UNIVERSITY OF AGDER

GPS-based Collision Avoidance for Moving Vehicles: Algorithm, Implementation and Testing

Zhaohan Jia

Supervisor
Frank Y. Li
Lei Jiao

This Master's Thesis is carried out as a part of the education at the University of Agder and is therefore approved as a part of this education. However, this does not imply that the University answers for the methods that are used or the conclusions that are drawn.

University of Agder, 2012
Faculty of Engineering and Science
Department of Information and Communication Technology

Abstract

The high incidence of traffic accidents propels our concerns on collision avoidance for mobile vehicles, particularly for marine vehicles. In this project, on the one hand, we investigate an up-to-date algorithm for predicting the location of mobile vehicles. We evaluate the performance of accuracy of the studied algorithms while taking consideration to apply it on collision avoidance. On the other hand, we design and implement a prototype based on Android platform. The prototype provides Vehicle-to-Vehicle (V2V) communications to share the GPS information, and provides the measurement of the distance between vehicles in order to avoid collisions. In addition, we survey on the real life maritime devices, and design a structure to extend our prototype into the marine devices.

Preface

This report is the Master Thesis in result of the project IKT-590 of my Master program at the Information and Communication Technology Department, University of Agder, Grimstad. The work on this project started from 5 Jan. 2012 and ended on 31 May. 2012, with 30 ECTS. The thesis is associated with NFR ECO-Boat MOL (Middle of Life) project. I have completed the main goals of my project "GPS-based Collision Avoidance for Moving Vehicles: Algorithm, Implementation and Testing", holding the motivation to improve safety of people and objects on mobile vehicles. MATLAB is utilized for the simulation, and Java is used to develop the prototype in this project. The report is accomplished by using LaTeX.

I would like to show my deepest gratitude first to my supervisor, Professor Frank Y. Li, a respectable and resourceful scholar. He offered this interesting project to me, and has provided me with valuable instructions in every stage of the writing of this thesis.

I would like to extend my gratitude to PhD fellow Mr. Lei Jiao. I feel very glad that he supervised my project. His impressively patient and responsible guidance has helped me achieve the targets favorably.

I gratefully acknowledge the help of my dear friends. With many friends' help, I could solve several problems while doing this project and accomplish this thesis.

Finally, my thanks would go to my parents. Their everlasting support and encouragement are the foremost source of my endeavor.

Zhaohan Jia
Grimstad
May 31, 2012

Abbreviations

AGPS	Assisted GPS
AP	Access Point
API	Application Programming Interface
CAN	Controller Area Network
CPS	Cellular Positioning System
C/S	Client/Server
DSC	Digital Selective Calling
GPS	Global Positioning System
MAC	Media Access Control
MMSI	Maritime Mobile Service Identity
MS	Mobile Station
P2P	Peer-to-Peer
PC	Personal Computer
RSS	Received Signal Strength
SDK	Software Development Kit
V2V	Vehicle-to-Vehicle
VHF	Global Positioning System
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Network
WPS	Wi-Fi Positioning System

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we research background and motivation of the project. The tasks of this project are also stated in this chapter. Furthermore, the last part of this chapter gives the outline of this report.

1.1 Background and Motivation

Nowadays, moving vehicles are the most elemental transport tools. No matter which kind of land, maritime and aerial transportation is, it always helps develop the society and humans' social life. Due to the increasing global communication, economic interaction, and fast pace of modern social life, the demand of transportation by vehicles grows fast.

However, the rapid development of the transportation brings some thorny problems, with the traffic crashes being regarded the severest one. Over million deaths are caused by traffic accidents every year. Collisions are responsible for about 50% of the reported crashes. For instance, Figure 1.1 [14] illustrates marine accidents statistics of Hong Kong in 2011. Collision and contract, which can be regarded as smaller collision, cover the largest parts of marine accidents in within and outside Hong Kong waters. Norway is a coastal country, and has the similar problem as well. Therefore, collision detection and avoidance for mobile vehicles

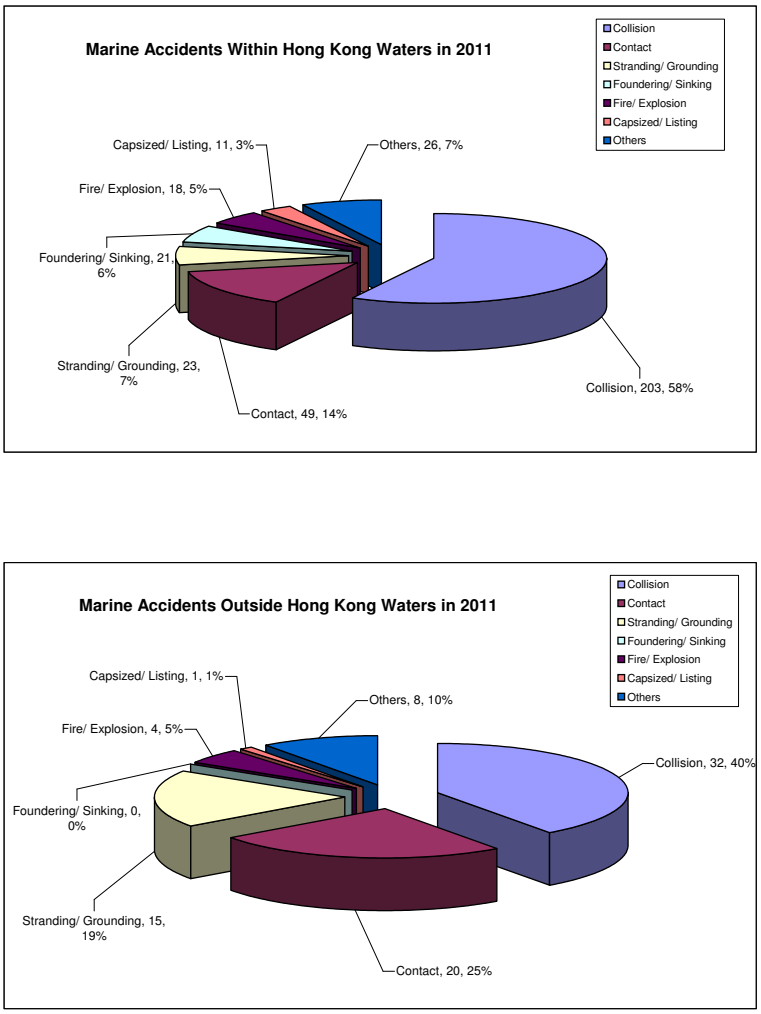


Figure 1.1: HK Marine Accidents in 2011 [14]

become the main concerns for transportation safety. Especially for maritime vehicles, collision detection and avoidance are very necessary, but the existing work is not enough in either Norway or other countries.

Driven by the situation mentioned above, we expect to develop a robust, real-time and accurate collision detecting and warning scheme for mobile maritime vehicles by using the popular GPS. The scheme may effectively predict a future collision risk for distributed mobile maritime vehicles by sharing information with each other.

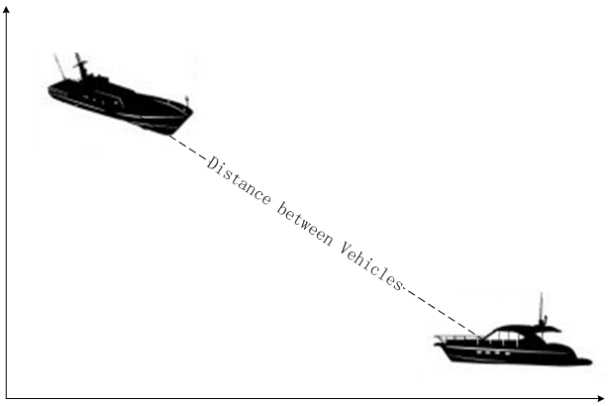


Figure 1.2: Distance between Two Vessels

1.2 Problem Statement and Goal

The purpose for the master thesis work is to detect and predict the risk of potential collision for mobile vehicles, in order to inform or warn the drivers or steersmen to take early action to avoid the potential collisions. The main problems which are expected to be solved come as follows.

How to get the position information of vehicles? We have to know the positions of the vehicles first, and then we could estimate if there is a risk of collision or not by using the position information.

How to identify a risk of collision? Apparently, the distance between vehicles is a direct and important parameter for collision. The distance can be obtained from the locations of vehicles, as shown in Figure 1.2. So one vehicle has to inform the others about its location in order to let the others calculate the distance between them. Since we focus on the maritime vehicles, the altitude can be regarded the same for all vehicles. Therefore, we only consider a two-dimension (2-D) space.

How to predict a potential collision in a future time? To estimate if there will be a potential collision in a future time considering the mobility of vehicles, the distance between mobile vehicles in the future is required, as illustrated in Figure 1.3. If there is a potential collision, the drivers or steersmen will be informed and warned to take action to avoid the possible collision. Therefore, we need to

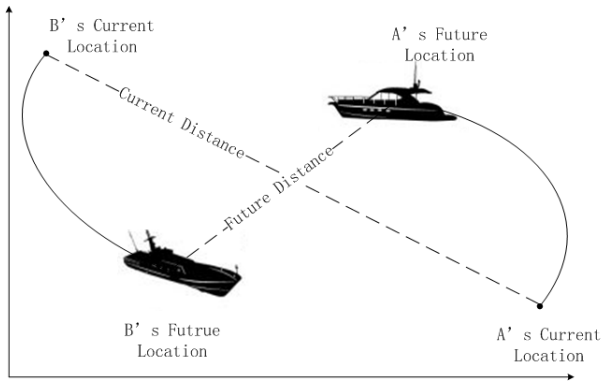


Figure 1.3: Current and Future Distance of Mobile Vessels

predict the routes and future locations of vehicles from the previous status of the vehicles.

Holding the purpose above, the tasks we need to accomplish in this project can be listed as follows:

- Survey on the potential positioning techniques used for mobile vehicles, particularly the GPS;
- Survey on the related communication patterns used for maritime vehicles;
- Study on the devices used for a prototype and for a real maritime vehicle;
- Study on location prediction algorithms for mobile nodes, and evaluate the performance of accuracy by proper simulations;
- Develop a prototype that can share locations and calculate the distance between mobile nodes on specific devices;
- Test the prototype for validation and performance evaluation;
- Discuss both of the theoretical and practical parts of the project.

1.3 Report Outline

Chapter 1 introduces the background and motivation of our project. Furthermore, the problems and tasks of the project are given. And the outline of this report is listed.

Chapter 2 has a basic survey of the technologies which are related to our project. The fields of the technologies include positioning, communications, and software development platform.

Chapter 3 states mainly the theoretical part of the project, including a study of proper mobility model for mobile vehicles in real life and an ideal location predication algorithm based on the mobility model. We evaluate the performance of the algorithm in the same chapter.

Chapter 4 describes a prototype we developed based on Android platform. The system requirements, and the ways we design and implement the prototype to fulfill the requirements are introduced in detail in this chapter.

Chapter 5 proofs the prototype we developed can work successfully. To examine the performance, various testing scenarios are designed in this chapter. And the testing results of different scenarios are discussed as well.

Chapter 6 introduces some communication devices equipped on the real maritime vehicles. An architecture that allows us to extend our prototype on these devices is illustrated. We also describe a design for our prototype on these devices in this chapter.

Chapter 7 concludes the thesis and states our main contributions in the project. Some other efforts we tried in this project are discussed. In addition, we discuss some future work in this field.

Chapter 2

Background

We introduce some background technologies related to our study in this chapter. Positioning technologies help us acquire the location of the mobile vehicles. Communications technologies provide the process to share the location information. In addition, the technologies for developing a prototype are also introduced.

2.1 Positioning

To detect and avoid a collision, the geographical locations of the mobile vehicles are imperatively. Positioning techniques are used to get the location of objects. There are several types of positioning techniques. Three of them, which are GPS, Cellular Positioning System (CPS) and Wi-Fi Positioning System (WPS), are introduced in this section.

2.1.1 Global Positioning System

GPS [1] [10] is a space-based satellite navigation system that provides instantaneous three-dimensional coordinates of location at any point on or near the Earth. At present, GPS becomes a typical navigation system, which is widely deployed for military, civil and commercial use around the world.

The position of a GPS receiver is calculated by the signals sent from at least three GPS satellites [19]. The message sent from a GPS satellite includes the precise transmitting time and the position of the satellite at the time. By using the message, a GPS receiver can calculate the distance to each satellite, so that the position of the GPS receiver can be calculated by the aid of trilateration. Usually, the speed and direction GPS receiver also can be calculated from position changes.

Due to the delay of satellite signals, the positioning time of GPS is longer than the other positioning systems, but it is still tolerable. GPS can provide a more precise positioning than other systems, especially in the marine environment. The precision of positioning is important in the application of collision avoidance, so we choose GPS as the positioning system for the mobile marine vehicles.

2.1.2 Cellular Positioning System

CPS is the technology for positioning a Mobile Station (MS) comprising a plurality of earth based broadcasting stations arranged geographically in a cellular pattern. By measuring power levels and antenna patterns, an MS can be located, since an MS always communicates with one of the closest base stations, and an MS can roughly estimate the distance to the base station. GSM localization is the use of multilateration to determine the location of GSM mobile phones, or dedicated trackers, usually with the intent to locate the user [17].

The precision of CPS depends on the density of the base stations in a way. In urban areas where the density of base stations is high, the CPS may achieve a relatively high precision. Nevertheless, CPS can not provide a satisfactory precision in rural and desolate areas, not to speak in the marine areas.

2.1.3 Wi-Fi Positioning System

WPS is a localization technique used for positioning with wireless Access Point (AP) based on measuring the intensity of the Received Signal Strength (RSS) and the method of “fingerprinting” [5][8]. The WPS can be used indoor that GPS is

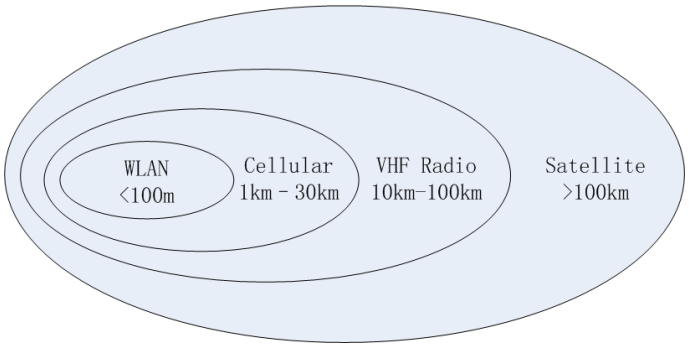


Figure 2.1: The Coverage of Different Communication Patterns

not adequate. The accuracy of WPS depends on the number of positions that have been entered into the database. The possible signal fluctuations that may occur can increase errors and inaccuracies in the path of the user.

2.2 Communications between Vehicles

Once a mobile vehicle has its location, it has to share its location with the other vehicles so that they can obtain each other's location information to avoid a collision. Several communication patterns may be utilized for mobile marine vehicles, and the normal coverage of them can be seen in Figure 2.1. We will discuss V2V communications via the following systems in this section.

2.2.1 Cellular and Satellite System

It is straightforward to use the service provided by mobile network operators. But the cellular system does not have coverage on the sea normally since it is difficult to construct a base station in the marine area. So the cellular network may be just used near the coast. The satellite communication can support the global mobile data communication and extend the coverage on the marine region. However, the high latency and cost of the satellite communication may not adapt to the real time application very well.

2.2.2 Marine VHF Radio

Marine Very High Frequency (VHF) radio is the most used equipment for marine communications. Almost every seagoing vehicle has the devices of marine VHF radio for communications. The VHF radio normally has the capability of voice communication and Digital Selective Calling (DSC). Thus, the VHF radio can be used for communication with marinas and other vehicles, and also for summoning rescue services.

2.2.3 Wi-Fi

Wi-Fi is a technology using MAC protocol of IEEE 802.11 to provide secure, reliable, fast wireless connectivity. The Wi-Fi Alliance certificates the Wireless Local Area Network (WLAN) products that are based on the IEEE 802.11 standards as Wi-Fi devices. The Wi-Fi devices can connect to each other, to wired networks which use Ethernet technology, or to the Internet. Legacy Wi-Fi usually provides a wireless connection through an AP to Wi-Fi devices. Thus, within a local coverage area, the Wi-Fi devices can connect to the network with some mobility. Beside the infrastructure mode connection with AP, Wi-Fi supports Ad-hoc mode connection without an AP, but Ad-hoc mode is not widely deployed in the market. Wi-Fi devices are popular in nowadays due to its high speed connection and free of charge characteristics.

2.2.4 Wi-Fi Direct

Based on the legacy Wi-Fi, a new standard named Wi-Fi Direct allows the Wi-Fi devices to connect to each other directly without an AP. By definition, a Wi-Fi CERTIFIED Wi-Fi Direct device is capable of a Peer-to-Peer (P2P) connection, and can support either an infrastructure or a P2P connection [18]. Wi-Fi Direct devices can connect by forming Groups in either one-to-one or one-to-many topology. One host device is in charge of the Group and can be treated as a virtual AP. The virtual AP has to support the Wi-Fi Direct and to control the discovery and

permission of a client. But the other devices in the Group need not to be Wi-Fi Direct certified, meaning they may just support the legacy Wi-Fi. Thus, without losing any advantages of the legacy Wi-Fi, Wi-Fi Direct provides the connection anytime and anywhere since an AP is not required. The mobility and portability have been significantly enhanced.

2.3 Android Development

Android is a software stack consisting of an operating system, middleware and key applications for mobile devices, such as smartphones and tablet computers. As an open source operating systems, Android is developed by Open Handset Alliance (OHA) led by Google, and is widely used in different mobile devices of many famous manufacturers. There are several versions of Android platform. Among this versions, Android 2.3 is the most popular one at present, and Android 4.0 is the latest one. The Android Software Development Kit (SDK) provides the tools and Application Programming Interface (API) to develop mobile applications for the Android platform using the Java programming language. The tools are classified into two groups listed below [4]:

- SDK tools: SDK tools are platform independent and are required no matter which Android platform you are developing on.
- Platform tools: Platform tools are customized to support the features of the latest Android platform.

Therefore, some API can be used for any version of Android platform, such as Location service. And some API dependents on a specific version of Android, for example, Wi-Fi Direct service is only supported by the Android 4.0 now.

2.4 NMEA Standard

The standard was defined by, and is controlled by, the US-based National Marine Electronics Association (NMEA). NMEA standard is the electrical and data specification for communication between marine electronic devices. The NMEA0183 standard is popularly used in many marine devices like echo sounder, sonars, anemometer, gyrocompass, autopilot, and GPS receivers which we care about most. The NMEA0183 standard uses serial communications protocol for data transmission, so that we could read the data from the NMEA devices to a Personal Computer (PC) by using serial data bus. The new version NMEA2000 standard using CAN bus is taking the place of NMEA0183 in marine applications slowly nowadays.

2.5 Chapter Summary

We can use all of the three positioning methods, but we choose GPS since its advanced accuracy. For communications of our prototype development, Wi-Fi wins due to the features of wide deployment and free of charge. In order to develop a prototype, we choose devices based on Android platform, which is easy to use and more and more popular today. For marine vehicles, a lot of commercial used marine devices are developed according to NMEA standard. Since we focus on collision avoidance for marine vehicles, knowledge of NMEA is necessary.

Chapter 3

Location Prediction Algorithm

We introduce a mobility model suitable for moving vehicles in this chapter. And we study on a location predication algorithm based on the mobility model. Afterwards, we evaluate the performance of the algorithm in our simulation scenes.

3.1 Gauss-Markov Mobility Model

In general, a mobile vehicle always travels with a destination. In addition, the mobility of a mobile vehicle within a short time is limited because of the physical restrictions, including the physical laws of acceleration, velocity and rate of change of direction.

Thus, the current velocity of a mobile vehicle partly depends on its previous velocity. Therefore, a mobile vehicle's future location/velocity is correlated with its past and current location/velocity in a way. Because the velocity of the mobile node has time correlation, this mobility characteristic is called temporal dependency of velocity [6].

Gauss-Markov mobility model, a popular mobility model with temporal dependency, is proposed in [12]. The velocity of mobile node is assumed to be correlated over time and modeled as a Gauss-Markov stochastic process in Gauss-

Markov mobility model. The Gauss-Markov mobility model is widely used in [7] [11]. We first introduce the one-dimension (1-D) Gauss-Markov mobility model in order to give a clear statement of its main idea. After that, we extend the 1-D Gauss-Markov mobility model to 2-D case which is suitable to the mobile marine vehicles.

3.1.1 1-D Case

For the sake of simplicity, we explain the Gauss-Markov mobility model in a 1-D system at first. The 1-D discrete Gauss-Markov process can be represented by the following recursive realization [12]:

$$v_t = \alpha v_{t-1} + (1-\alpha)\mu + \sigma\sqrt{1-\alpha^2}w_{t-1}, \tag{3.1}$$

where v_t is the mobile node's velocity at time slot t, α is the memory level whose value is between 0 and 1, $\{w_t\}$ is an uncorrelated Gaussian process with zero mean and unit variance and is independent of $\{v_t\}$, and μ and σ^2 are the asymptotic mean and asymptotic variance of $\{v_t\}$. Since there is no guarantee that v_0 has mean μ and variance σ^2, the process described by (3.1) is generally not stationary [12]. However, when t approaches infinity, μ is the asymptotic mean of v_t, and σ^2 is the asymptotic variance of v_t.

From the Eq. (3.1), we could know the velocity of a mobile node at time slot t is related to its previous velocity and a Gaussian random variable. The memory level parameter α reflects the degree of temporal dependency. By adjusting the memory level α, the Gauss-Markov mobility model represents different kinds of mobility patterns, including its two extreme cases, the random-walk and the constant velocity fluid-flow models:

1. When the Gauss-Markov mobility model is memoryless $(\alpha = 0)$, the Eq. (3.1) is

$$v_t = \mu + \sigma w_{t-1}, \tag{3.2}$$

which represents a drifting random-walk mobility pattern with mean μ and

standard deviation σ. In this case, the mobile node's velocity at time slot t is dependent on the fixed drift velocity and the Gaussian random variable.

2. When the Gauss-Markov mobility model is full memorial $(\alpha = 1)$, the Eq. (3.1) is

$$v_t = v_{t-1}, \tag{3.3}$$

which represents a constant velocity fluid-flow mobility pattern with $v_t = v_0$ for all t. In this case, the mobile node's velocity at time slot t is exactly the same as its previous velocity.

3. When the Gauss-Markov mobility model has some memory $(0 < \alpha < 1)$, the mobile node's velocity at time slot t is determined by its previous velocity and the Gaussian random variable. The memory level of α determines the degree of dependency or randomness. If α is large, the mobile node's previous velocity has more influence on the current velocity; if α is small, the Gaussian random variable affects more on the current velocity.

Therefore, the Gauss-Markov mobility model represents a wide range of mobility patterns with various degree of memory, which also include the random-walk and fluid-flow models as Gauss-Markov mobility model's two extreme cases.

3.1.2 2-D Case

By using vectors of Gauss-Markov processes, we can extend the Gauss-Markov mobility model to a multi-dimensional case easily. For the reason of fitting the mobility of mobile marine vehicles, we illustrate the Gauss-Markov mobility model in a 2-D system in this section.

In the 2-D case, the location of a mobile node is represented by the random vectors $\mathbf{s}_t = [s_t^x, s_t^y]^T$, where the superscripts denote the dimensions. The 2-D Gauss-Markov stochastic process can be represented as the following equation:

$$\mathbf{v}_t = \overline{\alpha} \odot \mathbf{v}_{t-1} + (1 - \overline{\alpha}) \odot \overline{\mu} + \overline{\sigma} \odot \sqrt{1 - \overline{\alpha}^2} \odot \mathbf{w}_{t-1}, \tag{3.4}$$

where $\mathbf{v}_t = [v_t^x, v_t^y]^T$ is the velocity vector at time slot t; $\mathbf{w}_{t-1} = [w_{t-1}^x, w_{t-1}^y]^T$ is the 2-D uncorrelated Gaussian variable vector with zero mean and unit variance, and is independent of $\mathbf{v}_t$; $\overline{\alpha} = [\alpha^x, \alpha^y]^T$ is the memory level vector; $\overline{\mu} = [\mu^x, \mu^y]^T$ and $\overline{\sigma} = [\sigma^x, \sigma^y]^T$ are the vectors of asymptotic mean and asymptotic standard deviation of the velocity; and $\odot$ denotes element-by-element multiplication. In (3.4) that is the velocity processes in both dimensions are uncorrelated. We can also rewrite the general form of (3.4) in a 2-D field as follows:

$$\begin{cases} v_t^x = \alpha^x v_{t-1}^x + (1-\alpha^x)\mu^x + \sigma^x\sqrt{1-\alpha^{x2}}w_{t-1}^x, \\ v_t^y = \alpha^y v_{t-1}^y + (1-\alpha^y)\mu^y + \sigma^y\sqrt{1-\alpha^{y2}}w_{t-1}^y. \end{cases} \tag{3.5}$$

3.2 Prediction Algorithm

The Gauss-Markov mobility model can capture the correlation of a mobile node's velocity in time, so it is widely utilized in the tracking system to predict the future location of mobile objects. When a mobile vehicle moves in Gauss-Markov mobility model, by estimating the Gauss-Markov parameters, μ, σ and α, the future location of the mobile vehicle is expected to be predicted accurately.

A Gauss-Markov parameter estimator called GMPE_MLH is proposed in [13]. The GMPE_MLH model uses a maximum likelihood technique to estimate the Gauss-Markov parameters. The GMPE_MLH model takes consideration of estimating parameters with few requirements. For the purpose of predicting a mobile marine vehicle's future location, we consider to use this new estimation model.

For a mobile marine vehicle in 2-D case, the GMPE_MLH model can be used in both the two dimensions. Because the method of calculation for the two dimensions is the same, we use one dimension to illustrate the GMPE_MLH model. Since we have known the parameters of Gauss-Markov mobility model, a summary of other notations used in the GMPE_MLH model is shown in Table 3.1 [13].

After a mobile node measuring the velocity of v_t, it uses the parameter estimator to evaluate $\hat{\mu}_t$, $\hat{\sigma}_t$, $\hat{\alpha}_t$ and $\bar{\tilde{\alpha}}_t$, based on the data $\hat{\mu}_{t-1}$, $\hat{\sigma}_{t-1}$, $\hat{\alpha}_{t-1}$ and $\bar{\tilde{\alpha}}_{t-1}$

Table 3.1: Summary of Notations [13]

α	A Gauss-Markov parameter used to vary the randomness of the Gauss-Markov equation
μ	A Gauss-Markov parameter used to denote the mean velocity as $t \to \infty$
σ	A Gauss-Markov parameter used to denote the stand deviation of velocity as $t \to \infty$
$\hat{\alpha}_t$	The value of α estimated at time slot t
$\hat{\mu}_t$	The value of μ estimated at time slot t
$\hat{\sigma}_t$	The value of σ estimated at time slot t
$\tilde{\alpha}_t$	The most likely value of α at time slot t
$\bar{\tilde{\alpha}}_t$	The mean of $\tilde{\alpha}_1$, $\tilde{\alpha}_2$,..., and $\tilde{\alpha}_t$

recorded from the previous time slot. The data v_t, $\hat{\mu}_t$, $\hat{\sigma}_t$, $\hat{\alpha}_t$ and $\bar{\tilde{\alpha}}_t$ are stored by the mobile node, and is prepared for the estimation of the next time slot. The GMPE_MLH model only requires the data of one previous time slot. Hence, it is efficient since few data need to be recorded and transmitted.

3.2.1 Estimation of μ and σ

The GMPE_MLH uses recurrence exists to calculate the $\hat{\mu}$ and $\hat{\sigma}$, which are the estimated values of parameters of μ and σ.

The $\hat{\mu}$ is calculated as follows:

$$\begin{cases} \hat{\mu}_1 = v_1 & (\text{if } t = 1), \\ \hat{\mu}_t = \dfrac{t-1}{t}\hat{\mu}_{t-1} + \dfrac{1}{t}v_t & (\text{if } t \geq 2). \end{cases} \tag{3.6}$$

And the $\hat{\sigma}$ is calculated as follows:

$$\begin{cases} \hat{\sigma}_1^2 = 0 & (\text{if } t = 1), \\ \hat{\sigma}_2^2 = \dfrac{(v_1 - \hat{\mu}_2)^2 + (v_2 - \hat{\mu}_2)^2}{2} & (\text{if } t = 2), \\ \hat{\sigma}_t^2 = \dfrac{t-1}{t}\hat{\sigma}_{t-1}^2 + \dfrac{1}{t-1}(v_t - \hat{\mu}_t)^2 & (\text{if } t \geq 3). \end{cases} \tag{3.7}$$

In the GMPE_MLH model, a mobile node records the parameters of the previous time slot $t-1$, including v_{t-1}, $\hat{\mu}_{t-1}$ and $\hat{\sigma}_{t-1}$. Once v_t is measured, $\hat{\mu}_t$ can be first calculated according to (3.6), and $\hat{\sigma}_t$ can be calculated according to (3.7).

3.2.2 Calculation of $\tilde{\alpha}_t$

The most likely value $\tilde{\alpha}_t$ means the α having the maximum probability that the velocity of the mobile object is changed from v_{t-1} to v_t at time slot from $t-1$ to t, given v_{t-1}, v_t, μ, and σ [13]. In the GMPE_MLH model, once the v_{t-1}, v_t, $\hat{\mu}_t$, $\hat{\sigma}_t$, and $\bar{\tilde{\alpha}}_{t-1}$ are obtained, $\tilde{\alpha}_t$ is achieved by using a maximum likelihood technique. The likelihood is evaluated by the function (3.8) [13]:

$$L(\alpha) = \frac{1}{\sqrt{1-\alpha^2}\hat{\sigma}_t\sqrt{2\pi}} e^{-\frac{(v_t-\alpha v_{t-1}-(1-\alpha)\hat{\mu}_t)^2}{2(1-\alpha^2)\hat{\sigma}_t^2}}. \tag{3.8}$$

There are several candidates of $\tilde{\alpha}_t$. The candidates are calculated by the following method. The interval $[0,1]$ is split into a finite number of subintervals $[\alpha_0,\alpha_1]$, $[\alpha_1,\alpha_2]$,...,$[\alpha_{m-1},\alpha_m]$ with $\alpha_0 = 0 < \alpha_1 < ... < \alpha_m = 1$. The candidates of $\tilde{\alpha}_t$ are $\frac{\alpha_i+\alpha_{i+1}}{2}$, for every i $(0 < j < m)$. The candidates of $\tilde{\alpha}_t$ as α in Eq. (3.8) are evaluated. The one which maximizes the value of $L(\alpha)$ is treated as the $\tilde{\alpha}_t$.

3.2.3 Estimation of α

The estimated value of parameter α_t, i.e., $\hat{\alpha}_t$, is evaluated by the relationship between $\bar{\tilde{\alpha}}$ and α_t. From the Table 3.1 we could know $\bar{\tilde{\alpha}}_t$ is the mean value of $\tilde{\alpha}_1, \tilde{\alpha}_2, ..., \tilde{\alpha}_t$, where $\tilde{\alpha}_t$ represents the most likely value of α at time slot t.

In order to get the $\hat{\alpha}_t$, the $\bar{\tilde{\alpha}}_t$ should be calculated before. As we know by the definition,

$$\bar{\tilde{\alpha}}_t = \frac{1}{t}\sum_{i=1}^{t}\tilde{\alpha}_i. \tag{3.9}$$

Only one step memory does GMPE_MLH model require, the Eq. (3.9) can be rewritten as the following recurrent equation:

$$\bar{\tilde{\alpha}}_t = \frac{1}{t}\tilde{\alpha}_t + \frac{t-1}{t}\bar{\tilde{\alpha}}_{t-1}. \tag{3.10}$$

A regression method is used to evaluate the relation between $\bar{\tilde{\alpha}}$ and α in [13]. As a result, a function to evaluate $\hat{\alpha}_t$ in the GMPE_MLH model is obtained:

$$\hat{\alpha}_t = \begin{cases} 0 & (\text{if } \bar{\tilde{\alpha}}_t \leq 0.35338), \\ \frac{1}{53.151}\ln(\frac{0.2}{-0.43403\bar{\tilde{\alpha}}_t} - \frac{1}{-0.43403}) & (\text{if } 0.35338 < \bar{\tilde{\alpha}}_t \leq 0.44602), \\ \frac{1}{0.37492}\ln(\frac{0.2}{-0.55069\bar{\tilde{\alpha}}_t} - \frac{1}{-0.55069}) & (\text{if } 0.44602 < \bar{\tilde{\alpha}}_t \leq 1), \\ 1 & (\text{if } \bar{\tilde{\alpha}}_t \geq 1). \end{cases} \tag{3.11}$$

3.2.4 Location Prediction

Referring to Eq. (3.1), the estimated velocity at time slot $t+1$ is:

$$\hat{v}_{t+1} = \hat{\alpha}_t v_t + (1-\hat{\alpha}_t)\hat{\mu}_t + \sqrt{1-\hat{\alpha}_t^2}W_t. \tag{3.12}$$

The expected value of W_t is $E[W_t] = 0$, due to the normal distribution of W_t. For this reason, the expected value of $\hat{v}_{t+1}$ can be represented as follows:

$$E[\hat{v}_{t+1}] = \hat{\alpha}_t v_t + (1-\hat{\alpha}_t)\hat{\mu}_t. \tag{3.13}$$

Therefore, the predicted location of the mobile node at time slot $t+1$ can be calculated by the following equation:

$$\hat{s}_{t+1} = s_t + E[\hat{v}_{t+1}] = s_t + \hat{\alpha}_t v_t + (1-\hat{\alpha}_t)\hat{\mu}_t. \tag{3.14}$$

3.3 Performance Evaluation

To check the performance of the GMPE_MLH algorithm, we simulate in 1-D case at first. In order to examine the prediction performance for mobile marine vehicles, we extend the simulation to 2-D space.

3.3.1 1-D Case

In the 1-D case, in each simulation trial, a node moves 2000 time slots according to a group of Gauss-Markov parameters, which are velocity randomness α expected mean μ, and variance σ. Firstly, three trials with different groups of parameters are simulated, illustrated in Figure 3.1.

Observing the simulation result, every curve converges to an estimated value while t increasing. As the time slot increases, the algorithm has a rational estimation of the Gauss-Markov parameters of α, μ and σ.

In addition, let us pay attention on the estimated value of the velocity expected mean $\hat{\mu}$, which has direct effect on location predication. There are different convergence rates of $\hat{\mu}$ referring to different values of α and σ. If a node moves with a larger σ, the velocity has a larger variation, so the convergence of $\hat{\mu}$ becomes slower. If a node moves with a larger α, the velocity has a better temporal memory. The curves converge is quicker since the estimated velocity at time slot $t+1$ is more influenced by the known velocity at time slot t.

In order not to lose the generality, much more trials of simulation are necessary. We simulate 200 trials with random values chosen from intervals of α, μ and σ respectively. The intervals of each parameters are: $\alpha \in [0, 1]$; $\mu \in [-10, 10]$; $\sigma \in [0, 5]$.

As shown in Figure 3.2, comparison of the differences between actual values and the estimated values at the $2000th$ time slots for each trial is performed, and the red line is the mean value of the differences of the 200 trials. From the result we can see the differences are in a very small scale compared with values of parameters.

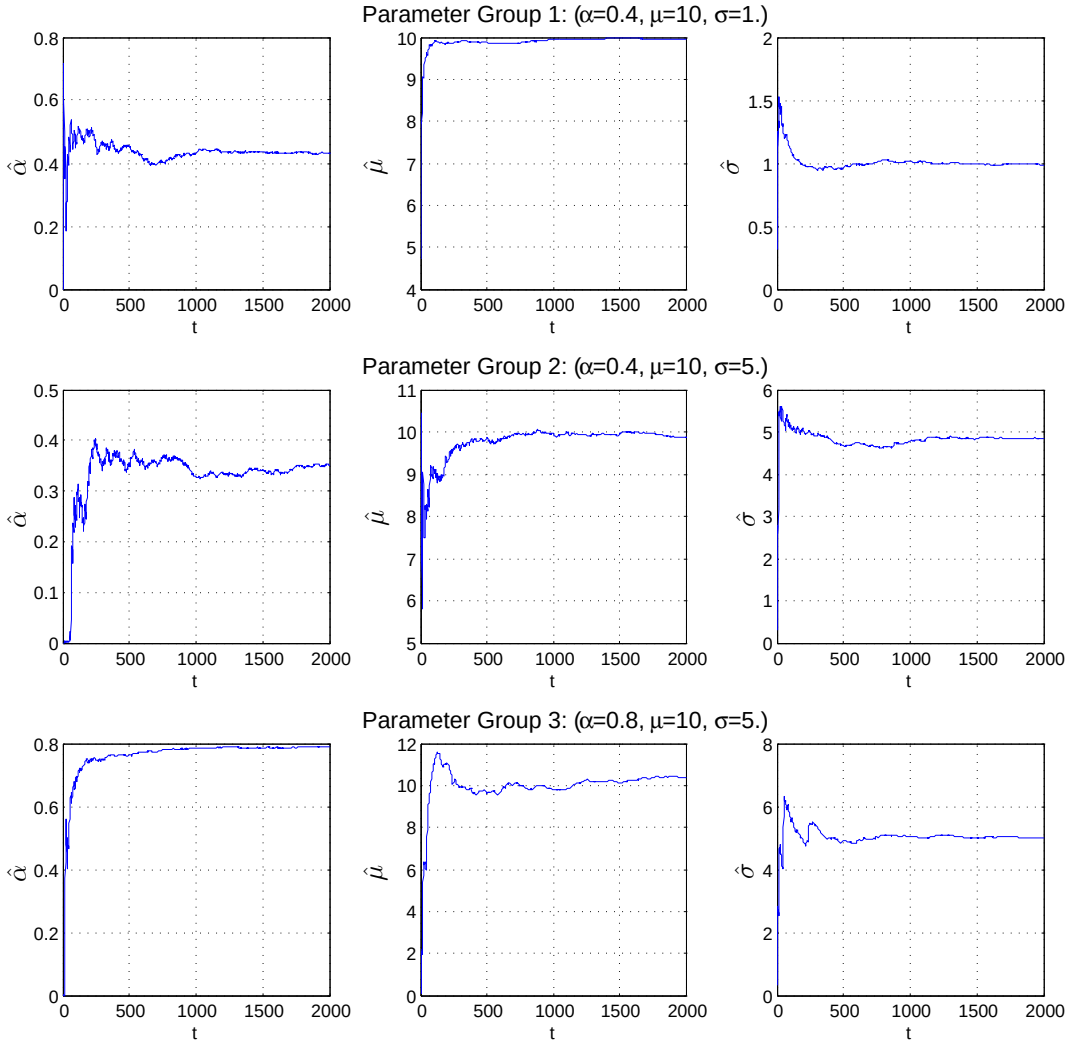


Figure 3.1: Estimated Parameters at Each Time Slot

3.3.2 2-D Case

In the 2-D case, a node moves 2000 time slots as well. It has two groups of parameters in x-dimension and y-dimension respectively. Once the location and velocity at the time slot t are measured, the location at time slot $t+1$ can be predicted by using the algorithm. To generate a track of the mobile node in the 2-D space, we have a node start moving from the coordinate (0,0). The Gauss-Markov mobility parameters of x-dimension are $\alpha = 0.5$, $\mu = -10$, and $\sigma = 5$;

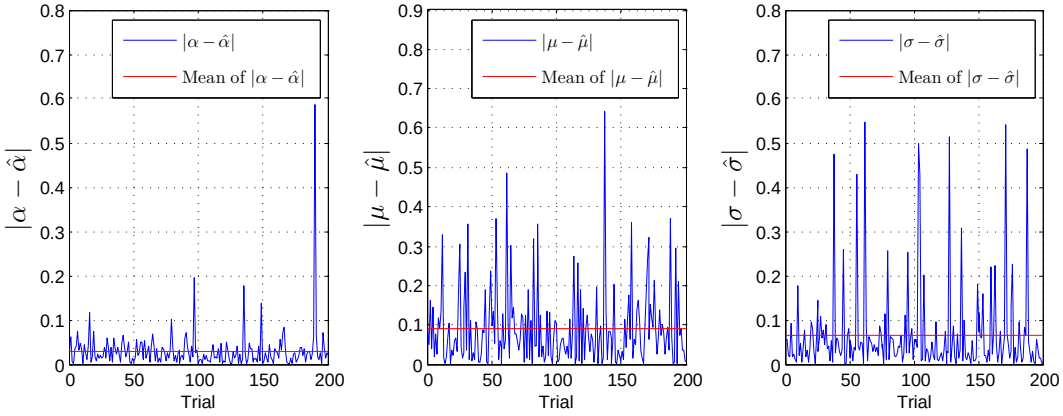
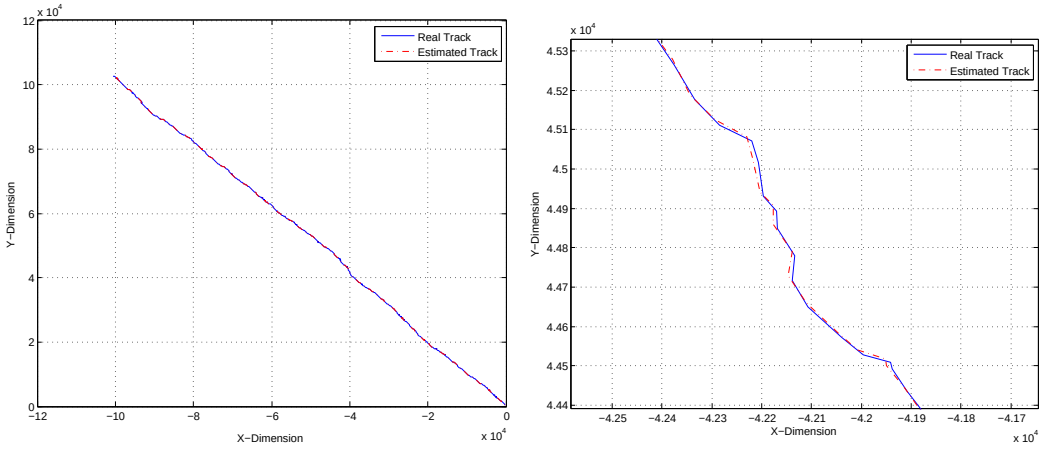


Figure 3.2: Estimation Error of Different Parameters

parameters of y-dimension are $\alpha = 0.5$, $\mu = 10$, and $\sigma = 5$. Thus, the expected velocity value of the mobile node is $10\sqrt{2}$. The track is shown in Figure 3.3(a) in blue line. And the red dash line in the same figure is the predicted track. The algorithm can properly predict the position of a mobile node in a 2-D space.



(a) (b)

Figure 3.3: Real Track and Estimated Track

Zooming in part of Figure 3.3(a), we have Figure 3.3(b). In Figure 3.3(b), the two tracks have some distances, which are the errors of the prediction. The error becomes relatively large whenever the mobile node has a obvious change of its direction. The obvious change of direction implies a large change of the velocity components in either x-dimension or y-dimension, and the GMPE_MLH model

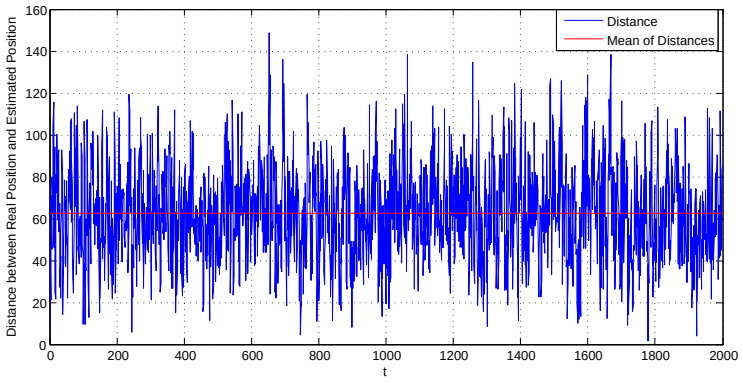


Figure 3.4: Distance between Real Position and Estimated Position at Each Time Slot

needs some time slots to correct its estimation.

To check the accuracy of prediction, we observe the distance between the real position and the predicted position. Figure 3.4 illustrates the distance at each time slot, and the red line is the mean value. With the expected velocity value of $10\sqrt{2}$, the error of position prediction is in a reasonable scale in the 2-D space.

3.4 Chapter Summary

We studied the mobility model and the prediction algorithm which are the main theoretical part of our project. We met some problems when studying the algorithm, and we communicated with the authors and got replies in time. Based on the understanding the algorithm, we accomplished simulation by our own programming in MATLAB. Furthermore, we extended the simulation to a 2-D case, which adapts to the mobile marine vehicles in real space. Then, we checked the accuracy of the algorithm in the 2-D case by examining the distance from the estimated position to the real position in the space.

Chapter 4

Design and Implementation of Prototype

In this chapter, the design and implementation details of a prototype platform we developed are introduced. The prototype is based on Android platform. In the prototype, we implement the functionalities of V2V communications, location acquisition, and distance measurement.

4.1 System Requirements

In the prototype, there are two main hardware modules are expected. First, a GPS module is for sure needed to get the GPS information of a vehicle. Second, taking the communications between vehicles into consideration, a module for communication is mandatory. Based on these demands, the smart mobile phones based on Android platform are chosen as the hardware for prototype.

In our prototype, two Android smart mobile phones are used. Since we know the distance is the basic idea to avoid collisions, the prototype captures the instantaneous locations of each device, and calculates the distance between the two devices in real time based on the locations.

The functionalities of our prototype are listed below:

- Get instantaneous location of local device,
- Send the local device's location information to the remote device,
- Listen and receive the remote device's location information,
- Calculate the distance between two devices on each device,
- Display the local and remote devices' locations, and the distance between the two devices on each device to inform the users.

4.2 Design

We would like to demonstrate some ideas of our design, including the network architectures for communications, and the user interfaces of the application.

4.2.1 Network Architectures

To implement the communications between the mobile devices, we utilize socket, which is a popular computer network communication technique based Internet Protocol. A socket API usually provided by the operating system, including Android. Socket is a Client/Server (C/S) based technique. The communication endpoint in socket is identified by an IP address and a port number.

In Android API, the class named "ServerSocket" provides a server-side socket, and another class named "Socket" represents a client-side socket. Binding a pair of "ServerSocket" and "Socket" to appropriate addresses and ports, data can be transmitted and received.

To transmit the location information, we consider active mode and passive mode. In the active mode, the time of location transmission is decided by the MS who transmits the location actively, the receiving MS just receives and does not request a location transmission. In the passive mode, an MS transmits the location

information only if it receives a location request. The active mode is time saving, so we decide to use active mode for our prototype. Nevertheless, the prototype may be transferred to passive mode from active mode via few changes.

Since socket is C/S based, we have to define client and server in the network during the communication process. We design three types of architecture for communication of the prototype, with or without the assistance of PC. The PC and the two MSs are connected to a same network with one or more APs. For simplicity, we suppose only one AP is used for the illustration.

Architecture 1 - PC Server/MS Clients

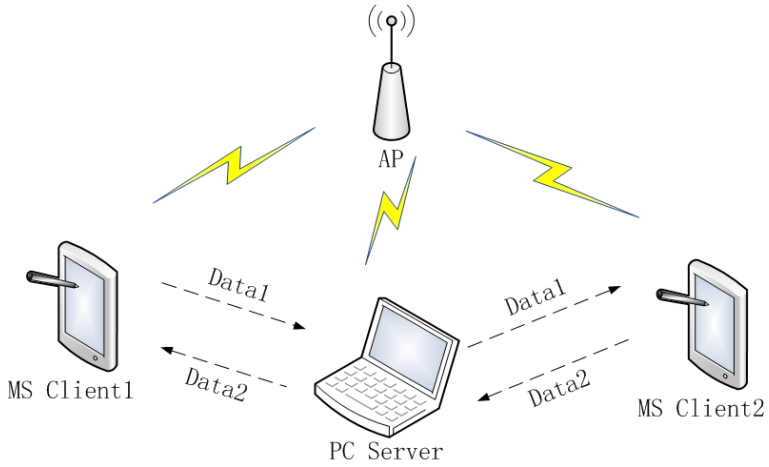


Figure 4.1: Architecture 1 - PC Server/MS Clients

At the beginning, because it is relatively easy to implement socket on a PC, we consider to use a PC to take the role of server, and the two mobile stations take the role of two clients of the PC. The structure is shown in Figure 4.1. Once an MS send data to the PC server, the server broadcasts the data to all its clients. Only can MS clients process the received data. Thus, the two MS clients can share the information with each other through the PC server. This architecture is relatively easy to implement, but the requirement of a PC's assistance makes it not convenient, especially for a mobile scenario.

Architecture 2 - MS1 Server/MS2 Client

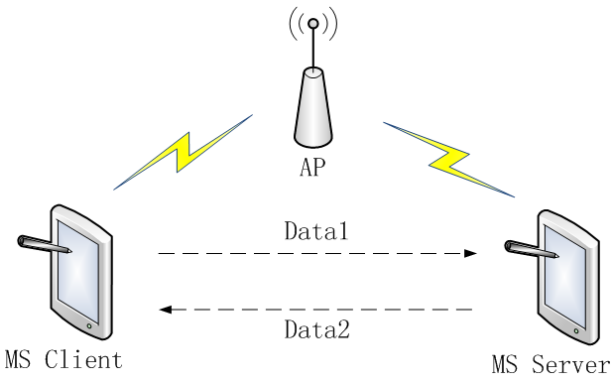


Figure 4.2: Architecture 2 - MS1 Server/MS2 Client

Taking consideration of mobility, we change the architecture by removing the PC server. In this architecture, one MS keeps the role of the client, and the other MS takes the role of the server, as Figure 4.2 shows. The client MS sends its data to the server at first and a link is constructed. The server MS sends its data using the same link. Hence, the two MSs can communicate. However, since the data should be processed by each MS, both the client and the server side should have the ability to handle with the data. What's more, two mobile devices have to install two different applications due to differences between the server and the client. All these problems may bring difficulties in the follow-up work.

Architecture 3 - MS Server/Client

For the purpose to eliminate the differences between the applications running on each MS, we would like to merge the server and the client into one application, so that the mobile devices can install the same application. As Figure 4.3 shows, an MS has a server part and a client part. The server part can just have function of listening on a port and receiving data, but not sending. The client part only owns the function of sending data, but not receiving. Once the server receives data, the processing procedure of the data on each device is the same. Therefore, one application can be installed on several devices, it is easy to extend the prototype to a structure of more than two MSs.

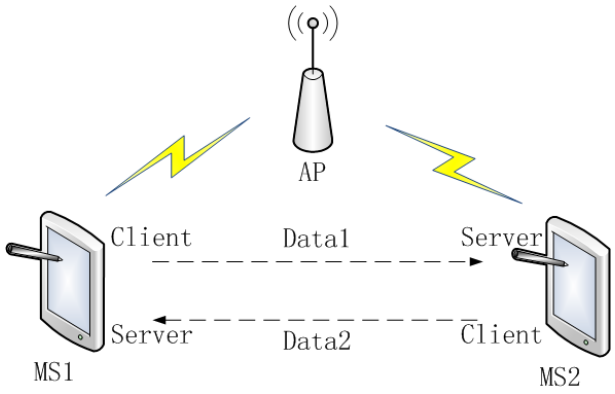


Figure 4.3: Architecture 3 - MS Server/Client

The three architectures are designed and developed step by step. We implement all three of them, and finally we use the third one for our prototype.

4.2.2 User Interface

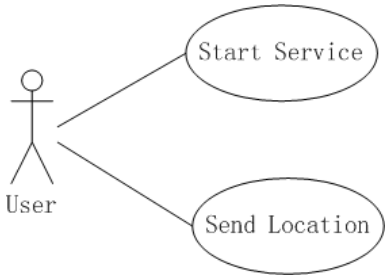


Figure 4.4: Use Case Diagram

We take into account of operating interface which can be provided to the users now. This user interface is related to layout design of the applications as well. It is better the operations for the user as few as possible, so that the application is not to complex for the user. We just provide two operations for the user, shown in the use case diagram in Figure 4.4.

The only actor in the use case is the user of the application. The user can have two kinds of operation, which are starting the service and sending location information to the others. The starting service operation allows the user to start

the main functions of the application, including starting to wait for a coming data and starting to acquire and update location. The sending location operation allows the user to send the location of the device to the other devices and inform the other users of this location. We design two buttons for the two operations on the layout of the application.

In a future design, the location transmission may be automatically triggered. The time intervals between each transmission can be different referring to different distances between the vehicles. At that time, the "Send" button can be removed, and the user only need to start the service.

4.3 Implementation Outline

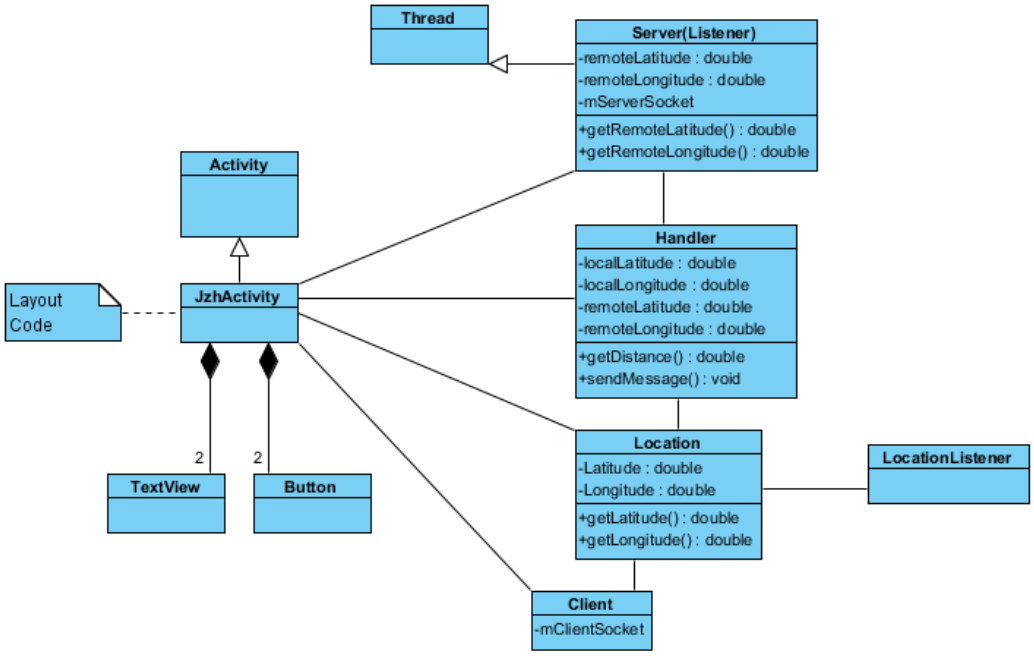


Figure 4.5: Class Diagram

The functionalities of prototype includes measurement and display of locations of the two devices and the distance between them. we use four modules to accomplish the application. The four modules are Server, Client, Location, and

Handler. The class diagram in Figure 4.5 describes the structure of the application. An activity is an application component that provides a screen with which users can interact in order to do something [2]. In our application, just one activity is needed. The four modules are all associated with the activity.

4.3.1 Server Part

First of all, let us see the Server part. The major structure of the Server part is described in the following codes.

```
class Server extends Thread
{
    private ServerSocket mServerSocket;
    private Socket reClientSocket;
    private InputStream mInputStream;
    private byte[] buf;
    private String remoteLocation;
    private Message msg;
    public void run();
}
```

The “mServerSocket” is a ServerSocket used for listening on a port, and the “reClientSocket” is the a client socket accepted by the the ServerSocket. The “mInputStream” and “buf’ are used for receiving the data from the client socket. The received data named “remoteLocation” contains the location information of the remote device. The “msg” is used for informing the Handler part to process the received data. Since the Server part is a thread always running after starting, a method “run()” is necessary.

The working procedure of the Server part is shown in Figure 4.6. Once the “Start” button is clicked by a user, the Server is called. The Server creates a ServerSocket, listens on a port and waits for the coming data. The data should contain the information of the location sent from the remote device. Once data comes, the Server sends a message to its handler to process the location information received. The Server keeps on listening on the port after submitting information to the handler.

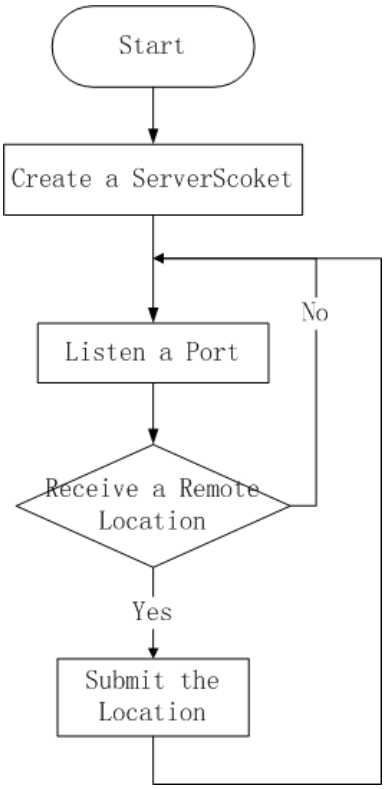


Figure 4.6: Flow Chart of Server Part

4.3.2 Handler Part

Then we come to the Handler, it handles with the message and measures the distance between two locations (latitudes and longitudes). We can also see the major structure of the Handler part as follows.

```
class Handler{
    private String remoteLocation;
    private double localLat;
    private double localLng;
    private double remoteLat;
    private double remoteLng;
    public void handleMessage(Message);
}
```

The four elements of “localLat”, “localLng”, “remoteLat”, and “remoteLng” are the location information of the local device and the remote device. The “re-

moteLocation" is the data received from the Server part. The method "handleMessage()" analyzes the data "remoteLocation" into "remoteLat" and "remoteLng", then it measures the distance between the two locations.

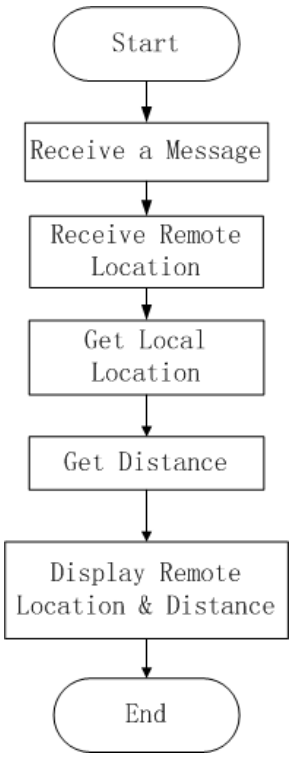


Figure 4.7: Flow Chart of Handler Part

Figure 4.7 illustrates the flow of the Handler part. The Handler is called when a server submits a remote location to it. When the Handler part receives the remote location from the Server Part. The handler requests the local location from the Location part. After the Handler part has the information of both local and remote locations, it uses the Android API to calculate the estimated distance between the two locations.

4.3.3 Location Part

The Location part is a very important part for our application. The following codes describe the major structure of the Location part (JzhLocation) in our system.

```
class JzhLocation {
    private Location mLocation;
    private LocationManager mLocationManager;
    private double mLat;
    private double mLng;
    public final LocationListener();
}
```

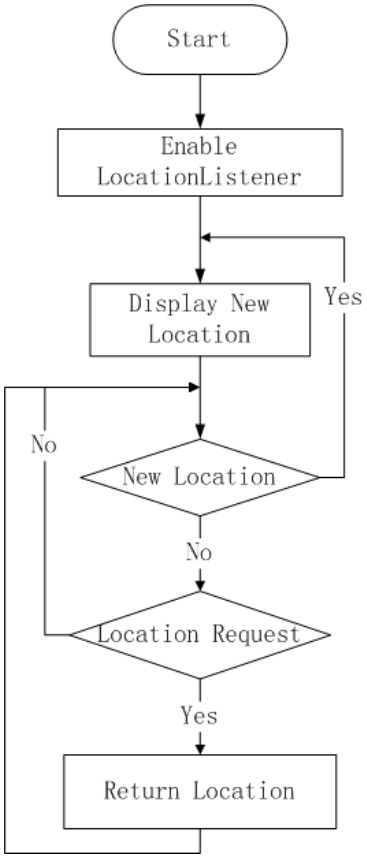


Figure 4.8: Flow Chart of Location Part

The class Location is a public class provided by Android API. We have an object of Location named “mLocation”. A “mLocation” consists of not only latitude and longitude, but also speed, bearing and so on. At present stage, we merely need latitude and longitude. The “mLocationManager” manages the data we need in “mLocation”. The “mLat” and “mLng” are the local latitude and longitude we need. The “mLocation” is associated with a “LocationListener” which can im-

ply the update of the location. Consequently, the latest location can always be displayed.

The working flow of the Location part is depicted in Figure 4.8. Once the "Start" button is clicked by a user, the "LocationListener" is enabled. The "LocationListener" keeps capturing the latest location. If another part in the system requests the location, the Location part will response and provide the location information to it.

4.3.4 Client Part

At last, we introduce the Client part. The "mLocation" contains the information of the local location. The "mClientSocket" is a client socket of the local device to connect the Server part of the remote device. The "mOutputStream" and "mBuffer" are used to transmit the "mLocation".

```
class Client{
     private String mLocation;
     private Socket mClientSocket;
     private OutputStream mOutputStream;
     private byte[] mBuffer;
}
```

The procedure of the Client part is shown in Figure 4.9. When a user clicks the "Send" button, the Client part is called. The Client part requests the location from the Location part. Then, it creates a client socket and sends the location via the client socket. After sending the client socket is closed. As we know Socket uses IP address and port number to identify the destination. In our prototype the IP address and port number were set up in advance by programming.

4.3.5 System

After we introduce the different parts, we can describe the procedure of the whole system in the sequence diagram in Figure 4.10. A button of "Start" enables both

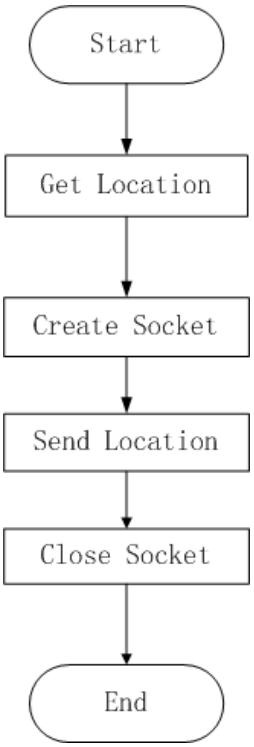


Figure 4.9: Flow Chart of Client Part

the Server and Location parts, the Server part can listen on a port, the Location part can provide the latest location. The Server and Location part last until the application is terminated. And a button of "Send" enables the Client, who can transmit the location. Both of the two devices should start the service at the beginning, which means before the Clients sending, both of the two Servers should have started. Then the two Clients can send their locations at any time whenever the user wants.

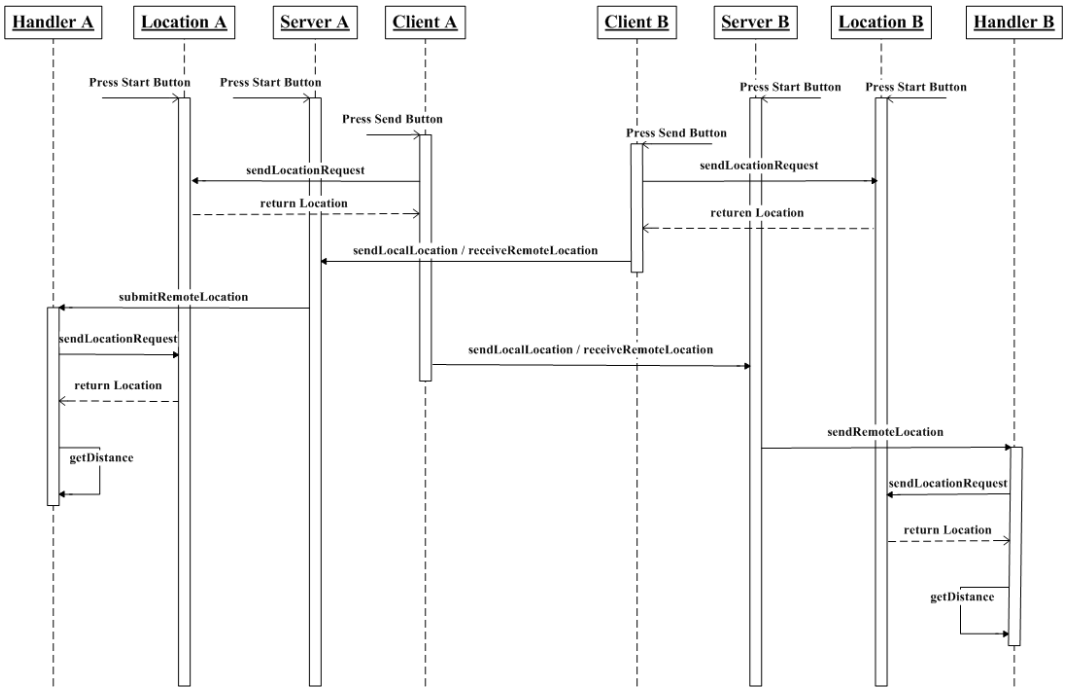


Figure 4.10: System Sequence Diagram

Chapter 5

Proof of Concept: Android-based Prototype

In this chapter, we validate the Android-based prototype we developed. Moreover, we design some scenarios to test our prototype. Adopting accuracy as a criterion, we discuss the testing results in different scenarios.

5.1 Validation

First of all, we would like to prove that our Android-based prototype can work successfully. The environment for testing is described in the following. Then we illustrate a testing result confirming the usability of the Android application.

5.1.1 Environment

In order to test the prototype, we use two Android smart phones, which are Motorola XT702 (Milestone) and ME525+ (Defy+). Both of them have Wi-Fi and GPS hardware support. We can see the devices in Figure 5.1 and the specifications of them can be seen in the appendix.

Figure 5.1: Devices for Test

Due to the Wi-Fi and GPS requirements, the place we do test should have wireless AP and be outdoor. In addition, the place should be wide enough. Therefore, the park outside the campus of University of Agder (Grimstad) is a proper place for our test. And we use a home wireless router as the AP for the two mobile devices to connect.

5.1.2 Application

Before testing, we have to ensure the application we developed can work properly. The two devices connect to an AP, and have been properly set up initially. As depicted in Figure 5.1, our application is running in the devices properly. Figure 5.2 is the screenshots of XT702, and Figure 5.3 is the screenshots of ME525+.

In the Figure 5.2 and Figure 5.3, we could see the first screenshots when launching the application. Once the "Start" button is clicked, the current location (latitude and longitude) is displayed on the screen, and it updates whenever the location changes, shown in the second screenshots in the two figures. Our program checks the location change every 2 seconds.

If we click the "Send" button, we could see the current location can be trans-

Figure 5.2: Screenshots of XT702 (Milestone)

Figure 5.3: Screenshots of ME525+ (Defy+)

mitted to the other device. The location of the other device and the distance between the two devices are displayed in the third screenshots. After that, every time the "Send" button is clicked, the new location is sent and the distance is updated.

From the third screenshots of both Figure 5.2 and Figure 5.3, we can obtain the data on the screen, and list the data in Table 5.1. The local latitude and longitude are measured by the GPS module (M.), and the remote latitude and longitude are received from Wi-Fi module (R.). Comparing the information displayed on each device, we can find each device can measure its location and transmit to the other device successfully.

Table 5.1: Information on Each Device

Device	XT702	ME525+
XT702's Latitude	58.33544969558546 (M.)	58.33544969558546 (R.)
XT702's Longitude	8.57585906982397 (M.)	8.57585906982397 (R.)
ME525+'s Latitude	58.33544969558546 (R.)	58.33544969558546 (M.)
ME525+'s Longitude	8.57588052749609 (R.)	8.57588052749609 (M.)
Distance	1.2569655 meters	18.617025 meters

However, we notice the measured distances on the device are not the same. The Android API provides calculation of the approximate distance in meters between two locations [3]. The algorithm may has an estimation, and the result may influenced by the order of parameters. Since we always treat the measured location as the first parameter, and the received location as the second parameter, the calculated distance on each device may be different. Due to this difference, we gather statistics on both of the two devices in each test trial.

5.2 Test Scenarios

Up to now, we have checked the availability of our prototype. We design some scenarios in this section, in order to test the Android-based prototype. Considering the characteristics of the vehicles, we design both motional and motionless scenarios, which are listed below:

Scenario 1: In Scenario 1, we want to test the locations and the distances for the two devices that are stable in a distance. We would like doing test for several kinds of distances to observe the differences of the results.

Scenario 2 In Scenario 2, one of the two devices is stable, and the other one is mobile. The mobile one moves around the stable one while keeping a distance, like a circle roughly. We would also like do tests for some different distances in this scenario. Due to the mobility, the practical distances for tests may be an approximate distance.

Scenario 3 In Scenario 3, we let both the two devices be mobile. While moving, the two devices also have an approximate distance. Similarly, we test several groups of different distances.

5.3 Experimental Results

For each of the scenario above, we test several groups of trials for each distance. As the constraints of wireless signal and the scale of the field, we do not test any larger scales of distances. Because the accuracy of distance is the main concern and our space is limited, we only list the results of distance (with four decimal places retained) measured by each device, but the detailed latitude and longitude information measured by the devices are not listed in detail any more.

5.3.1 Scenario 1

Table 5.2: Test Results of Scenario 1

Real Distance	2 m		10 m		30 m	
Device	XT702	ME525+	XT702	ME525+	XT702	ME525+
Test1 (m)	9.6220	22.6679	5.0278	20.9719	11.2631	20.2535
Test2 (m)	11.1342	17.4694	6.3976	21.5701	20.6722	44.4327
Test3 (m)	11.8724	7.5981	7.2356	10.7551	27.8574	33.7878
Average (m)	10.8762	15.9118	6.2203	17.7657	19.9309	32.8247
	13.3940		11.9930		26.3778	

In this scenario, the three distances for testing are 2m, 10m and 30m respectively. For each distance, we have three trials, Test1, Test2, and Test3, which are accomplished in three different areas around the AP. The two devices are stable in two

positions at each test trial, which means the real latitude and longitude of each device do not change. Once the two devices have their positions, the measured location and distance of each device do not change as well.

The measured distances on each device for each test are listed in the Table 5.2. From the testing results, we could observe that the measured distance has a deviation from the real distance. In this scenario, if the distance is measured on each device for one test, the same deviation always keeps, since the devices do not move and the measured locations by the devices do not change.

Looking further at the measured distances, we can find that each trail has a deviation of about 10 meters, and the deviations are from 2 meters to 15 meters. What's more, the deviations on different devices of a same test trial are apparently different. These are because the measured GPS information has some deviation from the real location of the device since the GPS errors. And based on the measured GPS information, the measured distance is calculated by using an estimated algorithm as we mentioned.

Due to the low speed of GPS measurement, if we keep stable or move in a very small range in a short period of time, the measured location updates very slowly. This factor makes the measured distance have a strong temporal memory. In addition, according to the Specifications of the XT702 and ME525+, they have the different hardware of GPS receivers. XT702 has the Standalone GPS, yet ME525+ has the Assisted GPS (AGPS) [9]. The measurement speed and accuracy have a device-dependent character.

In this scenario, if the real distance is in a small scale (like 2m, even 10m), the scale of the deviations seems to be large compared with the real distance. But if the real distance is a little bit larger (like 30m or more), the scale of the deviations look not very large. When we work the average values out, the same situation comes.

5.3.2 Scenario 2

To test in this scenario, we let the device XT702 play the role of mobile one, and the device ME525+ take the role of the stable one. Because the RSS has no effect on the result, we test in one area for the stable device. While moving, XT702 measured GPS information. And the measured distances of both the devices change. We refresh the data periodically, and record the measured distance of each device every time as each test trial. The real distance is approximate since we can not guarantee it exactly while one device having mobility.

Table 5.3: Test Results of Scenario 2

Real Distance (Approximate)	2 m		10 m		30 m	
Device	XT702	ME525+	XT702	ME525+	XT702	ME525+
Test1 (m)	4.7801	9.6424	11.6971	6.2850	36.6352	25.0521
Test2 (m)	4.7801	9.6424	15.0990	7.2799	35.1055	27.6797
Test3 (m)	4.7801	9.6424	5.2032	18.2754	14.6299	40.6217
Test4 (m)	4.7801	9.6424	21.6676	12.9930	24.9361	16.8102
Test5 (m)	11.6972	9.6424	17.3991	11.3970	17.9594	24.6190
Average (m)	6.1635	9.6424	14.2132	11.2461	25.8532	26.9565
	7.9029		12.7296		26.4049	

We test five trials for each distance. The results are listed in the Table 5.3.2. When the real distance is about 2 meters, the mobility happens within a small region. The devices cannot have distinct changes of the location, so the measured distances are almost the same for the tests.

5.3.3 Scenario 3

In this scenario, the two devices both have mobility. Still, an approximate distance between the two mobile devices exists, and we have 10 trials for each distance. In the same way of testing in the Scenario 2, we refresh the data periodically and record the measured distances for each test trail. In order to have enough space to move, we set the smallest approximate distance to 4 meters in this scenario.

Table 5.4: Test Results of Scenario 3

Real Distance (Approximate)	4 m		10 m		30 m	
Device	XT702	ME525+	XT702	ME525+	XT702	ME525+
Test1 (m)	21.6548	5.2032	23.9229	6.4824	37.8059	20.4290
Test2 (m)	21.6548	5.2032	29.1323	5.5671	27.5274	21.2719
Test3 (m)	19.0058	5.5671	15.0989	9.6424	37.7958	22.6575
Test4 (m)	19.0058	5.5671	11.8674	8.9292	42.8137	25.9525
Test5 (m)	19.0058	5.5671	18.2355	20.5533	38.8945	17.0258
Test6 (m)	16.5432	7.8091	25.0520	15.5755	32.3526	25.1399
Test7 (m)	16.5432	7.8091	20.9262	6.1752	32.6572	27.3439
Test8 (m)	16.5432	7.8091	31.6365	16.5432	37.6668	22.6260
Test9 (m)	16.5432	7.8091	29.6554	5.0280	34.1912	21.1401
Test10 (m)	16.5432	7.8091	22.7401	16.3408	24.5743	28.8915
Average (m)	18.3043	6.6153	22.8267	11.0837	34.6279	23.2478
	12.4598		16.9552		28.9379	

We can see the results in Table 5.3.3. Similarly, the devices cannot have frequent obvious changes of the location when the moving range is small. Once the device moves in a small range, the measured distance on each device does not fluctuate heavily.

5.3.4 Summary of Results

We can compare statistics of the test results of all the three scenarios now. Because of the error of GPS, the deviation between the measured distance and the real distance always exists. The scale of the deviation makes the prototype cannot recognize the accurate distance less than 10 meters. But if the real distance is larger, the error of the measured distance can be seen tolerable. Therefore, for vehicle collision avoidance, the measured distance can be treated accurate. Because when the distance is too small, the collision is difficult to be avoided at that situation. In addition, in the scenarios that the device has mobility, the measured data can be refreshed in time, so the error of measured data can be diminished by using mean value. Furthermore, the average value of the two devices can always neutralize the errors, and help to reduce the errors.

Normally, if the measured distance between vehicles is below a threshold, a alarm should be given to avoid collisions. On the one hand, if the threshold is too high, more empty alarms may happen; on the other hand, if the threshold is too low, it is dangerous since collisions may happen without alarms. A good value should be selected as a threshold. The errors of the measured distance are about 10m and not larger than 20m normally. For small vessels, the smallest threshold may be set as 30m. Thus, even though with a error of 20m, the actual distance of 10m is enough for small vessels to take action to avoid collision. But for safety, the threshold should be a little bit larger than 30m. For large vessels, the threshold should be much larger, even more than hundred meters, since large vessels need more time and distance to take action. So the error of the measured distance can be ignored for the collision avoidance for large vessels.

Chapter 6

Maritime Applications and Design

As we mentioned in Chapter 2, VHF radios are popularly deployed on the vessels in modern time. Raymarine is a manufacturer of marine electronic equipments, including VHF radio. In Norway, some big and solid boat manufacturers, such as Marex, use the electronic equipments supplied by Raymarine. We will give an introduction of some marine VHF radios produced by Raymarine. Apart from the introduction, we will state an idea to connect the Raymarine devices to a PC, for our further development preparation.

6.1 Marine VHF Radio Products

Raymarine has several types of fixed VHF marine radiotelephone, like Ray49, Ray55, Ray218, etc. The radios provide reliable two-way communications on all international marine channels. Using the VHF radiotelephones, people can have voice communications between vessels. We can take a look at the Ray218 with its microphone in Figure 6.1.

The Raymarine VHF radios accept NMEA0183 data from a particular position determining device (GPS) to provide the latitude and longitude position information, and the connection is shown in Figure 6.2 [15]. The GPS information can be displayed on the screen of the radio.

Figure 6.1: Raymarine Ray218 DSC VHF Radio

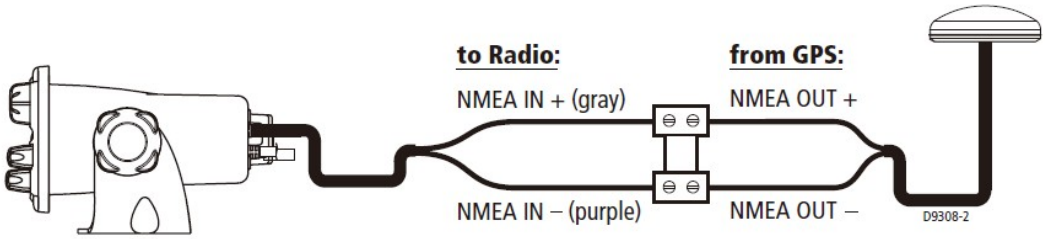


Figure 6.2: NMEA Connection from GPS to VHF Radio [15]

Moreover, the Raymarine VHF radio includes equipment for Class "D" Digital Selective Calling (DSC), which is a protocol used to send and receive pre-defined digital messages. The radio device is identified by a unique Maritime Mobile Service Identity (MMSI) number. The Raymarine radio can make six types of DSC calls, including DISTRESS, INDIVIDUAL, GROUP, ALL SHIPS, POSITION REQUEST, and RECEIVED CALLS. The POSITION REQUEST option enables the VHF radio user to request GPS information from any other VHF radio whose MMSI is known. Once a replied position in NMEA format is received by a RECEIVED CALL, the data is displayed on the screen and saved in the Position Log. Hence, by using the DSC function, a vessel can know the position of another vessel.

In addition, Raymarine produces a few VHF devices of Automatic Identification System (AIS), like AIS250 Receiver and AIS500 Transceiver, to provide advanced data communications between vessels or shore based stations to provide fast, automatic and accurate collision avoidance data. AIS is mandatory to

be carried just for large commercial vessels exceeding 300 tons that travel internationally. However, not all vessels are equipped expensive AIS, and it is not mandatory to use it even it is mandatory to carry for large vessels. Consequently, AIS can not display information from all vessels in an area [16].

6.2 Connection to PC

When Raymarine VHF radio users enable the DSC calls of POSITION REQUEST and receive replied position, the VHF radio outputs the NMEA format data in order to forward to other devices. The VHF radio has NMEA output wires, corresponding to the NMEA input wires, shown in Figure 6.3.

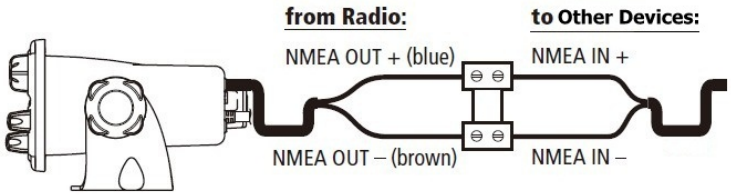


Figure 6.3: NMEA Output from VHF Radio

Up to now, Raymarine does not provide any NMEA programmable devices. If the NMEA data can be transmitted to a PC, we can apply the GPS information to our program. We have known that the NMEA0183 standard uses the serial bus and NMEA2000 uses CAN bus for data transmission. In order to let the PC read the NMEA data, we can use serial data bus, such as RS232. The normal PC has the interface of RS232 port. By hooking the right wires of RS232 and NMEA0183 cable up, the NMEA0183 data can be forwarded to RS232 port. The structure is illustrated in Figure 6.4. For NMEA2000 data, there exists CAN to RS232 converters providing a more simple way.

The local GPS information can be obtained from the boat's existing GPS, and the NEMA information from other vessels can be transmitted from the VHF radio. There exists some software running on a PC to read the NMEA data from RS232 port, such as "PuTTy". We can listen on the RS232 port by using our own

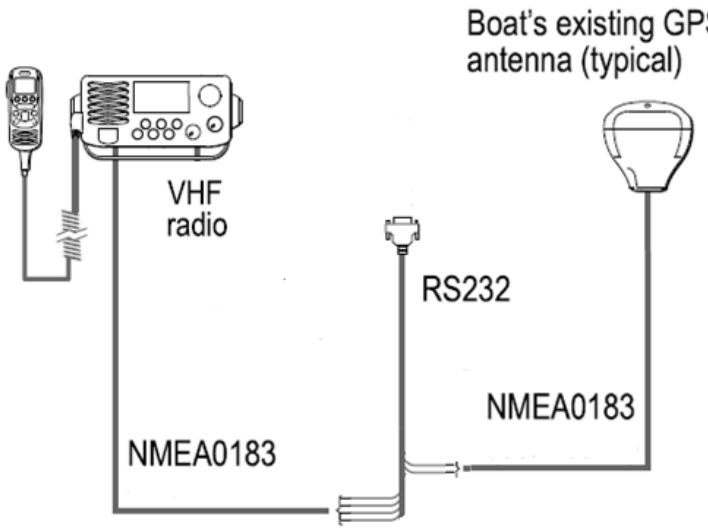


Figure 6.4: Connection Structure of NMEA Devices and PC Port

programming as well. Thus, the local and remote GPS information can be forwarded to a PC. Nevertheless, to request a location of other vessels, operations on the VHF radio need to be done manually but not automatically, because the Raymarine VHF radio is not programmable.

6.3 Prototype Extension into Marine Devices

In this section, we will discuss how to extend the Android-based prototype we developed into the maritime devices. In the structure of maritime devices, a marine GPS module takes the place of a GPS module in a smartphone for location acquisition; a marine VHF radio takes the place of a Wi-Fi module in a smartphone for communications; and a PC takes the place of a controller in a smartphone.

The maritime application needs to listen on the RS232 port instead of the network port of Wi-Fi, and needs a function to analyze the NMEA standard at the same time. Because the local location and the remote location are both forwarded to the PC via the RS232 port, the software should be capable to identify the local location and the remote location.

Moreover, the location transmission of the Raymarine device is in a passive mode through a POSITION REQUEST call. The user has to request the position manually due to the nonprogrammable VHF radio. The software needs to be suitable for these features. Therefore, the software does not control the transmission of a position, but only receives position information and computes the distance.

Chapter 7

Conclusion and Future Work

We summarize the results and conclude our project in this chapter. Referring to the results, we state the main contributions we have done in this project. And some efforts we tried in this project are discussed following on. Finally, we give some future work.

7.1 Conclusion

First, from the simulation result of the GMPE_MLH algorithm, we can see it has a satisfying accuracy to estimate the parameters of Gauss-Markov mobility model. When we extend the GMPE_MLH to 2-D case which is suitable for marine scenarios, the accuracy of location prediction for mobile node is satisfactory as well. Meanwhile, the advantage of message efficiency for transmitting parameters of GMPE_MLH model will help a lot, when several mobile vehicles need to communicate with each other to inform their status for collision avoidance.

Second, the Android-based prototype gives a fundamental model of collision avoidance for mobile vehicles. In a requisite scope for detecting risks and avoiding collisions, the prototype has enough accuracy of measurements. So in a normal or sparse environment of mobile vessels, the prototype can be used. But for a dense environment, like a docks or a narrow water channel, the prototype can not

be used since it may always alarm in this situation. Generally, the drivers keep attention in line-of-sight in a dense environment.

7.2 Contributions

In this project, we have surveyed most technologies related to collision avoidance of mobile vehicles. By studying on a location prediction algorithm, we applied this algorithm to multi-dimensional spaces and did simulation to evaluate the performance of the algorithm by ourselves.

What's more, we designed and developed a prototype on Android 2.3 platform by using Java programming for measuring location and distance of mobile stations. Testing the prototype in real environment, we have proofed that the prototype could work well.

Furthermore, we studied on the NMEA devices which are related to collision avoidance on real life vessels. After the communication with engineers from the boat manufacturer Marex and the marine electronic device manufacturer Raymarine, we designed a structure to connect the NMEA devices to the PC, for a further use of the NMEA data.

7.3 Discussion

Beside the work which are mentioned in this report, we have tried several other methods in both theoretical part and practical part. But some methods were discontinued due to inappropriateness, some work have not been accomplished due to time limitation. We discuss them in this section.

For the location prediction algorithm we mentioned in Chapter 3, we know that it just predicts the single location with the highest possibility. We considered to improve it further to adapt to the collision avoidance. A collision may be happen in an area but not in a single point. For avoiding collisions, we would like to predict an area with a high possibility where a mobile vehicle may occur in the next

time slot. Considering the overlap of the possible areas of two or more vehicles, a new collision avoidance scheme can be proposed. The main idea is to have a numerical interval with relatively high possibility according to the probability density function, but not just use the expectation value. This work is in process, but there is not result at present stage.

For the communication part of our prototype, in the beginning we tried AllJoyn technique, which is a new proximity based mobile P2P software framework developed by Qualcomm Innovation Center. Due to some problems of compatibility, we didn't successfully connect mobile devices with each other. Then, we managed to use traditional technique of Socket. And then we tried three network architectures mentioned in Chapter 4. We have implemented the three architectures to communicate successfully, but we only used the ideal one for the communication of our prototype finally.

Still for the prototype, we were wondering to try Wi-Fi Direct for the communication for our prototype as well. The Wi-Fi Direct is merely supported by the 4.0 version of Android platform up to now. At the beginning we only have two smartphones based on Android 2.3 platform. The department ordered some new devices based on Android 4.0, but it arrived very late in the semester. We transplanted our prototype in the new devices, but there is still some compatibility problems. The time limits our attempts to change communication part to Wi-Fi Direct. This work is expected to be done in the future.

For the marine NMEA devices, we were wondering to extend our prototype into the real onboard devices. Because the manufacture purchases the real devices are in a exact amount according to the amount of the boat. We could not get an external device to do test. Thus, we did not implement the design mentioned in Chapter 6 by using real marine devices.

7.4 Future Work

In our future work, we would like to propose a new collision avoidance scheme based on the GMPE_MLH model, which we discussed in the above section. Fur-

thermore, to improve the accuracy, we would like to combine Kalman Filter with the GMPE_MLH model, also with the new collision avoidance scheme.

In the implementation part, we would like to apply the GMPE_MLH algorithm and the new collision avoidance scheme to our Android-based prototype and test the new prototype. Another future work for the prototype, which we also discussed in the above section, is to use Wi-Fi Direct to communicate for the prototype. Moreover, we hope we can extend our prototype to marine onboard devices in the future.

Bibliography

[1] E. Abbott and D. Powell, "Land-Vehicle navigation using GPS," in *Proceedings of IEEE*, vol. 87, 1999, pp. 145–162.

[2] Activity. Android Developers. [Online]. Available: http://developer.android.com/reference/android/app/Activity.html

[3] Location. Android Developers. [Online]. Available: http://developer.android.com/reference/android/location/Location.html

[4] Tools. Android Developers. [Online]. Available: http://developer.android.com/guide/developing/tools/index.html

[5] P. Bahl and V. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," in *Proceedings of IEEE INFOCOM '00*, vol. 2, 2000, pp. 775–784.

[6] F. Bai and A. Helmy, *A Survey of Mobility Models in Wireless Adhoc Networks*. Kluwer Academic Publishers, 2004, ch. 1.

[7] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for Ad Hoc network research," *Wireless Communication and Mobile Computing (WCMC):Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, no. 5, pp. 483–502, 2002.

[8] Y. Chen and H. Kobayashi, "Signal strength based indoor geolocation," in *Proceedings of IEEE International Conference on Communications (ICC '02)*, vol. 1, 2002, pp. 436–439.

[9] G. M. Djuknic and R. E. Richton, "Geolocation and assisted GPS," *Computer*, vol. 34, no. 2, 2001.

[10] J. Georgy, A. Noureldin, M. J. Korenberg, and M. M. Bayoumi, "Low_Cost three-dimensional navigation solution for RISS/GPS integration using mix-

ture particle filter," *IEEE Transactions on Vehicular Technology*, vol. 59, pp. 599–615, 2010.

[11] Y.-C. Hu and D. B. Johnson, "Caching strategies in on-demand routing protocols for wireless ad hoc networks," in *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom '00)*, 2000.

[12] B. Liang and Z. J. Haas, "Predictive distance-based mobility management for multidimensional PCS networks," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 718–732, 2003.

[13] B. Liu, M. Chen, and M. Tsai, "Message-efficient location prediction for mobile objects in wireless sensor networks using a maximum likelihood technique," *IEEE Transactions on Computers*, vol. 60, no. 6, pp. 865–878, 2011.

[14] Reports or Summaries of Important Accident Investigation. Marine Department, Government of Hong Kong SAR. [Online]. Available: http://www.mardep.gov.hk/en/publication/ereport.html

[15] Raymarine, *Ray218 and Ray55 Marine VHF Radio Owner's Handbook*, 2007.

[16] ——, *AIS500 Transceiver Installation Instructions*, 2008.

[17] S. Wang, J. Min, and B. K. Yi. Location based services for mobiles: Technologies and standards. Keynote Speech of IEEE International Conference on Communication (ICC) '08, 2008.

[18] Wi-Fi Alliance. Industry White Paper: Wi-Fi CERTIFIED Wi-Fi Direct Personal, Portable Wi-Fi Technology, 2010.

[19] Y. Zhao, "Standardization of mobile phone positioning for 3G systems," *IEEE Communications Magazine*, vol. 40, no. 7, 2002.

Appendix A.

Device Specifications

A..1 Motorola ME525+ Specifications

- GENERAL:
  - Product Type – Smartphone ME525+
  - Operating System – Android 2.3.6
  - Physical Keyboard – No
  - Touch Screen – Yes
  - Display Size – 3.7 in
  - Display Resilution – FWVGA (480 x 854)
- PROCESSOR MEMORY GRAPHICS:
  - Processor – TI OMAP3620-1000
  - Processor Clock Speed – 300 $\sim$ 1000 MHz
  - RAM – 512 MB
  - Secure Storage (Approx) – 2 GB
  - GPU – Imagination PowerVR SGX530
- CONNECTIVITY:

  - WAN(Voice Bands) – GSM 850/900/1800/1900, W-CDMA 850/1900
  - WAN(Data Bearers) – GPRS/EDGE Class 12, HSDPA, HSUPA
  - WLAN – 802.11b/g/n
  - Bluetooth – Class 2, v2.1 + EDR
  - USB – Micro USB v2.0 High Speed
  - Location Services – Assisted GPS, E-Compass

- Camera:
  - Camera Resolution (Max) – 5 megapixels (2592 x 1936)
  - Camera Zoom (Max) – 6x

- MEDIA:
  - Media Format – Android core media formats, MPEG-4 encoder, WMA 9 decoder
  - Video Recording Resolution (MAX) – VGA (640 x 480)
  - Video Recording Frame Rate – 30 fps

- SENSORS:
  - Accelerometer – Yes
  - Magnetometer – Yes
  - Proximity Sensor – Yes
  - Ambient Light Sensor – Yes

A..2 Motorola XT702 Specifications

- GENERAL:
  - Product Type – Smartphone XT702
  - Operating System – Android 2.3.7

  - Physical Keyboard – Yes
  - Touch Screen – Yes
  - Display Size – 3.7 in
  - Display Resilution – FWVGA (480 x 854)

- PROCESSOR MEMORY GRAPHICS:
  - Processor – TI OMAP3430
  - Processor Clock Speed – 250 $\sim$ 1000 MHz
  - RAM – 256 MB
  - Secure Storage (Approx) – 512 MB
  - GPU – Imagination PowerVR SGX530

- CONNECTIVITY:
  - WAN(Voice Bands) – GSM 850/900/1800/1900, W-CDMA 850/1900/2100
  - WAN(Data Bearers) – GPRS/EDGE Class 12, HSDPA, HSUPA
  - WLAN – 802.11b/g
  - Bluetooth – Class 2, v2.1 + EDR
  - USB – Micro USB v2.0 High Speed
  - Location Services – Standalone GPS w/internal antenna, E-Compass

- Camera:
  - Camera Resolution (Max) – 5 megapixels (2592 x 1936)
  - Camera Zoom (Max) – 4x

- MEDIA:
  - Media Format – Android core media formats, H.264 encoder, WMA 9 decoder
  - Video Recording Resolution (MAX) – VGA (640 x 480)
  - Video Recording Frame Rate – 30 fps

- SENSORS:
  - Accelerometer – Yes
  - Magnetometer – Yes
  - Proximity Sensor – Yes
  - Ambient Light Sensor – Yes

Appendix B.

Matlab Codes

```
%Function of Estimating miu and sigma%%%%%%%%%%%%%%%%%%
function [ miu_eva, sig_eva ] = Eva_Miu( t, v , miu_eva_old
   , v_old, sig_eva_old )

sig2_eva_old = sig_eva_old.^2;

if t==1
    miu_eva = v;
    sig2_eva = 0.1;%%
elseif t==2
    miu_eva = (t-1)/t*miu_eva_old + 1/t*v;
    sig2_eva = ((v_old-miu_eva)^2+(v-miu_eva)^2)/2;
elseif t>=3
    miu_eva = (t-1)/t*miu_eva_old + 1/t*v;
    sig2_eva = (t-1)/t*sig2_eva_old + 1/t*(v-miu_eva)^2;
end

sig_eva = sqrt(sig2_eva);
end

%Function of Calculating MLH alpha_mean%%%%%%%%%%%%%%%%%%
```

```
function [ alpha_mean ] = Alpha_mean ( t, v, v_old, miu_eva
   , sig_eva, alpha_mean_old )

alpha = linspace(0,1,11);
like_hood = zeros(1,10);
maxlike = 0;

for i=1:10
    a = (alpha(i)+alpha(i+1))/2;
    like_hood(i) = integ(a, v, v_old, miu_eva, sig_eva);

    if like_hood(i)>maxlike
        index_1 = i;
        maxlike=like_hood(index_1);
    end
end
alpha_like = (alpha(index_1) + alpha(index_1+1)) / 2;

alpha_mean = 1/t * alpha_like + (t-1)/t * alpha_mean_old;
end

%Function of Estimating alpha%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ alpha_eva ] = Eva_Alpha( alpha_mean )

    if alpha_mean<=0.35338
        alpha_eva = 0;
    elseif 0.35338<alpha_mean && alpha_mean<=0.44602
        alpha_eva = 1/53.151 * log(0.2/-0.43403/alpha_mean
           - 1/-0.43403);
    elseif 0.44602<alpha_mean && alpha_mean<=1
        alpha_eva = 1/0.37492 * log(0.2/-0.55069/alpha_mean
            - 1/-0.55069);
    elseif alpha_mean>1
        alpha_eva = 1;
    end
```

```
end

%Funcution of Predicting Location%%%%%%%%%%%%%%%%%%%%%%%%
function [ next_loc ] = Loc_Pre( now_loc, now_v, alpha_eva,
    miu_eva)
next_loc = now_loc + alpha_eva*now_v +(1-alpha_eva)*miu_eva
   ;
end

%Simulation of 1-D Case%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ t, alpha_eva_array, miu_array, sig_array] =
   Testmain(in_para, step)
    TMAX = step;
    t=1:TMAX;

    GM_alpha = in_para(1);
    GM_mu = in_para(2);
    GM_sigma = in_para(3);
    v_array = zeros(1,TMAX);
    v_old = randn;
    v_old_array=zeros(1,TMAX);

    %initialization
    miu=0;
    miu_array = zeros(1,TMAX);
    sig=0;
    sig_array = zeros(1,TMAX);
    alpha_mean = 0;
    alpha_mean_array = zeros(1,TMAX);
    alpha_eva_array = zeros(1,TMAX);

    now_loc=0;
    next_loc_array=zeros(1,TMAX+1);

    for index=1:TMAX
```

```
85
        v_now = GM(v_old, GM_alpha, GM_mu, GM_sigma);
        v_array(index)=v_now;

        [miu,sig] = Eva_Miu(t(index),v_array(index),miu,
           v_old,sig);
90      miu_array(index) = miu;
        sig_array(index) = sig;

        [alpha_mean] = Alpha_mean ( t(index), v_array(index
           ), v_old, miu, sig, alpha_mean );
        v_old_array(index) = v_old;
95      v_old=v_array(index);
        alpha_mean_array(index) = alpha_mean;

        alpha_eva = Eva_Alpha(alpha_mean);
        alpha_eva_array(index) = alpha_eva;
100
        next_loc = Loc_Pre(now_loc,v_array(index),
           alpha_eva, miu);
        now_loc=next_loc;
        next_loc_array(index+1)=next_loc;
    end
105 end

%Simulation of 2-D Case%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear;clc;
TMAX = 1000;
110 TI=5; %time interval

%initialization
x1=0;
v_old_x1 = randn;
115 miu_old_x1 = 0;
sig_old_x1 = 0;
```

```
alpha_mean_old_x1 = 0;

y1=0;
v_old_y1 = randn;
miu_old_y1 = 0;
sig_old_y1 = 0;
alpha_mean_old_y1 = 0;

A_array = zeros(TMAX,2);
A_pre_array = zeros(TMAX,2);
A_err_array = zeros(TMAX,1);

%record of history(for observation,no need in practical)
miu_array_x1 = zeros(TMAX,1);
sig_array_x1 = zeros(TMAX,1);
alpha_mean_array_x1 = zeros(TMAX,1);
alpha_eva_array_x1 = zeros(TMAX,1);
miu_array_y1 = zeros(TMAX,1);
sig_array_y1 = zeros(TMAX,1);
alpha_mean_array_y1 = zeros(TMAX,1);
alpha_eva_array_y1 = zeros(TMAX,1);

for t=1:TMAX
    %------------x1---------------------
    v_now_x1 = GM(v_old_x1, 0.5, -5, 3);
    [ next_coo_x1, miu_x1, sig_x1, alpha_mean_x1] =
       Coordinate_Predict( t, v_now_x1, v_old_x1,
       miu_old_x1, sig_old_x1, alpha_mean_old_x1, x1);
    x1_pre = next_coo_x1;
    x1 = x1 + v_old_x1*TI;
    %store last status
    miu_old_x1 = miu_x1;
    sig_old_x1 = sig_x1;
    alpha_mean_old_x1 = alpha_mean_x1;
    v_old_x1 = v_now_x1;
```

```
    %%history%%
    miu_array_x1(t) = miu_x1;
    sig_array_x1(t) = sig_x1;
    alpha_mean_array_x1(t)=alpha_mean_x1;
    alpha_eva_array_x1(t) = Eva_Alpha(alpha_mean_x1);

    %------------y1---------------------
    v_now_y1 = GM(v_old_y1, 0.5, 10, 3);
    [ next_coo_y1, miu_y1, sig_y1, alpha_mean_y1] =
       Coordinate_Predict( t, v_now_y1, v_old_y1,
       miu_old_y1, sig_old_y1, alpha_mean_old_y1, y1);
    y1_pre = next_coo_y1;
    y1 = y1 + v_old_y1*TI;
    miu_old_y1 = miu_y1;
    sig_old_y1 = sig_y1;
    alpha_mean_old_y1 = alpha_mean_y1;
    v_old_y1 = v_now_y1;
    %%history%%
    miu_array_y1(t) = miu_y1;
    sig_array_y1(t) = sig_y1;
    alpha_mean_array_y1(t)=alpha_mean_y1;
    alpha_eva_array_y1(t) = Eva_Alpha(alpha_mean_y1);

    A_array(t,:)=[x1,y1];
    A_pre_array(t,:)=[x1_pre,y1_pre];

    %distance between real and prediction
    A_err_array(t) = DistanceAB (x1, y1, x1_pre, y1_pre);
end
```

Appendix C.

Java Codes

```
//Activity, Client and Handler
package com.jzh.project;

import java.io.IOException;
import java.io.OutputStream;
import java.net.Socket;
import java.net.UnknownHostException;
import android.app.Activity;
import android.content.Context;
import android.location.Location;
import android.location.LocationManager;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class JzhProjectActivity extends Activity
{
     private Button startButton = null;
```

```
    private Button sendButton = null;
    private TextView mTextView = null;
    private TextView mLocLocTextView = null;
    private Socket clientSocket = null;
    private OutputStream outStream = null;
    private ListenerThread mListenerThread = null;
    private double localLat;
    private double localLng;
    private JzhLocation locLoc = null;
    static Handler mHandler = null;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mTextView = (TextView)this.findViewById(R.id.
            retextview);
        mLocLocTextView = (TextView)this.findViewById(R.id
            .localposition);
        startButton = (Button)this.findViewById(R.id.
            startbutton);
        sendButton = (Button)this.findViewById(R.id.
            sendbutton);
        sendButton.setEnabled(false);
        final LocationManager locationManager;
        locationManager = (LocationManager)
            getSystemService(Context.LOCATION_SERVICE);
        this.locLoc = new JzhLocation(locationManager,
            mLocLocTextView);

        startButton.setOnClickListener(new View.
            OnClickListener()
        {
```

```
            @Override
            public void onClick(View v)
            {

                try {
                    mListenerThread = new
                        ListenerThread();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                mListenerThread.start();
                sendButton.setEnabled(true);
                startButton.setEnabled(false);
                locLoc.updateWithNewLocation(locLoc.
                    location);
                locationManager.requestLocationUpdates(
                    locLoc.provider, 2000, 10, locLoc.
                    locationListener);
            }
    });

        sendButton.setOnClickListener(new View.
            OnClickListener()
        {

            @Override
            public void onClick(View v)
            {
                try
                {
                    //Configure IP and port
                    clientSocket = new Socket("
                        192.168.1.101",8888);
                }
```

```
                catch (UnknownHostException e)
                {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                catch (IOException e)
                {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                // TODO Auto-generated method stub

                byte[] msgBuffer = null;
                String text = locLoc.latLongString;
                try {
                    msgBuffer = text.getBytes();
                    outStream = clientSocket.
                       getOutputStream();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }

                try {
                    outStream.write(msgBuffer);
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        });

        //Message Handler
        mHandler = new Handler()
        {
```

```
            @Override
115         public void handleMessage(Message msg)
            {

                String stri = (msg.obj).toString();
                int cut = stri.indexOf("\nLongtitude:");
120             String latString = stri.substring(9, cut);
                String lngString = stri.substring(cut+12);
                double remoteLat = Double.parseDouble(
                   latString);
                double remoteLng = Double.parseDouble(
                   lngString);
                double localLat = locLoc.location.
                   getLatitude();
125             double localLng = locLoc.location.
                   getLongitude();

                float[] dist = new float[3];
                Location.distanceBetween(localLat,
                   localLng, remoteLat, remoteLng, dist);
                mTextView.setText("The Remote Device's
                   Current Location:\n" + stri + "\
                   nDistance:" + dist[0] + " meter(s)");
130         }
        };

    }

135 @Override
    public void onStop()
    {
        super.onStop();
        if(this.mListenerThread != null)
140     {
          this.mListenerThread.stop = true;
```

```
            //this.mListenerThread.interrupt();
            this.mListenerThread.stop();
        }
    }

    @Override
    public void onDestroy()
    {
        super.onDestroy();
        if(this.mListenerThread != null)
        {
          this.mListenerThread.stop = true;
            this.mListenerThread.interrupt();
            this.mListenerThread.stop();
        }

    }
}

//Server(Listener)
import java.io.IOException;
import java.io.InputStream;
import java.net.ServerSocket;
import java.net.Socket;
import android.os.Message;

class ListenerThread extends Thread
{
     private ServerSocket mServerSocket;
     private Socket reclientSocket;
    private InputStream mInputStream = null;
    private byte[] buf ;
    private String str = null;
    public  boolean stop = false;
    private Message msg = null;
```

```
    public double lat = 0;
    public double lng = 0;

    public ListenerThread() throws IOException{
           //Create a ServerSocket
         this.mServerSocket = new ServerSocket(8888);
    }
    @Override
    public void run()
    {
        //Listen the port
     while(!stop)
        {
             try {
                 this.reclientSocket = this.mServerSocket.
                    accept();
                 this.mInputStream = this.reclientSocket.
                    getInputStream();
             } catch (IOException e) {
                 // TODO Auto-generated catch block
                 e.printStackTrace();
             }

            //Process a coming data
             msg = new Message();

            this.buf = new byte[256];
            try {
                this.mInputStream.read(buf);
            } catch (IOException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
            this.str = new String(this.buf);
            msg.obj = this.str;
```

```
                JzhProjectActivity.mHandler.sendMessage(msg);
                try {
                        this.reclientSocket.close();
                        this.mInputStream.close();
                  } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                  }
            }
      }
}

//Location
import android.location.Criteria;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.widget.TextView;

public class JzhLocation {
     private TextView locDisplay = null;
     private Criteria criteria = new Criteria();
     private LocationManager locationManager = null;
     protected String provider = null;
     protected Location location = null;
     protected double lat = 0;
    protected double lng = 0;
    protected String latLongString = null;

     public JzhLocation (LocationManager locM, TextView
        locDis){

          locationManager = locM;
          locDisplay = locDis;
```

```
        provider = locationManager.getBestProvider(
            criteria, true);
        location = locationManager.getLastKnownLocation(
            provider);

       criteria.setAccuracy(Criteria.ACCURACY_FINE);
       criteria.setAltitudeRequired(false);
       criteria.setCostAllowed(false);
       criteria.setPowerRequirement(Criteria.POWER_LOW);

    }

   public final LocationListener locationListener = new
      LocationListener() {
        public void onLocationChanged(Location location)
            {
                updateWithNewLocation(location);
       }
       public void onProviderDisabled(String provider){
            updateWithNewLocation(null);
       }
       public void onProviderEnabled(String provider){ }
       public void onStatusChanged(String provider, int
          status,Bundle extras){ }
    };

   public void updateWithNewLocation(Location location) {
       if (location != null) {
            double lat = location.getLatitude();
            double lng = location.getLongitude();
            latLongString = "Latitude:" + lat + "\
               nLongtitude:" + lng;
            locDisplay.setText("Your Current Location:\n
               " + latLongString);
       }
```

```
            else {
                latLongString = null;
                locDisplay.setText("No Current Location
                    Information\n");
            }
        }
}
```