will be running on MATLAB. Finally, the results will be displayed in a user-friendly interface with a map showing location of various entities in the system.

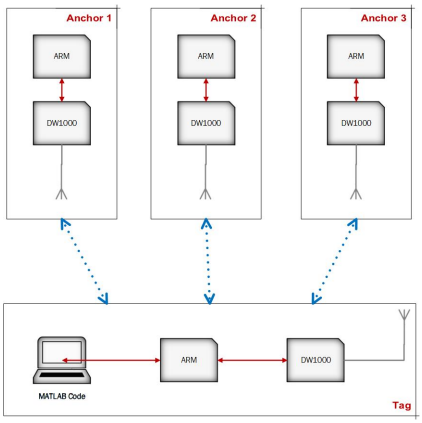

Figure 1: System design used on the project.

3.3 Hardware and Software specifications

The stations used are part of an evaluation kit (EVK1000) from decaWave. The kit, as seen on figure 2, contains two antennas, two micro USB cable 2.0, two perspex stands and two evaluation boards names EVB1000.

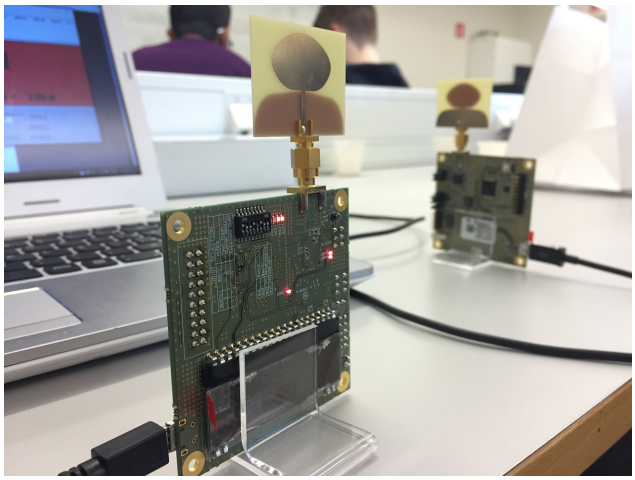

Figure 2: EVB1000 from decaWare's EVK1000.

The boards are built with a microprocessor ARM STM32F105, USB port for external connection, dip switch keys for user interaction, among other

components. More details can be found on the manual [2].

The radio integrated circuit built on the EVB1000 is a wireless transceiver called DW1000 [3]. It follows the IEEE Standard 802.15.4 - 2011 [6]. It uses Ultra-Wide Band technique, with the supported band from 3.5 GHz up to 6.5 GHz, transmit power of -14dBm or -10dBm and BPSK Modulation.

Following list presents some information about the hardware and software used in this project.

Radio unit : Decawave DW1000 [3]

- Supported Band : from 3.5 GHz to 6.5 GHz
- Transmit power : -14dBm or -10 dBm
- Modulation : BMP with BPSK
- SPI interface to host controller
- IEEE standard : 802.15.4 - 2011

Board & Processor: EVK1000, ARM [2]

- Microprocessor model : ARM STM32F105
- Microprocessor core : ARM® 32-bit Cortex® -M3 CPU
- I/O ports : Up to 80 I/O ports
- USB port from external control
- Power source : DC +3.6V to +5.5V

Software

- Programming language : C
- IDE : CooCox IDE

4 Project Management

The project is a part of the course Project in Wireless Communication, offered by KTH from March 16th, 2016 until May 27th 2016. The group was created by the course responsible based on personal and academic information.

4.1 Team

The team is consisting of six masters students from electrical engineerin. Team member's roles are provided below.

Name	**Responsibility**
Satyam Dwivedi	Project mentor
Adrien Anxionnat	Error analyst and code developer
Baptiste Cavarec	Coding expert and designer
Irlon Santos	Presentation and video in-charge
Navneet Agrawal	Project Manager and MATLAB developer
Raees Muhamad	Theorist and MATLAB developer
Zhang Yuqi	Code developer, focus: state machine

4.2 Initiation and planning

The first step in project management is to analyze the resources and prepare a plan to achieve project goals. During the first week of work, we analyzed our resource, team strengths and prepared a gantt chart to plan the 9 weeks duration of the project. The Gantt chart is shown in figure 3.

The project implementation has two main distinct field - Coding and Theory. Hence we divided two teams to focus on each part. Coding group includes - Baptiste, Yuqi and Adrien, while theory group includes - Irlon, Navneet and Raees. The group division is indeed loosely defined because the actual tasks performed during the project were overlapped and shared by everyone. Although the major responsibilities were assigned based on the group division.

Task division and description

We have divided the task in following fashion:

1. **MATLAB and theory**
   Task encompasses understanding the input and output requirements for device related to implementation of positioning method in MATLAB. That means responsible person should have an overall understanding of theory, specification of device and input measurements to the MATLAB program. He should also understand the working of "main.c" file where all the input and output parameters are set.

2. **Code Design and development**
   Software architecture and design is essential in our implementation. Responsible person should have a good understanding of working of entire code. He will work in tandem with the entire team to design structure of software and guide others in the implementation of the code. This basically involved almost all layers below the top application layer (main.c). Task in this category involves writing/editing callback functions, structuring messages based on IEEE standards and implementing State machine for communication. Debugging code will also be part of this task group. The person responsible should have a good understanding of C programming.

3. **Project demonstration and presentation**
   The final output of the project is a device that is capable of localizing in indoor space using five/three anchors. This output will be demonstrated using a prototype implementation on MATLAB. Also, to present the achievements of the project to more people, a video will be created, that will include all the important details and a demonstration of the prototype. The person responsible should have an overall understanding of the projects goal and presentation requirements. He should also have some basic knowledge of video editing and MATLAB's presentation functions.

Mid-term target

1. Successful ranging of tag using three anchors based on one of the ranging methods.

Implementation of ranging protocols and state machines on hardware board.

2. Centimeter accurate positioning
   Implementation of non-linear least square estimation methods on position measurements.

End-term target

1. Successful ranging of tag using Five anchors based on both two-way and Symmetric two-way ranging methods.
2. Implementation of both ranging protocols and state machines on hardware board.
3. Centimeter accurate positioning - Static and Dynamic positioning.
4. Implementation of non-linear least square estimation (NLS) and Extended Kalman Filtering methods on position measurements.
5. Live demonstration on Static and Dynamic filtering using five anchors based setup.
6. Project report and other resources compiled and published for future references.

4.3 Execution

The implementation phase ensures that the project management plan's deliverables are executed accordingly. To keep track of the progress and to plan for the next weeks work, we organized weekly team meetings on every Wednesday. Report from this meeting is written and published on the project's web-page.

4.4 Monitoring and Control

For the purpose of monitoring and sharing information within the team and keep version control on the project's development codes, we create a

ISO WEEK NUM	W14	W15	W16	W17	W18	W19	W20	W21	W22
MONTH	**APR**				**MAY**				
(MON)DAY	04	11	18	25	02	09	16	23	30
Getting acknowledged with the theory									
Ranging and positioning algorithm									
Estimation theory and filtering techniques									
Matlab algorithm									
Retrieving the data computed by DecaRange									
Processing the data									
Non-linear least square									
Extensive Kalman Filtering									
Result analysis									
Sources of errors									
Analysis of 3 anchor system									
Analysis of 5 anchor system									
C coding									
Understanding and coding a ranging system									
Implementing a ranging system									
Get distance output on USB interface									
Implementing the positioning protocol									
Two way ranging using 3 anchors									
Debugging									
Symmetric two way ranging using 3 anchors									
Debugging									
Ranging using 5 anchors									
Prototyping									
Real time system									
Implement positioning algorithm on MATLAB									
Graphic feature for the positioning									
Testing in real environement									
Administrative and Management									
Presenting the project									
Reports									
Video									
Homepage									
Communication									
Weekly meeting									

Figure 3: Project Gantt chart time plan

GitHub repository "WirelessPositioning" (`https://github.com/BaptCav/WirelessPositioning.git`). We also created a web page to publish information on KTH web (`https://www.kth.se/social/group/wireless-based-posit/`).

4.5 Ending phase

This is the "Publication" phase of the project. In this phase we prepare reports and archive all the files/documents useful in the project. We finalize all activities across both groups. Moreover, preparations for presentation/demonstration as well as the video are made in this phase of the project. On the final day of project, we will present our results (demonstration and video) to the project course in-charge and other course participants. That will conclude the project.

5 Theory

The project is mainly divided in two parts: ranging and positioning. Those parts combined will create the final system. In this section we will then present these two parts, the first layer to appear being the the ranging, as the ranges are needed in order to do positioning, it will be presented first. Then we will focus on explaining on how to find an approximation of the position knowing these ranges.

5.1 Ranging

The ranging process can also be described as the calculation of the distance between tag and anchor. Three main methods used to get ranging measurements are Received Signal Strength (RSS), Angle of Arrival (AOA) and Time of Flight (TOF). The first one is based on the fact that the amplitude of a signal decreases with the distance in a square proportion. The idea beside AOA is that the wavefront of a radio signal is orthogonal to the direction of propagation. TOF will be discussed in more detail on this document, since it was the chosen method for this project.

Time of Flight is defined as "the time interval between transmission time of an *epoch* to its reception at a distant receiver"[7], being *epoch* a particular instant, like the beginning of a frame.
When developing a radio system for communication, it is mandatory that a transmission frequency is set and known for all terminals, otherwise the communication become impracticable. We know that wavelength λ is proportionally inverse to its wave frequency f. Hence, the speed (v) that a specific wave propagate is constant according to the equation $\lambda = \frac{c}{f}$. For electromagnetic radiation this speed can be considered as the speed of light c.
As the Time of Flight definition implies, it is necessary to measure the interval between the send and receive time of a particular event. It is possible to do an analogy with a transport truck where this truck needs to leave the warehouse, arrive at a specific point and go back to the start point using the same path. Considering the truck has a constant speed v, and it took t seconds for the round trip, the distance (d) can be easily calculated by $2d = v \times t$.

However, this formula can be applied in case the emitted wave is sup-

posed to reflect immediately as is reaches the destination, as in a sonar for instance. On this application, the wave contains a modulated signal with information that need to be processed, thus, the time measured will include the travel time plus the necessary time to process the data. To ensure that the ToF calculation is correct, this time is a delay, purposely added in the code.

Furthermore, the measurements will also include time necessary to travel this distance between the antenna and the microprocessor itself as antenna delay (Ta). This distance must be considered in order to get a better precision on the final result. More information about this on section 6.1.

In our ranging system, two different ranging algorithms, two-way ranging and symmetric two-way ranging, are used to calculate time of flight. As we will discuss later, these two algorithms have distinct performances in accuracy, variance and measurement rate thus can satisfy different application situations.

5.1.1 Two-way Ranging

In our two-way ranging algorithm, two settings are made to support each measurement. First for each anchor, we set a known response delay. Then for tag, it will send Poll messages to anchors periodically to initialize a distance measurement. The detailed scheme of two-way ranging can be found in Fig. 4 and discussed below.

- First tag sends a poll message to the Anchor and record the time stamp of message sending as T_{SP}. If tag does not receive any response message in a next certain time period, it will time out and send another Poll message again.
- Second when anchor receives the Poll message tag sends, it waits for the known response time delay T_d and do some necessary operations. Then it sends a Response message to tag.
- Finally Tag receives the Response message from anchor and record the corresponding time stamp of receiving T_{RR}.

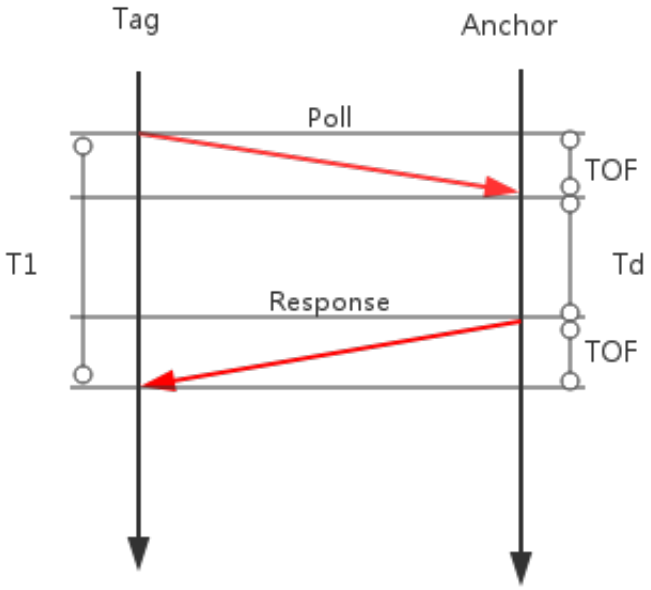

Figure 4: Scheme of two-way ranging.

The time measured within one experiment starting from the sending of the poll message and ending when a response message is received at T_1. All the delays should be subtracted from this time period.

$$\mathbf{TOF} = \frac{(T_1 - T_d)}{2} \tag{1}$$

Eq. 1 is a basic Time of Flight formula for two-way ranging, because the measurement is made based on a single poll-response arrangement. Another thing that need to be noticed is that in our systems, except for the response delay T_d, we also need to subtract the antenna delay T_a as mentioned above. But this delay is out of the scope of ranging algorithm and will be explained in detail in Sec. 7.2.

As discussed in Sec. 7, we get some ranging errors in two-way ranging algorithm because the average error is proportional to the response delay T_d. The average error can be large as T_d varies, that can be a problem if

not correcting in order to achieve a high accuracy positioning system. In order to improve ranging performances and to be less sensitive to the clock misadjustments, we introduce the symmetric two-way ranging process.

5.1.2 Symmetric Two-way Ranging

In addition to Poll message and Response message in former positioning system (Sec. 5.1.1), the symmetric two-way ranging algorithm uses a Final message sent from tag to anchor to decrease average ranging error compared with two-way ranging. The detailed scheme of symmetric two-way ranging is shown in figure 7 and discussed below.

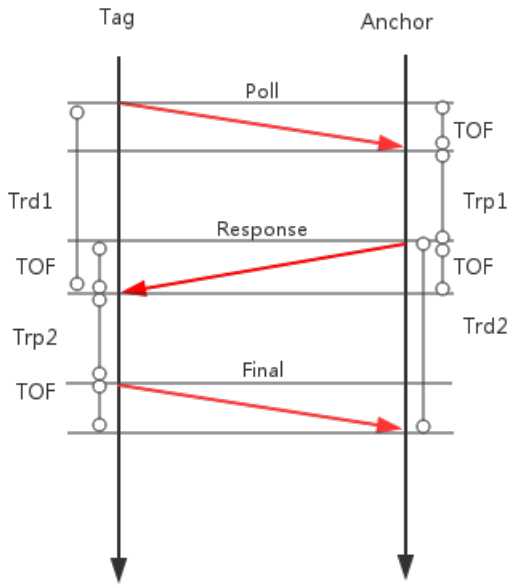

Figure 5: Scheme of symmetric two-way ranging.

- First tag sends a Poll message to the Anchor and record the time stamp of Poll sending as T_{SP}. If tag does not receive any Response message in a next certain time period, it will time out and send another Poll message again.

- Second when anchor receives the Poll message tag sends, it records the time stamp of receiving T_{RP}. Then it sends a Response message to tag and once more, record the time stamp of Response sending as T_{SR}. If anchor does not receive any Final message in a next certain time period, it times out and wait for a new Poll message again.

- Next, tag receives the Response message from anchor and record the corresponding time stamp of receiving T_{RR}. Then it waits for a delay time period and send a Final message where time stamp T_{RR} is attached to anchor. Notice that this delay time period is set by users so it is known both at tag and anchors.

- Finally when the Anchor receive the Final message, it extracts T_{RR} from the message and calculate time of fight. We define some notations as follows.

$$T_{rd1} = T_{RR} - T_{SP} \qquad T_{rp1} = T_{SR} - T_{RP} \tag{2}$$

$$T_{rd2} = T_{RF} - T_{SR} \qquad T_{rp2} = T_{SF} - T_{RR} \tag{3}$$

Another difference between the two-way ranging algorithm and symmetric two-way ranging algorithm is that in symmetric two-way ranging algorithm, time of flight is calculated at anchors. In order to run position algorithm at tag, every anchor should send distance between tag and itself which is calculated in last measurement to tag through Response message in current measurement.

Next we are able to use all the time stamps we get and the notations we define to derive the formula of TOF calculation in symmetric two-way ranging algorithm. Let's assume that **TOF** keeps the same in one measurement. According to the symmetrical property shown in figure 7, we have following range calculation equation[8].

$$\mathbf{TOF} = \frac{T_{rd2} \times T_{rd1} - T_{rp2} \times T_{rp1}}{T_{rd1} + T_{rd2} + T_{rp1} + T_{rp2}} \tag{4}$$

5.2 Positioning

The positioning stage is the last stage of the application and deals with estimating the position of the tag by using the measurements supplied by

the ranging stage. On these measurements we have to do trilateration i.e the method of estimating the position of a point given the distances of the point from three other known points. Geometrically it can be seen that the solution is the intersection of three spheres having these distances as radius and centered around the three points respectively. It must be mentioned here that we have two solutions having the same set of distances from the points(the actual tag position and its mirror image in the plane of the three anchors).Therefore in the solution its always assumed that the tag is below the plane. As indicated in the figure 6 the positioning algorithm uses the

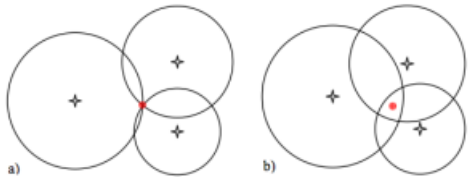

Figure 6: (a)Trilateration from exact measurements (b)Trilateration from noisy measurements

information in the distance measurements to arrive at an estimate that strives to minimize the error in the mean square sense. In our solution we assume the noise in the measurements to be Additive White Gaussian Noise. The positioning algorithm has been split into two cases, namely static positioning and dynamic positioning based on whether the tag is stationary or not. This was done because there exists an optimal solution (minimizing the mean square error) to the positioning problem assuming the tag is stationary while an implementable optimal solution for the dynamic case is not available. The solution implemented for the dynamic case would certainly work when the tag is stationary it will definitely be sub-optimal compared to the static positioning algorithm.

5.2.1 Static Positioning

Static Positioning is performed when we estimate a location of a stationary tag. The tag continuously receives the distance measurements from the three anchors. In this case we should be able to estimate the position at any time by using all the measurements received until that time. Since the distance between the tag and the anchors and the tag's position are related in a non-linear way, a non-linear weighted (weighting each anchor measurement) least

squares algorithm on data batches has been proven to be the optimal solution (in euclidian distance) by [9].

The algorithm used reduces the problem to finding the minimum of a quadratic cost function. By decomposing the cost function minimization problem into two steps we arrive at a solution that is optimal and recursive [9]. The first step is achieved using a Kalman filter while the second step uses an iterative Gauss-Newton algorithm.

5.2.1.1 Cost function

In order to describe the quadratic cost function used we first define the system as a vector s_k pointing to the position of the tag at any time instant t_k.

$$s_k{}^T = \begin{bmatrix} x_k & y_k & z_k \end{bmatrix}$$

Since the tag is stationary we can define the state of the system by:

$$s_{k+1} = s_k \tag{5}$$

Let z_k refers to the vector containing measurements of the three anchors at a time step t_k .

$$z_k = \begin{bmatrix} (m_{a1}(t_k))^2 & (m_{a2}(t_k))^2 & (m_{a3}(t_k))^2 \end{bmatrix}$$

Let $f(s_k)$ is a function which returns a vector containing the actual distance between the tag and the three anchors and is given by:

$$f(s_k)^T = \begin{bmatrix} (x_k - x_{a1})^2 + (y_k - y_{a1})^2 + (z_k - z_{a1})^2 \\ (x_k - x_{a2})^2 + (y_k - y_{a2})^2 + (z_k - z_{a2})^2 \\ (x_k - x_{a3})^2 + (y_k - y_{a3})^2 + (z_k - z_{a3})^2 \end{bmatrix}$$

The error in the measurements at a time step t_k be given by:

$$e_k = \begin{bmatrix} e_{a1}(t_k) & e_{a2}(t_k) & e_{a3}(t_k) \end{bmatrix}$$

The measurements received any time instant are related with the state and the measurement noise e_k by 6.

$$z_k = f(s_k) + e_k \tag{6}$$

We then can combine all the vectors from all time steps untill the current time step t_n to obtain the following matrices

$$Z_n{}^T = \begin{bmatrix} z_1 & z_2 & \dots & z_n \end{bmatrix}$$

$$F(s_n)^T = \begin{bmatrix} f(s_1) & f(s_2) & \dots & f(s_n) \end{bmatrix}$$
$$E_n{}^T = \begin{bmatrix} e_1 & e_2 & \dots & e_n \end{bmatrix}$$

Therefore we can write 7

$$Z_k = F(s_k) + E_k \tag{7}$$

Using this description we will be able to describe the optimality of our estimate with respect to a cost function J_n defined over all measurements till time step t_n. Using a minimum mean square error criterion we are able to arrive at the following cost function which gives a measure of this error where R is the weighing matrix containing the variance of the measurement of the three anchors[9] .

$$J = \frac{1}{2}(Z_n - F(s_n))^T R^{-1}(Z_n - F(s_n)) \tag{8}$$

The static positioning algorithm tries to find an estimate which minimizes 8

$$\frac{\partial J}{\partial s}|_{s=\widehat{s}} = 0 \tag{9}$$

this is achieved in two steps, a linear first step where we estimate non-linear combinations of the first step using a linear least square fit and a non-linear second step which uses a non-linear squares fit of the first step estimates to obtain an estimate of the unknown state.

5.2.1.2 First Step Optimization

Let Y_n and y_k represent $F(s_n)$ and $f(s_k)$ respectively and rewrite 6.

$$Z_n = Y_n + E_n \tag{10}$$

A new cost function in terms of Y_n can be minimized with with respect to Y_n. This cost function is:

$$J_y = \frac{1}{2}(Z_n - Y_n)^T R^{-1}(Z_n - Y_n) \tag{11}$$

We now find a estimate $\widehat{Y_n}$ which minimizes this cost function. Note that this cost function is linear with respect to Y_n. The weighted least squares batch solution is the optimal solution and it can be shown that the Kalman filter is a recursive,iterative equivalent [9]. The Kalman filter would have only a

measurement step as the state of the system is stationary. The estimate of y_k is given by 12

$$\widehat{y}_{k+1} = \widehat{y}_k + P_k \cdot R^{-1} \cdot (z_k - \widehat{y}_k) \tag{12}$$

$$P_{k+1} = ({P_k}^{-1} + R^{-1})^{-1} \tag{13}$$

5.2.1.3 Second Step Optimization

The estimates of the first step are used as measurements in the second step.

$$\widehat{Y}_n = F(s) + V_n \tag{14}$$

V_n is defined as the 'measurement noise' which covariance matrix is given by P_n, i.e. the covariance of the first step states. The second step cost function to minimize becomes:

$$J_s = \frac{1}{2}(\widehat{Y}_n - F(s))^T P_n^{-1}(\widehat{Y}_n - F(s)) \tag{15}$$

The estimate of the position for the time instant t_k at the s_k is given by 16

$$\widehat{s}_k = \widehat{s}_{k-1} - \Delta \cdot H_{k-1}^{-1} \cdot r(\widehat{s}_{k-1}) \tag{16}$$

Where H_{k-1}^{-1} is the Jacobian of the function $f(\widehat{s}_{k-1})$

$$H_{k-1} = \begin{bmatrix} \hat{px}_{k-1} - x_{a1} & \hat{py}_{k-1} - y_{a1} & \hat{pz}_{k-1} - z_{a1} \\ \hat{px}_{k-1} - x_{a2} & \hat{py}_{k-1} - y_{a2} & \hat{pz}_{k-1} - z_{a2} \\ \hat{px}_{k-1} - x_{a3} & \hat{py}_{k-1} - y_{a3} & \hat{pz}_{k-1} - z_{a3} \end{bmatrix}$$

$r(\widehat{s}_{k-1})$ is the residual of the estimate

$$r(\widehat{s}_{k-1}) = \begin{bmatrix} (\hat{px}_{k-1} - x_{a1})^2 + (\hat{py}_{k-1} - y_{a1})^2 + (\hat{pz}_{k-1} - z_{a1})^2 \\ (\hat{px}_{k-1} - x_{a2})^2 + (\hat{py}_{k-1} - y_{a2})^2 + (\hat{pz}_{k-1} - z_{a2})^2 \\ (\hat{px}_{k-1} - x_{a3})^2 + (\hat{py}_{k-1} - y_{a3})^2 + (\hat{pz}_{k-1} - z_{a3})^2 \end{bmatrix} - y_{k-1} \tag{17}$$

5.2.2 Dynamic Positioning

In dynamic positioning the tag is assumed to be in motion and a real-time estimate of the tag's location is required.

The Extended Kalman Filter has been extensively used in navigation systems and can be regarded as the 'defacto' standard for such purposes. The linear Kalman filter is considered a good solution for linear systems

models where both the measurement and state transition models are affected by additive independent white Gaussian noise. The extended Kalman filter works by adapting this technique to non-linear systems by linearizing the model about a working point using the Taylor Series expansion [10]. The Extended Kalman filter also uses a recursive approach that is comprised of two phases, the predict phase and the update phase.

The non-linear system is modelled in the following manner: The state of the tag at a time instance t_k denoted by s_k is defined by its position x_k, y_k and z_k, and its velocities $\dot{x}_k$, $\dot{y}_k$ and $\dot{z}_k$ in the three co-ordinates x,y and z respectively.

$$s_k{}^T = \begin{bmatrix} x_k & y_k & z_k & \dot{x}_k & \dot{y}_k & \dot{z}_k \end{bmatrix}$$

We model the non-linear state transformation by the following where w_k is the process noise.

$$s_k = f(s_{k-1}) + w_k \tag{18}$$

The measurement at any time instant t_k can be modelled similarly as a non-linear function of the state in the following way where v_k is the observation noise given by $\mathcal{N}(0, R_k)$.

$$z_k = h(s_k) + v_k \tag{19}$$

Here $h(s_k)$ is the observation function which gives the vector containing the euclidean distances between the state and the anchors, and x_{ai}, y_{ai}, z_{ai} the x,y and z co-ordinates of the anchor i.

$$h(s_k) = \begin{bmatrix} \sqrt{(x_k - x_{a1})^2 + (y_k - y_{a1})^2 + (z_k - z_{a1})^2} \\ \sqrt{(x_k - x_{a2})^2 + (y_k - y_{a2})^2 + (z_k - z_{a2})^2} \\ \sqrt{(x_k - x_{a3})^2 + (y_k - y_{a3})^2 + (z_k - z_{a3})^2} \end{bmatrix} \tag{20}$$

5.2.2.1 Prediction Phase

The predict phase uses a non-linear state tranformation to calculate the estimate of the apriori state vector from the estimated posteriori state vector. The transformation is then 'linearized' to obtain the apriori state estimate from the posterori state estimate .

$$\hat{s}_{k|k-1} = F_k \cdot \hat{s}_{k-1|k-1} \tag{21}$$

Here F_k is called the linearized state transition matrix

$$F_k = \begin{bmatrix} 1 & 0 & 0 & \Delta t_k & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t_k & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t_k \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

where Δt_k is the time elapsed between the previous estimation and the current estimate.

$$\hat{P}_{k|k-1} = F_k \cdot \hat{P}_{k-1|k-1} \cdot F_k^T + Q \tag{22}$$

The acceleration of the tag reflects the process noise and is modelled here. The acceleration along x, y and z axis are assumed to be independent white gaussian noise and having variances σ_x, σ_y and σ_z respectively. Therefore the Q matrix representing the process noise covariance matrix is given by.

$$Q = A \cdot \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_z^2 \end{bmatrix} \cdot A^T \tag{23}$$

where A is given by:

$$A = \begin{bmatrix} \frac{\Delta t_k^2}{2} I_3 \\ \Delta t_k I_3 \end{bmatrix}$$

5.2.2.2 Update Phase

The update phase updates the posteriori state vector from the apriori state vector and the measurements. Firstly the innovation vector is calculated using the measurement z_k at the k^{th} step and the observation function.

$$\tilde{y}_k = z_k - h(\hat{s}_{k|k-1}) \tag{24}$$

The covariance matrix of the innovation vector is then computed.

$$S_k = H_k \cdot \hat{P}_{k|k-1} \cdot \hat{H}_k^{\ T} + R \tag{25}$$

where R_k is the covariance matrix of the measurements. And H_k is the Jacobian computed around the apriori state estimate $\hat{s}_{k|k-1}$ given by:

$$H_k = \begin{bmatrix} \frac{\hat{x}_{k|k-1} - x_{a1}}{dist(\hat{s}_{k|k-1}, p_{a1})} & \frac{\hat{y}_{k|k-1} - y_{a1}}{dist(\hat{s}_{k|k-1}, p_{a1})} & \frac{\hat{z}_{k|k-1} - z_{a1}}{dist(\hat{s}_{k|k-1}, p_{a1})} & 0 & 0 & 0 \\ \frac{\hat{x}_{k|k-1} - x_{a2}}{dist(\hat{s}_{k|k-1}, p_{a2})} & \frac{\hat{y}_{k|k-1} - y_{a2}}{dist(\hat{s}_{k|k-1}, p_{a2})} & \frac{\hat{z}_{k|k-1} - z_{a2}}{dist(\hat{s}_{k|k-1}, p_{a2})} & 0 & 0 & 0 \\ \frac{\hat{x}_{k|k-1} - x_{a3}}{dist(\hat{s}_{k|k-1}, p_{a3})} & \frac{\hat{y}_{k|k-1} - y_{a3}}{dist(\hat{s}_{k|k-1}, p_{a3})} & \frac{\hat{z}_{k|k-1} - z_{a3}}{dist(\hat{s}_{k|k-1}, p_{a3})} & 0 & 0 & 0 \end{bmatrix}$$

where $dist(\hat{s}_{k|k-1}, p_{ai})$ returns the Euclidean distance between the apriori state estimate and the i^{th} anchor We now compute the Kalman gain represented by K_k.

$$K_k = \hat{P}_{k|k-1} \cdot {H_k}^T \cdot S_k^{-1} \tag{26}$$

We compute the refined posteriori state estimate

$$\hat{x}_{k|k-1} = \hat{x}_{k|k-1} + K_k \cdot \tilde{y}_k \tag{27}$$

Finally we update the posteriori state covariance estimate matrix

$$\hat{P}_{k|k} = (I_6 - K_k \cdot H_k) \cdot \hat{P}_{k|k-1} \tag{28}$$

6 Development

6.1 C Code

On this part, we focus on explaining the challenges behind the implementations of a ranging system in hardware. As mentionned in Sec. 3, our positioning system has been realized on DecaWave Ltd. EVK1000 board. The ARM processor has been programmed in C using the IDE CooIDE distributed by CooCox Ltd.. The description of the ranging processes that we implemented can be found in Sec. 5.1.

As mentioned in previous section, our positioning process aims at knowing the position of a device (the tag) by estimating its distances with other devices (the anchors). If nothing else is stated, in the following part of the document N refers to the number of anchors with a minimal number of 3 anchors (if we have less anchors we then wouldn't be able to perform 3-Dimensional positioning).

We implemented two ranging methods in our hardware. These two methods use the same applicative top layer, that we described in this part. What we call applicative layer is what drives the algorithm. We designed and used this method presented as the top layer of both methods:

- First we initialize each value and determine the parameters and different operating modes of the board, by analyzing the values obtained from the board switches S_1.
- Then we enter the main loop (as we intend to do positioning with no finite time the loop is an infinite loop), in this loop we first check for the device interruption request, because all the messages coming from the antenna are handled by the hardware as interruptions.
- Then we run the state machine to process the interruption. If a distance is reported by the state machine we display it and send it to the front end layer (Matlab in our case) through the USB link.

The loop ends there providing a decent way of making the system running in real time, handling interruptions and data supply to the front end.

6.1.1 Two Way Ranging

In this part we focus on the specific implementation of the two way ranging (TWR) and the problems we met during the implementation of this ranging

process.

Timeline The two way ranging was our first trial of implementation on these devices so it took us some time to get aknowledged with the envrionements and devices. We then designed the state machine for TWR based on the one designed by DecaWave Ltd for their STWR demonstration application. On top of this we also designed a communication protocol to be able to supply proper communication between the tag and the N anchors.

Communication protocol The idea behind the design of the communication protocoll is pretty simple. Each of the anchor has its address implemented on the tag memory, this is encoded in a table $T_{Anchors}$ of length N. The scheme we present was thought for 3 anchors so it makes sure that all the anchors implied in the process are addressed and replied. Our scheme uses an index i_{Anchor} (initialised at 0), we then address the anchor situated at i_{Anchor} in $T_{Anchors}$, if the process time out, or any error happens during the process, we keep on addressing this anchor (to make sure we perform ranging with any of the anchors). Once the ranging has been made between the tag and this anchor, i_{Anchor} is increased modulo N. this method is self sufficient to provide the set of ranges with each anchors, but has the drawback of failing if one anchor is not working or the communication link deficient (loss of line of sight, distance too high...) in the case of using more than 3 anchors.

Interfacing the top layer and the hardware As in our board, the EVK 1000, the DW1000 has its own processor, we needed to create an interface between the top layer (State Machine) and the radio. For this most of the functions were implemented in the API, but things were left to do so that the communications are performed the way we want them to. The way we communicate with the radio is through interruption requests (IRQ). This interruptions have to be analyzed and rendered as events that can be analyzed in our state machine, these events would the be stack into an event stack that is handled by the state machine. The functions in charge of this analysis and interface are the callback functions in our implementation. We implemented two of them : the transmission callback and the reception callback function.

Transmission callback This callback function is called anytime a transmission IRQ is received. It analyses the sending timestamp of the message

and puts the event in the event stack.

Reception callback This callback function is more complex as it handles the analysis of the received message. It first stores the timestamp of the event (sent through the IRQ). Once this is done it analyses the header to know if the message is a poll or a response message (see Sec. 5.1.1 for vocabulary on TWR). Then it checks if the destination of the message is our coressponding adress, if not the message is not refered. If it is the device address, the message is analyzed and the message event is transmitted to the state machine. In the case of a poll message being received the timestamp t_{Rx} is used to directly set the sending time of the response message at $t_{Rx} + D$, D being the reply delay, the response message is then sent using the callback functions, it is not the responsibility of the state machine.

State Machine for TWR As mentioned in the above paragraph, the implementation of our state machine is inspired by the one used by DecaWave Ltd. in their demonstration application. The state machine of the TWR is one of the simplest one that can be achieved for a communication protocol between two devices. It can then be used to develop more complex systems as a basis to understand how to build a State Machine.

The state machine does two-way ranging by forming the messages for transmit(TX), commanding their transmission, by commanding the receive(RX) activities, by recording the different timestamps from different callback event, by calculating the time of flight according to those timestamps.

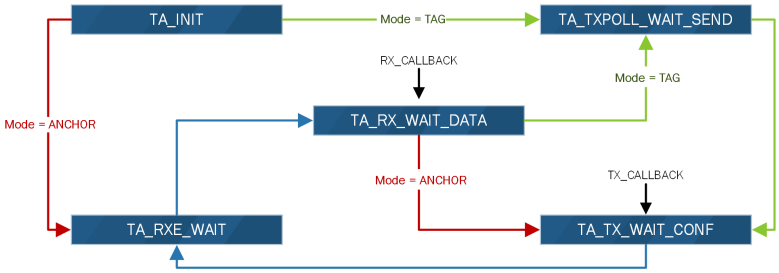

Figure 7: Two-way ranging state machine

In this state machine transition, we have totally 5 states to complete one measurement.

The tag It consists of all the 5 states and the description of tag execution is as follows,

- **State: TA_INIT** The initial state case TA_INIT performs initialisation and determines the next state to run depending on whether the inst-¿mode is selecting tag or anchor operation. In the case of a tag we want to send a Poll message to allow an anchor to communicate with the tag and then send back a response message, thus the state is changed to TA_TXPOLL_WAIT_SEND.

- **State: TA_TXPOLL_WAIT_SEND** In this state we want to send the Poll message, so we call function setupmacframedata(), which sets up the all the other parameters/bytes of the Poll message. And We also configure and enable the RX frame wait timeout, so that if the response is not coming, the tag times-out and restarts the ranging. Then we start the transmit.

- **State: TA_TX_WAIT_CONF** In this state, we await the confirmation that the Poll message transmission has completed. We get the interrupt event by calling a transmit call back function. After a confirmation of a successful transmission, we will read and save the TX time of Poll message and then proceed to the next state (TA_RXE_WAIT) to turn on the receiver and await a response message.

- **State: TA_RXE_WAIT** This is the pre-receiver enable state. Here the receiver is enabled and the state will then proceed to the TA_RX_WAIT _DATA where it will wait to process any received messages or will time-out.

- **State: TA_RX_WAIT_DATA** This is the biggest state in our state machine. But for a tag, we only need to consider the situation where we receive a response message from an anchor. We can see if we receive a good response message or not through the event we get from the receive call back function. If it is the correct response message from the anchor, we record the receive time of this response message. Then we can do time-of-flight calculations considering the delay of anchor is known at tags. Finally we go to the TA_TXPOLL_WAIT_SEND state to start next ranging.

The anchor It consists of 4 states and the description of flow of execution is as follows,

- **State: TA_INIT** In the case of a anchor we want to receive a Poll message from one tag and then send back a response message, thus the state is changed to TA_RXE_WAIT after some necessary operations such as set the address and disable frame filtering.
- **State: TA_RXE_WAIT** The same thing happens here in this state as a tag. The receiver is enabled and the state will then proceed to the TA_RX_WAIT_DATA where it will wait to process any received messages or will timeout.
- **State: TA_RX_WAIT_DATA** For an anchor, the only correct message we may receive in this state is the Poll message from a tag. So after getting a receive callback function, we first record the receive time of Poll message(not used for time-of-flight calculation) and make state transition. But this time we directly go to the TA_TX_WAIT_CONF state since in the receive callback function, we finish the task of setting a constant delay of response and sending this response.
- **State: TA_TX_WAIT_CONF** In this state, we record the transmit time of response message and then calculate the overflow of our anchor's antenna delay. Then we may reset this delay in next measurement in receive callback function to make our response transmit time more accurate. After this, we go back to TA_RXE_WAIT state to wait for next measurement.

Symmetrical TWR

In this part we will describe the Symetrical Two Way Ranging positioning scheme (STWR) described in Sec. 5.1.2.The driving idea of this process is quite similar to the one concerning the TWR (Sec. 6.1.1), we will then mainly focus on the main differences on both implementation: only one more message is exchanged in the communication scheme but both the Tag and the Anchors can know their respective distance (actuallly only the Anchor computes it but it uses the data frame of the response message to send this time of flight estimation back to the Tag).

Timeline This positioning process has been developed directly after the completion of the TWR, we were then knowing the challenges of the implementation and the specificities of the hardware implementation. It has then been a faster developing processus for us. The developement of this process was done simultaneously with the dynamic positioning algorithm developement.

Communication protocol The communication protocol is exactly the same as the one used in TWR, hence one can refer to Sec. 6.1.1 to see how it had been designed and implemented.

Interfacing the top layer and the hardware In this section there are few differences between the implementations of TWR and STWR. The concept is still the same in order to process messages from the radio we process IRQs through the two callback functions.

Transmission callback The transmission callback function for the STWR is exactly the same as the one used in TWR, hence one can refer to Sec. 6.1.1 to see how it had been designed and implemented.

Reception callback As for the TWR the aim of this function is to provide an analysis of the received message (correct destination and source, etc), carry the analysis and provide information to the event stack. The main difference between this callback and the previous one is that, in case of a response poll being received, the response is set immediatly (the device sends as soon as it can). Then when a response message is recieved it is reported to the state machine which then set the delayed final message.

State Machine for STWR The state machine for symmetric two-way ranging contains 6 states to complete one measurement.

For **tag**, it consists of all the 6 states and the description of tag execution is as follows,

- **State: TA_INIT** The initial state case TA_INIT performs initialisation and determines the next state to run depending on whether the mode is selecting to tag or anchor. In the case of a tag we want to send a Poll message to allow an anchor to communicate with the tag

and then send back a response message, thus the state is changed to TA_TXPOLL_WAIT_SEND.

- **State: TA_TXPOLL_WAIT_SEND** In this state we want to send the Poll message, so we call function setupmacframedata(), which sets up the all the other parameters/bytes of the Poll message. And We also configure and enable the RX frame wait timeout, so that if the response is not coming, the tag times-out and restarts the ranging. Then we start the transmit. The next state is TA_TX_WAIT_CONF.

- **State: TA_TX_WAIT_CONF** In this state, we await the confirmation that the Poll message and Final message transmission is completed. We get the interrupt event by calling a transmit callback function (instance_txcallback()). After a confirmation of a successful transmission, we read and save the TX time of Poll message(not for Final message) and then proceed to the next state. The next state here can be TA_RXE_WAIT for Poll message to turn on the receiver and await a response message or TA_TXPOLL_WAIT_SEND for Final message to start next measurement.

- **State: TA_RXE_WAIT** This is the pre-receiver enable state. Here the receiver is enabled and the state will then proceed to the TA_RX_WAIT _DATA where it will wait to process any received messages or will time out.

- **State: TA_RX_WAIT_DATA** This is the biggest state in our state machine. But for a tag, we only need to consider the situation where we receive a response message from an anchor. We can see if we receive a good response message or not through the event we get from the receive call back function. If it is the correct response message from the anchor, we record the receive time of this response message. Then go to state TA_TXFINAL_WAIT_SEND.

- **State: TA_TXFINAL_WAIT_SEND** In this state we want to send Final message. The time of sending the final message depends on the receive time of response message and the delay time we set. Also these two time values and time stamp of sending Poll message are attached to the Final message in this state so they can be known at anchors. After sending Final message, the state machine go to TA_TX_WAIT_CONF to switch target anchor to communicate with in next measurement.

For **anchors**, they consist of 4 states and the description of flow of execution is as follows,

- **State: TA_INIT** In the case of a anchor we want to receive a Poll message from one tag and then send back a response message, thus the state is changed to TA_RXE_WAIT after some necessary operations such as set the address and disable frame filtering.
- **State: TA_RXE_WAIT** The same thing with anchors happens here in this state as a tag. The receiver is enabled and the state will then proceed to the TA_RX_WAIT_DATA where it will wait to process any received messages or will time out.
- **State: TA_RX_WAIT_DATA** For an anchor, the correct messages we may receive in this state can be Poll message and Final message from a tag. So after getting a receive callback function, we first record the receive time of the message, do calculations and make state transition. If the received message is a Poll message, we directly go to the TA_TX_WAIT_CONF state which is totally the same as we do in two-way ranging. The corresponding response sending is done in the received callback function. However if the received message is a Final message, we extract all the time stamps attached in the Final message and calculate time of flight. Then the state machine switch to TA_RXE_WAIT to wait for next Poll message.
- **State: TA_TX_WAIT_CONF** In this state, we record the transmit time of response message and go back to TA_RXE_WAIT state to wait for Final message.

6.2 MATLAB scripts

In this section, we describe the implementation of the theory behind positioning discussed in Sec. 5.2. We have already discussed about positioning of target device (Tag) using range measurements from three or more fixed anchors. We have created simulations and real-time implementations for two models for positioning. One is for static positioning and other for Dynamic positioning. Simulations are performed to verify the theoretical model before the real-time implementations, that ultimately provides the practical solution to obtain accurate positioning.

Property	Value	Comments
Serial port	COMXX	Specific to OS
Input Buffer size	110	> Size of output chars (11)
Output Buffer size	110	> Size of output chars (11)
Terminator	char('t')	Set in TAG's USB output
Timeout	100 Seconds	Set to a large value

Table 1: List of properties configured

Obtaining real-time data

Range measurements are sent to MATLAB over serial COM port. Serial port connection is established in MATLAB by creating a `serial` object, and connection to this object is established using `fopen` function. Every time a connection to the serial port is opened, it has to be closed before starting a new connection on the same serial port object.

Serial port connection has specific properties which can be modified based on the program requirements. Table 1 shows the list of properties that are configured in our program. Other properties are left as default. The data from serial port is read from the buffer using `fscanf` command. `fscanf` reads the data from buffer till it reaches the terminator character. It returns the set of characters. In our case, we obtain a set of chars in the a particular format given by : <Anchor Index no. - 1 char >(space - 2char) <range in meter - 5 chars (float)>.

To obtain position of the Tag in x,y and z directions, we need range measurements from at least 3 anchors nodes. Hence our script reads the serial port for range measurements corresponding to all three anchor index values. If one of the values are corrupted or failed to reach the serial port, we skip the complete set of readings and move to obtain the next valid set.

Two Way Ranging (TWR)

As two way ranging is faster that Symmetric Double-Sided Two Way Ranging (SDS-TWR), we have to wait shorter amount of time to get same number of reading. Although, TWR has worse precision and variance of error is quite high as compared to STWR. In TWR we obtain one set of 3 range measure-

ments in 0.035 seconds, which means that we get approximately 100 readings in 3.5 seconds. After accounting for failed readings and buffer loss, we, in practice, obtain 200 readings in 8-9 seconds.

Symmetric Double-Sided Two Way Ranging (SDS-TWR)
SDS-TWR gives much better precision, but at the expense of time. One set of 3 range measurements take approximately 0.4 seconds. This amounts for around 40 readings in 10 secs.

Static Position Estimation

As discussed in previous sections, the static position estimation is based on the Non-Linear Least Square estimation (NLS). In static positioning, we place the tag at a static position and wait for a number of reading (f value for centimeter precision is found to be 200 readings). The more reading we get at a position, the more accurate we can be in positioning, but at the expense of time.

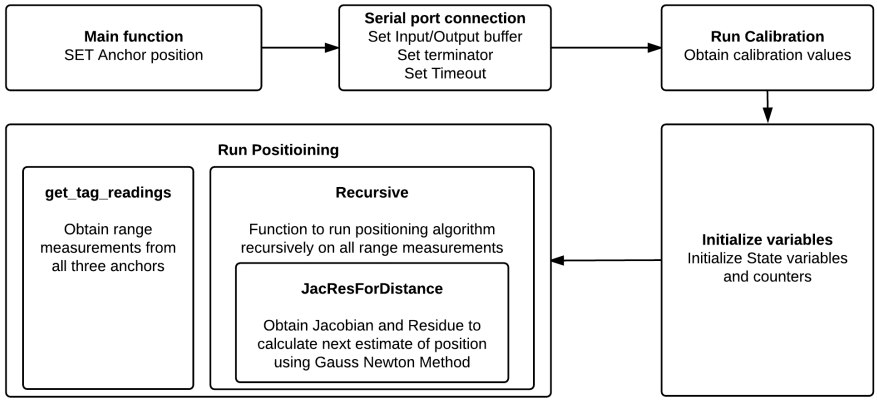

Figure 8: Matlab function flow for Static positioning

Implementation of Static positioning algorithm is done in a recursive fashion using a two-step method, which estimates the correct ranges first, and then estimate the position. We make a call to `recursive` function, which in turn calls to `get_tag_readings` to get range data, and `jacresfordistance`

for getting jacobian and residual values that help estimating positioning. The process will become more clear from the figure 8.

Dynamic Position Estimation

In dynamic positioning, we use the Extended Kalman Filtering technique to dynamically obtain the estimate of position of the TAG. The theory is discussed in section 5.2. Here we create a model of movement of TAG and estimate the position based on this model. The movement of the TAG will be defined by its position, velocity and acceleration values. Our model only predicts till first order derivative i.e. velocity.

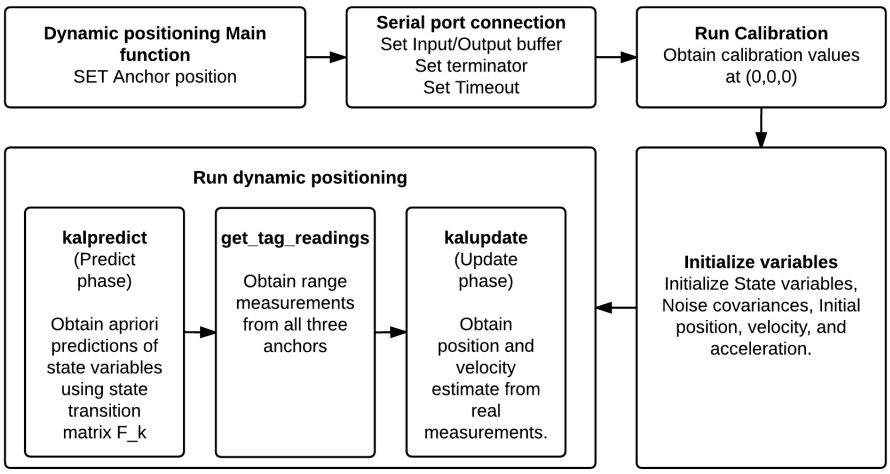

Figure 9: Matlab function flow for Dynamic positioning

The important aspect of dynamic positioning is that the predicted position should follow a path within the range of our model. Kalman filtering enables us to get better estimates using a state space model. Initial steps are similar to Static positioning case where we have to connect to the serial port for obtaining data. However, in case of dynamic positioning, we only take one set of range measurements at time to get position estimate. Initialization of state model variables is also another important factor that can affect the error in our estimates if not matched properly. The first step in EKF is the Prediction phase where we predict new state variables based on initial position estimate. In the second step, we use the values from prediction phase

to obtain an Updated value of position and velocity estimate. The complete flow of the algorithm can be seen in figure 9.

Complete set of MATLAB codes are available in the APPENDIX I section. Scripts are thoroughly commented and easy to understand. The scripts for Static and Dynamic positioning are organized separately.

Executing MATLAB codes

Make sure that all required files are available in the PATH of MATLAB.
For Static positioning, open "Static_Positioning_main.m" file, and make changes according to the setup. Run this file to execute Static positioning.
Similarly for dynamic positioning, open "Dynamic_Positioning_main.m" file, and make changes according to the setup. Run this file to execute Dynamic positioning.

6.3 Prototype

The final user is able to operate and visualize the outputs from the positioning system through a user interface developed on MATLAB. When running the algorithm it is possible to choose from three different parameters as shown on figure 10.

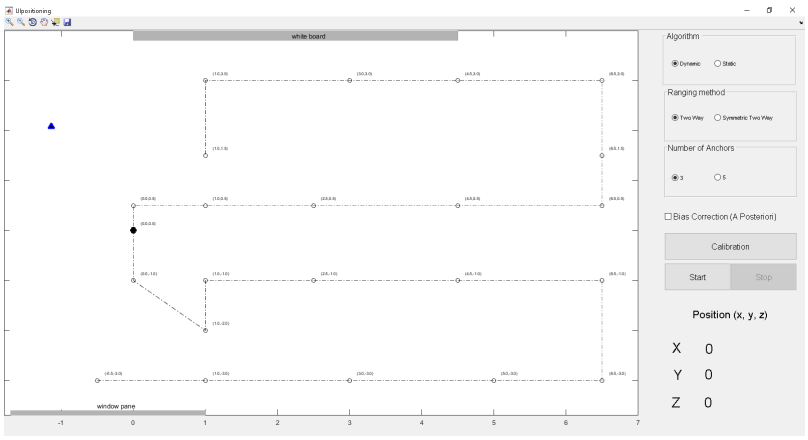

Figure 10: Start screen of the user interface on final prototype.

Related to the positioning algorithm, it is possible to select between dynamic or static method. There are also two supported operational modes for ranging, two-way or symmetric two-way. Furthermore, it is possible to run the algorithm considering five anchors, in case the user posses enough devices.

On the same screen it is also possible to calibrate system. In this case, the tag should be placed at the coordinate plane origin.

After selecting the desired options the system can be initialized and the results will be available on screen. In other words, it is possible to see the actual tag coordinates in 3 dimension and a 2 dimension map with a path suggestion, based on points marked on the floor of the room during experimentation. On the top of the screen map is is possible to see the position of the white board, such as the windows on the bottom. These are references to help on the location in the map. As the user moves around the room the UI keeps track of the movement.

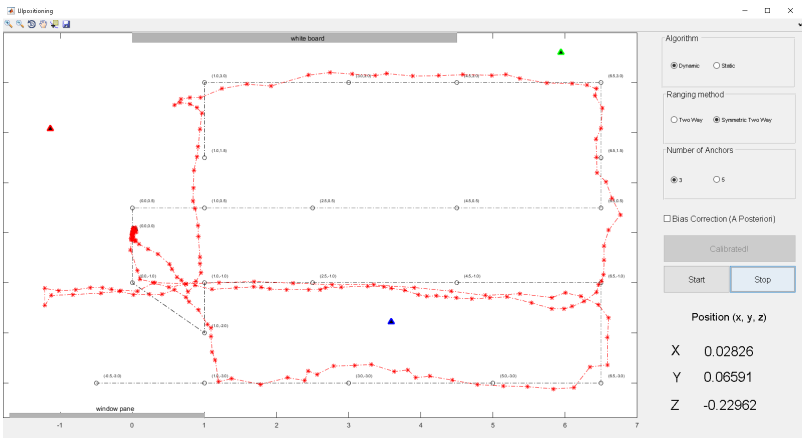

Figure 11: User interface after running the dynamic algorithm for symmetric two-way ranging with three anchors.

Figure 11 shows the user interface after someone walked on the path with tag on its hand.

7 Results

7.1 Positioning

7.1.1 Simulation results

Static Positioning

The Gauss Newton algorithm is used in the second step of the static positioning algorithm. The algorithm will not be defined for the estimates where the Jacobian of the residual function is singular (as we calculate its inverse) and is evaluated as the solution of 29.

$$\begin{bmatrix} \hat{px} - x_{a1} & \hat{py} - y_{a1} & \hat{pz} - z_{a1} \\ \hat{px} - x_{a2} & \hat{py} - y_{a2} & \hat{pz} - z_{a2} \\ \hat{px} - x_{a3} & \hat{py} - y_{a3} & \hat{pz} - z_{a3} \end{bmatrix} = 0 \tag{29}$$

Eq.29 is the equation of a plane which contains the three anchors. If the tag is placed in the plane, the algorithm will diverge as the estimate gets close to the actual location of the tag. However this is not a problem if the anchors are placed in the roof our room and the tag in its intended operation will never reach the plane.

There are still points in the operation area where the Jacobian is badly-conditioned and causing divergence. For example the performance between two points having different condition numbers associated with their Jacobian is shown in Fig. 12. The algorithm diverges as the estimate gets closer to

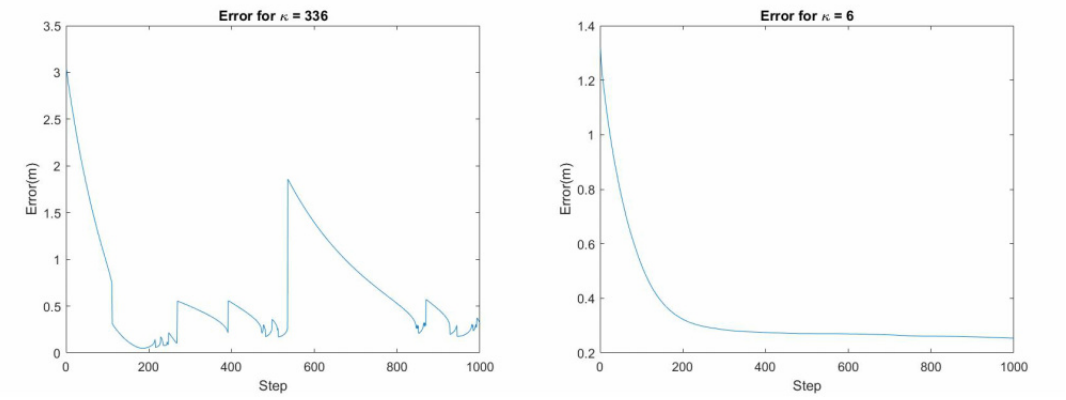

Figure 12: Effect of condition number on convergence

the tag location with a high condition number. The condition number of the

intended area of operation is plotted as a heatmap in Fig. 13.

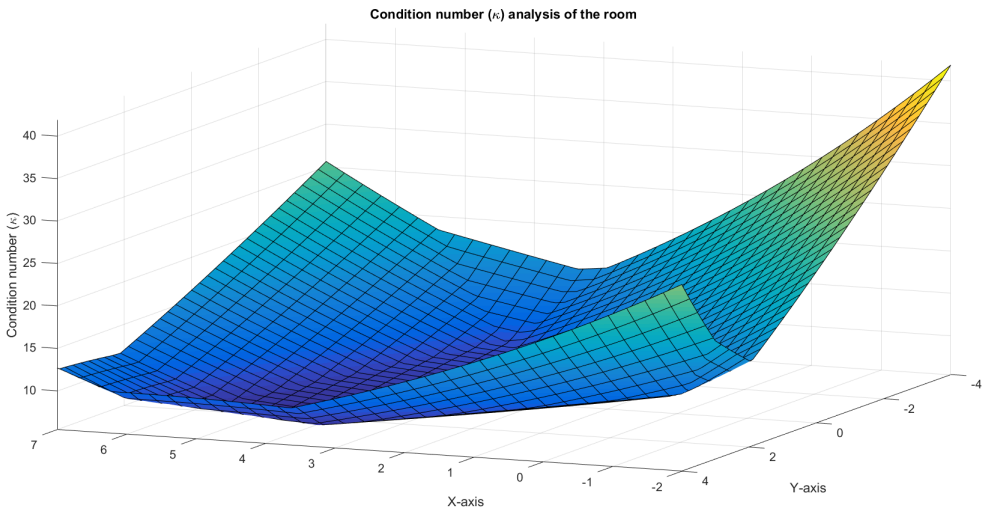

Figure 13: Condition number of operation area

The efficiency of the Gauss Newton algorithm is depending on the quality of the initialization of the parameters. Fig. 14 shows different initialization values used for estimating the position. One can see in Fig. 14 that the algorithm is robust to misadjustment on the initialization of the position estimate, provided that enough iterations are completed.

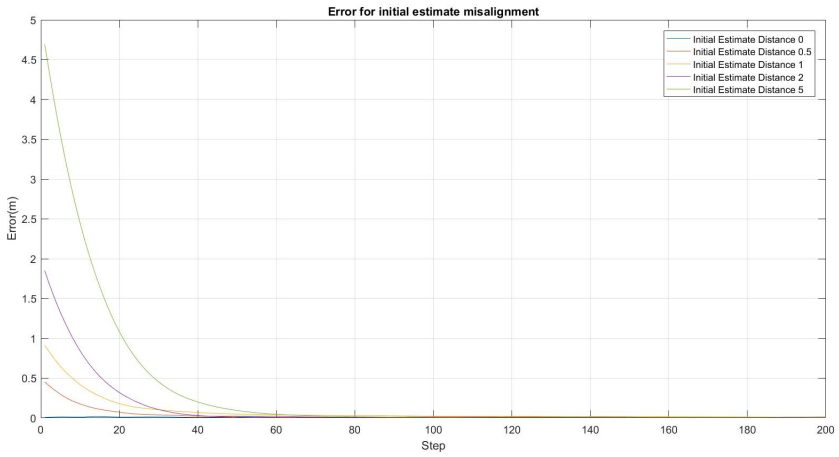

Figure 14: Parameter Initialization Error

Dynamic Positioning

As explained in Sec. 5, we use two different methodes, each having its specifities. TWR offers superior update rate of 0.03s but with less accuracy with a variance of 0.10 m^2 and more bias. The STWR offers an update rate of 0.27s but a better accuracy and a variance of $0.004m^2$.

As it is difficult to know the exact time a point is reached while using dynamic popsitioning, we are unable to carry a quantitative analysis of the dynamic positioning in a real environment. We then used simulations to evaluate the quality of the algorithm. The simulation works by generating path models for the tag by defining the coordinates of the system in parameters of time t which enables us to generate any arbitrary paths.

$$x(t) = f_1(t)$$
$$y(t) = f_2(t)$$
$$z(t) = f_3(t)$$

The distance measurements are taken from these coordinates at the corresponding time-intervals and an Additive White Gaussian noise is generated on top of these values.

Straight Line Path : We model a straight line path with a speed of 5km/hr(average human walking speed). The results obtained are shown in

Fig. 15.

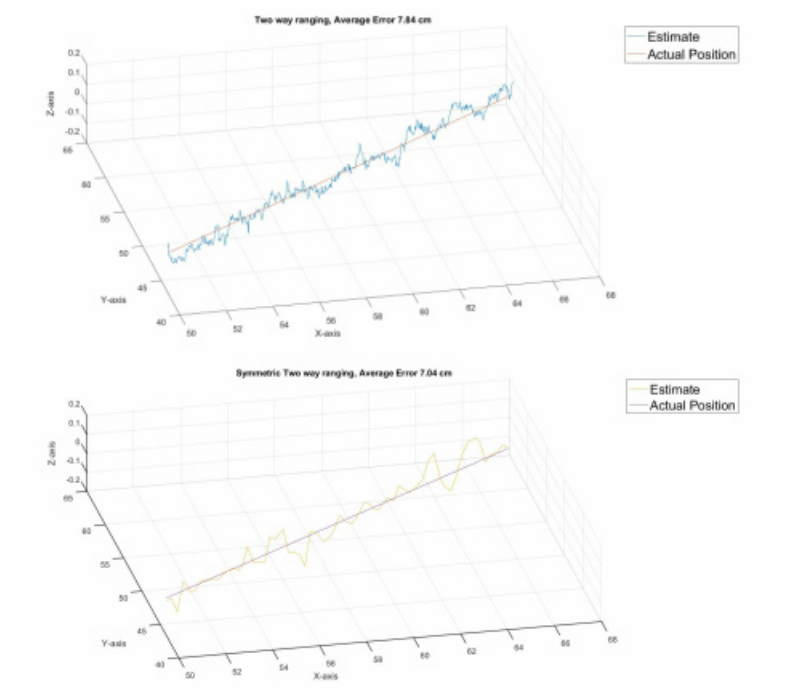

Figure 15: Dynamic Positioning Simulation for straight line path

Circular Path : We model a circular path for the tag with the same speed to reflect the movement of a human taking a turn. The results obtained are shown in Fig. 16.

For both scenarios, our simulations of STWR offers better performance in the than the simulated TWR.

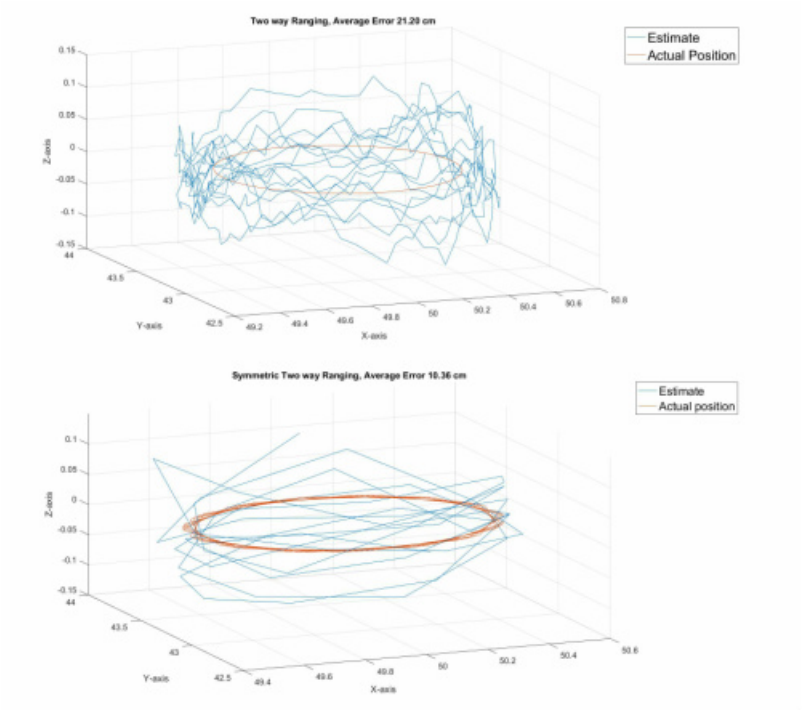

Figure 16: Dynamic Positioning Simulation for circular path

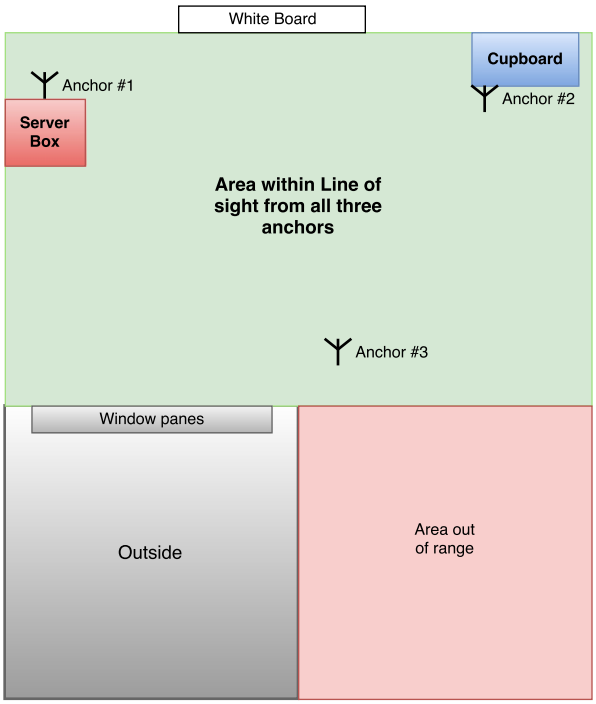

Figure 17: Room B230 setup for experimentation

7.1.2 Experiment setup

The real experiments for positioning are performed in room number B-230. Devices used in the setup consist of 3 Anchors and 1 Tag. Position of the anchors is fixed, while Tag can move freely within the line of sight from all three anchors, in all three dimensions. The figure 17 shows the structure of the experiment space, position of the anchors and the region available for positioning.

We created a set of points and a path to obtain range measurements for static and dynamic experiments respectively. The figure 18 shows these points and path used.

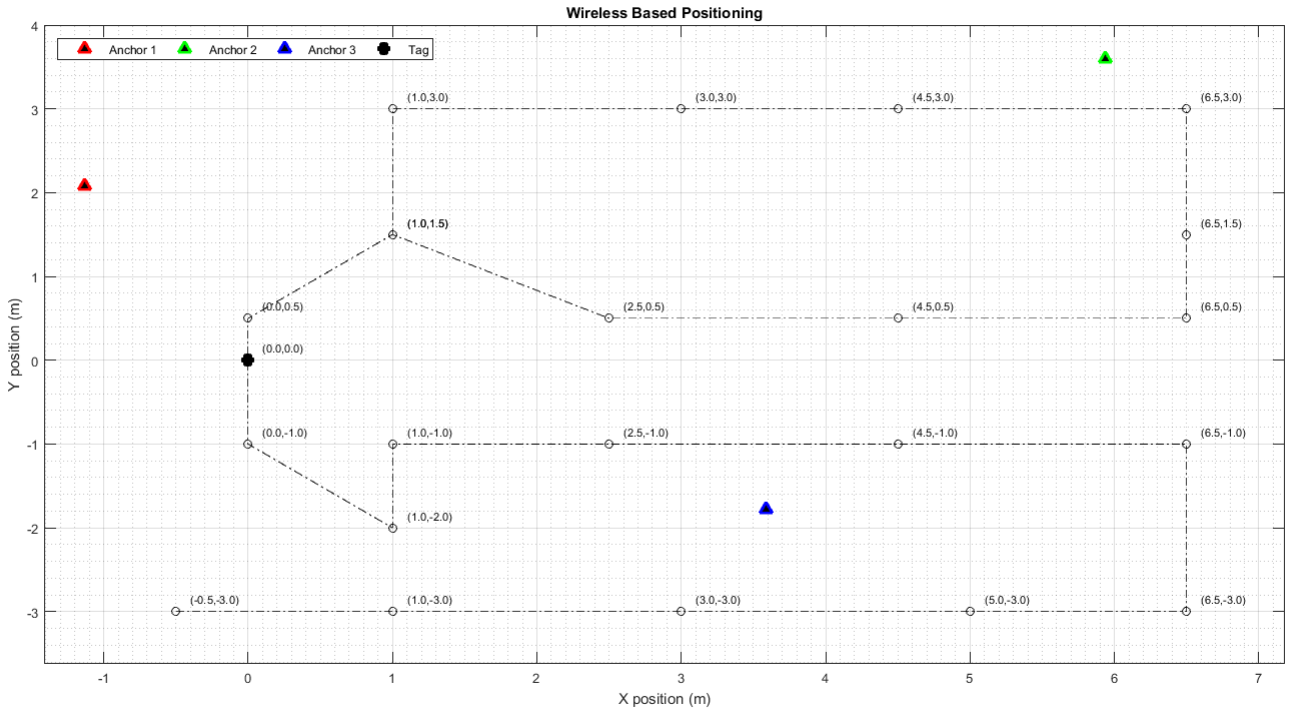

Figure 18: Measurement points and path used for experiments

7.1.3 Initialization

7.1.3.1 Static Positioning parameter initialization

The static positioning algorithm5.2.1 have parameters to be configured before operation. These parameters are initial position estimate ($\widehat{y}$, $\widehat{s}$), initial posteriori error covariance matrix $\widehat{P}$, measurement co-variance matrix (R) and step size (Δ).

Initial position and posteriori error covariance matrix: An estimate of the position needs to be initialized in the beginning. Ideally, initial value of $\widehat{y}$ and $\widehat{s}$ should be close to the actual square of the distances of the tag with the anchors and the actual co-ordinate position respectively. Since the position of the tag is unknown we initialize the tag at the center of all three anchor locations. $\widehat{P}$ is initialized by assuming the tag can be anywhere with uniform probability in the operating area and calculating the mean square error between the position of the tag and the initial position estimate.

Measurement Co-variance matrix and Step size: The measurement co-variance matrix, R is the variance in range measurement values from the ranging stage. This parameter has to be initiated with an approximate value close to the real variance in measurement data. The step size determines the speed of convergence of the Gauss-Newton algorithm. If the value is too small, it will take lot of time to converge to actual value, while if the value to too high, the algorithm might not converge at all. The algorithm is analyzed for different values of measurement co-variance and step size and optimal values are selected by choosing the configuration which minimizes the mean square of error, this is done for the two different ranging methods, which have different nature of measurement noises associated with them and accordingly have different parameters.

For TWR the performance analysis for the measurement variance and stepsize is carried in Fig. 19. One cannot find the optimal configuration in Fig.19 as the plot shows a lot of discontinuities. We then choose an appropriate step size of 0.01 and a measurement variance of 0.1 m^2.

For STWR, the performance analysis for the measurement variance and stepsize is carried in Fig. 20. In Fig. 21, one can see that it is unappropriate to choose a low stepsize value as the resulting error in distance would be higher than 1m. One can extract form Fig. 22 that the error is almost constant in the area studied that corresponds to the border of Fig. 20. Hence the

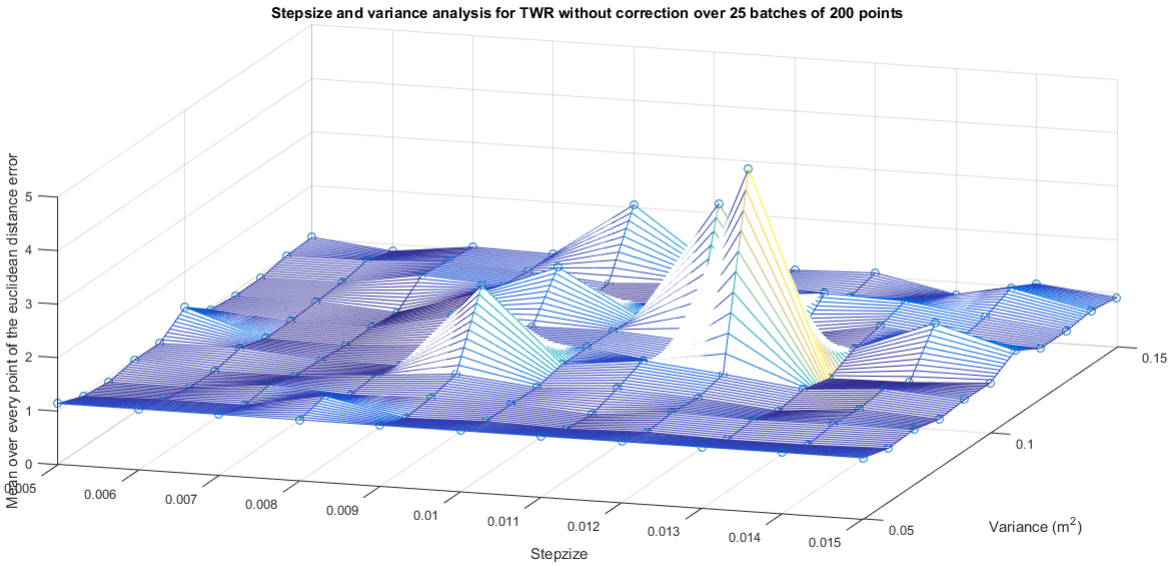

Figure 19: Performance of TWR for different configuration values

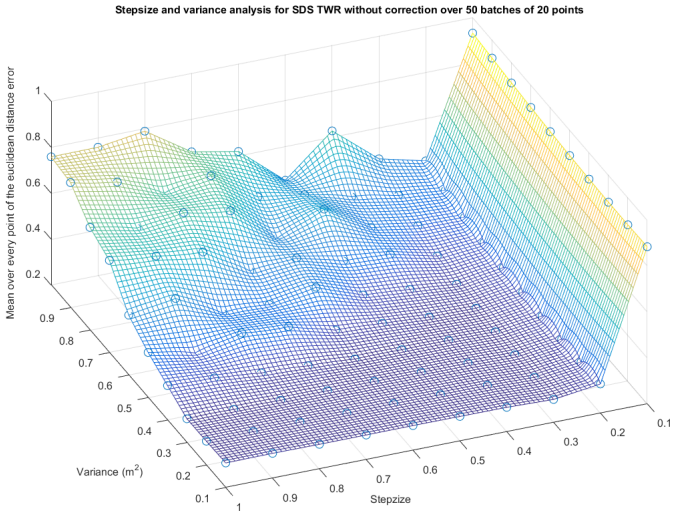

Figure 20: Performance of SDS for entire configuration values

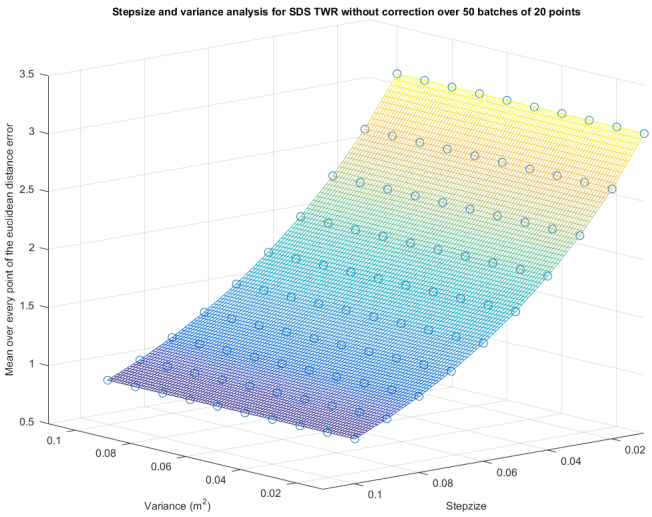

Figure 21: Performance of SDS for low step size

choosen parmaters are taken in this area and are 0.004 m^2 for the variance and 0.7 for the stepsize.

7.1.4 Static positioning results

Convergence Analysis

The convergence of the static positioning algorithm is studied in Fig. 23 for TWR , using the found as good step size and measurement covariance matrix. We then can see that the convergence is reached in 100 measurements.

In Fig. 24, we carry the convergence analysis for STWR using step size and measurement co-variance matrix found previously. The convergence is reached for 20 measurements.

This difference in the number of iterations needed is lying on the fact that STWR is way more accurate. However since the time taken for each measurement is higher in STWR compared to TWR, the times are similar to reach convergence. This can be seen on Fig. 25 where the curves are displayed with respect to the acquisition time.

From Fig. 25. , one can extract that STWR is still better as the speed and convergence is approximately the same for both but STWR is more accurate.

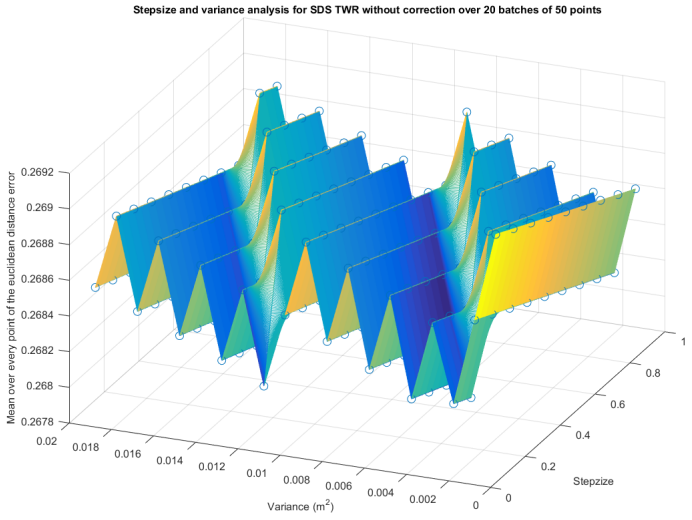

Figure 22: Performance of SDS for optimal configurations

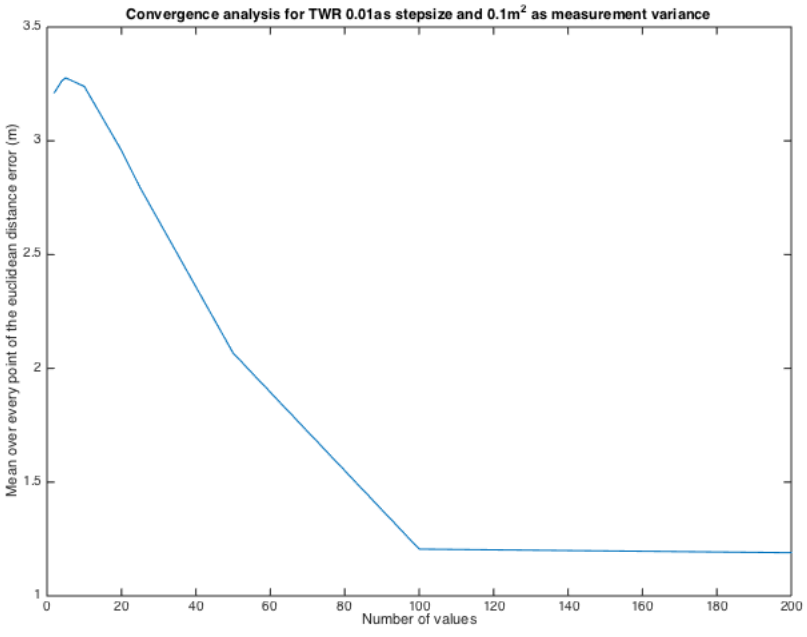

Figure 23: Convergence analysis for TWR

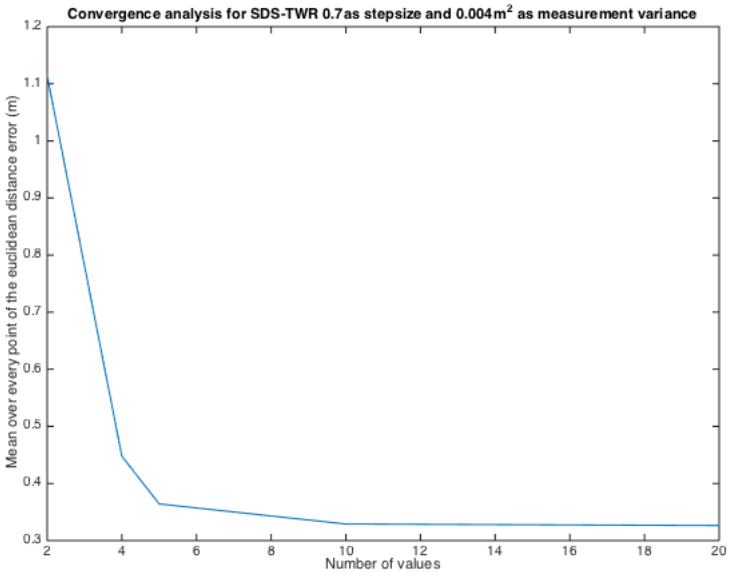

Figure 24: Convergence for SDS

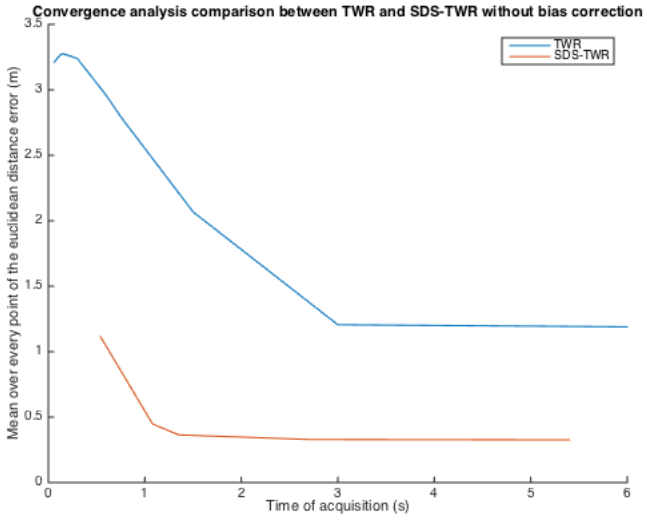

Figure 25: Convergence SDS vs TWR in time

7.1.4.1 Static Positioning Final Results

Results for the TWR The studies performed in the other sections allowed us to find a good operating point for our system. In this section we present the results obtained for this system on the experience conditions.

Fig. 26 shows the behaviour of our TWR system with no correction being performed, in functions of the x and y coordinates of the room for a constant altitude z. One can see that the results are outside the 10cm range fixed by our soecifications. in order to reduce these errors, we performed an in depth error analysis in Sec. 7.2, in order to reduce those errors.

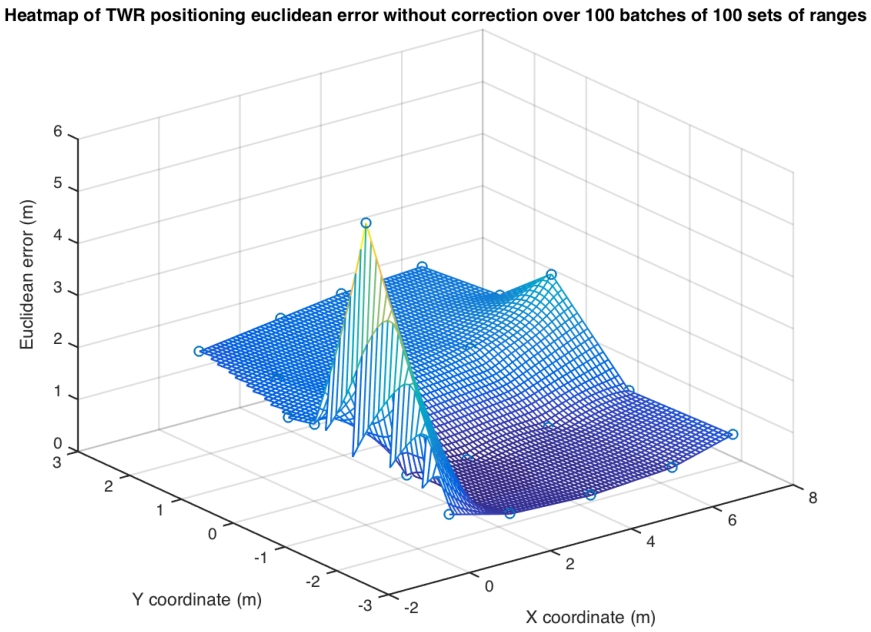

Figure 26: Heatmap for TWR without any correction being performed

After the error analysis we were able to provide a correction processus. These led to very accurate results in Fig. 28 and Fig. 29.

Results for the STWR As for the TWR the results are presented here for STWR in the same fashion. First, Fig. 30 shows the error induced by

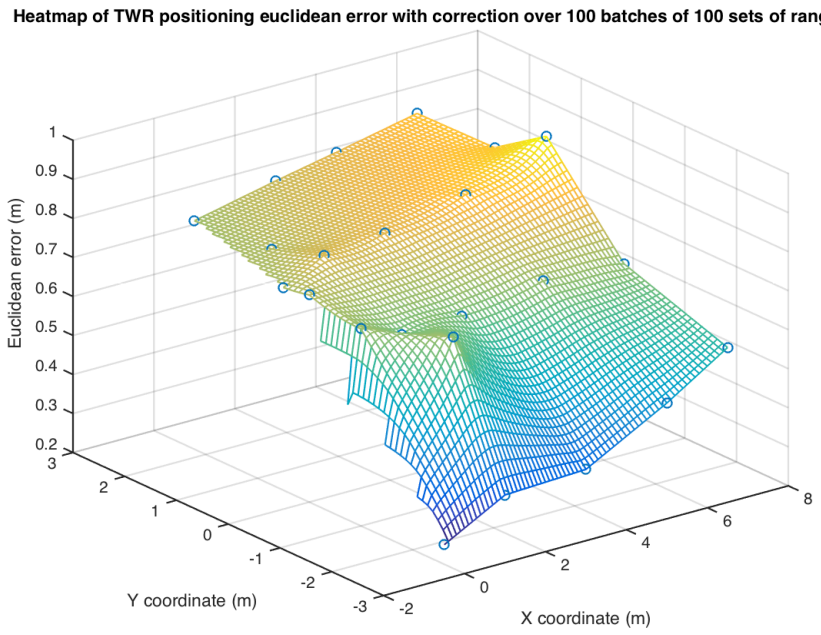

Figure 27: Heatmap for TWR with correction

the STWR process without any corrections. Then applying the correction method developed, we obtained the corrected results in Fig. 31 and Fig. 32.

7.1.5 Dynamic positioning results

Dynamic positioning is based on the EKF algorithm. The algorithm estimates the position and velocity of the tag using state space model of tag's movements. The parameters that needs to be initialized in this case are initial position matrix, covariance matrix of the initial position, acceleration covariance matrix and measurement covariance matrix.

In order to analyze the results in dynamic positioning, we need some device that can follow an exact path, with determined velocity and acceleration. This will enable us to study the predicted position estimates in comparison with the actual values of position and velocity. Although, unfortunately, we don't have such device to our disposal. Hence we could only perform some manual tests, and resorted to simpler visual analysis for errors.

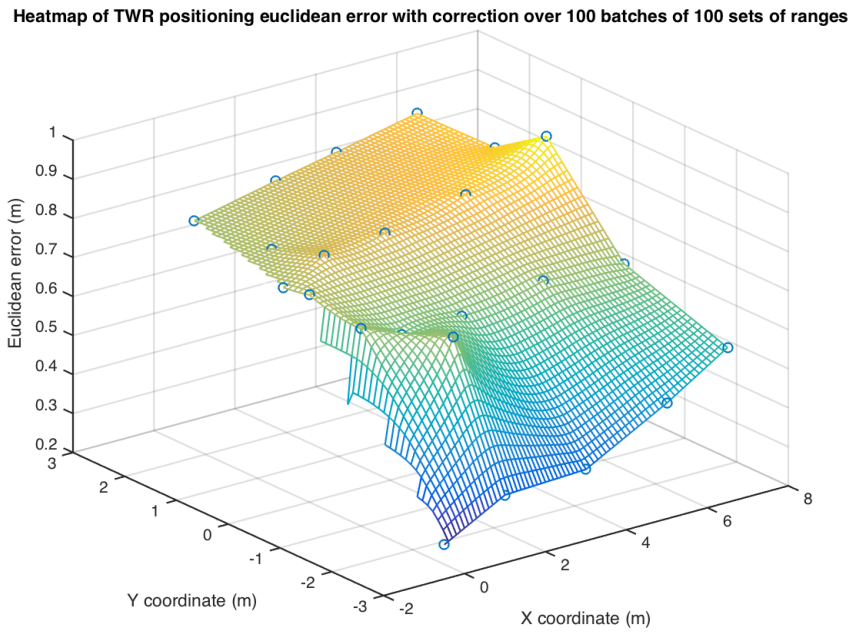

Figure 28: Heatmap for TWR with correction

Fig: 33 and Fig: 34 shows the output from one of the manual tests performed on two way ranging and symmetric two way ranging. Tag was moved around different locations in the room at normal human speed (around 3-4 km/hr). Average time taken between any two predictions is around 0.25 seconds in case of SDS-TWR and 0.12 seconds in case of TWR. Although this causes a delay between the actual point and the point that is displayed, but this delay is not very significant compared to the normal speed of a human.

7.2 Sources of error

The sources of error in the positioning algorithm are derived from the ranging error made in the previous step. In order to study the ranging error, several experiments have been carried out.

The first protocol that has been developed was to test the robustness of TWR algorithm in different environment. The tag was set in a fixed position

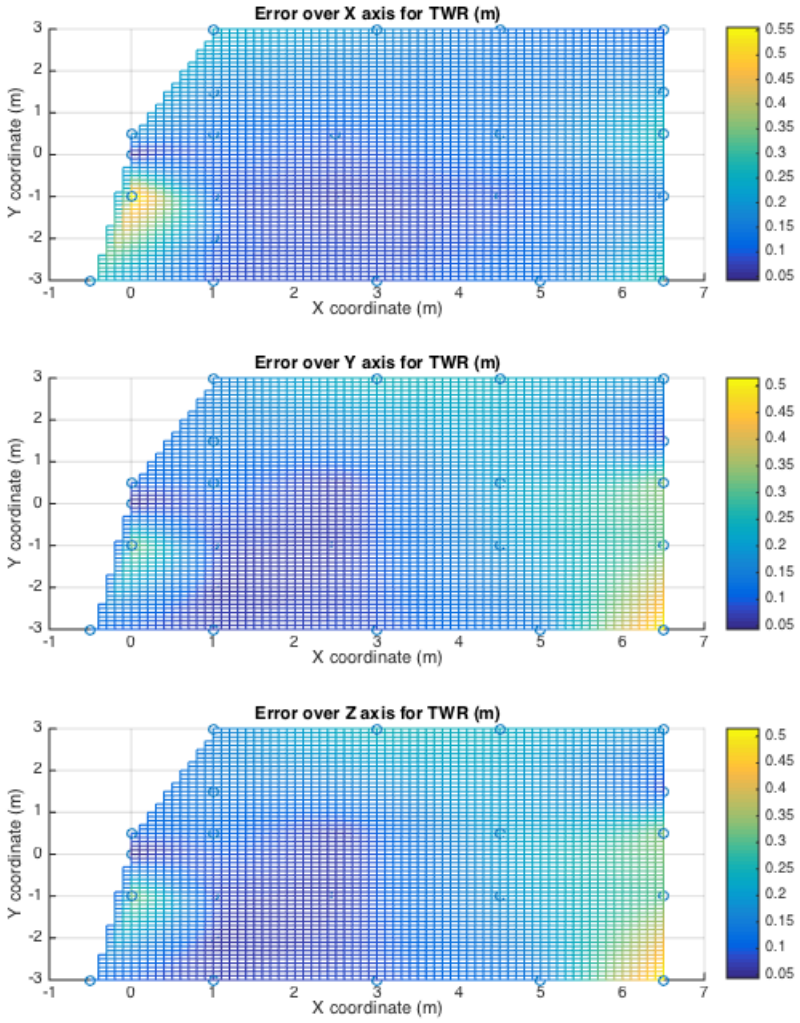

Figure 29: Heatmap for TWR with correction for each coordinate

and the anchors were moved step by step from 1m distance to 6m distance with a 1 meter-step. This experiment has been carried out in two indoor spot: one close to a wall and the other in the middle of the room. The results of the tests can be found in Fig 35.

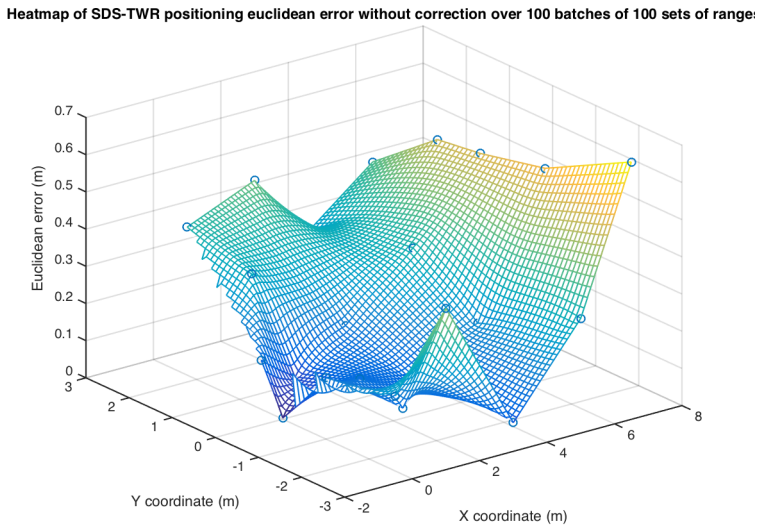

Figure 30: Heatmap for STWR without any correction being performed

This result is crucial because it is three-fold: the range error depends on anchor, on the distance from the tag to the anchor and on the environment itself.

7.2.1 Dependency on the clock system of each anchor

The dependency of the ranging error on each anchor is obviously linked to an inherent difference of clock system in each antenna. The Tab. 2 shows the mean of the actual and the relative bias between antennas over 23'000 acquisitions in 23fa different spots of the room studied. The relative bias is computed comparatively to anchor 2.

Measurement	Anchor 1	Anchor 2	Anchor 3
Actual bias (m)	1.64	0.53	0.76
Relative bias(m)	1.11	0	0.23

Table 2: Studying of the range bias for each anchor

This bias is related to the hardware differences between each antenna. It is particularly high because of the choice of the TWR protocol (see Fig. 4 p. 16). Indeed, if the clock system of the anchor is slightly different from the clock system of the tag there will be some inherent bias to the ranging. The DW1000 clocking scheme operates at a frequency of 38.4 MHz [4]. Let e_{tag} and e_{anchor} be the relative frequency error of the tag and of the anchor. Let TOF and t_d be respectively the time of flight and the time of response of the anchor (see Fig. 4 p. 16) The following equations can be proved:

$$\begin{aligned} \hat{TOF} - TOF &= TOF \times e_{tag} + \frac{t_d.(e_{tag} - e_{anchor})}{2} \\ &\approx t_d \times \frac{(e_{tag} - e_{anchor})}{2} \end{aligned}$$

$$\hat{d} \approx d + c \times \frac{t_d}{2}.(e_{tag} - e_{anchor}) \tag{30}$$

The bias can then be explained by the fact that each clock crystal of each antenna is different and operates at slightly different pulses. In order to illustrate this bias we designed two different protocols.

The first protocol was to change the response reply t_d in Eq. 30. The bias $b = d - \hat{d}$ is then a linear function of t_d: $b = b(t_d)$ with $\frac{(e_{tag} - e_{anchor})}{2}$ as slope. This experiment has been performed for anchor 2, the result can be seen in Fig 36

The second protocol was to measure the crystal signal directly on the 3rd pin of the DW1000 [4]. The evolving of the frequency of each crystal has been studied in Fig 37. The device used to measure was a frequency counter of 350 MHz (Agilent 53230 [5])

Using the actual frequency measured, the clock related bias has been computed and compared to the relative bias that was presented in Tab. 2. The result can be shown in Tab 3. One can see that the clock related bias is close to the relative bias measured.

A calibration of the data is done before applying the positioning algorithm in order to address the clock system error. The calibration is performed in (0,0,0) point (see Part. 7.1). Nevertheless, the bias related to the environment is not curbed. The next paragraph tends to show and explain the bias due to the environment.

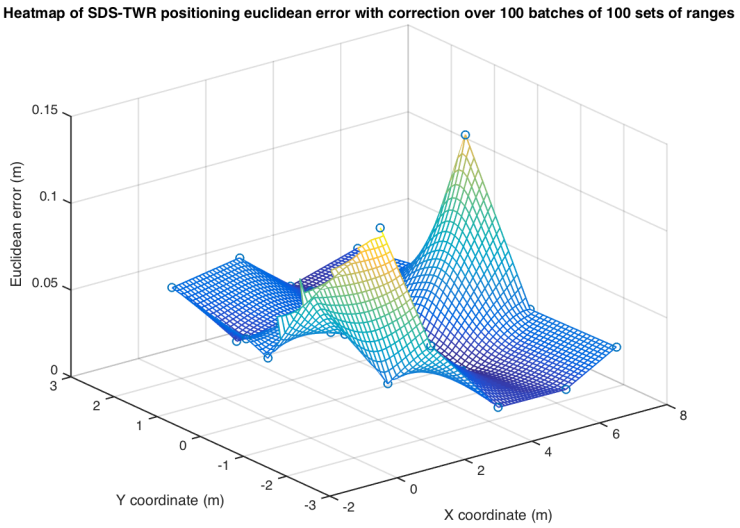

Figure 31: Heatmap for STWR with correction

Measurement	Anchor 1	Anchor 2	Anchor 3
Relative bias measured (m)	1.11	0	0.23
Clock related bias (m)	0.93	0	0.24

Table 3: Comparison between the relative bias measured and the clock related bias

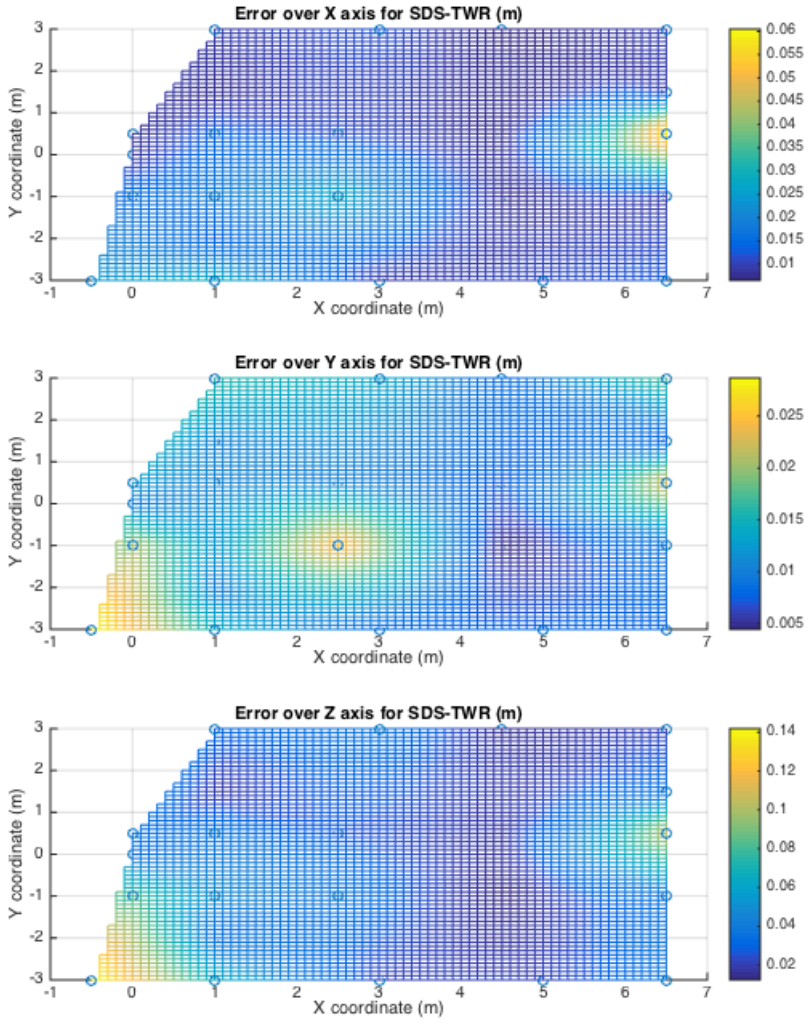

Figure 32: Heatmap for STWR with correction for each coordinate

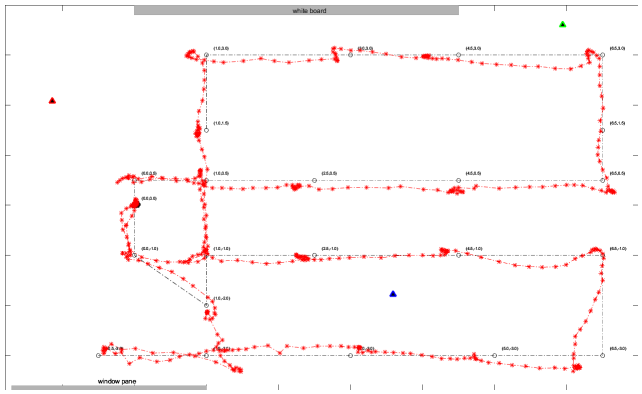

Figure 33: Symmetric Two way ranging with Dynamic positioning.

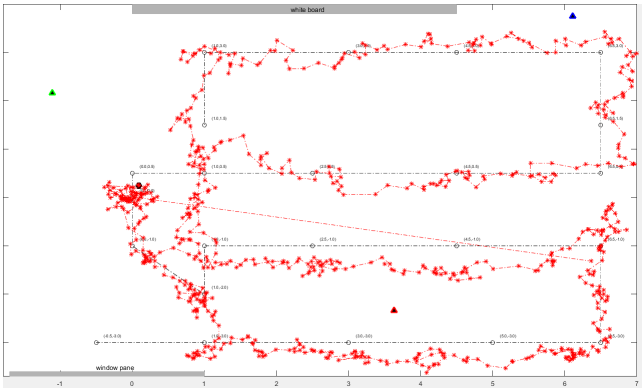

Figure 34: Two way ranging with Dynamic positioning.

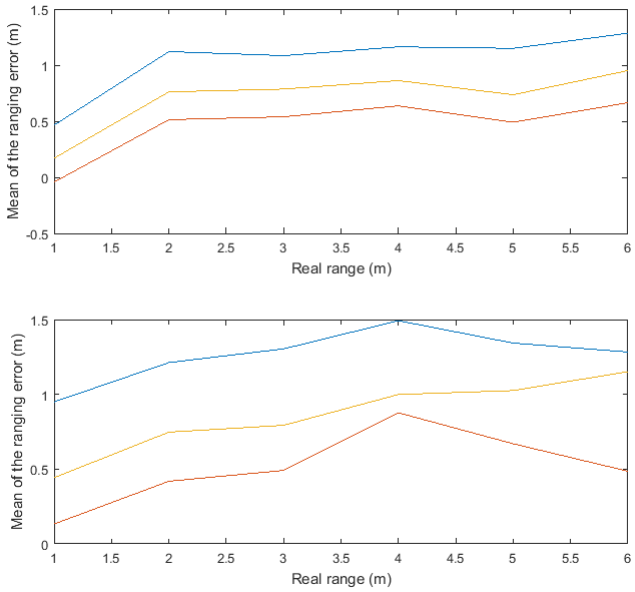

Figure 35: Ranging error in two different environment

7.2.2 Dependency on the environment and the range itself

The ranging accuracy has been shown to depend on the environment and on the range itself. This model can be confirmed by the heatmaps of the bias that we produced see Fig 38- 41. The setup of the experiments is described in Fig 17.

As one can assess, the main cause of error is due to the fact that the bias is changing within the position in the room. We coded a function to correct the ranges in real time before applying the positioning algorithm i.e. before knowing the position of the point. The 23 biases related to each 23 measurement points are saved in a chart. The simple idea of the function is hence to find the closest point in the range domain from three input ranges and then correct all ranges by the saved biases.

It is crucial to precise that the bias correction is possible only if a heatmap of the room has been made beforehand. If only one calibration is possible the heatmaps show that it should be performed in the middle of the room.

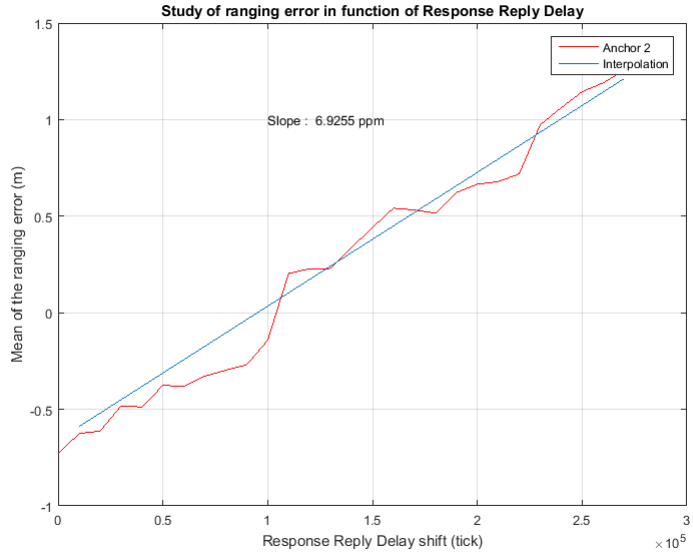

Figure 36: Ranging bias in function of t_d for anchor 2

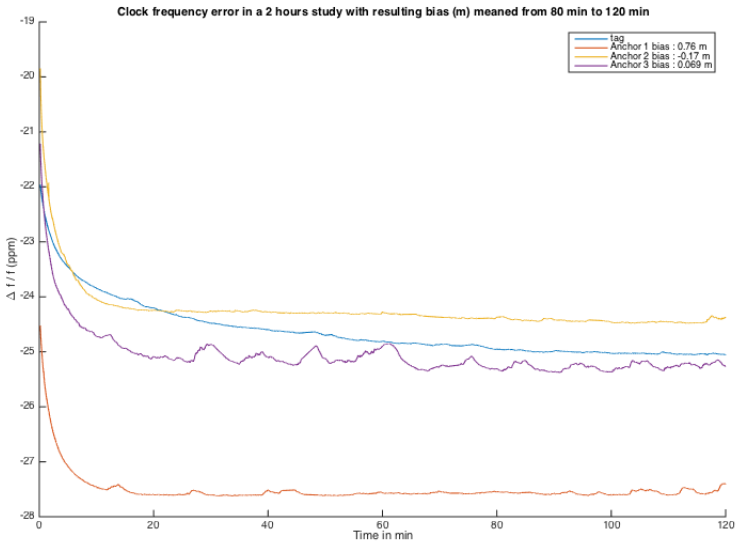

Figure 37: Studying of the crystal frequency in function of time

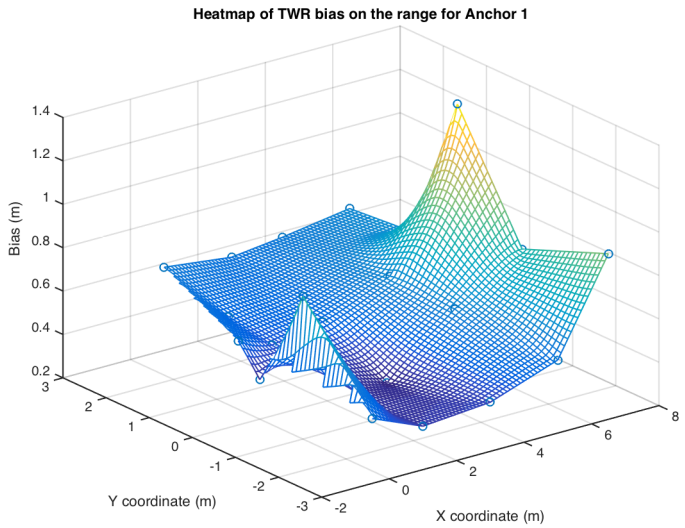

Figure 38: Heatmap of TWR range bias for anchor 1

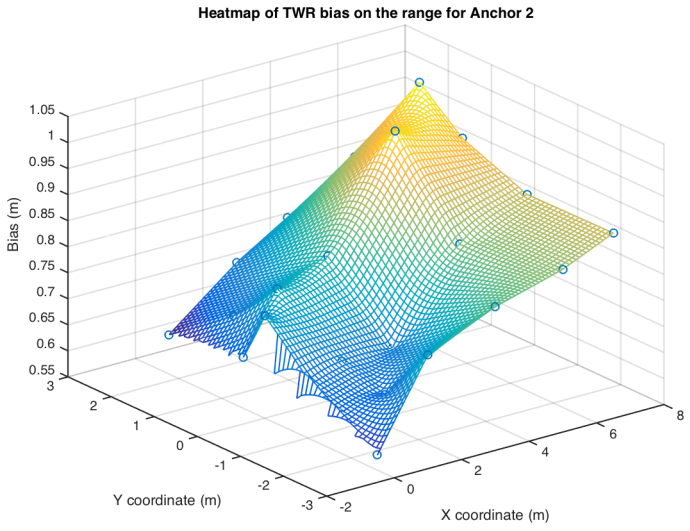

Figure 39: Heatmap of TWR range bias for anchor 2

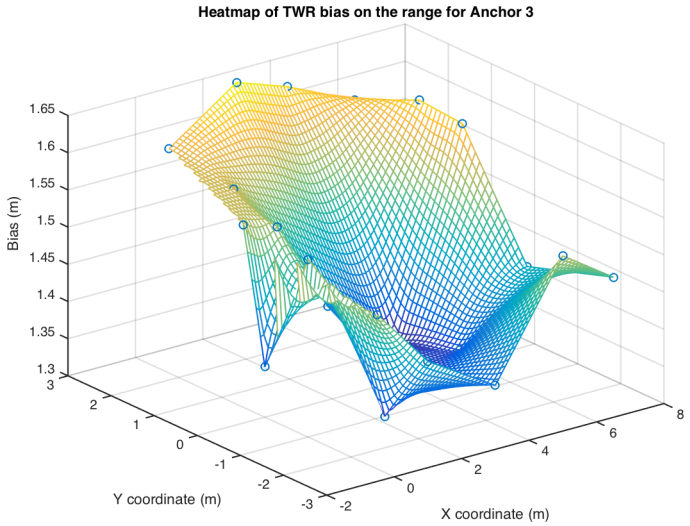

Figure 40: Heatmap of TWR range bias for anchor 3

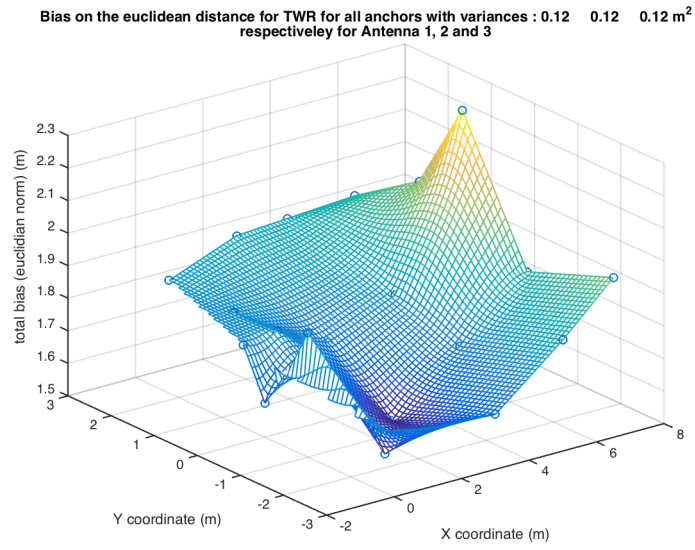

Figure 41: Heatmap of range bias for all anchors

According to Decawave datasheet, the dependency of the range error on the distance and on the environment could be merged into one unique dependency on the received signal level. This phenomena is consistent with the measurements gathered in Fig 38- 41. The persistent and substantial bias on the right hand of each heatmap can be explained by a loss of power due to the open space and very architecture of the room.

8 Conclusion

We have successfully achieved project goals of accurate positioning using Decawave DW1000 radio devices. The accuracy of the system depends on type of ranging and positioning methods are used, as well as many environmental factors. However, for our environment specifically, we analyzed these factors and calibrated the system for best performance. The achieved accuracy is significant for positioning in both static and dynamic case. We will further discuss conclusions drawn from different phases of the project.

In ranging phase, we implemented two separate protocols - Two-way ranging (TWR) and Symmetric Double-sided Two-way ranging (SDS TWD). TWR is faster, but with lower accuracy. While SDS-TWR is slower with higher accuracy. In conclusion, TWR can be used in scenarios which require real-time position updates with low accuracy. However, in scenarios where accuracy is of high priority, SDS-TWR will prove to be a better option. The variance of measurement errors in ranges in TWR case is around 0.12m and for SDS-TWR case 0.04m. Moreover, due to added error by antenna delay in TWR case, there is always some constant bias in measurements of around 1m to 0.6m depending on the antenna pair. In case of SDS-TWR, bias is around 0.4m-0.01m. For circumventing this issue, we added a calibration step in our implementation where devices can be calibrated at the origin to account for this constant bias. Later, we created a heatmap of the bias values in different location of the room, and used that to further improve our readings. This resulted in significant improvement in accuracy.

In positioning phase, we implemented Static and Dynamic positioning algorithms. Static algorithm uses Non-Linear Least square estimation (NLS), which gives very accurate measurements of static location, but its takes a long time to gather enough data to achieve specific accuracy. In case of TWR, convergence is achieved after almost 500 readings, which takes around 17 seconds at one point. For SDS-TWR, as it gives more accurate ranges, it takes only 40 readings to converge, which is equivalent to 12 seconds. We analyzed different initialization parameters used, to come up with values that leads to minimum error and faster convergence of the algorithm. We also studied the error in positioning at different locations in the room and created heatmaps to understand the behavior of the system. This helped us to understand how positioning is affected by the location of the tag relative to the anchors.

Dynamic positioning uses Extended Kalman Filtering techniques to give position (and velocity) updates in real-time. Although in dynamic case the accuracy is not as good as static case, but dynamic positioning can track changes in position quite accurately. EKF algorithm is analyzed by visual inspection of the deviation of tag's estimated track from the actual track. In real world scenarios, dynamic positioning can track human movements very accurately.

In error analysis phase, we studied sources of error in both ranging and positioning phases. We found out that many factors such as clock drifts, antenna delays, environmental factors etc. adds to the error. Hence it is important to pre-analyze the environment and other factors, and reduce the errors due to these factors.

The prototype created for demonstration in this project is a simple user interface running on MATLAB. A user have to connect the tag device to the USB port of PC running MATLAB first. Once the tag boots up, one should wait at least 30 secs for the core to get heated in order to prevent errors due to cpu clock. After that user can start the positioning system by simply clicking the start button. Estimated position of the tag will be shown in a figure in the user interface. We provided two different methods of calibration or bias-error correction. One is the calibration in which bias is corrected by calibrating for the errors in positioning at the origin. And other is the bias correction using heatmap generated for the entire room.

Finally, it can be concluded that UWB antenna (Decawave DW1000) with high speed clock processors (ARM) provides a good hardware to get centimeter level accuracy in positioning. Time-of-flight method works well in regions which are within line of sight.

This project has provided us a valuable practical experience implementing theory in real life. We learned about the implementation of communication protocols.

References

[1] Thuy Mai (2015) **NASA: Global Positioning System History.** Accessed april 2016. [Online]. Available: http://www.nasa.gov/directorates/heo/scan/communications/policy/GPS_History.html

[2] decaWave (2014) **EVK1000 USER MANUAL (version 1.07).** Retrieved from: http://www.decawave.com/sites/default/files/product-pdf/evk1000_user_manual_v107.pdf

[3] decaWave (2013) **PRODUCT INFORMATION: DW1000.** Retrieved from: http://www.decawave.com/sites/default/files/product-pdf/dw1000-product-brief.pdf

[4] decaWave (2014) **DATASHEET: DW1000.** Retrieved from: http://thetoolchain.com/mirror/dw1000/dw1000_datasheet_v2.04.pdf

[5] decaWave (2016) **DATASHEET: AGILENT 53230A.** Retrieved from: http://literature.cdn.keysight.com/litweb/pdf/5990-6283EN.pdf?id=1942617

[6] IEEE Computer Society Sponsored by the LAN/MAN Standards Committee (2011) **802.15.4-2011 - IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)** Retrieved from: https://standards.ieee.org/findstds/standard/802.15.4-2011.html

[7] Bensky, Alan. **Wireless Positioning Technologies and Applications**. Norwood, MA: Artech House, INC., 2008.

[8] Decawave (2015) **DecaRanging ARM Source Code Guide**.

[9] Haupt, Kasdin, Keiser and Parkinson(1995) **An Optimal Recursive Iterative Algorithm for Discrete Nonlinear Least-Squares Estimation**.Retrieved from:https://einstein.stanford.edu/content/sci_papers/papers/HauptG_1995_57.pdf

[10] Khan, Sottile and Spirito (2013) **Hybrid Positioning through Extended Kalman Filter with Inertial Data Fusion**.Retrieved from:http://www.ijiee.org/papers/281-N012.pdf

[11] Croeze, Pittman and Reynolds **Solving Nonlinear Least-Squares problems with the Gauss-Newton and Levenberg-Marquardt methods** .Retrieved from:https://www.math.lsu.edu/system/files/MunozGroup1%20-%20Paper.pdf