**POLITECNICO DI MILANO**
**Master of Science Degree in Automation and Control Engineering**
**SCHOOL OF INDUSTRIAL AND INFORMATION ENGINEERING**

# Online Localization Using UWB Devices

**Supervisor: Prof. Marcello Farina**
**Co-supervisor: Prof. Luca Bascetta**

**Candidate:**
**Dong Liang**
**Matricola: 898356**

**Academic Year 2019-2020**

# Abstract

In this thesis, we propose an optimized trilateration positioning approach based on Kalman Filter and on data measured using Decawave DWM1001 UWB modules. First of all, we develop a custom function interface to obtain the data from DWM1001 in real-time. The interface is divided into an on-board C program to send original data to the host computer via serial port and a MATLAB script to collect original data and store them synchronously in the form of an array. Secondly, the UWB sensor and the accelerometer of the DWM1001 module are analyzed and evaluated in detail. We show that the measurement noises of both the UWB ranging sensor and the accelerometer are Gaussian white noises. The third part of the work consists of the design and implementation of a Kalman Filter. With the original UWB data and acceleration data, we performed Kalman Filtering in MATLAB. With the Kalman Filter, measurement noise effect is significantly reduced, and the optimized distance estimate can be obtained. After that, two trilateral positioning methods are proposed to calculate the coordinates of the UWB module in 3D space. The first method uses the idea of least squares to find the coordinates by minimizing a suitable cost function. The second method calculates the space coordinates by a numerical analysis approach. Finally, experiments in a real environment are carried out with all methods mentioned above. The experimental results prove that the methods proposed in this work can achieve centimeter-level positioning.

# Sommario

In questa tesi viene proposto un algoritmo di localizzazione basato su un filtro di Kalman, per applicabile a dati ottenuti tramite moduli Decawave DWM1001 UWB.

Prima di tutto è stata sviluppata un'interfaccia per ottenere i dati da DWM1001 in tempo reale. L'interfaccia è stata divisa in due parti: una parte è caratterizzata da un programma C integrato sul dispositivo per inviare i dati originali al computer attraverso una porta seriale; la seconda parte è caratterizzata da uno script di MATLAB che raccoglie i dati in modo sincrono memorizzandoli sotto forma di array. In secondo luogo, è stato analizzato e valutato in dettaglio il sensore di UWB e l'accelerometro del modulo DWM1001, e si è rilevato che i rumori di misura, sia del sensore UWB che dell'accelerometro, sono rumori bianchi Gaussiani. Il passo successivo è stato la progettazione e l'implementazione di un filtro di Kalman, i cui parametri del filtro e sono calcolati in base alle analisi condotte nella parte precedente. Applicando il filtro di Kalman ai dati originali di UWB e dell'accelerometro si è verificata una riduzione significativa del rumore di misura ed sono stata ottenute stime di distanza ottimali.

Successivamente, sono stati anche proposti due metodi di posizionamento trilaterale per il calcolo delle coordinate del modulo UWB nello spazio tridimensionale. Il primo metodo utilizza l'idea dei minimi quadrati per trovare le coordinate riducendo al minimo la funzione obbiettivo. Il secondo metodo consiste nel calcolare le coordinate spaziali mediante un approccio di analisi numerica.

Infine, tutti i metodi sopra menzionati sono stati messi alle prove in un ambiente reale. Dai risultati ottenuti dagli esperimenti compiuti si dimostra che tali metodi forniscono buoni dati di posizionamento con un range di errore entro i 10 cm.

# Contents

# List of Figures

# List of Tables

# List of abbreviations

| Abbreviation | Corresponding Word/Phrase |
|---|---|
| GPS | Global Positioning System |
| IPS | Indoor Positioning System |
| A-GPS | Assisted GPS |
| WPS | Wi-Fi Positioning System |
| RSSI | Received Signal Strength Indication |
| UWB | Ultra-wideband |
| MEMS | Microelectromechanical Systems |
| I2C | Inter-Integrated Circuit |
| SPI | Serial Peripheral Interface Bus |
| API | Application Programming Interface |
| RTLS | Real-Time Location System |
| TWR | Two-Way Ranging |
| PCB | Printed Circuit Board |
| EDF | Empirical Distribution Function |
| PANS | Positioning and Networking Stack |
| UART | Universal Asynchronous Receiver/Transmitter |
| BLE | Bluetooth Low Energy |
| CSV | Comma-Separated Values |
| IDE | Integrated Development Environment |
| CFM | Cost Function Minimizing Method |
| MS | Matrix Solution Method |
| RSS | Root Sum Square |

# Chapter 1

# Introduction

## 1.1 Background

In the field of positioning, the Global Positioning System (GPS) is nowadays a mature technology after decades of development and application. By communicating with satellites, GPS can quickly determine the location and altitude of the user terminal on the earth surface. With just a smart-phone, ordinary people can obtain positioning information with an accuracy up to 3 meters. High-precision positioning information is used in many applications, such as navigation. This greatly facilitates the lives of ordinary people. In the business field, positioning information has the potential to leverage commercial applications such as customized advertising and social networks. A wide range of applications have made positioning technology attract more attention. However, satellite positioning can only be used in an outdoor environment. Once indoor, satellite positioning cannot be used because the navigation signal decays fast. To solve the problem of how to locate objects and people in the buildings, Indoor Positioning Systems (IPS) are proposed.

Indoor positioning is a field that has continued to flourish in the past decade. Accurate indoor localization is becoming more and more important due to the increased use of augmented reality, health care monitoring, personal tracking, inventory control and other indoor location-aware applications. For more than a decade, in order to solve the "last mile" problem of positioning and navigation, technology giants and research institutions have carried out a lot of research on indoor positioning technology. A variety of indoor positioning technologies have been developed. For example, Assisted GPS (A-GPS) technology that combines a GPS signal and mobile phone base station signals for indoor

positioning. Another example is Wi-Fi positioning system (WPS). WPS uses the characteristics of nearby Wi-Fi hotspots and other wireless access points to locate a device. Based on measuring the intensity of the received signal (received signal strength indication or RSSI) and the method of "fingerprinting", the accuracy of WPS reaches the meter level. The equipment used in this thesis applies an alternative indoor positioning technology, i.e., Ultra-wideband (also known as UWB). UWB is a positioning technology with high transmission rate, low transmit power and strong penetrating ability. Under ideal circumstances, the positioning accuracy of UWB is generally up to ten centimeters. There are also many other indoor positioning techniques such as ZigBee positioning, Bluetooth positioning, RFID positioning, acoustic positioning, geomagnetic positioning, etc. The accuracy of these positioning technologies ranges from centimeters level to meters level. Figure 1.1 shows the accuracy of these technologies.

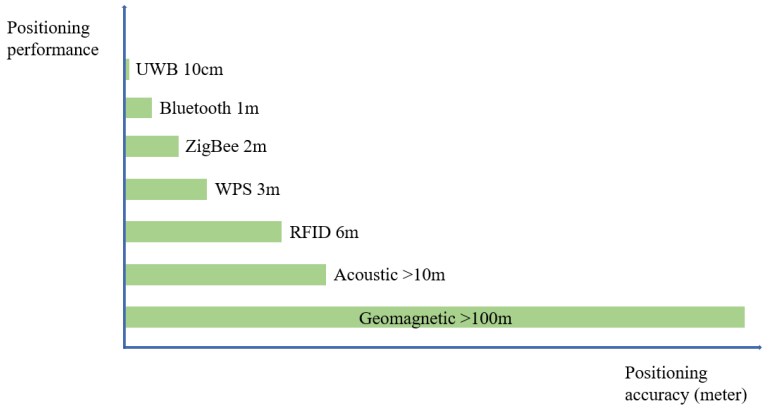



**Figure 1.1: Common indoor positioning technology accuracy**

The development of indoor positioning technology has two directions: wide-area indoor positioning technology and local indoor positioning technology. Wide-area indoor positioning technology, for example A-GPS and geomagnetic positioning, usually requires the transformation of equipment modules such as base stations and mobile phone chips, which is costly and has a long time period. Local indoor positioning technology has low cost, short cycle as well as higher positioning accuracy, which is a better option for commercial promotion and operation. In the current research progress, the positioning accuracy of the wide-area indoor

positioning technology is far worse than the local indoor positioning technology. Among the available options, UWB positioning is a highly accurate indoor positioning method. The UWB positioning delay time is much shorter than other indoor positioning technologies such as Bluetooth positioning or WPS. With the reduction of UWB chip cost in recent years, UWB has become the first choice for indoor high-precision positioning. Apple iPhone 11 series products launched in 2019 have built-in UWB chips.

Although UWB technology has great advantages in accuracy, compared to other indoor positioning technologies, complicated indoor positioning environment will have a serious impact on positioning accuracy. For example, as it is can be seen in Figure 1.2, the UWB frequency bands allowed in EU region is 6000MHz to 9000MHz. At such a high frequency, the pulse has poor penetration to walls and floors. When the UWB device is facing away from the base station, or obstacles exist between the UWB device and the base station, the positioning results may display errors.

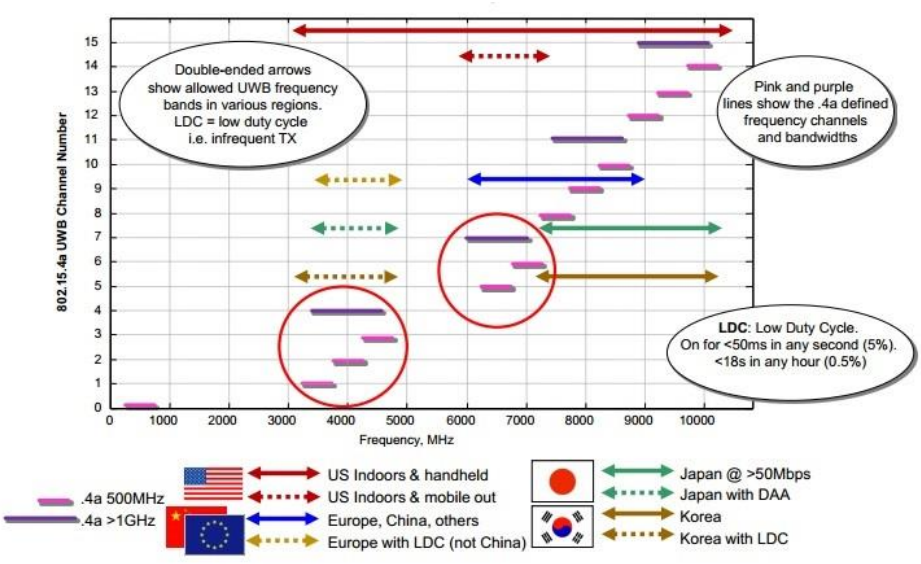



**Figure 1.2: Allowed UWB frequency bands in various regions**

In order to optimize the positioning result, this thesis propose an algorithm based on Kalman filter to process historical data, combining UWB measurement results and accelerometer data.

## 1.2 Brief description of the work

In this thesis we used Decawave UWB devices. First of all, in order to obtain the original ranging and acceleration data of the UWB device, we developed a custom interface for the device. This interface is divided into two parts. The first part is an on-board user application in C language. With the embedded on-board user application, the original ranging and acceleration data can be sent to the host computer via USB serial communication. The other part of the interface is a MATLAB script. With the MATLAB script, the original data on the serial port can be collected and stored synchronously in the form of an array. In Chapter 3, the UWB sensor and the accelerometer of the DWM1001 module are analyzed and evaluated. The original UWB and acceleration data are processed by an ad-hoc Kalman predictor to obtain an optimal estimate of the distance. Then, two trilateral positioning methods are used to obtain the coordinates of the UWB device in the three-dimensional space. The first method uses the idea of least squares to find the coordinates of the UWB device by minimizing a cost function. The second method calculates the space coordinates of UWB devices by suitable numerical analysis approach. Finally, an experiment in real environment has been carried out to test the techniques mentioned above.

## 1.3 Structure of the thesis

The thesis is organized in the following way: in this chapter, a brief introduction to the topic is given, along with an introduction to the work-flow of this thesis; in Chapter 2, the Decawave UWB device is described in detail; in Chapter 3, original data obtained from DWM1001 module is analyzed; in Chapter 4, the data communication interface is illustrated; in Chapter 5, the principles of the Kalman filter is described; in Chapter 6, two trilateral positioning algorithms are discussed; in Chapter 7, the experiment setups and results are presented; finally, Chapter 8 concludes this thesis work, where also the possible future developments are clarified.

# Chapter 2

# Hardware Introduction

In this chapter, the UWB device DWM1001 development board is introduced. The introduction discusses performance and parameters of the UWB chip and embedded accelerometer. A simple test is also performed to show DWM1001 actual performance and to motivate the necessity of data optimization, subject of the following chapters.

## 2.1 DWM1001 Development Board

The UWB positioning device used in this thesis is Decawave MDEK1001 Development Kit (MDEK1001). This development kit includes 12 DWM1001 Development Boards (DWM1001). Figure 2.1 shows the DWM1001 module and its case.

**Figure 2.1: DWM1001 and MDEK1001 case**

Decawave DWM1001 is an Ultra-Wide-bandwidth (UWB) module based on Decawave DW1000 IC. DWM1001 module integrates antennas and an on-board Microelectromechanical Systems (MEMS) digital output motion sensor (accelerometer) LIS2DH12.

In the table 2.1, main parameters of DWM1001 are listed.

**Table 2.1: Main parameters of DWM1001**

| | |
|---|---|
| Voltage Supply | 3.6-5.5V |
| Current Level | 500mA |
| Channel | 5-6.8MHz |
| Tag Update Rate | Up to 10Hz |
| Cluster Update Rate | Up to 150Hz |
| Anchors Number in a Cluster | Up to 30 anchors |
| Distance Measurement Range | Up to 30 meters |
| Location Accuracy | typical <10 cm |

The LIS2DH12 is an ultra-low-power high performance three-axis linear accelerometer with digital I2C/SPI serial interface standard output. In normal mode, its accuracy reaches 4$mg$ (approximately equal to 0.039$m/s^2$). Figure 2.2(a) shows the default three-axis pointing of the accelerometer. Figure 2.2(b) shows the welding position and direction of accelerometer of the DWM1001 module.

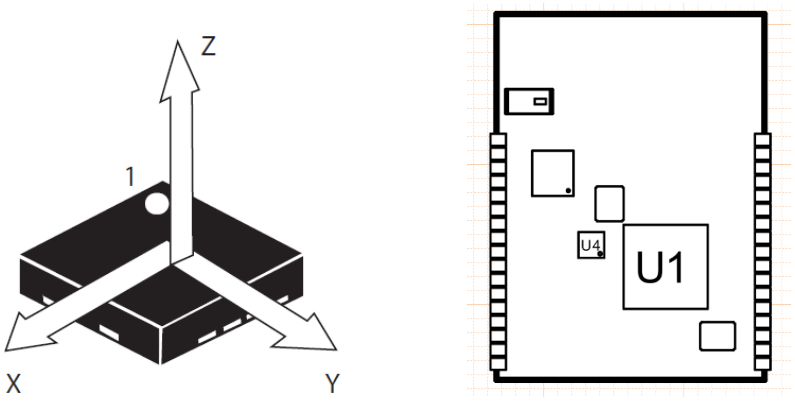



**Figure 2.2: Accelerometer LIS2DH12 (a) default three-axis pointing of the accelerometer LIS2DH12; (b) the welding position (U4) and direction (dot on the bottom right) of accelerometer LIS2DH12 of the DWM1001 module**

Accompanying the UWB positioning hardware device DWM1001, Decawave also provides software and development environment such as embedded firmware binary location stack and gateway firmware to quickly evaluate its features and performance. With the Decawave Android app, users can view the location and configure the system on an Android device. Users can also customize their own functions via APIs and re-flash custom firmware onto boards over USB.

By configuring DWM1001 units in a different way, such as anchor mode and tag mode, a wireless Real-Time Location System (RTLS) can be established. The tag units can transmit signals to nearby anchor units and use the UWB ranging algorithm in the original firmware to calculate the distance with nearby anchor units.

According to the unit type definition of Decawave, a unit can be defined as a tag, an anchor, or an initiator anchor. A tag unit is an object whose coordinates are unknown. The main task in this thesis is to locate the position of the tag. An anchor unit is a fixed base station, whose position is artificially set. A tag unit can communicate and execute range measurements with surrounding anchors with UWB signals. An anchor unit can also do it vice versa. This two-way signal communication called Two-Way Ranging (TWR), would effectively improve ranging accuracy. An initiator anchor is a necessary main anchor of a positioning network. If there is no existing network, the initiator anchor will start a new one. Then one by one the other anchors will join.

# Chapter 3

# Analysis and Verification of DWM1001 Sensors

In this chapter, the original data of DWM1001 module are analyzed. Section 3.1 shows a set of original data including UWB distance measurement and acceleration data. Calibration and Mean Normalization are executed for distance measurement and acceleration measurement. In Section 3.2, the Anderson's Whiteness Test is briefly introduced and implemented on the normalized datasets. In Section 3.3, the distribution of the measurement noise is analyzed. We also compute the empirical distribution function and its variance.

## 3.1 Original measurement data analysis

In this section, we perform a static test for DWM1001 module. We analyze the obtained data. We use two DWM1001 modules, one as a tag unit, the other as an anchor unit.

Two units stay on the same plane, facing each other. Both units keep still. The true distance between two units is 2.54m. The test duration is about 45s, update rate is 10Hz. A total of 451 sets of original data are obtained. Table 3.1 lists, as an example, 10 sets of original data.

**Table 3.1: An example of original dataset**

| Number | Time ($s$) | Distance ($m$) | $A_x(m/s^2)$ | $A_y(m/s^2)$ | $A_z(m/s^2)$ |
|---|---|---|---|---|---|
| 1 | 15.2776 | 2.5060 | 9.9568 | -0.2058 | 0.1862 |
| 2 | 16.2776 | 2.5620 | 9.9568 | -0.2352 | 0.1960 |
| 3 | 17.2776 | 2.4820 | 10.0940 | -0.2352 | 0.2156 |
| 4 | 18.2776 | 2.5380 | 10.0940 | -0.2352 | 0.1862 |
| 5 | 19.2776 | 2.5060 | 10.0940 | -0.1274 | 0.0784 |
| 6 | 20.2776 | 2.5100 | 10.0646 | -0.2254 | 0.1666 |
| 7 | 21.2776 | 2.5670 | 10.0940 | -0.0882 | 0.1862 |
| 8 | 22.2776 | 2.5010 | 10.0744 | -0.1078 | 0.1960 |
| 9 | 23.2776 | 2.4960 | 10.0744 | -0.2254 | 0.1568 |
| 10 | 24.2776 | 2.5200 | 10.0940 | -0.1960 | 0.1960 |

For simplicity of display, all data keep four decimal places.

In Table 3.1 Time represents the timestamps when the data are generated. The timestamps depend on the module system start-up time. The time interval between two datasets is exactly 0.1s. The Distance is the UWB measurement result. In the complete dataset, the maximum distance data is 2.6230m and the minimum distance data is 2.4350m. The peak-peak value therefore is 0.1880m. $A_x$, $A_y$, $A_z$ represent the accelerations in three directions. As described in Section 2.1, $A_x$ points to the earth center. $A_y$ points straight ahead. In this test, $A_y$ of the tag unit pointed to the anchor unit. $A_z$ points to the side of a module, perpendicular to $A_x$ and $A_y$.

In Figure 3.1 and 3.2, plots of the time varying data of both distance measurements and 3-axis acceleration measurements are depicted.

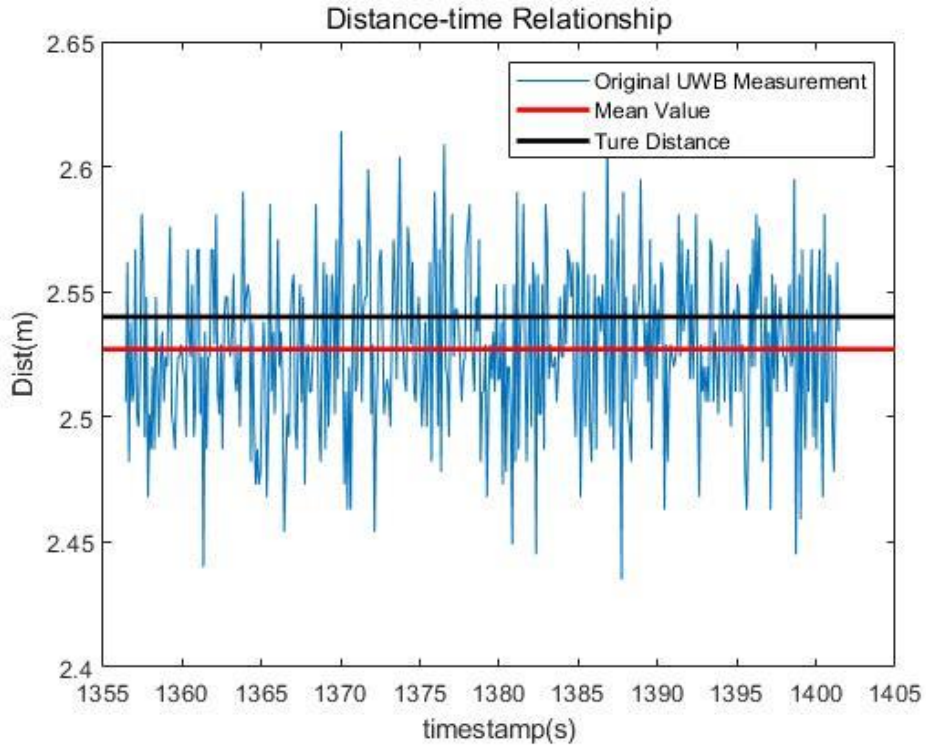



**Figure 3.1: Original measurement, mean value and true value of the distance measurements**

In Figure 3.1, the mean value of distance measurement is represented by the red straight line. The true value is represented by the black straight line. This error may be caused by the thickness of the DWM1001 case. And because the measurement noise affect, the distance measurement fluctuates.

In order to quantify the degree of dispersion of the data, we need to calculate the variance of the data set. Considering the UWB ranging data as an example, the variance is calculated as:

$$Var(Dist) = E[(Dist - mean(Dist))^2] \tag{3.1}$$

The following Table 3.2 shows the statistical results of the distance measurement.

**Table 3.2: Distance measurement statistical results**

| Term | Value |
|---|---|
| Mean | 2.5271m |
| Variance | 0.0010m2 (or 1033.5mm2) |
| True Value | 2.54m |

In the following Sections 3.2 and 3.3, the measurement noise will be analyzed in detail.

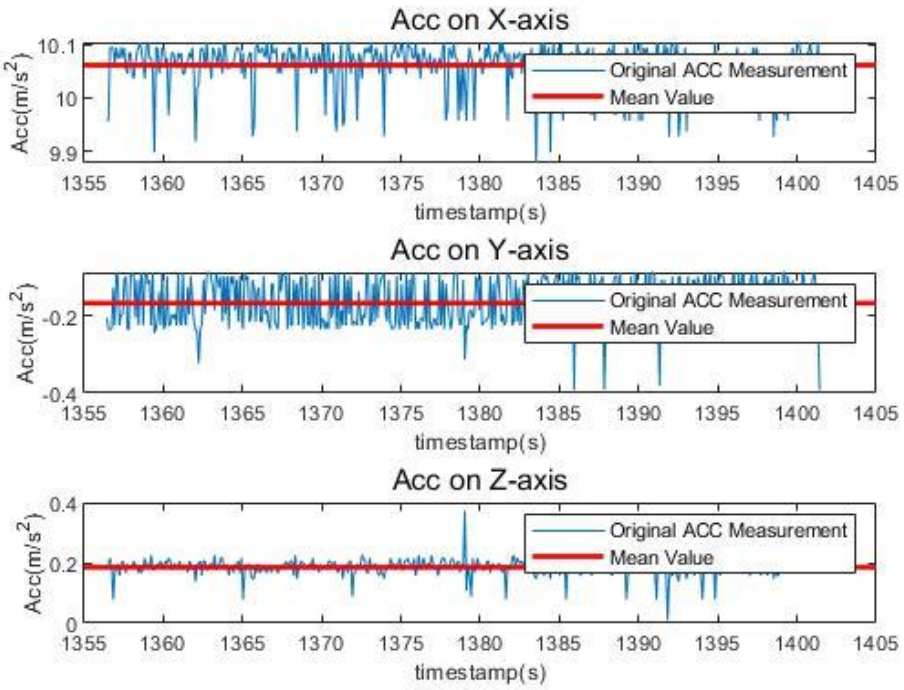



**Figure 3.2: Original measurement and mean value of the 3-axis acceleration**

In Figure 3.2, we can see that the mean values of $\boldsymbol{A_y}$ and $\boldsymbol{A_z}$ are biased. This may be due to the angle deviation when soldering the accelerometer to the PCB plus the experimental plane was slightly tilted. To calibrate the accelerometer, the module needs to be kept still in different postures, upright, flat or side i.e., so that acceleration data can be obtained for each axis, without gravitational acceleration effect. The expression of the calibration is:

$$A^{C}_{one\ axis} = A_{one\ axis} - mean(A_{one\ axis}) \quad (3.2)$$

Laying the module flat, face up, we can obtain measurement for x axis. The original mean acceleration of three axis listed in Table 3.3.

**Table 3.3: Original mean 3-axis acceleration**

| Axis | Original Mean Acceleration $(\boldsymbol{m/s^2})$ |
|---|---|
| X | 0.3332 |
| Y | -0.1678 |
| Z | 0.1853 |

Following Figure 3.3 shows the calibrated 3-axis acceleration data.

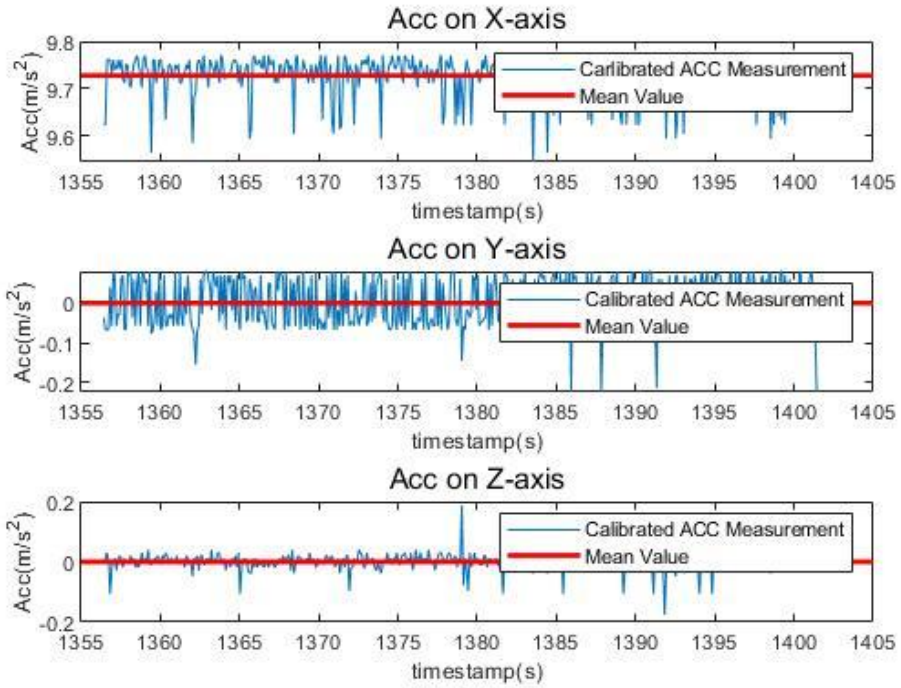



**Figure 3.3: Calibrated measurement and mean value of 3-axis acceleration**

After calibration, the mean value of the accelerations in the Y and Z reduced to 0. Also, the mean value of the acceleration in the X axis is 9.7279. The calibration process will not change the variance of the acceleration dataset. The variance for 3-axis are listed in Table 3.4.

**Table 3.4: Variance of 3-axis acceleration measurement**

| Axis | Measurement Variance |
|---|---|
| X | 0.0017 |
| Y | 0.0039 |
| Z | 7.2548e-04 |

We can see that noises exist in both measurements. To identify the type of the noises, the normalization can be done to signal series. Here we use distance measurement data as an example. The expression of Mean Normalization of the distance measurement is:

$$Dist^N = \frac{Dist - mean(Dist)}{\max(Dist) - \min(Dist)} \tag{3.3}$$

The distance measurement signal after mean normalization can be plotted as Figure 3.4.

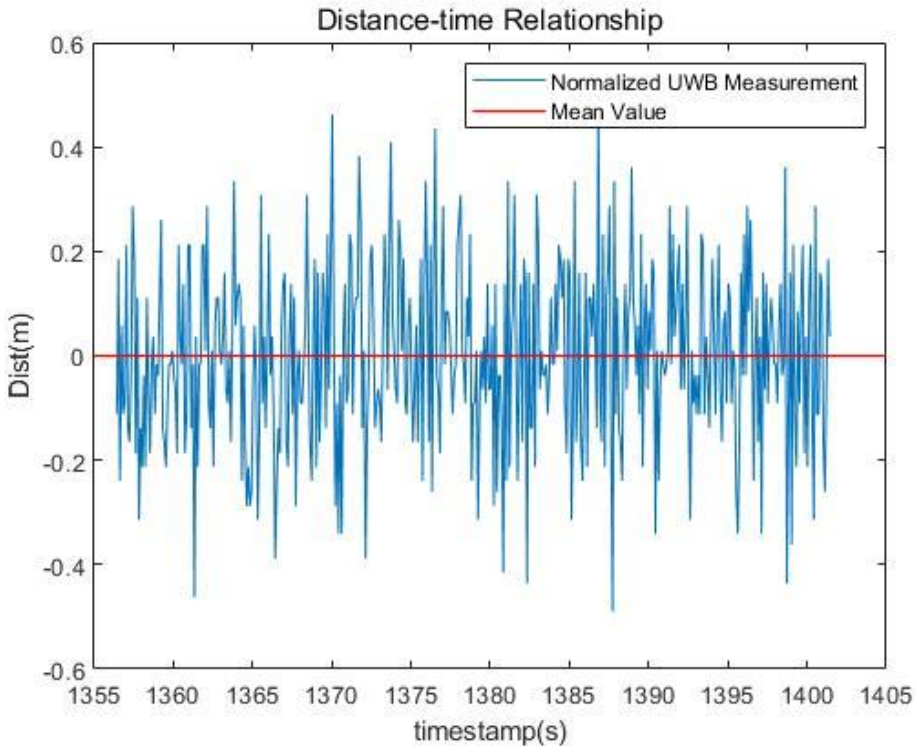



**Figure 3.4: Normalized distance measurement signal**

With the normalized signals, we can identify the type of the noises which

affect the distance measurement and acceleration measurement. To determine the whiteness of the noise, in next section, an Anderson's Whiteness Test is carried out.

## 3.2 Anderson's Whiteness Test

To apply the Kalman Filter and optimize the positioning accuracy, the type of noises must be identified. If and only if the noises are Gaussian White Noise, the Kalman Filter can be applied to reduce the effect of the noises. In this section, the Anderson's Whiteness Test is introduced and performed to identify the noises of distance measurement signal and acceleration signal.

### 3.2.1 Principle of Anderson's Whiteness Test

Assume there is a signal $\mathcal{E}(\cdot)$ under test with N samplings, the correlation function of the signal can be computed as:

$$\hat{\gamma}_{\mathcal{E}}(\tau) = \frac{1}{N}\sum_{t=\tau+1}^{N} \mathcal{E}(t)\mathcal{E}(t-\tau)\,, 0 \le \tau \le M, M \ll N-1 \quad (3.4)$$

The normalized sample correlation function can be computed as:

$$\hat{\rho}_{\mathcal{E}}(\tau) = \frac{\hat{\gamma}_{\mathcal{E}}(\tau)}{\hat{\gamma}_{\mathcal{E}}(0)}, \tau > 0 \quad (3.5)$$

If the signal $\mathcal{E}(\cdot)$ is a zero-mean white noise, the normalized correlation function $\hat{\rho}_{\mathcal{E}}(\tau)$ is asymptotically normally distributed:

$$\sqrt{N}\hat{\rho}_{\mathcal{E}}(\tau) \sim As\mathcal{N}(0,1), \forall \tau > 0 \quad (3.6)$$

and $\hat{\rho}_{\mathcal{E}}(\tau), \tau = 1, 2, \ldots$ can be considered as independent values of a

Gaussian variable:

$$\hat{\rho}_{\mathcal{E}}(i) \perp \hat{\rho}_{\mathcal{E}}(j), \forall i \neq j \tag{3.7}$$

The probability that $-\beta \leq \sqrt{N}\hat{\rho}_{\mathcal{E}}(\tau) \leq \beta$ is given by the solution of integer of the grey area in the Figure 3.5.

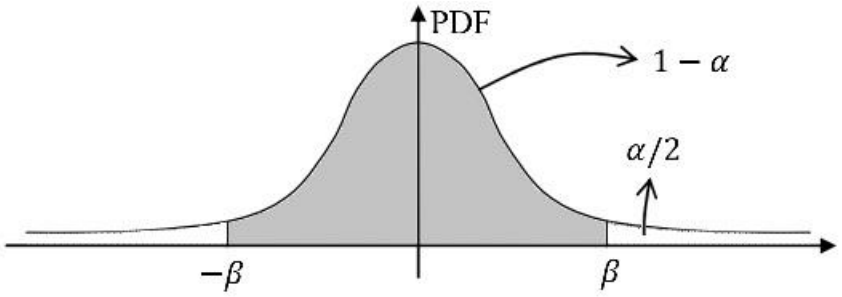



**Figure 3.5: Probability Density Function**

Set a possible small confidence interval $\alpha \in (0,1)$, i.e. the probability $\alpha$ that $\hat{\rho}_{\mathcal{E}}(\tau) < -\beta$ or $\hat{\rho}_{\mathcal{E}}(\tau) > \beta$, and compute the $\beta$. Count the number of samples which $\sqrt{N}\hat{\rho}_{\mathcal{E}}(\tau) \notin [-\beta, \beta]$ and denote as $N_\alpha$. If $N_\alpha/N < \alpha$, the signal $\mathcal{E}(\cdot)$ is assumed as white noise, other it is not.

## 3.2.2 Implementation of Anderson's Whiteness Test

We perform an Anderson's Whiteness Test for the normalized distance measurement data we obtained in Section 3.1. [15]

The MATLAB function script fragment is as follows:

```
alpha=0.1;   %confidence level
gamma=covf(x,floor(N/10));
%covariance calculation with System Identification Toolbox
rho=gamma(2:end)/gamma(1);
%normalized correlation function
beta=norminv(1-alpha/2);
```

```
nalpha=length(find(sqrt(N)*rho>beta))+length(find(sqrt(N)
*rho<-beta));
f=nalpha/length(rho);
%f=Nalpha/N
```

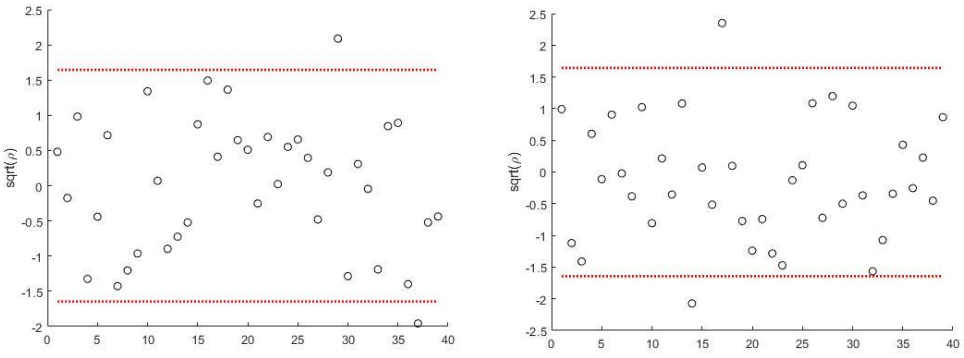
For complete MATLAB script, please refer to Appendix A.

After importing the normalized distance measurement data and acceleration data into Anderson's Whiteness Test script, we have following results in Table 3.5:

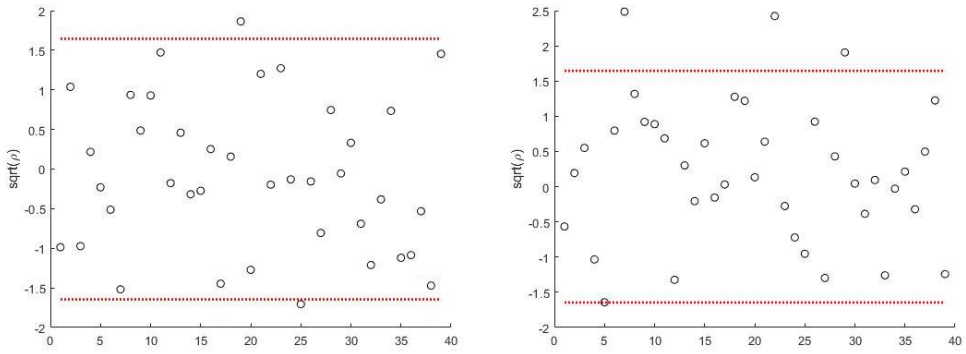
**Table 3.5: Results of Anderson's Whiteness Test**

| **Terms** | $\boldsymbol{\alpha}$ | **Frequency of violation** | **Result** |
|---|---|---|---|
| Distance | 0.10 | 0.051282 | Pass |
| $\boldsymbol{A_x}$ | 0.10 | 0.051282 | Pass |
| $\boldsymbol{A_y}$ | 0.10 | 0.076923 | Pass |
| $\boldsymbol{A_z}$ | 0.10 | 0.076923 | Pass |

The following Figure 3.6 shows the noise points distribution of $\sqrt{N}\hat{\rho}_{\mathcal{E}}(\tau)$. The upper and lower red dotted lines indicate $\beta$ and $-\beta$ respectively. Subplots (a), (b), (c), (d) are respectively drawn from dataset distance,$\boldsymbol{A_x}$, $\boldsymbol{A_y}$ and $\boldsymbol{A_z}$.




**(a): Distance** **(b): x-axis**




**(c):y-axis** **(d): z-axis**

**Figure 3.6: Anderson's Whiteness Test result**

These results prove that the measurement noises of the UWB ranging measurement and the accelerometer measurement are both White Noises. The results ensure the feasibility of implementation the Kalman filter in the subsequent chapter.

## 3.3 Analysis of Empirical Distribution Function

In Section 3.2, we used the Anderson's Whiteness Test to confirm the whiteness of both the measurement noises of UWB ranging sensors and accelerometer. In this section, the empirical distribution functions of both measurements are plotted and analyzed.

### 3.3.1 Empirical Distribution Function

In statistics, an Empirical Distribution Function (EDF) is the distribution function associated with the empirical measure of a sample. Its value at any specified value of the measured variable is the fraction of observations of the measured variable that are less than or equal to the specified value. In our cases, the plots of EDF visually show the distribution of noises.

### 3.3.2 EDF plots of the noises

In MATLAB, command "`ecdf`" draws a stairstep graph of the evaluated function by using the stairs function. Specify "`'Bounds','on'`" to draw the confidence bounds in the graph. Take the UWB ranging data as an example, the MATLAB function script fragment is:

```
ecdf(dist-mean(dist)); hold on
%plot the noise of original distance measurement
ecdf(noise,'Alpha',0.1,'Bounds','on'); grid on
%plot ecdf of the theoretical white noise
%set confidence interval 10%
```

Besides the original data, a theoretical white noise signal is necessary to do the comparison between the actual experiment signal and ideal simulated signal. Generate a white noise with the same mean value and same variance of distance measurement, we use command "`randn`" as follow:

```
noisex=0.04123105626*randn(451,1);
%0.04123…is the standard deviation of the dist signal, equals to sqrt(var)
```

From this, we get the following four EDF figures.

Figure 3.7 is the EDF plots of the UWB distance measurement. In which the blue line is the empirical data and the red line is a theoretical Gaussian white noise with mean value 0 and variance 0.001. The red dotted line is the bounds of 10% confidence interval of the theoretical white noise. Comparing with the theoretical white noise, the noise affecting the UWB ranging has very similar shape. The distance measurement noise can be therefore considered to be asymptotically normally distributed.

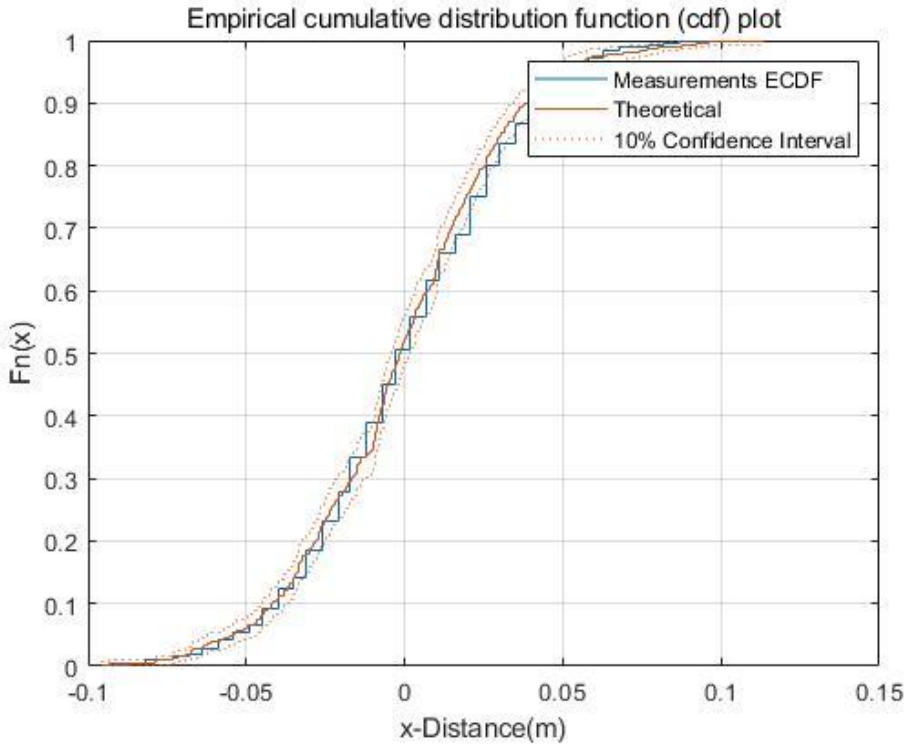



**Figure 3.7: Distance measurement EDF**

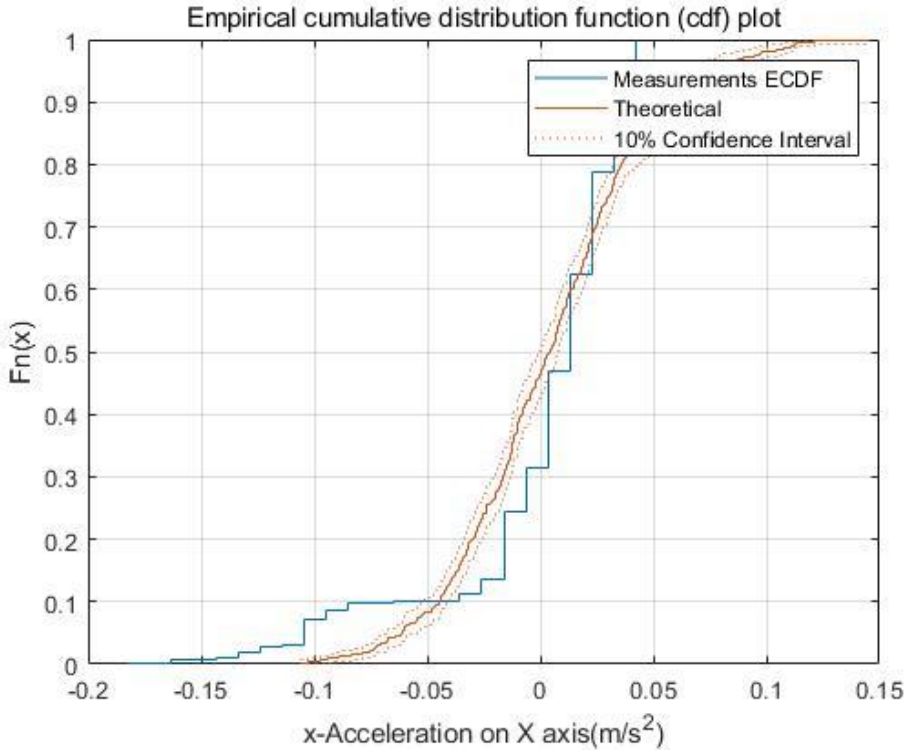



**Figure 3.8: X-axis acceleration measurement EDF**

Figure 3.8 is the EDF plot of the X axis (up and down) acceleration. More than 90% noise distribute within the interval of $-0.04m/s^2$ to $0.05m/s^2$. The right side of the data converges quickly, but the left side is significantly slower than the simulated theoretical value.

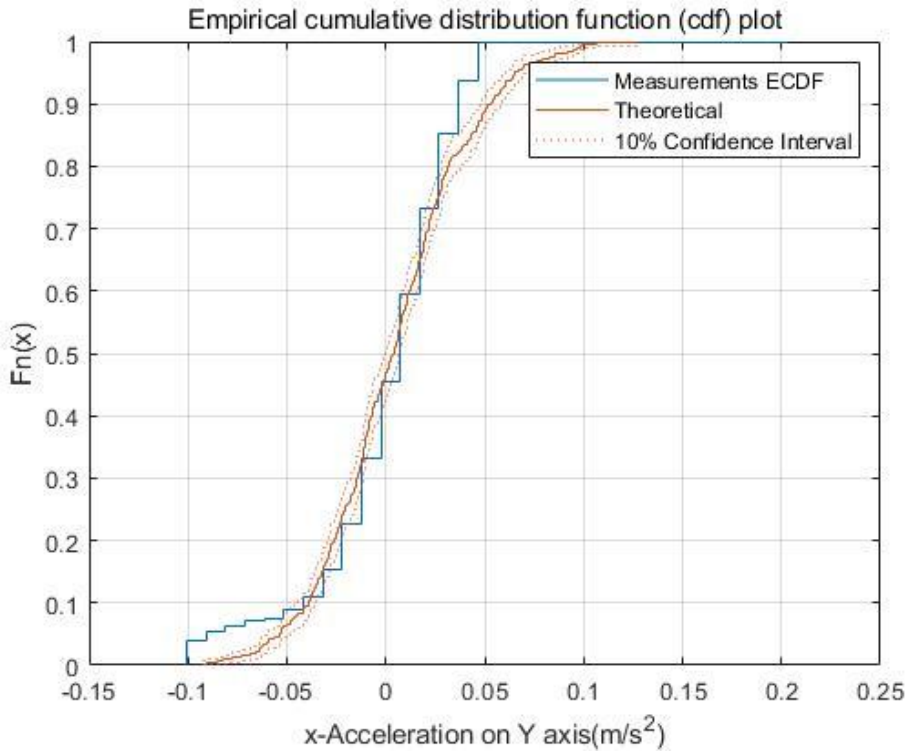



**Figure 3.9: Y-axis acceleration measurement EDF**

Figure 3.9 is the EDF plot of the Y axis (front and back) acceleration. Comparing with the X axis, we can see that the noise on Y axis is more symmetrical distributed. The parts greater than zero and less than zero are close to 50%. The noise data converges quickly on both sides.

Figure 3.10 is the EDF plot of the Z axis (sides) acceleration. Comparing to the simulated theoretical white noise, it fits well in the central area with a normal distribution. The noise is distributed within or very close to the confidence interval. The convergence rate on the right is also faster than theory.

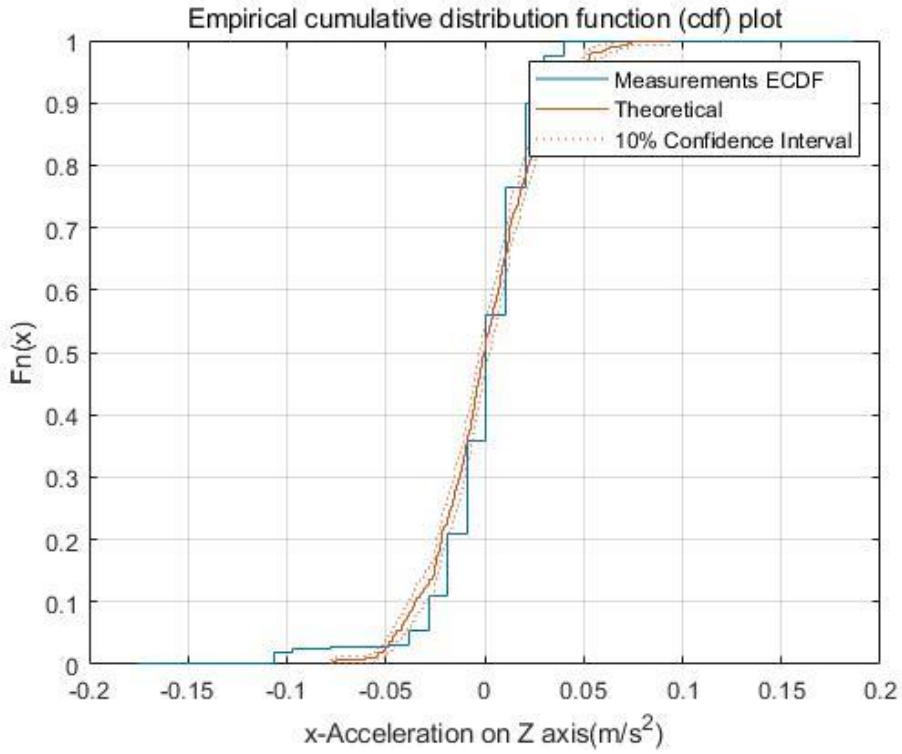



**Figure 3.10: Z-axis acceleration measurement EDF**

From Figure 3.8, 3.9 and 3.10, we can see that the noise distribution of the accelerometer basically fits the theoretical curve. Among the three axes, the noises on the Y and Z axis are more normally distributed than the X axis. The X axis points to the center of the earth when the DWM1001 module is upright. In this thesis, the horizontal movement of objects is studied. Therefore, we can obtain the accelerations in the horizontal directions from the Y and the Z axis.

# Chapter 4

# Data Communication Interface

To communicate with DWM1001 unit, a data communication interface is mandatory. In this chapter, we use C language and MATLAB to write a custom data communication interface. This interface divides into two parts. The first part is an embedded on-board program. The second part is a MATLAB script.

In Section 4.1, the factory firmware of DWM1001 unit is introduced. It also contains some application programming interfaces (API) introduction. In Section 4.2, we decide what kind of data we need, and we determine the function to be realized by the interface. In Section 4.3, the function and the development process of the on-board C program is described. In Section 4.4, the MATLAB script is presented.

## 4.1 Introduction of DWM1001 firmware

The DWM1001 module comes pre-loaded with embedded firmware, including the firmware Positioning and Networking stack (PANS) library for on-board development. The firmware embedded in the DWM1001 module basically provides two types of functions: the PANS API, and the PANS library which provides lower level functions. With the factory firmware, DWM1001 can be configured and controlled via its APIs. Two types of communication methods are provided to calling APIs. They are integrated access, such as C API, and external access, including Universal Asynchronous Receiver/Transmitter (UART) (or Serial Peripheral Interface, (SPI)) API and Bluetooth Low Energy (BLE) API. Figure 4.1 shows the architecture of DWM1001 factory firmware.

In this thesis, C API and UART API are used by two parts of the customized interface separately.

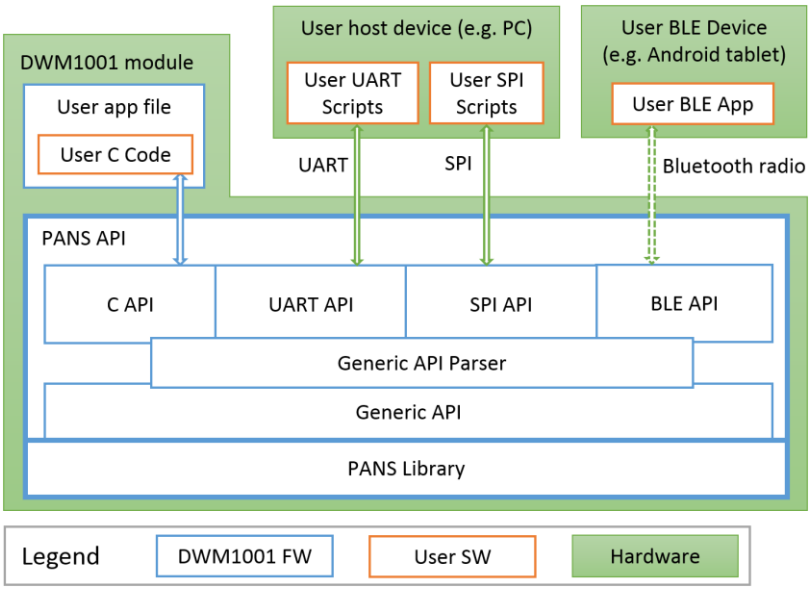



**Figure 4.1: Architecture of DWM1001 factory firmware**

### 4.1.1 UART API introduction

To call UART API, DWM1001 unit must communicate with a host device, such as a PC or a Raspberry Pi, via serial port. In this section, PC is the host device. To communicate with DWM1001 on a PC with Windows operating system, a terminal program is mandatory. In this work, Tera Terminal is recommended. Main parameters of serial communication are:

- Baud rate: 115200 bps
- Data bits: 8
- Stop bits: 1
- Check bits: none

After establishing connection with DWM1001 module, DWM1001 module would be in generic mode. The host device is acting as the initiator, while the DWM1001 module is the responder. A wakeup procedure, double clicking “Enter” key within one second, has to be executed before SPI/UART starts accepting commands: the module will switch into shell mode. In shell mode, UART APIs (SPI APIs) can be called in shell command form. Figure 4.2 shows the shell mode user interface of DWM1001 module.

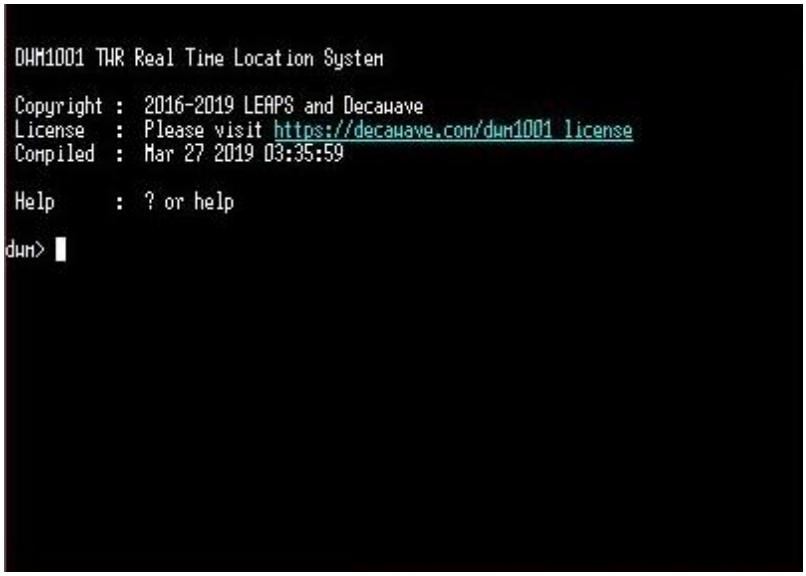



**Figure 4.2: Shell mode user interface of DWM1001 on terminal**

To retrieve real-time data from DWM1001 module, Table 4.1 lists several shell commands (UART/SPI APIs) which may be used in configuration step or data transmitting step.

**Table 4.1: List of shell command used**

| ***Shell command*** | ***Description*** |
|---|---|
| *les* | *Show distances to ranging anchors. Sending this command multiple times will turn on/off this functionality.* |
| *lec* | *Show distances to ranging anchors in CSV format. Sending this command multiple times will turn on/off this functionality.* |
| *av* | *Read ACC values.* |
| *nmi* | *Configures node to as anchor - initiator, active and reset the node.* |
| *nmt* | *Configures node to as tag, active and reset the node.* |
| *aurs* | *Set position update rate.* |
| *aps* | *Set position of the node.* |
| *aid* | *Read ACC device ID.* |

### 4.1.2 C API introduction

Besides UART API, we can code and make use of the C API functions in our own C files. The Decawave factory firmware reserves memory space for user software and provide source code for creating and starting threads. In this way, users are able to add custom functions inside the module firmware. To code and compile user programs, Segger Embedded Studio is used as an integrated development environment (IDE). To flash user C program to DWM1001 module, Segger J-Flash is used. Figure 4.3 and Figure 4.4 shows the interface of Segger Embedded Studio and the Segger J-Flash Lite.

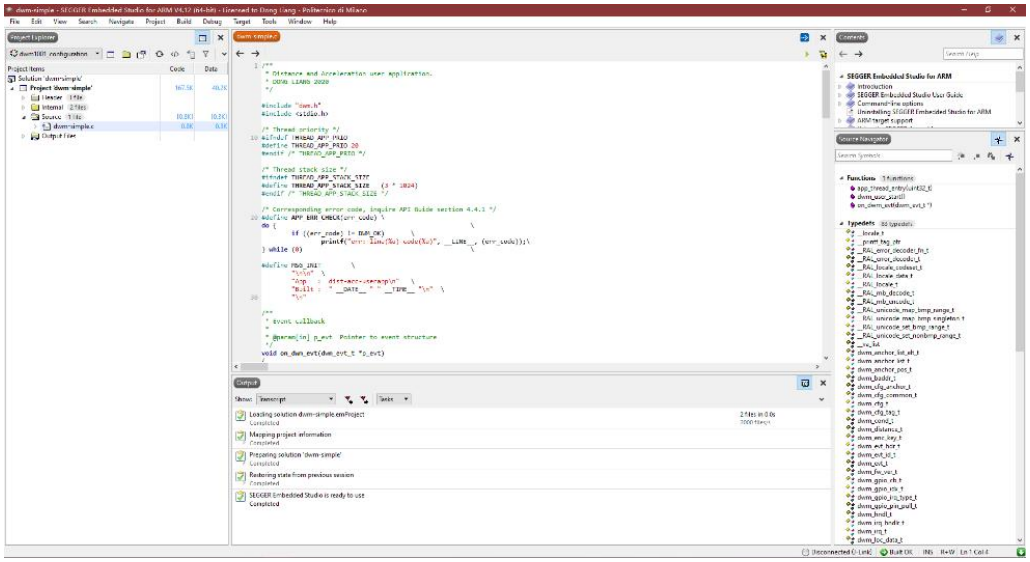


**Figure 4.3: Segger Embedded Studio IDE**

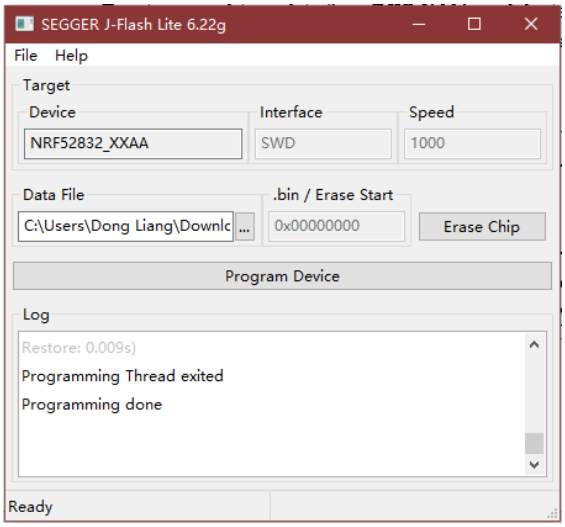



**Figure 4.4: Segger J-Flash Lite**

Table 4.2 lists several C APIs which may be used in data transmission step.

**Table 4.2: List of C API used**

| *C API* | *Description* |
|---|---|
| *dwm_loc_get* | *Get last distances to the anchors (tag is currently ranging to) and the associated position.* |
| *dwm_i2c_read* | *This API function read data from I2C slave, such as accelerometer.* |
| *dwm_i2c_write* | *This API function writes data to I2C slave.* |

## 4.2 Function definition for the interface

To locate the coordinates of a tag unit in three-dimensional space, the necessary data are distance measurements from tag units to at least three nearby anchor units, and corresponding anchor coordinates.

In Chapter 3, we show that the original UWB ranging data of DWM1001 module is affected by measurement noise. To optimize the ranging data, a filtering algorithm such as Kalman Filter or Particle Filter can be implemented with a sensor fusion method. The sensor fusion method requires two kinds of measurement data, from two independent sensors. In this thesis, the two considered sensors are the UWB chip DW1000 and the I2C accelerometer LIS2DH12, and the corresponding measurements are distance and acceleration, respectively.

Due to the above two reasons, our function is defined as follows

- Host device can read UWB measurement and corresponding anchor coordinates in real-time;
- Host device can read acceleration value in real-time;
- Two data above should be obtained synchronously.

## 4.3 On-board C program

As we know, MATLAB can communicate directly with the serial port. Therefore, it seems to be possible to use shell command “les” and “av” to obtain the two measurements. However, the shell commands “les” and “av” act in different ways. Shell command “les” sends the ranging results via serial port synchronously. The host device can receive the latest ranging data in real time. The shell command “av” acts asynchronously. Two data at a certain moment cannot be sent to the host computer synchronously. To avoid the problem of data out of sync, a custom on-board C program has

been proposed.

The code snippet to retrieve UWB measurements is:

```
case DWM_EVT_LOC_READY:
// predefined case in header file dwm.h line 969
    for (i = 0; i < p_evt->loc.anchors.dist.cnt; ++i) {
// loop for i times, i=number of anchors
        printf("DIST%d:", i);
        if (i < p_evt->loc.anchors.an_pos.cnt) {
            printf("[%ld,%ld,%ld]",
                p_evt->loc.anchors.an_pos.pos[i].x,
                p_evt->loc.anchors.an_pos.pos[i].y,
                p_evt->loc.anchors.an_pos.pos[i].z);
// print anchor coordinates, in which loc is a bool value predefined in header file dwm.h line 989
        }
        printf("=[%lu] ", p_evt->loc.anchors.dist.dist[i]);
// print distance between the tag and the anchor
}
```

With the code above, UWB measurements and corresponding anchor coordinates can be sent via serial port in real-time.

The code snippet to retrieve acceleration values is:

```
int8_t data[6];
// declare an array (pointers) as buffer zones for data storage
const uint8_t addr = 0x19;
// 7-bit address of the I2C accelerometer
data[0] = 0xAA; data[1] = 0xBB; data[2] = 0xCC; data[3] =
0xDD; data[4] = 0xEE; data[5] = 0xFF;
// preset buffer zone addresses for the pointers
dwm_i2c_write(addr ,data, 6, true);
// this API allocate the presetting address for the buffer zone in DWM1001 register
dwm_i2c_read(addr, data, 6);
```

```
// this API read values from the address preset before
printf("A:%d,%d,%d",
(data[0]<<8)+data[1],(data[2]<<8)+data[3],(data[4]<<8)+da
ta[5]);
//original acceleration value is 16-bit, connect two bytes to obtain value for one axis
```

With the code above, acceleration data can be sent in real-time.

In addition, to evaluate the sampling frequency, we also added timestamps to each row of data with the following code:

```
printf("\nT:%lu ", dwm_systime_us_get());
```

For the complete on-board C program code please refer to Appendix A.

## 4.4 MATLAB script

With the on-board C program, our DWM1001 module can send data including timestamp, UWB measurements, anchor coordinates and tag acceleration to an host device via serial port.

To communicate with serial port in MATLAB, command “serial” can be used in order to create a serial communication object. It is worth mentioning that with serial setting “s.ReadAsyncMode = 'continuous’” , it is possible to let MATLAB continue to receive data. However, the program will remain in “busy” mode until the data receiving step ends. During the “busy”, it is impossible to process data. Therefore, a callback function must be proposed.

With a callback function, data collection and storage will loop as an independent thread. Every time the callback triggers, data received and stored in buffer zone would be moved to a global variable. The global variable should be pre-allocated a large space to avoid performance drop due to multiple reallocations of space. The callback thread would remain

in “busy” mode, so that “inputting” data from serial port and “outputting” data to the global variable are in real-time, and we can read the real-time data from the global variable and execute data processing steps like position calculation or plotting a figure. These later steps should be written in another script as a new thread. The code snippet of callback function is:

```
s = serial('COM3');
set(s,'BytesAvailableFcnMode','byte');
set(s,'BytesAvailableFcnCount',27);
% set the interrupt trigger: when receives 27 bytes of data, interrupt occurs.
s.BytesAvailableFcn =@ReceiveCallback;

function ReceiveCallback( ~,~)
% customize a callback function
str = fscanf(s);
% read data from buffer
pattern = '(-)?\d{1,10}';
temp = regexp(str, pattern, 'match');
% use regular expression to extract time/distance/acc data
data_tag = [data_tag; temp];
% save historical data in a matrix for later data processing
end
```

For the complete MATLAB code please refer to Appendix B.

With the custom interface described in this chapter, data can be transmitted and saved on a PC at 10Hz, which is the maximum update frequency of DWM1001 module. This ensures the feasibility of subsequent synchronous/asynchronous data processing.

# Chapter 5

# Implementation of a Kalman Filter

## 5.1 Introduction

In Chapter 3, we use the On-board C User Program and the MATLAB serial communication script to obtain distance measurement between a tag and surrounding anchors. As shown in the Figure 3.1, the ranging data fluctuates in a range of amplitude 180 $mm$ while the tag remains stationary. The variance of the distance measurement is equal to 0.01 $m^2$(or 1033.5 in unit $mm^2$). As described in Chapter 2, the distance measurement results of Decawave DWM1001 unit entirely depend on UWB ranging algorithm. The official encapsulation of the positioning algorithm in the firmware cannot be modified nor optimized. So that we can only use additional filtering algorithms to optimize the original data. In addition to the original UWB ranging information, we also used acceleration data obtained from the I2C accelerometer.

In order to improve ranging and positioning accuracy, we choose Kalman filter to optimize the ranging results. Kalman filter is an optimal autoregressive filter. Due to the non-mutable nature of physical quantities, we can use the measurement data and the acceleration measurements in Kalman filter to improve the positioning data accuracy.

In Section 5.1, a brief introduction to the principle and workflow of Kalman filter is given. In Section 5.2, a Kalman filter model in a simple scenario is created and simulated.

## 5.2 Principle of the Kalman Filter

Kalman filter estimates the state of a process by two steps. The first step is the prediction step. The new state of the process can be estimated in the prediction step using the system model. No new measurement is involved in this step. The second step is the correction step. In the correction step, new measurement information is used as a feedback to reduce the random interference.

### 5.2.1 State-space representation

The goal of Kalman filter is to reconstruct the state vector of the system from the measured value. For describing a system or a process, Kalman filter uses a linear model composed of a state and an output equation.

The state equation represents the relationship between the current state and the previous state, the control input variable and some (Gaussian) process disturbance. The model has the following form:

$$\mathbf{x_{k+1}} = \mathbf{A x_k} + \mathbf{B u_k} + \mathbf{w_k} \tag{5.1}$$

In (5.1), $\mathbf{x_k}$ is the state vector at time instant $k$. As explained later in this chapter, the state vector, in our application, includes the distance and the velocity of the tag with respect to each anchor. Matrix $\mathbf{A}$ is state-transition matrix, and $\mathbf{u_k}$ is the input control vector at time instant $k$. In our scenario, $\mathbf{u_k}$ is the measured acceleration of the tag with respect to the anchor. Matrix $\mathbf{B}$ is the input metrix and indicates the effect of the input control vector $\mathbf{u_k}$ on the system state. $\mathbf{w_k}$ is a zero-mean gaussian process noise, whose covariance is denoted with $Q$, i.e., the process covariance matrix

The output equation represents the relationship between the state vector and the measured output. The model has the following form:

$$\mathbf{z_k} = \mathbf{H x_k} + \mathbf{v_k} \tag{5.2}$$

$\mathbf{z_k}$ is the observation vector at time instant $k$. $\mathbf{H}$ is the observation matrix. $\mathbf{v_k}$ is a zero-mean gaussian measurement noise, whose covariance is denoted with $R$.

Two prerequisites must be met to use the Kalman filter. The first is the linearity of both the state and the output equations. The second is the assumption that both $\mathbf{w_k}$ and $\mathbf{v_k}$ are white Gaussian noises.

## 5.2.2 Discrete Kalman Filter equations

A Kalman filter is composed by a prediction step and a correction step.

In the prediction step, the priori estimate $\hat{\mathbf{x}}_\mathbf{k}^-$ is obtained by the information available up to time instant $k-1$.

The prediction step has the following form:

$$\hat{\mathbf{x}}_\mathbf{k}^- = \mathbf{A}\hat{\boldsymbol{x}}_{\boldsymbol{k-1}} + \mathbf{B u_k} \tag{5.3}$$

in which $\hat{\boldsymbol{x}}_{\boldsymbol{k-1}}$ is the posterior estimate of the state vector obtained at time instant $k-1$.

The covariance of the prediction error $\mathbf{x_k} - \hat{\mathbf{x}}_\mathbf{k}^-$ is denoted with $\mathbf{P}_\mathbf{k}^-$ and is computed as:

$$\mathbf{P}_\mathbf{k}^- = \mathbf{A P_{k-1} A^T} + \mathbf{Q} \tag{5.4}$$

Where $\mathbf{P_k}$ is the covariance of the posterior estimate error $\mathbf{x_k} - \hat{\mathbf{x}}_\mathbf{k}$ obtained as:

$$\mathbf{P_k} = (\mathbf{I} - \mathbf{M_k H})\mathbf{P_K^-} \tag{5.5}$$

$\mathbf{P}_\mathbf{k}^-$ represents the degree of uncertainty of the system. This degree of

uncertainty will be large when the Kalman filter is initialized. As more and more data being injected into the filter, the degree of uncertainty will become smaller.

In the correction step, the posterior estimate is computed as:

$$\hat{\mathbf{x}}_{\mathbf{k}} = \hat{\mathbf{x}}_{\mathbf{k}}^{-} + \mathbf{M}_{\mathbf{k}}(\mathbf{z}_{\mathbf{k}} - \mathbf{H}\hat{\mathbf{x}}_{\mathbf{k}}^{-}) \tag{5.6}$$

where the Kalman Gain can be computed as:

$$\mathbf{M}_{\mathbf{k}} = \mathbf{P}_{\mathbf{k-1}}\mathbf{H}^{\mathbf{T}}(\mathbf{H}\mathbf{P}_{\mathbf{k-1}}\mathbf{H}^{\mathbf{T}} + \mathbf{R})^{-\mathbf{1}} \tag{5.7}$$

From equation (5.6) we can see that a posteriori estimation is a sum of a priori estimate $\hat{\mathbf{x}}_{\mathbf{k}}^{-}$ and a term which is proportional to the residual $\mathbf{z}_{\mathbf{k}} - \mathbf{H}\hat{\mathbf{x}}_{\mathbf{k}}^{-}$, i.e. the priori estimation error. [2, 6, 7]

## 5.3 Application of the Kalman Filter

In this section, we apply the Kalman filter to the case study considered in this thesis. By filtering distance measurement data and acceleration data obtained by two independent sensors, we aim to obtain accurate ranging results.

### 5.3.1 Test setup and Kalman Filter parameter setting

We now consider, for simplicity, the case where there is one tag unit and one anchor unit. [12] The tag unit moves linearly on the connection between the two units. Figure 5.1 shows the setup of the test. Via serial communication, we obtain a set of data including distance measurements between the tag and the anchor, and acceleration of the tag unit with a sampling interval of $\Delta t = 0.1s$. As calculated in Chapter 3, the variance of the distance measurement noise is 0.001 $m^2$ (more accurate 1033.5 $mm^2$), while the variance of the acceleration measurement is 0.0021 $(m/s^2)^2$.

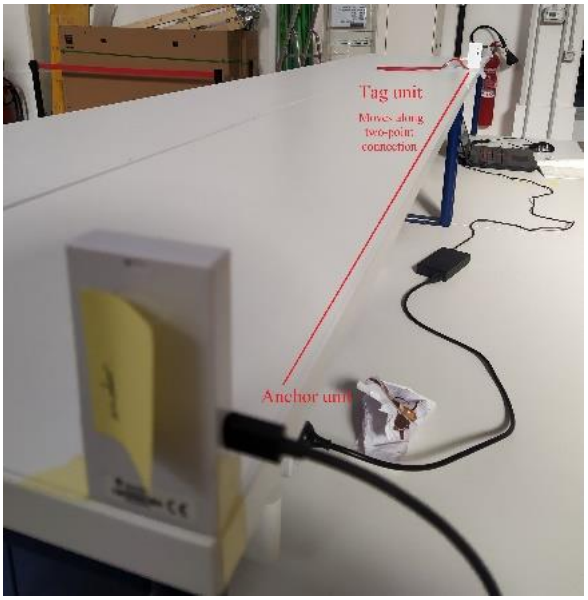



**Figure 5.1: Tag linear movement test**

When the tag unit moves in a straight line with acceleration $a_t$, the following discrete-time equations of motion can be written by discretization of the double-integrator equation $\ddot{s}(t) = a(t)$, i.e.

$$\begin{cases} s_{k+1} = s_k + \Delta t v_k + \frac{\Delta t^2}{2} a_k \\ v_{k+1} = v_k + \Delta t a_k \end{cases} \tag{5.8}$$

where $s_k$ is the distance between the tag and the anchor and $v_k$ is the tag velocity at time instant $k$.

We reorganize equations (5.8) into matrix form and we have:

$$\begin{bmatrix} s_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} s_k \\ v_k \end{bmatrix} + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} a_k \tag{5.9}$$

The state vector of this model is therefore:

$$\mathbf{x_k} = \begin{bmatrix} s_k \\ v_k \end{bmatrix} \tag{5.10}$$

The matrix $\mathbf{A}$ in equation (1.1) is:

$$\mathbf{A} = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \tag{5.11}$$

on which $\Delta t$ is the time interval between two measurements. In this experiment it is equal to 0.1s.

In this section, the acceleration is regarded as an input of the dynamic model (5.9). However, $a_k$ is affected by measurement noise, and therefore we write:

$$a_k = u_k + w_k^Q$$

Where $u_k$ is the acceleration measurement obtained at instant $k$, while $w_k^Q$ is the corresponding noise, assumed gaussian, with zero mean, and variance equal to 0.0021 $(m/s^2)^2$.

This allows to rewrite equation (5.9) as:

$$\mathbf{x_{k+1}} = \mathbf{A}\mathbf{x_k} + \mathbf{B}u_\mathrm{k} + w_\mathrm{k} \tag{5.12}$$

The matrix $\mathbf{B}$ in equation (5.12) is:

$$\mathbf{B} = \begin{pmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{pmatrix} \tag{5.13}$$

While the noise $w_\mathrm{k}$ is:

$$w_k = B w_k^Q \tag{5.14}$$

In view of this, we obtain that:

$$Q = w_k w_k^T = \sigma_w^2 B B^T \tag{5.15}$$

The measurement is the output of the UWB ranging sensor and therefore the observation matrix in equation (5.5) is:

$$\mathbf{H} = (1 \quad 0) \tag{5.16}$$

while the Gaussian measurement noise is:

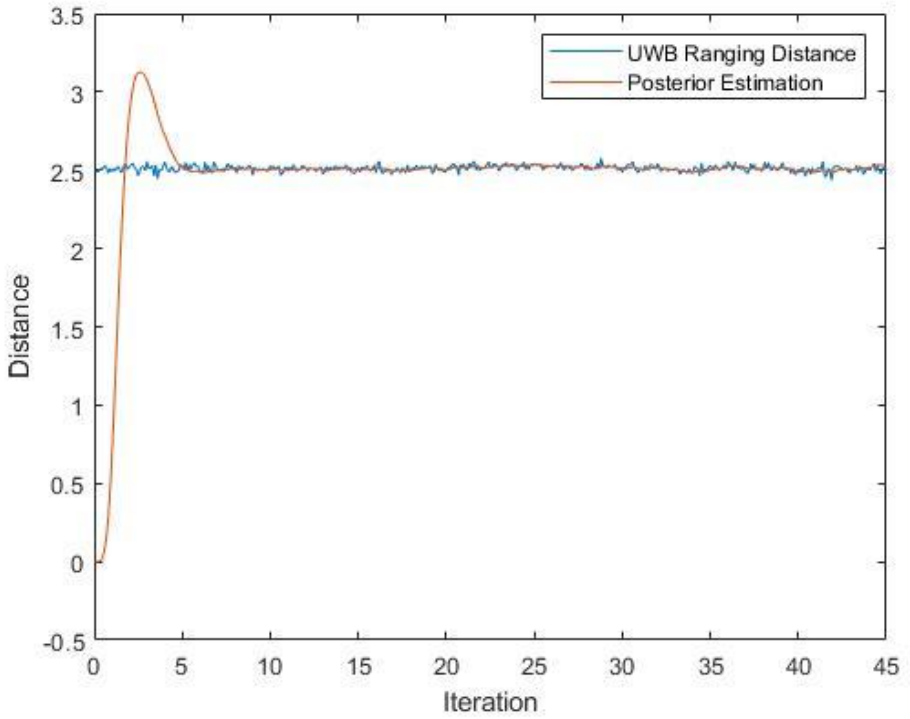
$$\mathbf{v_k} \sim WGN(0, R) \tag{5.17}$$

where $R = 1033.5\, mm^2$.

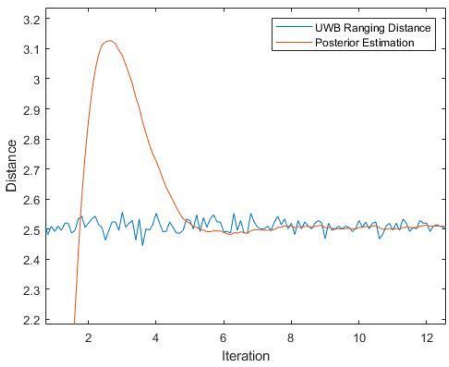
## 5.3.2 Kalman Filtering results

In this section we implement the Kalman Filter with the parameters obtained in Section 5.3.1 on the original UWB distance measurement data.

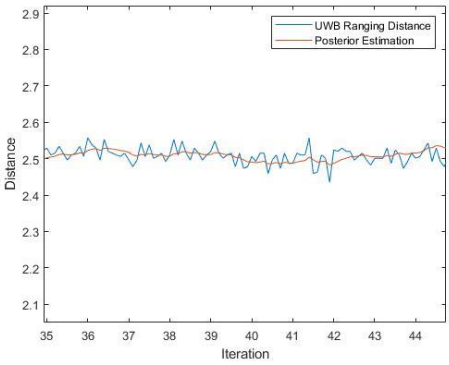
The first case is both the anchor unit and the tag unit remaining stationary. As we proved in Chapter 3, when both modules remain stationary, the measurement noise of the UWB distance measurement is a Gaussian white noise. This result ensures the feasibility of the use of Kalman Filter. The original distance measurement data and the filtered data is shown in Figure 5.2.




**(a): global view**




**(b): the first 10 seconds zoom**




**(c): the last 10 seconds zoom**

**Figure 5.2: Stationary test Kalman Filtering result**

As we can see in Figure 5.2, the iteration starts from the second time instant, 0.2s. The posteriori estimate reaches peak value at the twenty second iteration and start convergence immediately. Since the 8 second, the distance posteriori is equal to 2.5m. We can see that the filtered curve is smooth, and the fluctuation range is small. For analysis purpose, we select the data of the last ten seconds and calculate their variance. The

variance of the original distance measurement data is 0.00047521 $m^2$, or 475.21 $mm^2$, and the variance of the filtered data is 0.00017022 $m^2$.

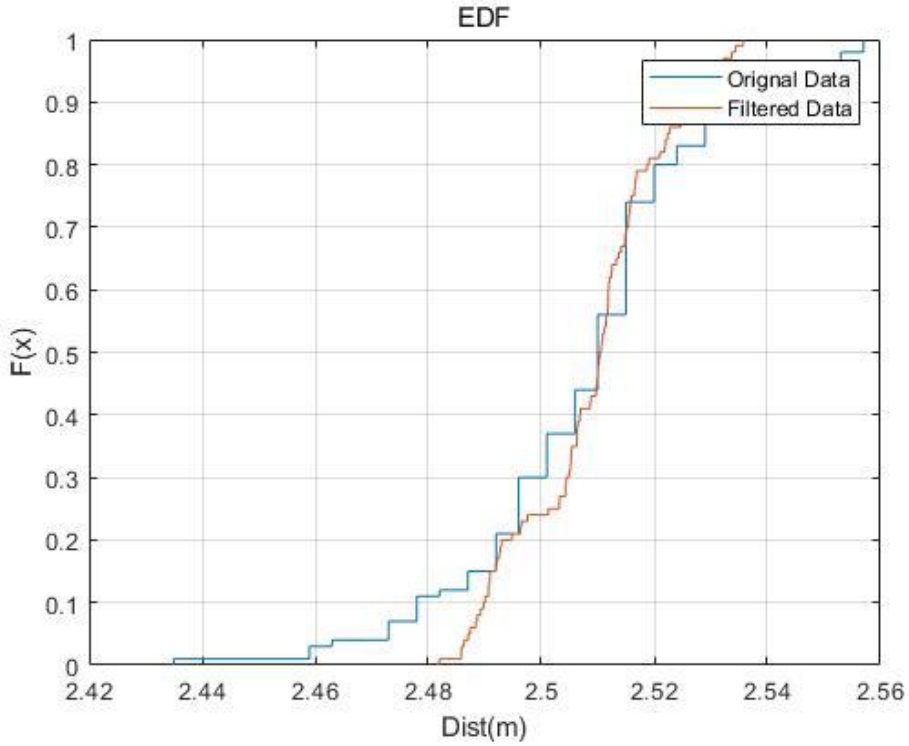
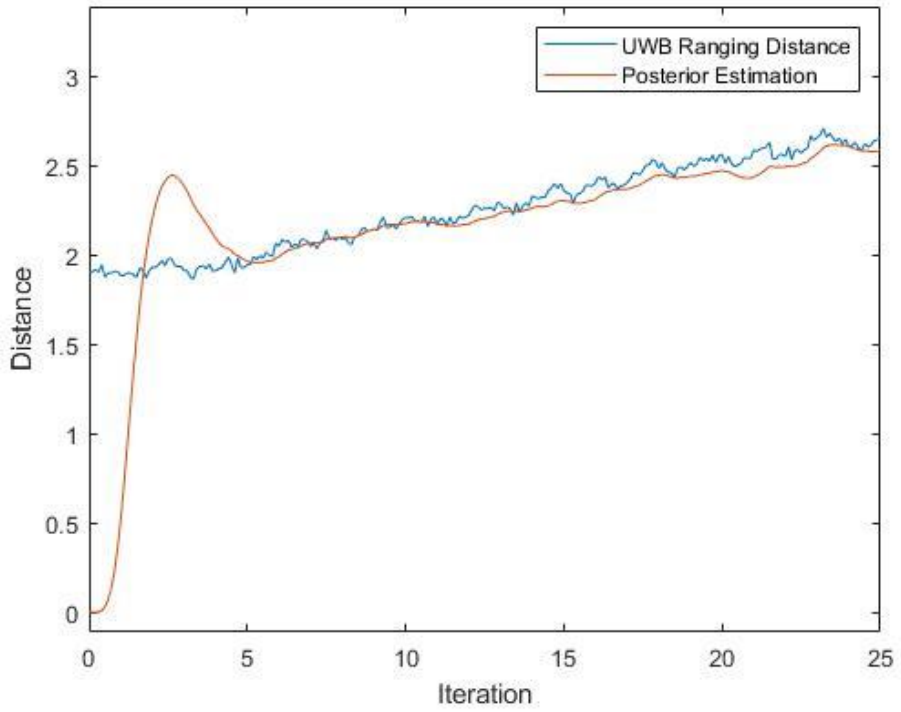


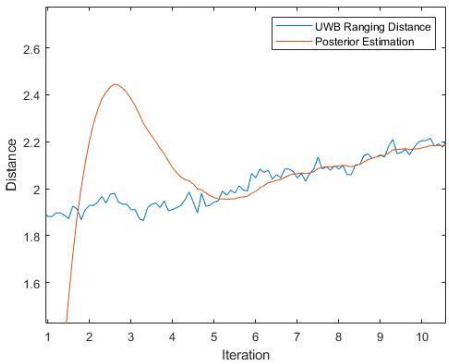

**Figure 5.3: EDF before/after filtering**

In Figure 5.3, we can see that the data distribution is more concentrated to 2.5m.

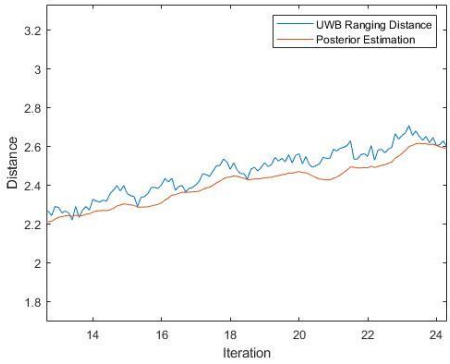
The second test consists of moving the tag unit forward, leaving the anchor unit . The experimental data comes from the real environment, so that we do not know the exact value of the distance or the acceleration during the movement. However, we can still observe the original measurement data and the filtered data to judge the effect of Kalman filter. The distance data can be plotted as the following Figure 5.4:




**(a): global view**




**(b): the first 10 seconds zoom**




**(c): the last 10 seconds zoom**

**Figure 5.4: Tag straight movement test Kalman Filtering result**

We can see that the Kalman filter has similar performance as in the previous experiment. The filtered data start to converge at the 3rd second. Within the last ten seconds, the original UWB distance measurement data fluctuates repeatedly. The Kalman filter reduces these fluctuations and maintains a smooth rise trend. [1]

# Chapter 6

# Principle and Application of the Trilateration Algorithm

In this chapter, we discuss how to locate an object in the three-dimensional space. In Section 6.1, the principle of trilateration algorithm is introduced. In Section 6.2 and Section 6.3, two methods to solve the trilateral positioning problem are proposed. The first method is denoted as Cost Function Minimizing Method. The second method is denoted as Matrix Solution Method. Two methods have been simulated separately. And the performance of the two methods have been evaluated in Section 6.4.

## 6.1 Principle of the Trilateration Algorithm

In this section we discuss about how to calculate the coordinates of the tag unit in the three-dimensional space.

As show in Figure 6.1, we assume that the following data are available: at least three anchor points coordinates $P_1(x_1, y_1, z_1)$, $P_2(x_2, y_2, z_2)$, $P_3(x_3, y_3, z_3)$, and the corresponding distance measurements $s_1$, $s_2$, $s_3$ between tag and anchors.

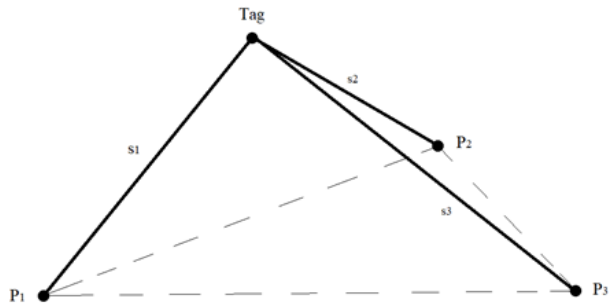



**Figure 6.1: Tag unit and anchor units in three dimensions space**

In Chapter 5, the Kalman filter is implemented to smooth raw range measurement data, i.e. to obtain a new dataset of range measurements with minimal variance. In this chapter, these estimates are used in the trilateral positioning method.

Determining the coordinates $(x, y, z)$of a tag is equivalent to finding the solutions to the following system of quadratic equations.

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 = {s_1}^2 \\ (x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 = {s_2}^2 \\ (x - x_3)^2 + (y - y_3)^2 + (z - z_3)^2 = {s_3}^2 \end{cases} \tag{6.1}$$

As shown in Figure 6.2, the three spheres, respectively determined by three anchor points coordinates and corresponding distance measurements, intersect at one point. By solving the system of equations (6.1), we can get an accurate estimate of the coordinates of the tag location.

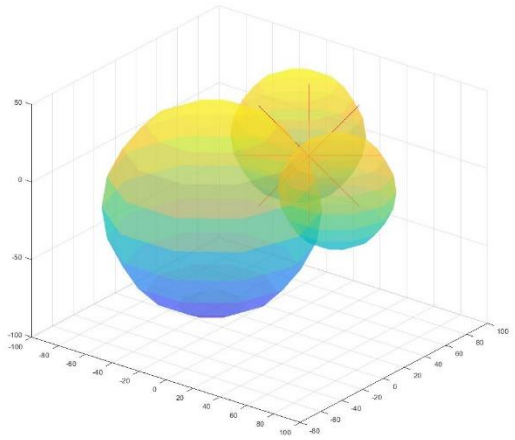


**Figure 6.2: Trilateral Positioning**

Due to the presence of noise and errors in the distance measurement process, and to account for the probability of considering more than three anchor points, in the following we consider two alternative approaches, i.e., the Cost Function Minimization Method and the Matrix Solution Method.

## 6.2 Cost Function Minimization Method

### 6.2.1 Principle of the Cost Function minimizing Method

We assume that the tag is located at $P_t(x_t, y_t, z_t)$. The distances between tag and anchors are computed according to:

$$\begin{cases} s_{t1} = \sqrt{(x_1 - x_t)^2 + (y_1 - y_t)^2 + (z_1 - z_t)^2} \\ s_{t2} = \sqrt{(x_2 - x_t)^2 + (y_2 - y_t)^2 + (z_2 - z_t)^2} \\ s_{t3} = \sqrt{(x_3 - x_t)^2 + (y_3 - y_t)^2 + (z_3 - z_t)^2} \end{cases} \quad (6.2)$$

In the functions above, $s_{ti}$ indicates the distance between the estimated tag position $P_t$ and the $i\text{-}th$ anchor real position $P_i$.

To find the optimal solution, we define a cost function expressed as follows:

$$C = \sum_{i=1}^{3} (s_{ti} - s_i)^2 \quad (6.3)$$

The cost function shown in (6.3) is the Residual Sum of Squares (RSS or Sum Squared Error(SSE)). RSS is the sum of the squared differences between predicted distances $s_{ti}$ and measured distances $s_i$. [4]

The MATLAB command `fminsearch` can be used to find a local minimum of the cost function. Command `fminsearch` uses a derivative-free method to find minimum of unconstrained multivariable function using derivative-free method. To improve the performance of `fminsearch`, a starting point is set to an anchor point whose measuring distance from this anchor point to the tag is the shortest.

### 6.2.2 Simulation of the Cost Function Minimizing Method

In order to verify the feasibility of the cost function minimizing method, several simulations are carried out in this section.

Assume that the coordinates of three base stations are $P_1(27.297, -4.953, 1.470)$, $P_2(25.475, -6.124, 2.360)$, $P_3(22.590, 0.524, 1.200)$. Three distance measurements between tag and anchors are $s_1 = 3.851$, $s_2 = 3.875$, $s_3 = 3.514$. Import the above data into the following MATLAB program:

```
dist_fun = @(pos)
sqrt(sum((bsxfun(@minus,P,pos(:)')).^2,2));
cost_function = @(pos) sum((dist_fun(pos)-
S(:)).^2);
initial_guess = [22.59 0.524 1.2];
position=fminsearch(cost_function,initial_guess);
```

we can obtain the coordinate of the tag unit which is $Tag(24.3123, -2.5204, 1.5365)$.

Calculate the distances between anchor points and the estimated tag point, we obtain $\hat{s}_1 = 3.851$, $\hat{s}_2 = 3.875$, $\hat{s}_3 = 3.514$. The posteriori measurement is exactly equal to input distance ranging data when it keeps three decimal places. When we keep five or more than five decimal places, we can discover very subtle differences. This can be regarded as a calculation error.

The following three views and the three-dimensional perspective diagram of Figure 6.3 show the sphere composed of anchors coordinates plus distance measurements, as well as the calculated position of the tag unit which is represented by a red asterisk.

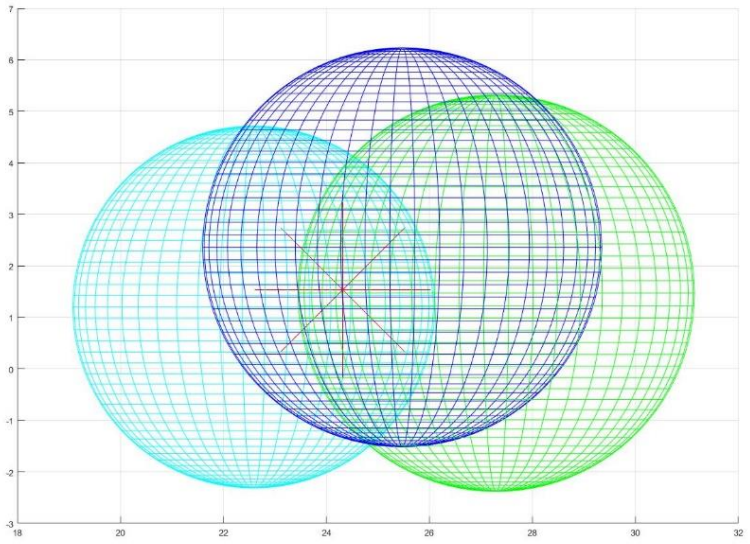



**Figure 6.3(a): Front view**

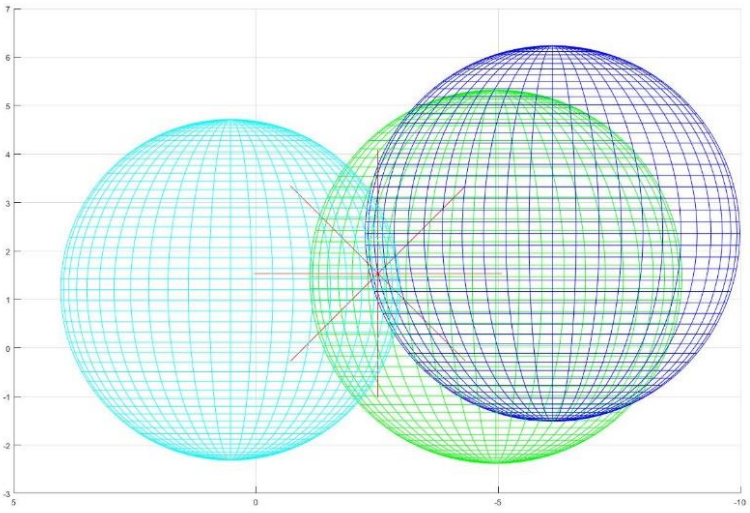



**Figure 6.3(b): Left view**

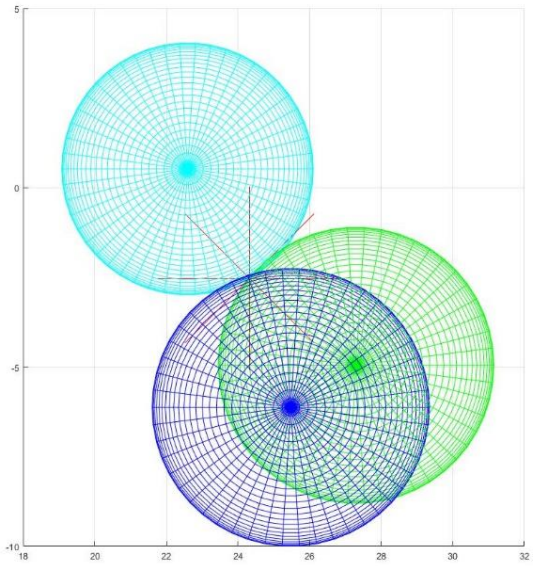



**Figure 6.3(c): Top view**

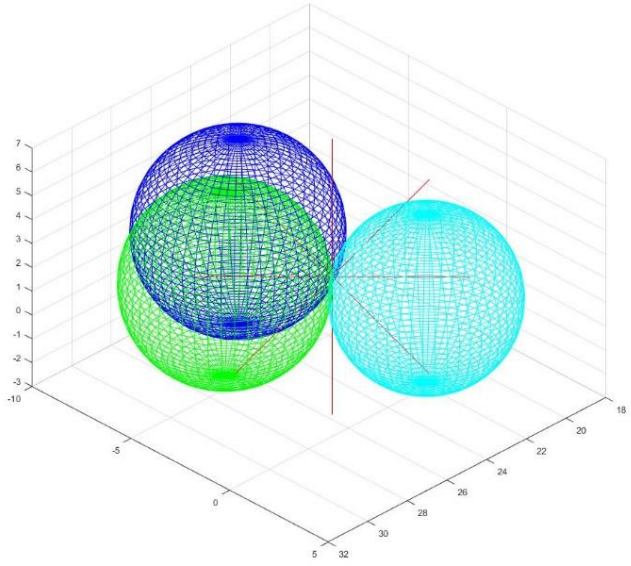



**Figure 6.3(d): 3D view**

Using the method and commands described in this section, code has low space complexity.

## 6.3 Matrix Solution Method

### 6.3.1 Principle of the Matrix Solution Method

The equations of system (6.1) can be expanded and reformed as:

$$\begin{cases} x^2+y^2+z^2-2x_1x-2y_1y-2z_1z = {s_1}^2-{x_1}^2-{y_1}^2-{z_1}^2 \\ x^2+y^2+z^2-2x_2x-2y_2y-2z_2z = {s_2}^2-{x_2}^2-{y_2}^2-{z_2}^2 \\ x^2+y^2+z^2-2x_3x-2y_3y-2z_3z = {s_3}^2-{x_3}^2-{y_3}^2-{z_3}^2 \end{cases} \quad (6.4)$$

The equation system above can be rewritten in matrix form as:

$$\begin{bmatrix} 1 & -2x_1 & -2y_1 & -2z_1 \\ 1 & -2x_2 & -2y_2 & -2z_2 \\ 1 & -2x_3 & -2y_3 & -2z_3 \end{bmatrix} \begin{bmatrix} x^2+y^2+z^2 \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} {s_1}^2-{x_1}^2-{y_1}^2-{z_1}^2 \\ {s_2}^2-{x_2}^2-{y_2}^2-{z_2}^2 \\ {s_3}^2-{x_3}^2-{y_3}^2-{z_3}^2 \end{bmatrix} \quad (6.5)$$

If we define $\boldsymbol{x} = [x^2+y^2+z^2, x, y, z]^T$, system (6.5) is a linear system having a familiar form $A \cdot \boldsymbol{x} = \boldsymbol{b}$ under the constraint that $\boldsymbol{x} \in E$, being $E = \{(x_0, x_1, x_2, x_3)^T \in \mathbb{R}^4 | x_0^2 = x_1^2 + x_2^2 + x_3^2\}$.

We can see that matrices $A$ and $\boldsymbol{b}$ consist of only known elements. Unknown tag's coordinates are all moved to the matrix $\boldsymbol{x}$.

The solution to the constrained problem can be found in a numerically efficiency way, see [5, 14].

### 6.3.2 Simulation of the Matrix Solution Method

Same as in Section 6.2.2, we assume that $P_1(27.297, -4.953, 1.470)$, $P_2(25.475, -6.124, 2.360)$, $P_3(22.590, 0.524, 1.200)$, $s_1 = 3.851$, $s_2 = 3.875$, $s_3 = 3.514$. Import these data to MATLAB and we can obtain two solutions after calculation. $Tag_1(24.3506, -2.4811, 1.6667)$ and $Tag_2(24.3123, -2.5205, 1.5365)$. Calculate theirs residual sum of

squares, $RSS_{Tag_1} = 1.1566 \times 10^{-10}$, and $RSS_{Tag_2} = 6.1801 \times 10^{-9}$. We can find that the solution given by the Minimizing Cost Function Method is one of the solutions of matrix calculation. Although its RSS is significantly small, another better solution may exist. In this case, the coordinates of solution $Tag_1$ would be more accurate.

The following three views and the three-dimensional perspective diagram of Figure 6.4 show the spheres and the calculated positions. The $Tag_2$, same solution of the minimizing cost function, is represented by a red asterisk. And the more accurate point $Tag_1$ is represented by a black asterisk.

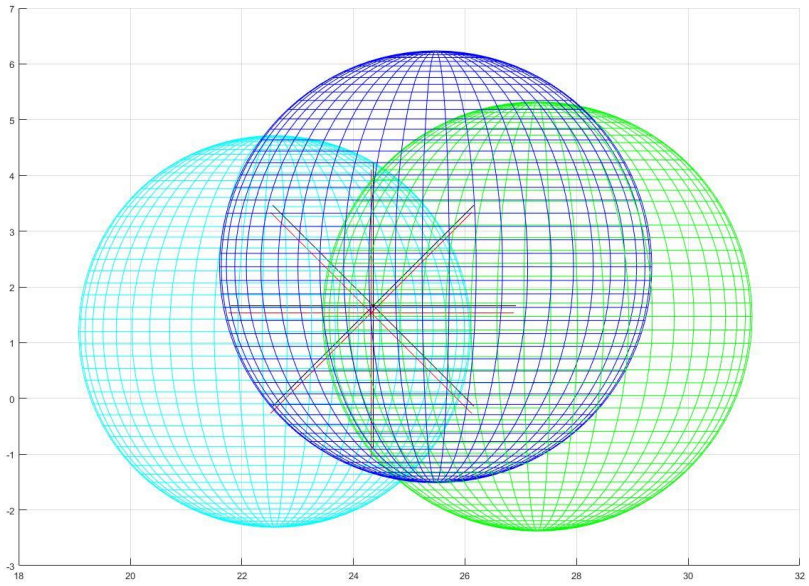



**Figure 6.4(a): Front view**

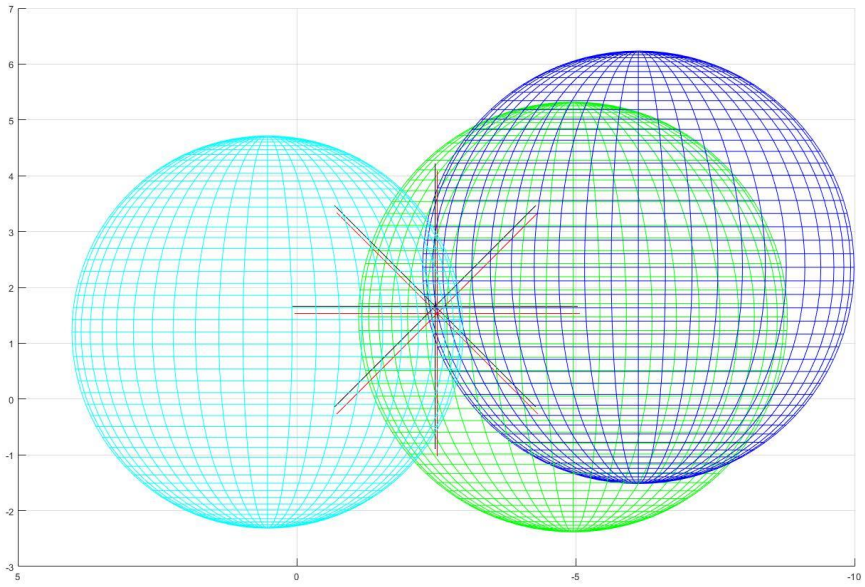



**Figure 6.4 (b): Left view**

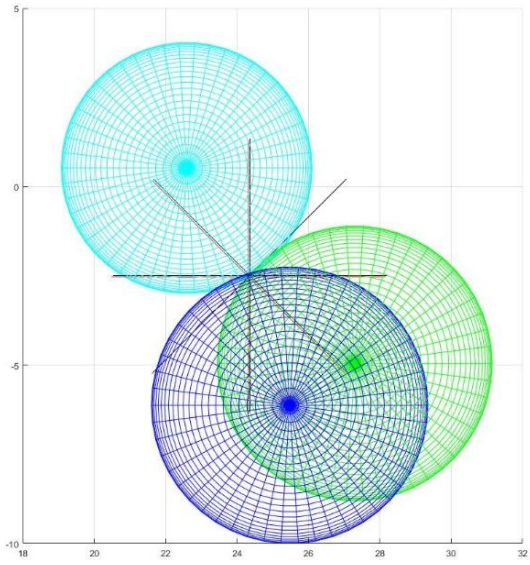



**Figure 6.4(c): Top view**

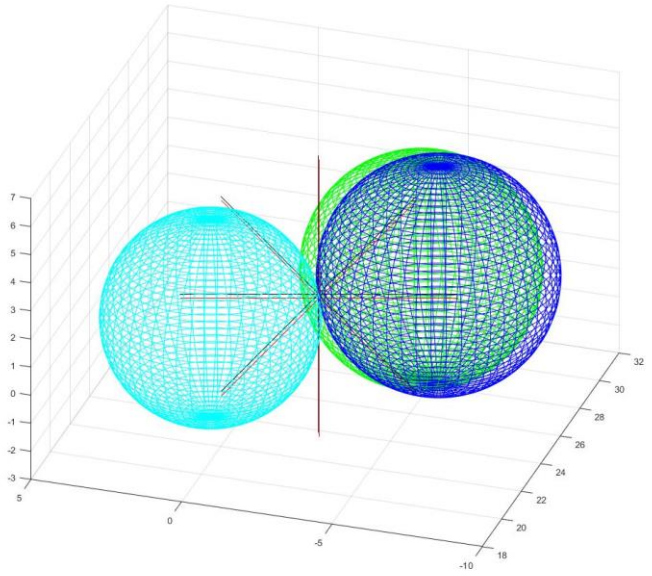



**Figure 6.4(d): 3D view**

## 6.4 Performance evaluation

In this section, the performance of the Cost Function Minimizing Method and the Matrix Calculation Method are evaluated. Both methods have been applied to a test dataset with white noise ranging data whose variance being 0.03 for trilateral positioning. In the Table 6.1, some original data and the positioning results of both methods are displayed.

As shown in the Table 6.1, comparing to the Cost Function Minimizing Method, the Matrix Solution Method can obtain estimated positions with similar or smaller root-mean-square deviation in most conditions. And the program execution time reduces by 63%. On the other hand, the Cost Function Minimizing Method behaves stable with all data. For correctness consideration, the Cost Function Minimizing Method would be a better trade-off.

**Table 6.1: Performance evaluation for two methods**

| Anchor Coordinates | Tag Coordinates | Distance Measurement | Estimated Position by CFM Method | Program Execution Time of CFM Method (s) | RMSD of CFM Method | Estimated Position by MS Method | Program Execution Time of MS Method (s) | RMSD of MS Method |
|---|---|---|---|---|---|---|---|---|
| P1(0.4130,0.7635,0.2335)<br>P2(9.8951,10.3126,10.0916)<br>P3(4.4851,5.4746,0.1535) | T(3.0676,3.2576,3.1307) | S1=4.6259<br>S2=12.0302<br>S3=3.9693 | (2.9844,3.3215,3.1297) | 0.042 | 0.0606 | (2.9786,3.3109,3.1574) | 0.016 | 0.0602 |
| P1(-0.2523-0.6353,-0.19129)<br>P2(10.3243,10.4129,9.4925)<br>P3(4.76455.0685,-0.1459) | T(3.1509,3.2000,2.5350) | S1=5.8019<br>S2=12.2604<br>S3=3.6789 | (3.2733,3.0938,2.5807) | 0.058 | 0.0972 | (3.2613,3.0747,2.6355) | 0.022 | 0.0928 |
| P1(-0.2223,-0.0780,0.1380)<br>P2(9.8694,10.2217,10.1959)<br>P3(4.3747,4.5260,-0.3706) | T(2.7461,2.8397,3.0062) | S1=4.9639<br>S2=12.5134<br>S3=4.1481 | (2.6534,2.8571,3.0143) | 0.048 | 0.0547 | (2.6293,2.8345,3.1006) | 0.022 | 0.0455 |
| P1(-0.5334,0.4669,0.1752)<br>P2(9.9855,10.0912,9.2175)<br>P3(4.9577,5.8020,0.0492) | T(3.0207,2.6329,2.9846) | S1=5.0285<br>S2=11.9704<br>S3=4.7229 | (3.0911,2.5761,2.9501) | 0.0500 | 0.0559 | (1.8654,3.8396,3.0312) | 0.021 | 0.9649 |
| P1(0.1199,-0.3452,-0.3258)<br>P2(10.5961,9.1941,9.9878)<br>P3(1.0256,5.5102,0.4309) | T(3.0006,2.9646,1.7569) | S1=4.8744<br>S2=12.7501<br>S3=2.9783 | (3.0781,2.9892,1.7260) | 0.046 | 0.0502 | (3.0803,2.9600,1.7698) | 0.024 | 0.0354 |
| **MEAN VALUE** | | | | 0.0058 | 0.0637 | | 0.021 | 0.2398 |

# Chapter 7

# Experiment and Result

In this chapter, the experiment within real scenario is carried out. The Kalman Filter and two trilateral positioning methods are implemented on the experimental data. In Section 7.1, the experiment setup is introduced. In Section 7.2, the results of the positioning are analyzed.

## 7.1 Experiment setup

To evaluate the Kalman Filter and the trilateral positioning methods proposed is Chapter 5 and 6, distance measurement in a closed field were performed with the DWM1001 modules. There is no obstacle in the experiment field. The sight line between the tag unit and the anchor unit is not blocked. The communication and control between the tag unit and the host device is via USB serial port and the custom interface developed in Chapter 4.

Figure 7.1 demonstrates the location if the anchor units as well as the tag unit on T, where the y-axis points to west direction, z-axis points to south direction and the x-axis points to the center of the earth.

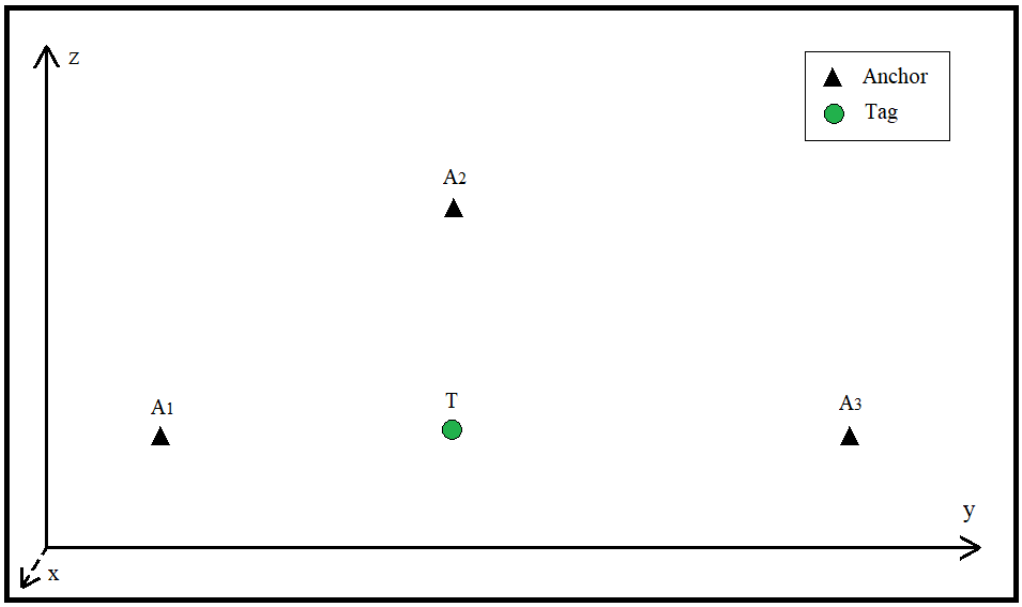



**Figure 7.1: Experiment filed layout**

## 7.2 Experiment Result

The true coordinate of the unknown tag point is (0.0470,3.0000,1.0000). The coordinates of three anchor units and the true distances between the tag unit and the anchor units are listed in Table 7.1.

**Table 7.1: Coordinates of the anchor units and the distance to the tag unit**

| Anchor | X[m] | Y[m] | Z[m] | $Dist_{true}$[m] |
|---|---|---|---|---|
| **A1** | 0.0470 | 0.4510 | 1.0000 | 2.5490 |
| **A2** | 0.0470 | 3.0000 | 2.5220 | 1.5220 |
| **A3** | 0.0470 | 6.0490 | 1.0000 | 3.0490 |

### 7.2.1 Trilateral positioning solutions without the Kalman Filter

Due to the measurement white noises existence of both the UWB ranging chip and the accelerometer, the high frequency original distance

measurement data fluctuate within a certain range. As shown in Chapter 3, the distance measurement variance reaches $1033.5mm^2$. In this section, we use original distance measurement data to calculate the tag position with both the Cost Function Minimizing method and the Matrix Solution method. We pick the original data on the 15th second, 25th second, 35th second and 45th second as input. The distance measurements at these time instants shows in Table 7.2.

**Table 7.2: Original distance measurement on 15/25/35/45 second**

| To anchor | $D_m$ 15s[m] | $D_m$ 25s[m] | $D_m$ 35s[m] | $D_m$ 45s[m] |
|---|---|---|---|---|
| **A1** | 2.4310 | 2.4210 | 2.4260 | 2.4070 |
| **A2** | 1.4470 | 1.4230 | 1.4470 | 1.4330 |
| **A3** | 3.0870 | 3.0820 | 3.0770 | 3.0580 |

With both trilateral methods proposed in previous chapter, we obtain following coordinates in Table 7.3:

**Table 7.3: Trilateral positioning results without the Kalman Filter**

| Method | Results | 15s | 25s | 35s | 45s |
|---|---|---|---|---|---|
| **CFM method** | X[m] | 0.0471 | 0.0470 | 0.0470 | 0.0470 |
| | Y[m] | 2.9219 | 2.9194 | 2.9245 | 2.9245 |
| | Z[m] | 1.0749 | 1.0978 | 1.0744 | 1.0867 |
| | $\text{Dist}_{error}$[m] | 0.1082 | 0.1267 | 0.1060 | 0.1150 |
| | $\text{Mean}_{Dist_{error}}$[m] | 0.1140 | | | |
| | RSS | 0.0522 | | | |
| **MS method** | X[m] | 0.0470 | 0.0470 | 0.0470 | 0.0470 |
| | Y[m] | 2.9267 | 2.9251 | 2.9300 | 2.9322 |
| | Z[m] | 1.0029 | 1.0122 | 0.9893 | 0.9687 |
| | $\text{Dist}_{error}$[m] | 0.0734 | 0.0759 | 0.0708 | 0.0747 |
| | $\text{Mean}_{Dist_{error}}$[m] | 0.0737 | | | |
| | RSS | 0.0217 | | | |

Comparing to the cost function minimizing method, the matrix solution method obtains more accurate tag position coordinates. We can see that

the coordinate mean estimate error of the MS method is 0.0737m, or 7.37cm. The mean error of the CFM method is not that good, does not reach the centimeter level.

### 7.2.2 Trilateral positioning solutions with the Kalman Filter

To improve the positioning accuracy, I implement the Kalman Filter on the distance measurement. Table 7.4 lists the filtered distance measurement data, that at the same time distance as Table 7.2.

**Table 7.4: Filtered distance measurement on 15/25/35/45 second**

| To anchor | $D_m$ 15s[m] | $D_m$ 25s[m] | $D_m$ 35s[m] | $D_m$ 45s[m] |
|---|---|---|---|---|
| **A1** | 2.4303 | 2.4387 | 2.4470 | 2.4561 |
| **A2** | 1.4430 | 1.4391 | 1.4458 | 1.4386 |
| **A3** | 3.0530 | 3.0638 | 3.0656 | 3.0488 |

Same step with the previous section, I use both the CFM method and the MS method to calculate the coordinates and obtain results as shown in Table 7.5.

We can see that the sum of the squared residual error of both methods decreasing. The RSS of the CFM method reduced to 0.0377 from 0.0522. The RSS of the MS method reduced to 0.01334 from 0.0217. We can also observe that the accuracy of the CFM method reaches the centimeter level with the Kalman Filter. The accuracy of the MS method reaches 5.67cm. Moreover, with the increasing of the Kalman Filter's iteration, we can see that the distance measurement error of the MS method keeps falling. It means that the Kalman Filter keeps reducing the effect of the measurement white noise, and trust more on its estimate posterior.

**Table 7.5: Trilateral positioning results with the Kalman Filter**

| Method | Results | 15s | 25s | 35s | 45s |
|---|---|---|---|---|---|
| **CFM method** | X[m] | 0.0470 | 0.0470 | 0.0470 | 0.0470 |
| | Y[m] | 2.9386 | 2.9374 | 2.9407 | 2.9536 |
| | Z[m] | 1.0741 | 1.0814 | 1.0750 | 1.0814 |
| | $\text{Dist}_{error}$[m] | 0.0962 | 0.1027 | 0.0956 | 0.0937 |
| | $\text{Mean}_{Dist_{error}}$[m] | 0. 091 | | | |
| | RSS | 0.0377 | | | |
| **MS method** | X[m] | 0.0470 | 0.0470 | 0.0470 | 0.0470 |
| | Y[m] | 2.9267 | 2.9428 | 2.9454 | 2.9586 |
| | Z[m] | 1.0029 | 0.9957 | 0.9983 | 0.9977 |
| | $\text{Dist}_{error}$[m] | 0.0734 | 0.0574 | 0.0546 | 0.0415 |
| | $\text{Mean}_{Dist_{error}}$[m] | 0.0567 | | | |
| | RSS | 0.01334 | | | |

# Chapter 8

# Conclusion and Future Work

In this thesis, an optimized approach to locate objects in the three-dimensional space is presented. This approach is based on a Kalman Filter and a suitable trilateration algorithm. The main contributions of this thesis work are:

1) Pre-process and analyze the performance of Decawave DWM1001 UWB module, including processing and analysis the distance measurement data obtained by the UWB chip DWM1000 IC and acceleration data obtained by the embedded I2C accelerometer LIS2DH12, for further studies;

2) Develop a function-customized serial port communication interface for DWM1001 module. The interface enables distance measurement data and acceleration data to be sent by the module and received by the host device in real time. With simple modifications, more kinds of data transmission function can be added in short time;

3) Design a Kalman Filter able to improve UWB ranging accuracy, based on sensor fusion of the UWB position sensor and the accelerometer;

4) Propose two approaches of trilateration positioning.

Experimental results obtained from a location dataset in the real environment illustrate the validity and the performance of the proposed approaches, showing that the usage of the Kalman Filter improve the ranging accuracy of DWM1001 module. The presented Cost Function Minimization Method and Matrix Solution Method are able to achieve the higher positioning accuracy comparable to DWM1001 datasheet.

From the author’s point of view, the topic addressed by this thesis is worth future development and implementation on the following possible directions:

- Enhancement of the positioning accuracy, by implementing Particle Filter to the continuous positioning results, so as to further smooth the data curve; [3]

- Extend the diversity, by introducing the Extended Kalman Filter for non-linear movement model development; [13]

- Extend the scenario applicability, by implementing the proposed model on datasets of multiple environments, open environment, environment with many obstacles or electromagnetic interference, base station with wall blocking, i.e.

# Appendix A

## Anderson's Whiteness Test

```
alpha=0.1;
gamma=covf(x,floor(N/10)); % compute the covariances
rho=gamma(2:end)/gamma(1);   %   compute   the   normalized
correlations (for tau>0)
beta=norminv(1-alpha/2);

nalpha=length(find(sqrt(N)*rho>beta))+length(find(sqrt(N)
*rho<-beta));
f=nalpha/length(rho);

figure
hold on
plot(1:length(rho),beta*ones(length(rho),1),'r:','linewid
th',2)
plot(1:length(rho),-
beta*ones(length(rho),1),'r:','linewidth',2)
plot(1:length(rho),sqrt(N)*rho,'ko')
ylabel('sqrt(\rho)')

disp(['Frequency     of     violation:     ',num2str(f),',
alpha=',num2str(alpha)])
if f<=alpha
   disp('Anderson test passed')
else
   disp('Anderson test failed')
end
```

# Appendix B

# On-board C Program

```
/**
 * Distance and Acceleration user application.
 * DONG LIANG 2020 */

#include "dwm.h"
#include <stdio.h>
/* Thread priority */
#ifndef THREAD_APP_PRIO
#define THREAD_APP_PRIO   20
#endif /* THREAD_APP_PRIO */
/* Thread stack size */
#ifndef THREAD_APP_STACK_SIZE
#define THREAD_APP_STACK_SIZE   (3 * 1024)
#endif /* THREAD_APP_STACK_SIZE */
/*  Corresponding  error  code,  inquire  API  Guide  section
4.4.1 */
#define APP_ERR_CHECK(err_code) \
do {                        \
   if ((err_code) != DWM_OK) \
      printf("err:    line(%u)    code(%u)",    __LINE__,
(err_code));\
} while (0)                  \
#define MSG_INIT   \
   "\n\n"\
   "App   :  dist-acc-userapp\n"\
   "Built :  " __DATE__ " " __TIME__ "\n"\
   "\n"
/**
```

```
 * Event callback
 * @param[in] p_evt  Pointer to event structure */
void on_dwm_evt(dwm_evt_t *p_evt)
{
   int len;
   int i;
   switch (p_evt->header.id) {
   /* New location data */
   case DWM_EVT_LOC_READY: /* check API Guide section
4.4.11 evt_id_map for more detail */
            printf("\nT:%lu ", dwm_systime_us_get());
      for (i = 0; i < p_evt->loc.anchors.dist.cnt; ++i) {
                  printf("D:");
                  printf("%lu                          ",
p_evt->loc.anchors.dist.dist[i]);    /* Distance to the
anchor. There is only 1 anchor in the network now*/
      }
            /* Acc Value Obtain */
            int8_t data[6];
            const uint8_t addr = 0x19; // some address of
the slave device
            data[0] = 0xAA;    // Pre-allocated pointer
address by dwm_i2c_write API
            data[1] = 0xBB; data[2] = 0xCC; data[3] = 0xDD;
data[4] = 0xEE; data[5] = 0xFF;
            dwm_i2c_write(addr ,data, 6, true);
            dwm_i2c_read(addr, data, 6);
            printf("A:%d,%d,%d",
(data[0]<<8)+data[1],(data[2]<<8)+data[3],(data[4]<<8)+da
ta[5]);
      break;
   default:
      break;
   }
}
```

```
/**
 * Application thread
* @param[in] data  Pointer to user data
 */
void app_thread_entry(uint32_t data)
{
   dwm_cfg_t cfg;
   uint8_t i2cbyte;
   dwm_evt_t evt;
   int rv;
   uint8_t label[DWM_LABEL_LEN_MAX];
   uint8_t label_len = DWM_LABEL_LEN_MAX;
       dwm_pos_t pos;
   /* Initial message */
   printf(MSG_INIT);
   /* Get node configuration */
   APP_ERR_CHECK(dwm_cfg_get(&cfg));
   /* Update rate set to 0.1 second, stationary update rate
set to 0.1 seconds */
   APP_ERR_CHECK(dwm_upd_rate_set(1, 1));
   /*  Sensitivity  for  switching  between  stationary  and
normal update rate */
   APP_ERR_CHECK(dwm_stnry_cfg_set(DWM_STNRY_SENSITIVITY_
NORMAL));
   /* Register event callback */
   dwm_evt_listener_register(
          DWM_EVT_LOC_READY | DWM_EVT_USR_DATA_READY |
          DWM_EVT_BH_INITIALIZED_CHANGED |
          DWM_EVT_UWBMAC_JOINED_CHANGED, NULL);
   /* Test the accelerometer */
   i2cbyte = 0x0f;
   rv = dwm_i2c_write(0x33 >> 1, &i2cbyte, 1, true);
   if (rv == DWM_OK) {
      rv = dwm_i2c_read(0x33 >> 1, &i2cbyte, 1);
      if (rv == DWM_OK) {
```

```
            printf("Accelerometer chip ID: %u\n", i2cbyte);
        } else {
            printf("i2c: read failed (%d)\n", rv);
        }
    } else {
        printf("i2c: write failed (%d)\n", rv);
    }
    rv = dwm_label_read(label, &label_len);
    if (rv == DWM_OK) {
        printf("LABEL(len=%d):", label_len);
        for (rv = 0; rv < label_len; ++rv) {
            printf(" %02x", label[rv]);
        }
        printf("\n");
    } else {
        printf("can't read label len=%d, error %d\n",
label_len, rv);
    }
    while (1) {
        /* Thread loop */
        rv = dwm_evt_wait(&evt);
        if (rv != DWM_OK) {
            printf("dwm_evt_wait, error %d\n", rv);
        } else {
            on_dwm_evt(&evt);
        }
    }
}
/**
 * Application entry point. Initialize application thread.
* @warning ONLY ENABLING OF LOCATION ENGINE OR BLE AND
CREATION AND STARTING OF
 * USER THREADS CAN BE DONE IN THIS FUNCTION
 */
void dwm_user_start(void)
```

```
{
   uint8_t hndl;
   int rv;
   dwm_shell_compile();
   //Disabling ble by default as softdevice prevents
debugging with breakpoints (due to priority)
   //dwm_ble_compile();
   dwm_le_compile();
   dwm_serial_spi_compile();
   /* Create thread */
   rv         =         dwm_thread_create(THREAD_APP_PRIO,
app_thread_entry, (void*)NULL,
          "app", THREAD_APP_STACK_SIZE, &hndl);
   APP_ERR_CHECK(rv);
   /* Start the thread */
   dwm_thread_resume(hndl);
}
```

# Appendix C

# MATLAB script of the interface

```
%% S1.define globally variable
global s str temp data_tag;
%% S2.Serial settings
s = serial('COM3');
set(s,'Databits',8);
set(s,'Stopbits',1);
set(s,'Baudrate',115200);
set(s,'Parity','none');
%% S7.Serial event callback settings (1st method (bytes)
Event-driven method interrupt)
set(s,'BytesAvailableFcnMode','byte');
set(s,'BytesAvailableFcnCount',27);     %set the interrupt
trigger: when receives 27 bytes of data, interrupt occurs.
s.BytesAvailableFcn =@ReceiveCallback;%customize         a
callback function

%% Callback function
function ReceiveCallback( ~,~)
global s str temp data_tag;
global str;
global temp;
global data_tag;
str = fscanf(s);  %read data from buffer
pattern = '(-)?\d{1,10}';
temp  =  regexp(str,  pattern,  'match');    %use  regular
expression to extract time/dist/acc data
data_tag = [data_tag; temp];
end
```

# Bibliography

[1] Jwo D J, Cho T S. A practical note on evaluating Kalman filter performance optimality and degradation[J]. Applied Mathematics and Computation, 2007, 193(2): 482-505.

[2] Welch G, Bishop G. An introduction to the Kalman filter[J]. 1995.

[3] Gustafsson F. Particle filter theory and practice with positioning applications[J]. IEEE Aerospace and Electronic Systems Magazine, 2010, 25(7): 53-82.

[4] McDonald C. Machine learning fundamentals (I): Cost functions and gradient descent[J]. Towards Data Science, 2017, 27.

[5] Norrdine A. An algebraic solution to the multilateration problem[C]//Proceedings of the 15th international conference on indoor positioning and indoor navigation, Sydney, Australia. 2012, 1315.

[6] Li X, Wang Y, Khoshelham K. A robust and adaptive complementary kalman filter based on Mahalanobis distance for Ultra Wideband/Inertial Measurement Unit fusion positioning[J]. Sensors, 2018, 18(10): 3435.

[7] Bishop G, Welch G. An introduction to the kalman filter[J]. Proc of SIGGRAPH, Course, 2001, 8(27599-23175): 41.

[8] Capra M. UWB Tracking System for Patient Monitoring in Home Environment[D]. Politecnico di Torino, 2018.

[9] Embedded F. ultra-low-power high-performance 3-axis" femto" accelerometer[J].

[10] IENI M. Realization and performance evaluation of a hybrid UWB/WiFi indoor localization system[J]. 2018.

[11] ZHANG K, YANG J, DU X. Design and Implementation of Serial Communication Based on DSP [J][J]. Microprocessors, 2010 (6): 9.

[12] LIU C, DU D, PAN J. Predictive control for lane control systems using a small deviation model[J]. Journal of Tsinghua University (Science and Technology), 2015 (10): 8.

[13] ZHAN H, YANG R, ZHOU X. Buoy Passive Tracking System Model[J]. Computer Engineering, 2009, 2009(20): 101.

[14] Quarteroni A, Saleri F, Gervasio P. Scientific computing with MATLAB and Octave[M]. Berlin: Springer, 2006.

[15] Sergio Bittanti. Model Identification and Data Analysis Slides. http://corsi.dei.polimi.it/IMAD/IMAD_MI_AUT/, 2017-2018

# Acknowledgment

I would like first to express my great gratitude to my supervisor Prof. Farina and the co-supervisor Prof. Bascetta, for their in-depth and very patient guidance throughout this work. Without their help, it is not possible for this thesis to be completed. In fact, their solid professional knowledge, accurate scientific intuition, and more importantly their selfless efforts all left great impression on me. The time spent with them on this thesis means a lot to me, and this experience is for sure lifetime benefiting.

I would like to express my thanks to my family, my father Dong Jianqing, my mother Wang Hongyu and my grandparents Dong Xuetian and Jin Ping. I would like to thank them for their academic support to me. Their strong support allowed me to study in a foreign country when I was in my twenties.

I also want to thank my classmates and friends who have accompanied me for many years. My roommates Long Yuxiang, Hu Liang, Tang Wenshuai helped me a lot during this thesis work. We encouraged each other, we cursed each other more often. We are friends, for always. Also, my ex-roommate and good friends Zhou Jingyu, Li Mingju and Yang Zongshuai. Thanks everyone for being my friends for seven years.

Thanks to all the professors, teachers, classmates, those who have ever helped me, and all the guys I know, or I do not, during my Master Degree years in Politecnico di Milano.