Three-Dimensional Localisation using Cricket System

TECHNICAL REPORT: CES-516

ISSN 1744 - 8050

Sen Wang[1] and Huosheng Hu[2]

School of Computer Science and Electronic Engineering
University of Essex

December 27, 2011

[1]Email: swangi@essex.ac.uk
[2]Email: hhu@essex.ac.uk

Abstract

The Cricket system is a kind of wireless sensor networks used for indoor localisation, which is developed by MIT and Crossbow. It can be deployed to compute the location information for various devices, such as computer and cell phone. This report firstly describes the initialisation of Cricket and the additional resources. Then, a three-dimensional localisation algorithm is designed based on the optimisation method. Its performance is analysed in simulation and compared with the two existing localisation algorithms. The simulation results demonstrate its high localisation accuracy. Moreover, a new program, namely CricketMFC, is developed to initialise the localisation system by deploying Java, Matlab and Microsoft Foundation Classes (MFC). It also can be used to compute the three-dimensional position and display the results in three-dimensional coordinates in real time. The real experiment clearly verifies that the designed algorithm provides very good localisation results and the CricketMFC is an efficient localisation system.

Contents

Chapter 1

Introduction

1.1 Cricket system

The Cricket System is a low-power wireless sensor network (WSN) for indoor localisation. It is intended for the areas where the Global Positioning System (GPS) cannot work. Each Cricket hardware unit consists of an ultrasound transmitter and receiver as well as the wireless communication module (see Fig. 1.1). Although the maximum ranging distance is short (only 10m), the accuracy can be as high as 1cm to 3cm.

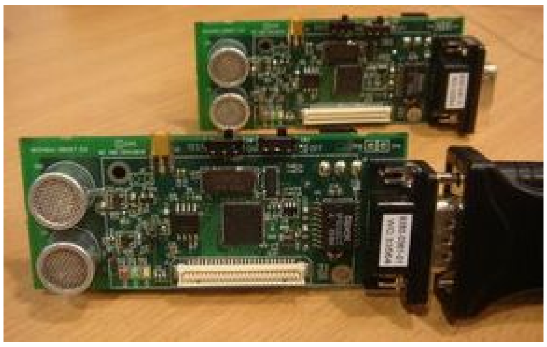


Figure 1.1: Cricket Hardware

1.1.1 Principal

System frame

In Cricket system, there are two types of nodes - beacon and listener. Each node can be set as either of them by the software control.

- Beacon serves as the reference for the listener. It is artificially deployed at a known location, and it should not be moved once fixed. The coordinate value (x,y,z) of each beacon should be initialised before running the localisation.

- Listener is constantly moving in the area monitored by the beacons. With the estimated distance between the listener and the beacons, the listener location can be calculated by the localisation algorithm. Therefore, the device location can be found out as long as a listener is attached.

The architecture of the Cricket is shown in Fig. 1.2. It can be seen that the listener receives the data transmitted by the beacons and then sends it to the host device (for example, laptop) by serial interface. When the localisation application running on the host device collects all the required information, the listener location can be computed.

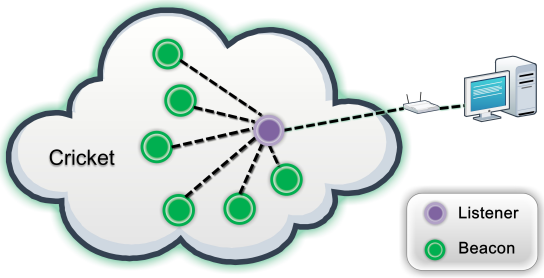


Figure 1.2: Cricket architecture

Distance measurement

Cricket adopts both the radio frequency (RF) and ultrasound techniques to determine the distance. The distance between the listener and the beacon is measured by Time Difference of Arrival (TDOA), which is one of the typical ranging techniques. Specifically, each beacon transmits a RF signal and an ultrasonic sound simultaneously. The RF signal is usually considered to be received immediately since it propagates as fast as light. However, the ultrasonic sound takes some time to arrive at the listener. Consequently, the listener receives the two signals at different times. When both the RF and ultrasonic sound transmitted by a beacon are received, the listener can estimate the distance between them by

$$dist = V \cdot T = V \cdot (Time_{Sound} - Time_{RF}), \tag{1.1}$$

where $dist$ is the estimated distance, V is the sound propagation velocity, and $Time_{RF}$ and $Time_{Sound}$ are the arrival times of the RF and sound respectively.

1.1.2 System deployment

The Cricket system must be pre-deployed before running. The detailed procedure is as follows:

- Several beacons must be fixed to the steady positions. The number of beacons of two-dimensional or three-dimensional localisation has to be more than three or four respectively, and they cannot be collinear or coplanar.

- One listener must be attached to the host device. Besides, it is important to ensure the listener can always satisfy the requirement of the minimum number of beacons during its movement. It is worth mentioning that the listener should have the line-of-sight connectivity to the beacons and be within the maximum range.

1.2 Software installation

1.2.1 Cygwin

Although the Windows system cannot run the applications developed in Linux directly, the Cygwin provides substantial Linux API for programmes developed for Linux to run on Windows.

Note : During the Cygwin installation, don't forget to select the "Devlp" item, which will install the programming development tools, such as Make, gcc.

1.2.2 Cricketd

As the basic program of the whole Cricket system, Cricketd is designed to receive all the listener data through the serial port, which is similar to PuTTY. It binds to port 2947 and can be accessed by telnet. The Cricketd is configured as follows:

1. Compiling: the following commands should be run in `Cricket2.0/src/cricketd/` directory:

   ```
   autoconf
   ./configure
   make
   ```

2. Running: the command `./cricketd` launches the Cricketd:

   ```
   ./cricketd -p <serial port> -s 115200
   ```

   where `<serial port>` is address of the serial port connected with Cricket. In my computer, because the serial port is `ttyS3`, the corresponding command is `./cricketd -p /dev/ttyS3 -s 115200`.

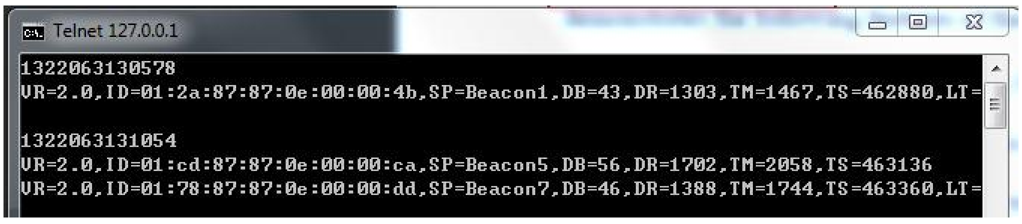
3. Connecting: the telnet can access the Cricketd by connecting to port 2947. The procedure is:

   `CMD` and type `telnet 127.0.0.1 2947` is run in windows,

   type `r` can enable or disable the connection.

   If the host connects a listener, the CMD should start to receive Cricket messages as shown in Fig. 1.3.

   Note : The telnet service has to be manually enabled in Windows 7 operating system.



```
1322063130578
VR=2.0,ID=01:2a:87:87:0e:00:00:4b,SP=Beacon1,DB=43,DR=1303,TM=1467,TS=462880,LT=

1322063131054
VR=2.0,ID=01:cd:87:87:0e:00:00:ca,SP=Beacon5,DB=56,DR=1702,TM=2058,TS=463136
VR=2.0,ID=01:78:87:87:0e:00:00:dd,SP=Beacon7,DB=46,DR=1388,TM=1744,TS=463360,LT=
```

Figure 1.3: Telnet connected to Cricketd

1.2.3 BeaconConfigDemo

The BeaconConfigDemo is also an application for Cricket system. It displays all the beacons in both a list and a two-dimensional picture and processes the data received by Cricketd. When no fewer than three beacons are mounted on a flat surface, the BeaconConfigDemo can calculate the two-dimensional location of the listener. Its setting parameters can be changed in the file `NMSDemo/2004/BeaconConfigDemo/launchall.sh`. The `NUMBEACONS` and `CRICKETD_IPADDR` should be set in advance. The BeaconConfigDemo is operated as follows:

1. Launching. Cricketd must be running on the host device to which a listener is attached. Then, the BeaconConfigDemo can be launched by
   `cd NMSDemo/2004/BeaconConfigDemo` and ./launchall.sh commands.
   Two programs will be launched. One is Beacon Finder, which displays the measured distance of each beacon; the other is Beacon Configuration, which configures the beacon coordinate system and tracks the listener position.
2. Running. There are two steps. (a) Configuring the coordinate system. Steadily holding a listener underneath a reference beacon lets the system measure the distances to all other beacons with respect to the reference beacon. The first reference beacon that is being calibrated by the listener defines the origin of the coordinate system in the lower left corner. Then, the second reference beacon is calibrated. The line extended between the first and second beacons becomes the horizontal axis of the beacon coordinate system. (b) Tracking and displaying mode. When the coordinate system is created, the BeaconConfigDemo begins to localise. The real time position of listener is represented as a red square in the Beacon Configuration.

Chapter 2

Localisation Algorithm

The localisation is discussed in this chapter, especially the ranging techniques and localisation algorithms. At first, a cyclical method which aims at designing, analysing and improving the localisation algorithm together is introduced. Then, the main section - localisation algorithm - is presented after the deployment environment is described.

2.1 Methodology

An approach combining simulation with real experiment is adopted to explore the drawbacks of the algorithm and system. There are three steps: simulation, real experiment and improvement. It is shown in Fig. 2.1.

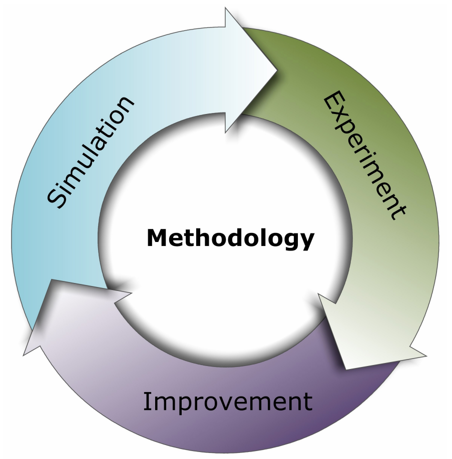


Figure 2.1: Methodology

First, the algorithm is designed and evaluated by simulation according to the two-step localisation procedure - ranging and localisation. The ranging estimates the distance between the beacon and listener. It plays a critical role in the whole system because the original distance values have considerable influences on the accuracy of the result. Because the propagation path suffers from various disturbances, it is impossible to measure the distance accurately. Therefore, localisation algorithm becomes increasingly significant since it can dramatically improve the localisation accuracy. Its primary purpose is to compute the listener position close to the real one as much as possible even though the estimation distances are poor. Hence, the algorithm should be robust to various disturbances and uncertainties, which is evaluated and improved by simulation.

The designed algorithm will be tested in real world to assess the performance and find out the shortages. This is the second phase - real experiment. Since some disturbances cannot be emulated entirely, the algorithm with good simulation results is not always adequate to the real situation. During the experiment, the statistic data can be analysed to find the causes of the problem.

Based on the analysis, some solutions will be investigated to improve the algorithm. A new process may be required or the existing technique should be enhanced. When the algorithm is improved by simulation, it will be validated in practice again.

2.2 Experiment environment

All the beacons should be fixed according to the requirements described in Subsection 1.1.2 before algorithm investigation. Three beacons are attached to the ceiling while the other one is deployed on the bookshelf (see Fig. 2.2). The cartesian coordinates are defined as illustrated in the figure. The origin is the corner where the walls and floor meet near the door. Then, the coordinate values of beacons are listed in Table 2.1. These positions are deliberately imprecise because some errors can be introduced. By this way, the robustness of the localisation algorithm can be tested more effectively. During the localisation, the ultrasound receiver and transmitter of listener have to face the beacons all the time in order to receive the data from beacons. In other words, listener must have line-of-sight connectivity to the beacons.

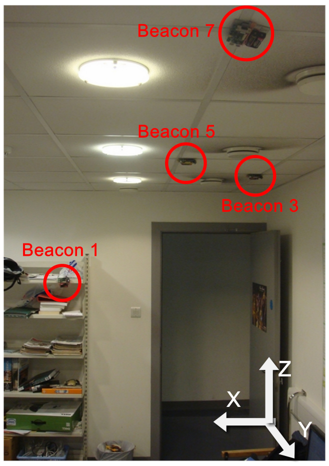


Figure 2.2: Deployment in lab

Beacon Mark	Location (m)		
	X	Y	Z
Beacon1	1.78	0.42	1.65
Beacon3	0.4	1.2	2.36
Beacon5	0.9	1.8	2.36
Beacon7	0.9	3.8	2.36

Table 2.1: Beacon Locations

2.3 Localisation algorithm

A brief introduction to the typical ranging and location computation techniques is presented in this section. Besides, a new localisation algorithm is proposed.

2.3.1 Introduction

The localisation can be divided into two types: range-free and range-based. The range-free methods usually rely on connectivity. For example, the algorithm in [1] considers the centroid of all the beacons that can communicate with listener directly as the location. In contrast, the range-based algorithms adopt the distance estimation as well as the theories of geometry and statistics to calculate the location. Generally, the range-based algorithms are more complicated than range-free ones but better in terms of the accuracy.

2.3.2 Ranging techniques

There are several kinds of ranging techniques, such as Time of Arrival, Time Difference of Arrival and Received Signal Strength Indicator. Because of their own characters as well as advantages and disadvantages, the method which can keep a balance between the requirement on accuracy and constrain on hardware, etc. should be chosen for a WSN localisation system.

Time of Arrival

Time of Arrival (TOA) is the signal travel time from the transmitter to receiver. According to the propagation theory, the distance is equal to

$$Dist = V \cdot T \tag{2.1}$$

where V is the spreading speed, and T is travel time. Sound is the common signal because the velocity of radio frequency is so fast that the time is too short to be detected. The range accuracy of this method is high but the time synchronisation is required, which is difficult for a large scale WSN.

Time Difference of Arrival

Time Difference of Arrival (TDOA) has been introduced in Subsection 1.1.1. Although TDOA can estimate the distance as good as TOA, it needs more hardware and energy.

Received Signal Strength Indicator

Received Signal Strength Indicator (RSSI) estimates the distance by measuring the radio signal strength. The signal strength has the mathematical relationship with distance since the RF is attenuated during the propagation. The weaker the signal is, the longer the distance will be. The signal strength can be measured during the communication without extra hardware and process. Therefore, RSSI is an energy-saving and low-cost method. But it is so sensitive to the disturbances that the distance estimation is very imprecise.

2.3.3 Localisation

Two basic techniques named trilateration and maximum likelihood estimation are often used for range-based localisation.

Trilateration

The trilateration determines the location using the geometry of circles and spheres. The location is the intersection point of the three circles or four spheres in two-dimensional or three-dimensional geometry respectively as illustrated in Fig. 2.3 and Fig. 2.4. However, there must be errors between the estimated distances and the true values in the real environment, changing the intersection from a point to an overlapped area or solid. Consequently, some additional technologies should be used.

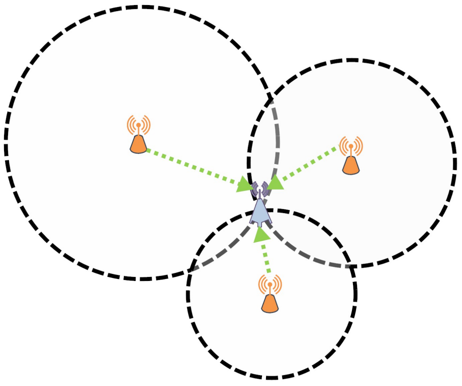


Figure 2.3: Two-dimensional trilateration

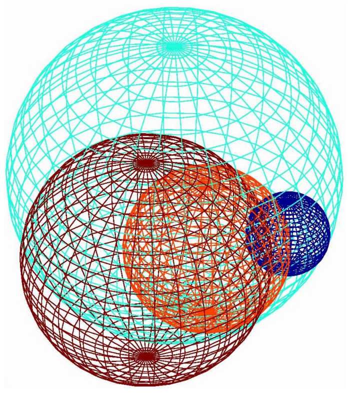


Figure 2.4: Three-dimensional trilateration

Maximum likelihood estimation

By contrast, maximum likelihood estimation (MLE) is better than trilateration [2]. Because of the error and noise, it is difficult for the trilateration to perform the localisation with good accuracy. As shown in Fig. 2.5, there are n beacons which can communicate with listener. The B_i is the ith beacon with the corresponding coordinate value (x_i, y_i) and estimated distance d_i. The set of equations is

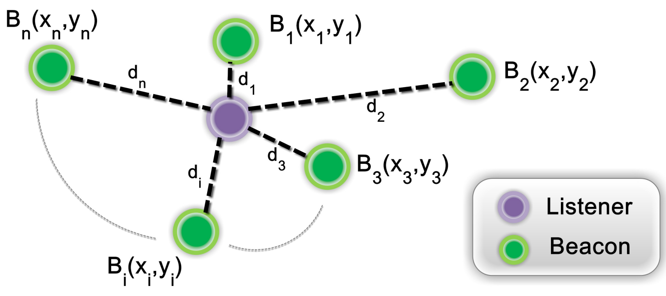


Figure 2.5: Maximum likelihood estimation

$$\begin{cases} \sqrt{(x-x_1)^2+(y-y_1)^2}=d_1 \\ \sqrt{(x-x_2)^2+(y-y_2)^2}=d_2 \\ \vdots \\ \sqrt{(x-x_n)^2+(y-y_n)^2}=d_n \end{cases} \tag{2.2}$$

where (x, y) is the unknown listener location. After being simplified, the equations can be described as

$$AX = b \tag{2.3}$$

where $X = [\ x \quad y\ ]^T$ and

$$A = \begin{bmatrix} 2(x_1-x_n) & 2(y_1-y_n) \\ \vdots & \vdots \\ 2(x_{n-1}-x_n) & 2(y_{n-1}-y_n) \end{bmatrix} \quad b = \begin{bmatrix} {x_1}^2-{x_n}^2+{y_1}^2-{y_n}^2+{d_n}^2-{d_1}^2 \\ \vdots \\ {x_{n-1}}^2-{x_n}^2+{y_{n-1}}^2-{y_n}^2+{d_n}^2-{d_1}^2 \end{bmatrix} \tag{2.4}$$

The optimal solution of (2.3) is the (x, y) that reaches the minimum of $\|AX - b\|_2$. $\|AX - b\|_2$ is

$$\|AX-b\|_2^2 = (AX-b)^T(AX-b) = X^TA^TAX - 2X^TA^Tb + b^Tb \tag{2.5}$$

Let the gradient of (2.5) equal zero

$$2A^TAX - 2A^Tb = 0 \tag{2.6}$$

So

$$\hat{X} = (A^TA)^{-1}A^Tb \tag{2.7}$$

the $\hat{X}$ is the optimal listener location. The three-dimensional case can be derived by referring to the above analysis.

Optimisation algorithm

A novel refinement algorithm which can restrain the errors and improve the location accuracy is presented. Because the ranging measurement suffers from many disturbances, the estimated distance cannot approximate to the real value. For example, the results of trilateration and MLE are not accurate. Therefore, some additional processing, such as refinement, should be introduced to improve the accuracy of the preliminary outcome.

The refinement algorithm is based on the optimisation theory, especially the Steepest Decent (SD) and Newton method. The main idea is to transform the localisation refinement into an optimisation problem. An error cost function $f(x, y)$ is constructed in order to use the two optimal methods. According to (2.2), it can be

$$\begin{aligned} f(x,y) &= \sum_{i=1}^{n}(\|P - p_i\| - d_i)^2 \\ &= \sum_{i=1}^{n}(\sqrt{(x-x_i)^2 + (y-y_i)^2} - d_i)^2 \end{aligned} \tag{2.8}$$

where P is the unknown location, and p_i is the position of the ith beacon. The smaller the (2.8) is, the more accurate the location will be. Then, the optimisation method is used to find the solution that can reach the minimum of the cost function. When the optimisation problem is resolved, the result is considered as the optimal location.

Steepest Decent SD is one of the classical unconstrained optimisation algorithms. Although it converges on the optimal point slowly, it needs low computational cost. Each result is searched along the negative gradient direction of $f(x)$:

$$\begin{aligned} d_k &= -\nabla f(x_k) \\ &= -(\frac{\partial f(x_k)}{\partial x_1}, \frac{\partial f(x_k)}{\partial x_2}, \cdots, \frac{\partial f(x_k)}{\partial x_n}) \end{aligned} \tag{2.9}$$

When $\|d_k\|$ is less than the required precision, a local optimal result can be acquired. However,

$$\nabla f(x_{k+1})^T \nabla f(x_k) = 0 \tag{2.10}$$

which means the iteration approaches the optimal result in the shape of zigzag. When it is far away from the optimal result, the convergence rate is fast. However, each iterative step will take a very long time in the later period. As a consequence, the SD is unsuitable to be used directly because the localisation of WSN requires the real-time update.

Fast optimisation algorithm

In order to overcome the drawback of SD, the Newton method can be utilised. It is much faster than SD, but it has many strict requirements on the initial value. It may not be convergent with a bad initial number. Hence, a fast algorithm is proposed to have the advantages of both SD and Newton methods.

The algorithm is divided into three steps. First, a location is calculated by MLE. Then, the SD is used to optimise it with a low precision. Because of the SD character described in Subsubsection 2.3.3, a value which often satisfies the initial requirement of Newton is quickly converged. With it, Newton method can be launched to achieve a higher precision. The optimal result can be calculated very fast. If the Newton is divergent, the whole algorithm will be executed by SD method. The three-dimensional localisation of Fast optimisation algorithm is shown in 1.

Algorithm 1 Fast optimisation algorithm

```
1:  Get an unknown node index;
2:  Find all the neighbour beacon nodes of this node;
3:  if Number of neighbour beacons is not fewer than 4 then
4:      Calculate a location by MLE;
5:      Optimise the location by SD with a low precision;
6:      Go on optimising the location by Newton with a much higher precision;
7:      if Iterative time of Newton exceeds the maximum time then
8:          Use SD with a high precision;
9:      end if
10: else
11:     Failure;
12: end if
```

2.3.4 Simulation

The designed algorithm is simulated and analysed to evaluate the performance. The three-dimensional diagrams of localisation results are illustrated to describe the errors more directly.

The node placement is shown in Fig. 2.6. The red stars are beacons, which are deployed according to Section 2.2. The blue circles are the locations of listener, which are distributed in the whole space stochastically. The number of beacons is 100 and they are in the interior of a $3m \times 5m \times 2.36m$ cube.

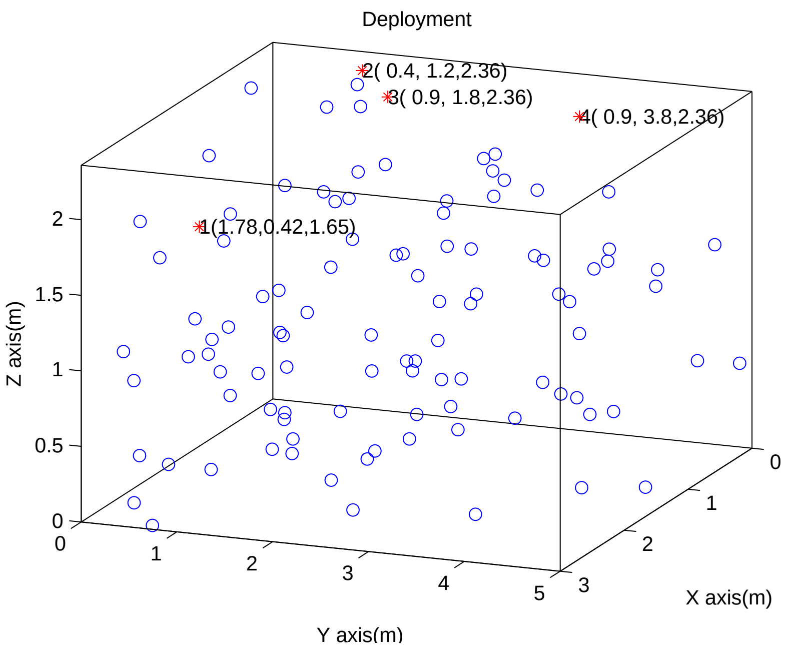


Figure 2.6: Deployment of the simulation

The noises whose probability density function is Gaussian distribution are imposed on the estimated distances to simulate the influences of the uncertain disturbances on the accuracy as well as the algorithm performance. If the algorithm is good enough, it can restrain the disturbances greatly and localise the unknown nodes accurately.

Results

Two existing algorithms - MLE and Multidimensional Scaling Map (MDS-MAP) [3,4] - are used to compare the algorithm performance.

MLE is chosen because it is a classical localisation algorithm and better than trilateration [2]. The localisation result of the deployment in Fig. 2.6 is shown in Fig. 2.7. The blue line connects the true and estimated locations of a listener. The blue circle is the estimated position and the other end is the real one. Hence, the length of the line stands for the magnitude of localisation error. In other words, the longer the line is, the bigger the error is. Therefore, the MLE localisation result is bad since many locations are estimated out of the space. Besides, the errors of the majority of listeners are large. This also indicates the influence of the disturbance is serious.

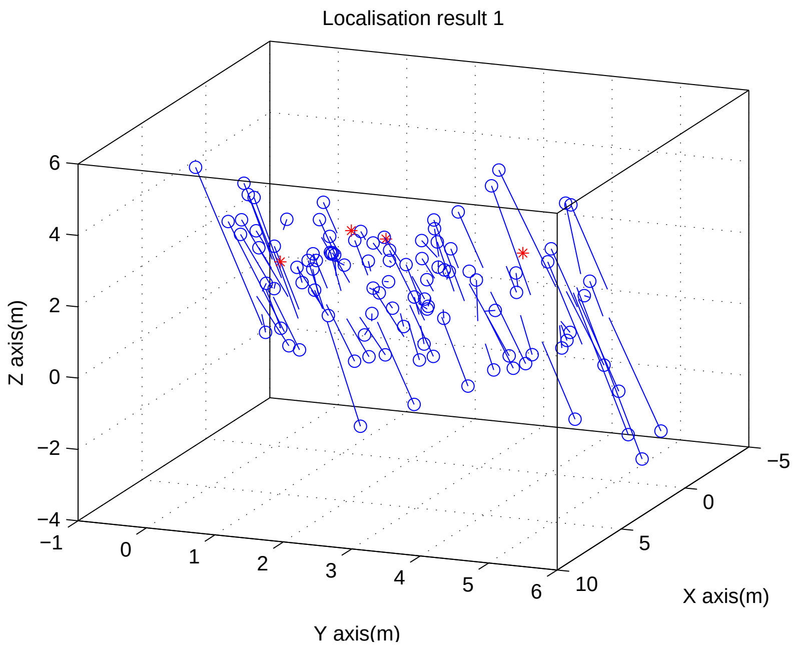


Figure 2.7: MLE result

MDS-MAP algorithm can localise very accurately based on multidimensional scaling (MDS) technique. It is also recognised as a robust algorithm [5]. During the process of solving the localisation problem, two maps - relative map and absolute map - need to be constructed. First, the shortest path between each pair of nodes is created. Then, the relative map is produced by executing MDS. At last, the relative map is transformed to the absolute map by only three or four beacons. The absolute map is the localisation result as demonstrated in Fig. 2.8. It can be seen that the outcome is better than that of MLE in terms of the errors, which verifies that the MDS-MAP is robust to the noise.

Fast optimisation algorithm designed in Subsubsection 2.3.3 is simulated and the localisation result is shown in Fig. 2.9. It clearly demonstrates that the result is much better than the previously mentioned two algorithms. Most of the unknown nodes are estimated near their own real locations and the errors are small. The result proves the excellent performance of the proposed algorithm.

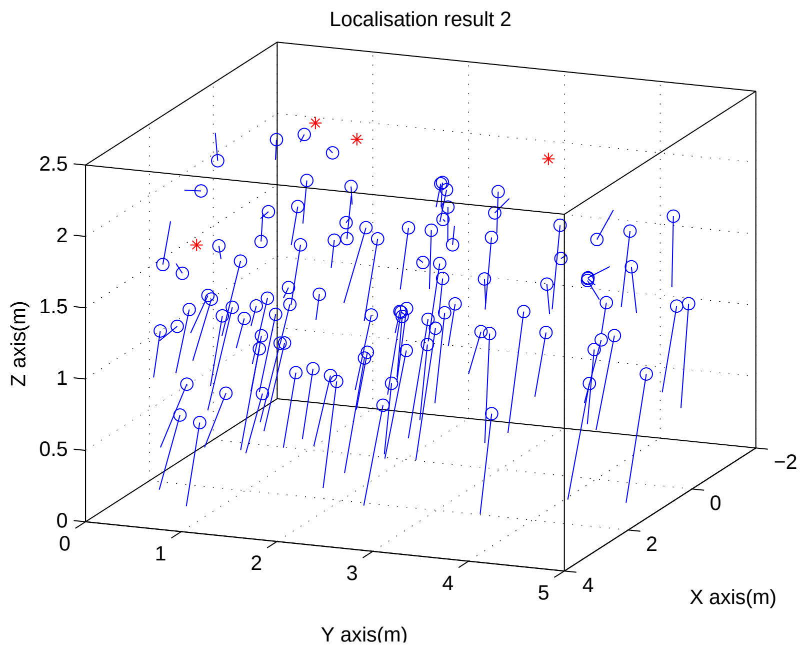


Figure 2.8: MDS result

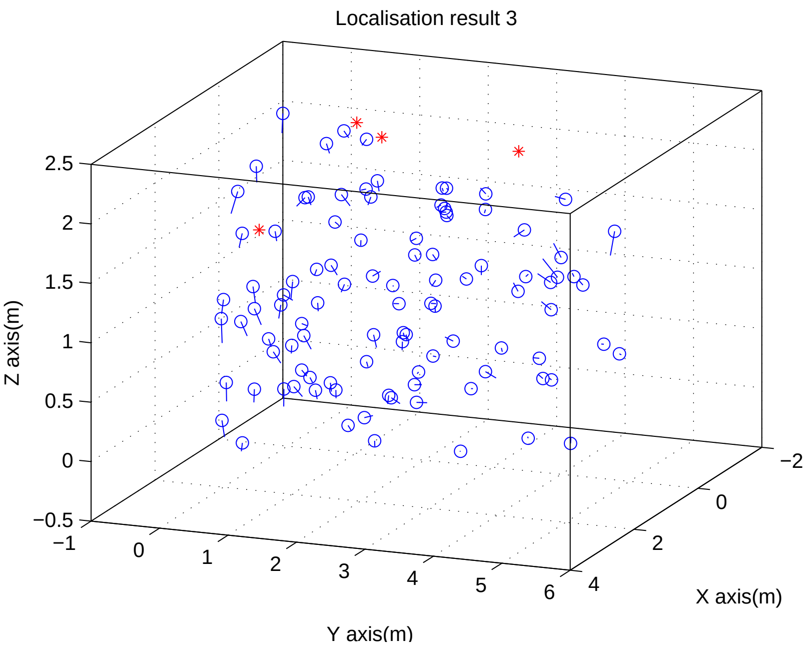


Figure 2.9: Fast optimisation algorithm result

Comparison

The mean errors are described in a bar chart (see Fig. 2.10). The height of the bar represents the magnitude of the mean error, which is the average localisation error of all the unknown nodes. The mean error can evaluate the algorithm performance more objectively. The Fig. 2.10 illustrates that the fast optimisation algorithm has the smallest error, which is just about a tenth and one third of MLE and MDS-MAP respectively. The comparison means that the proposed algorithm improves the accuracy dramatically.

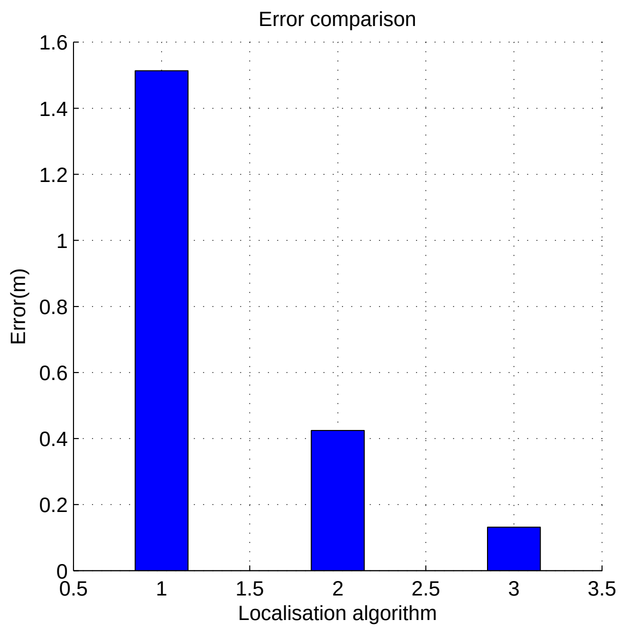


Figure 2.10: Error comparison of the three algorithms

Chapter 3

CricketMFC Development

A program named CricketMFC is developed. It combines the benefits of Java, Matlab and MFC to initialise the localisation system. Moreover, it can compute the three-dimensional location by the algorithm proposed in Subsubsection2.3.3 and display it in three-dimensional coordinate in real time. It is based on BeaconConfigDemo but better than it in terms of accuracy and dimension.

3.1 Architecture

The CricketMFC is developed based on the existing resources of Cricket, such as Cricketd and BeaconConfigDemo. Therefore, it is more efficient and stable. The program mainly consists of system initialisation, data acquisition, data segmentation, location estimation and user interface. The CricketMFC architecture is shown in Fig. 3.1. The left part is Cricket hardware, while the right area is the software - the Java class, CricketMFC and Matlab algorithm.

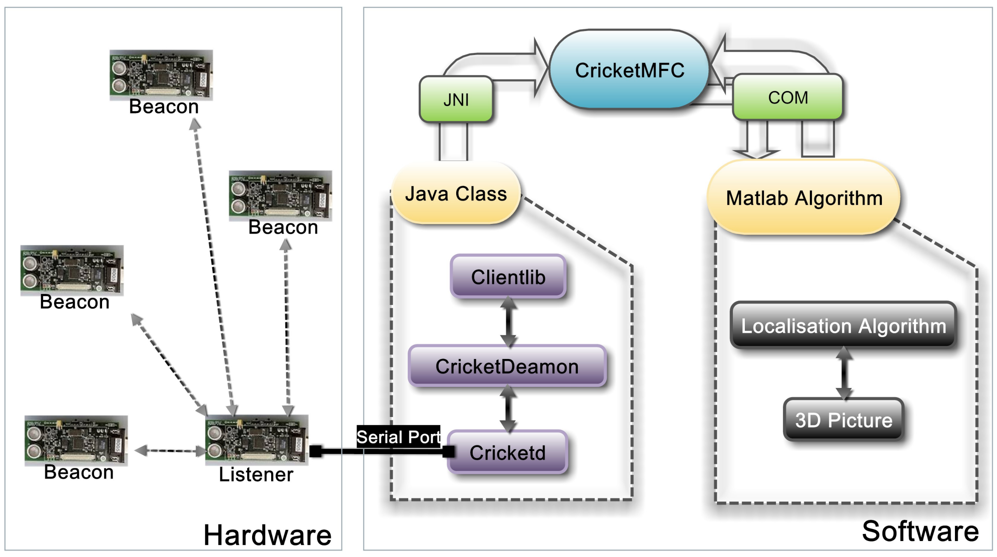


Figure 3.1: CricketMFC architecture

3.2 System initialisation

System initialisation is the foundational part of the software. The initialisation of CricketMFC is more important owing to its combination with Java and Matlab. Besides the typical initialisation (for example, MFC initialisation), CricketMFC has to configure the Java Native Interface (JNI) and COM component. The JNI and COM are introduced in Subsection 3.3.2 and 3.5.2 respectively.

3.2.1 JNI initialisation

It is necessary for the JNI to be initialised successfully because CricketMFC relies on it to get the original data from the Java class. If the initialisation fails, CricketMFC cannot work. The initialisation procedure is shown in Fig. 3.2.

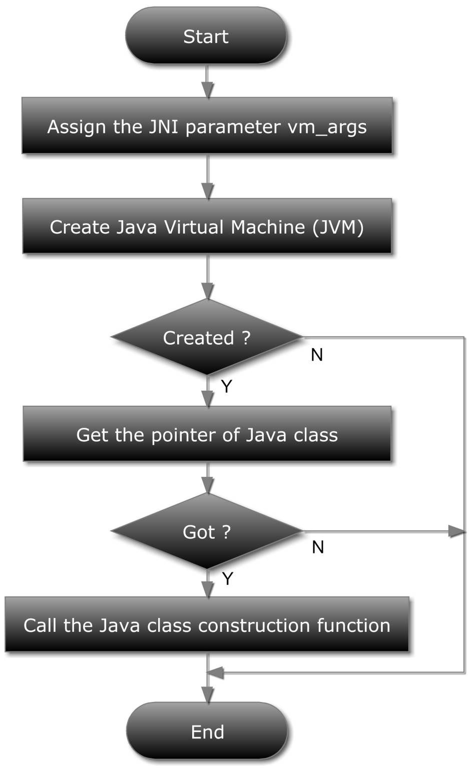

Start
Assign the JNI parameter vm_args
Create Java Virtual Machine (JVM)
Created ?
N
Y
Get the pointer of Java class
Got ?
N
Y
Call the Java class construction function
End

Figure 3.2: Java Native Interface Initialisation

Note: The `jar` files used in Java class must be included by setting vm_args.

3.2.2 COM component initialisation

The Matlab COM component is the channel of using Matlab localisation algorithm directly in CricketMFC. The COM component can be created and initialised by the `CoCreateInstance` function with a specified class identifier (CLSID) in MFC. The algorithm cannot return the correct results if the Matlab COM component is not created correctly.

3.3 Data acquisition

The data is transmitted from the listener to CricketMFC by a self-defined Java class, which is based on the Cricket API library. When the Cricket system is running, the Java class is periodically receiving the measurement information from serial interface connected with listener. Then, the Java class and CricketMFC can exchange the messages via JNI.

3.3.1 Java class

A Java class is utilised to communicate with Cricket and CricketMFC directly, which means it serves as a bridge between them (see Fig. 3.1). It transfers the data received from Cricket to CricketMFC.

It uses the Cricket Client library (Clientlib). Clientlib feeds back location information by callback mechanism. Specifically, an interface named ServerBroker is instantiated to dispatch the callback handlers whenever it receives a location update from CricketDaemon. When a callback handler is triggered, the corresponding handler function will be called to deal the message. In this Java class, the handler function of location update saves the new location and returns it. Therefore, the CricketMFC can get the latest location as long as the Java class handler function is called, which can be implemented by JNI.

3.3.2 JNI in CricketMFC

JNI Introduction

"The Java Native Interface (JNI) is a programming framework that enables Java code running in a Java Virtual Machine (JVM) to call and to be called by native applications (programs specific to a hardware and operating system platform) and libraries written in other languages such as C, C++ and assembly." [6] Therefore, the Java class, object, etc. can be defined and invoked to perform their functions by other languages through JNI.

JNI in CricketMFC

The data produced by Cricket system is essential for the program and localisation algorithm. But one bridge between Java and C++ has to be created since the Clientlib is written in Java while the CricketMFC in MFC. In CricketMFC, the JNI is used as an interface.

First of all, JNI has to be initialised (see Subsection 3.2.1). Then, a timer is defined in CricketMFC by registering a `WM_TIMER` message. The timer message handler function `OnTimer()` is periodically triggered to read the latest location by calling Java class through JNI. Actually, all the original measurement data is acquired by Java class and then sent to CricketMFC by JNI. The flow chart of this part is shown in Fig. 3.6.

3.4 Data segmentation

The original location data should be processed after acquisition. Data segmentation plays an important role in this period. It consists of two steps - string segmentation and storage.

1. String segmentation. The location data obtained from JNI is string type variable. It includes several segments which have the same structure as described in Fig. 3.3. One segment belongs to one beacon. In each segment, there is information about the sequence number of the beacon, the distance away from the listener and so on. When the whole string is got by CricketMFC, it is split into several segments according to the number of beacons.
2. Storage. Depending on the difference of the beacon number in each segment, the data is saved in the corresponding Beacon class. The beacon class contains the order number, the coordinate values (x,y,z) and the latest distance. When all the segments are analysed and saved successfully, the data segmentation is finished.

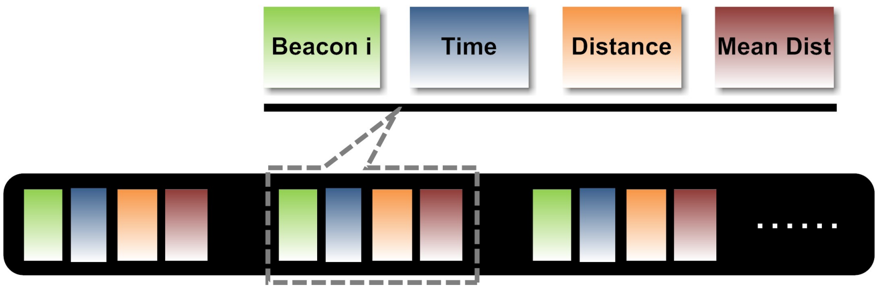


Figure 3.3: Data structure

3.5 Location estimation

CricketMFC uses the localisation algorithm proposed in Subsubsection 2.3.3 to calculate the listener location. In order to conduct the algorithm efficiently, CricketMFC introduces the hybrid programming between C++ and Matlab. There are three phases - algorithm design in Matlab, the COM component compiler and hybrid programming implementation in VC.

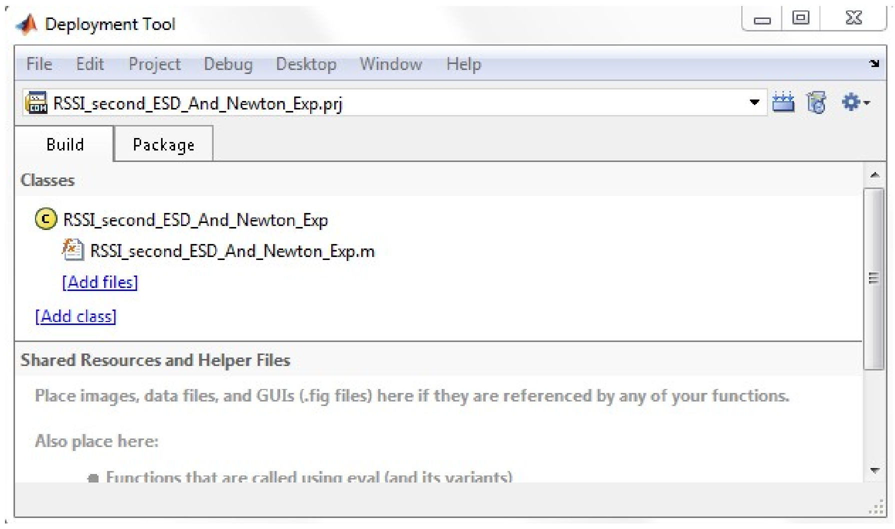
3.5.1 Design in Matlab

Because Matlab is good at numerical computing and analysis, it is an excellent approach to design, implement and analyse the algorithm together. Therefore, the localisation algorithm used in CricketMFC is produced by Matlab. The localisation principal and mathematic theory have been comprehensively introduced in Section 2.3.

A Matlab file should be created to define a function. The function has some input and output parameters. The input is the data stored in the beacon class while the output is the three-dimensional coordinate (x,y,z) estimated by localisation algorithm. Once the algorithm is successful in Matlab, the function file can be used to do next.

3.5.2 COM component compiler

There are several techniques can realise hybrid programming between Matlab and other programming languages. The COM component method is adopted by CricketMFC since it is more convenient to be migrated to the other computers. When a localisation algorithm is packaged as a function file, it can generate the COM component by the `deploytool` tool supplied by Matlab. The `deploytool` is the graphical user interface for Matlab Compiler as illustrated in Fig. 3.4.


Deployment Tool
File Edit Project Debug Desktop Window Help
RSSI_second_ESD_And_Newton_Exp.prj
Build
Package

Classes

RSSI_second_ESD_And_Newton_Exp

RSSI_second_ESD_And_Newton_Exp.m

[Add files]

[Add class]

Shared Resources and Helper Files

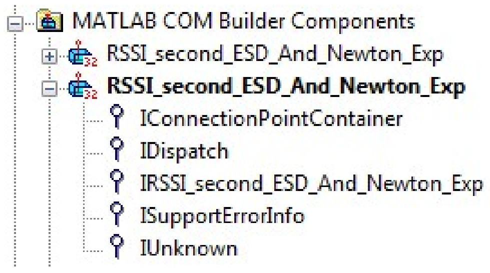
Place images, data files, and GUIs (.fig files) here if they are referenced by any of your functions.

Also place here:

Functions that are called using eval (and its variants)

Figure 3.4: Matlab Deploytool

The COM component can be registered in the system after it is successfully created. By the `OLE/COM Object Viewer` tool, the registered Matlab COM component can be found. The COM used in CricketMFC is shown in Fig. 3.5. The information in `OLE/COM Object Viewer` is essential for COM Initialisation.



MATLAB COM Builder Components
- RSSI_second_ESD_And_Newton_Exp
- **RSSI_second_ESD_And_Newton_Exp**
  - IConnectionPointContainer
  - IDispatch
  - IRSSI_second_ESD_And_Newton_Exp
  - ISupportErrorInfo
  - IUnknown

Figure 3.5: COM component in OLE/COM Object Viewer

3.5.3 Hybrid programming

When the works described in Subsection 3.5.1, 3.5.2 and 3.2.2 are finished, the COM component can be used to realise hybrid programming in MFC. The core idea is to deliver the values saved in Section 3.4 to the Matlab function and then receive the outcome returned from the algorithm. The Fig. 3.7 is the flow chart of location computation.

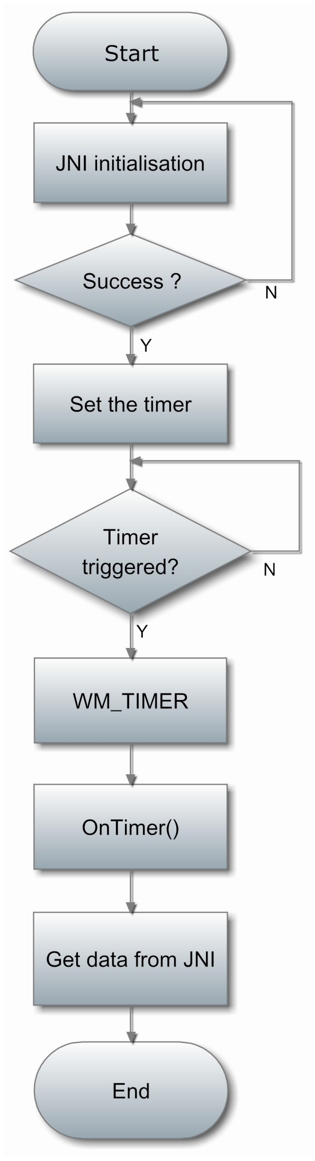


Figure 3.6: JNI flow chart

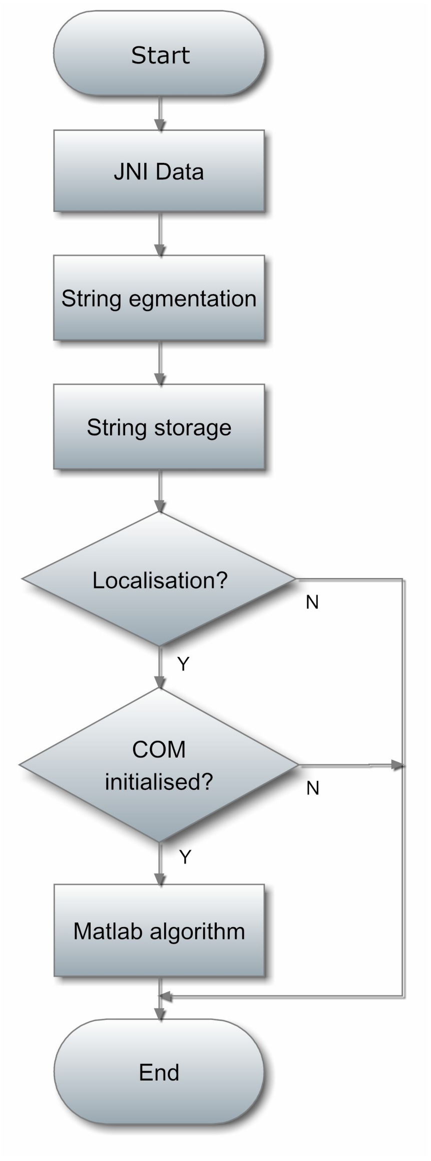


Figure 3.7: Matlab flow chart

3.6 User interface

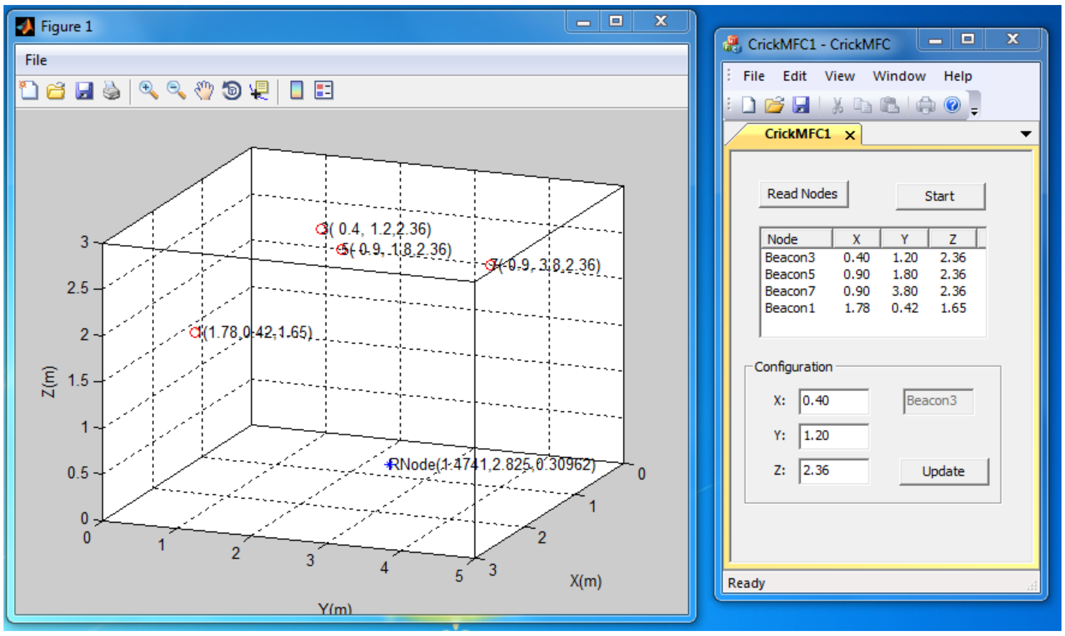
The user interface of CricketMFC enhances the operation and control of the program.

3.6.1 System setting

The beacon details are listed in the MFC List Control when the `Read Nodes` button is clicked. The format is `Beacon Number, x value, y value, z value` as illustrated in Fig. 3.8.

Meanwhile, each beacon location can be changed. When the specific beacon node number is double clicked, the existing three coordinate values are displaying in the corresponding configuration editor controls. When the new beacon location is set, the click of `Update` button can accomplish the position updating.

At last, the `Start` button initialises and launches the program.



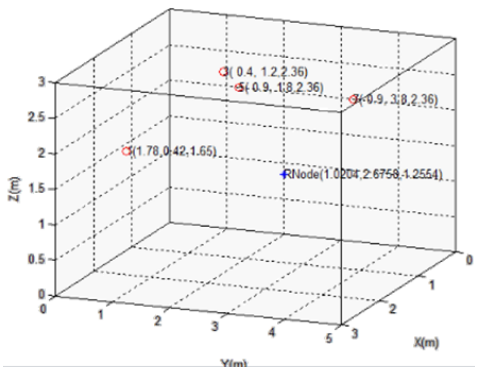
Node	X	Y	Z
Beacon3	0.40	1.20	2.36
Beacon5	0.90	1.80	2.36
Beacon7	0.90	3.80	2.36
Beacon1	1.78	0.42	1.65

X: 0.40
Y: 1.20
Z: 2.36
Beacon3

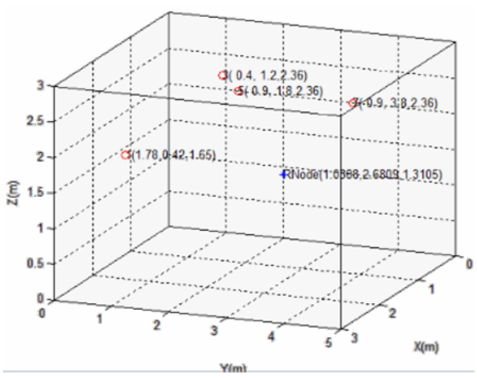
Figure 3.8: CricketMFC is running

3.6.2 Three-dimensional dispaly

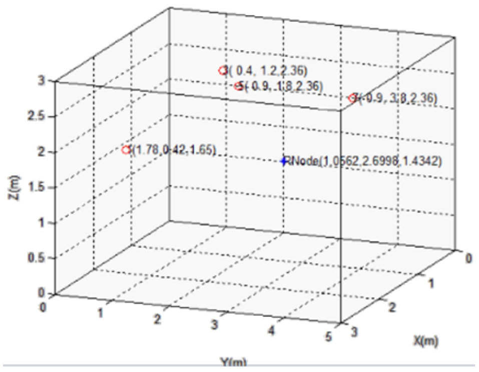
Because the three-dimensional space is hard to be imaged directly, it is necessary to display the three-dimensional localisation result in a stereograph as the user interface. The CricketMFC can describe the time varying listener location in real time as shown in Fig. 3.8. In the picture, the red circles are beacons with coordinate values, and the blue star is listener. Therefore, the blue star position stands for the estimated location. While the listener is moving, every location update is being plotted in the figure. Fig. 3.9 shows some frames of the continuous movement along Z axis.



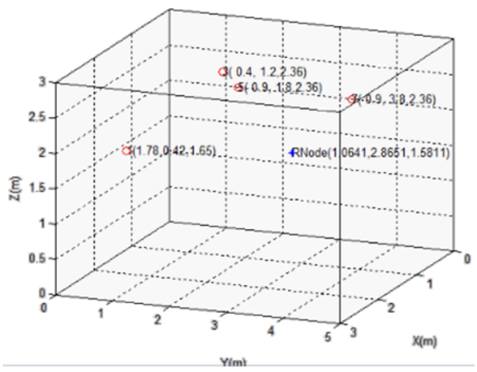
(a) (1.02,2.67,1.26)



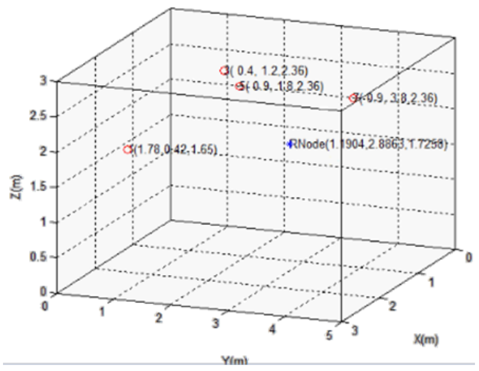
(b) (1.03,2.68,1.31)



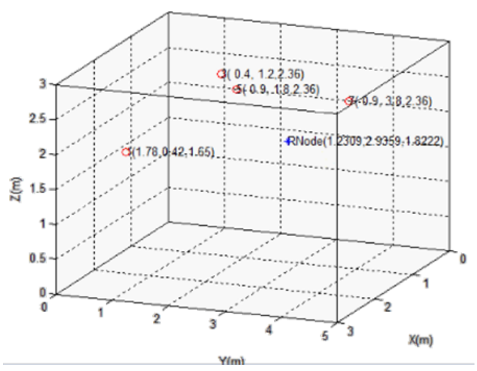
(c) (1.07,2.70,1.43)



(d) (1.06,2.87,1.58)



(e) (1.19,2.88,1.73)



(f) (1.23,2.94,1.82)

Figure 3.9: Localisation of moving along Z axis

Chapter 4

Conclusion and Future Work

In this report, the initialisation of Cricket and the additional resources are firstly described. A three-dimensional localisation algorithm is then designed, and its performance is analysed in simulation to demonstrate its high localisation accuracy. Moreover, a new program CricketMFC is developed to initialise the localisation system by using Java, Matlab and MFC. It can be used to compute the three-dimensional position and display the results in three-dimensional coordinates in real time. The real experiment clearly verifies that the designed algorithm provides good localisation results and the CricketMFC is an efficient localisation system.

Although the proposed algorithm has been evaluated by both simulation and in practice, the performance in the real environment has not been deeply investigated. In the future, this will be researched. Moreover, several localisation algorithms will be utilised to localise the Cricket simultaneously and the real-time results will be analysed more comprehensively. The theories and problems of acoustic communication will also be investigated.

Bibliography

[1] N. Bulusu, J. Heidemann, and D. Estrin, "Gps-less low-cost outdoor localization for very small devices," *Personal Communications, IEEE*, vol. 7, no. 5, pp. 28–34, 2000.

[2] C. Wang and L. Xiao, "Sensor localization under limited measurement capabilities," *Network, IEEE*, vol. 21, no. 3, pp. 16–23, 2007.

[3] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz, "Localization from mere connectivity," in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing.* ACM, 2003, pp. 201–212.

[4] Y. Shang, W. Rumi, Y. Zhang, and M. Fromherz, "Localization from connectivity in sensor networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 15, no. 11, pp. 961–974, 2004.

[5] K. Whitehouse and D. Culler, "A robustness analysis of multi-hop ranging-based localization approximations," in *Proceedings of the 5th international conference on Information processing in sensor networks.* ACM, 2006, pp. 317–325.

[6] [Online]. Available: http://en.wikipedia.org/wiki/Java_Native_Interface