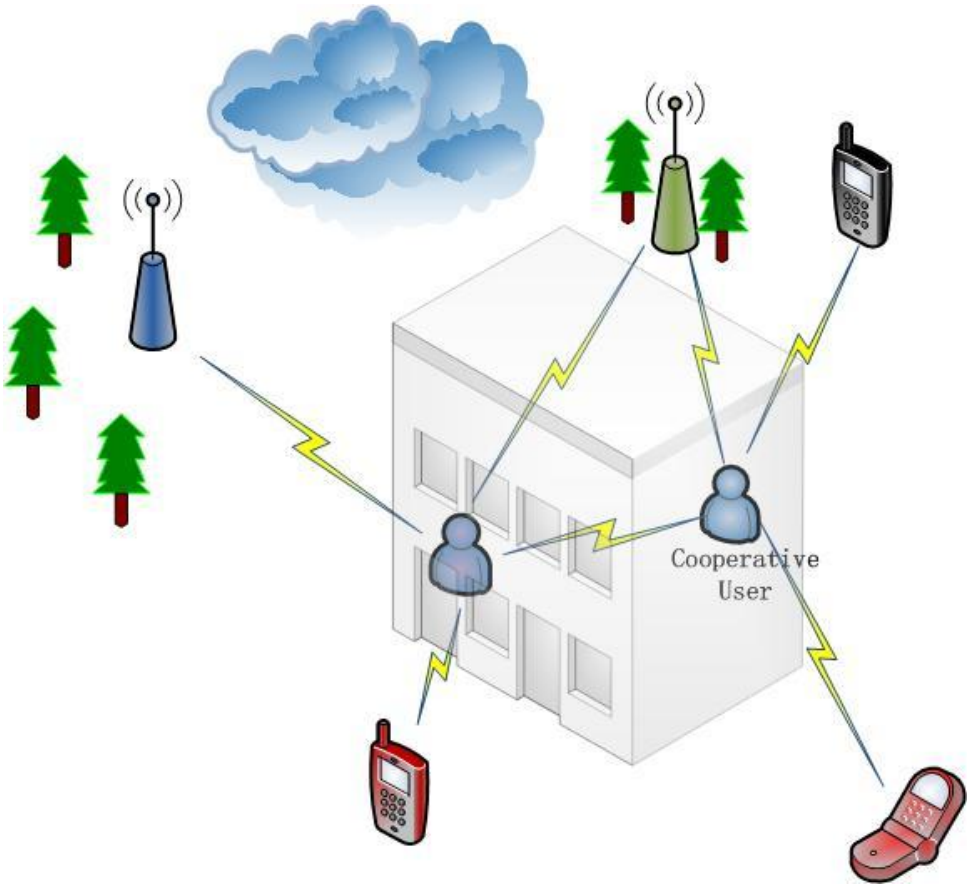
CHALMERS



Consistency Verification in Location-aware Wireless Networks

Master of Science Thesis [in the Master Degree Programme, communication engineering]

YI LI

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2008
REPORT NO.

REPORT NO.

English Title

Consistency Verification in Location-aware Wireless Networks

Author, Program and Year
Yi Li, Communication Engineering, 2010

Instructor
Henk Wymeersch, Assistant Professor, CTH

Credits
30.0

Language
English

Examination
This graduation work is a part of the requirements for a Degree of Master in Communication Engineering

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
SE-41296, Göteborg, Sweden
Telephone + 46 (0)31-772 1000

Abstract

Nowadays, the cooperative communication technology is evolving in a high speed. However, in practical cooperative networks, devices may fail and may inject false position information into the network. This thesis project aims to design and implement algorithms to verify the consistency of the wireless networks. In the first half of the research, three different algorithms will be designed upon a wireless network without cooperative communication. These three algorithms use Global Positioning System (GPS) integrity monitoring method, linear programming method and factor graph method respectively. In the second half of the research, a wireless communication scenario with cooperative communication will be considered, and we choose a algorithm which has the best performance in the first half of the research. All of the simulation results will be compared together, and we will come to a conclusion using factor graph method with cooperative communication can achieve the best consistency verification performance.

Keywords: Cooperative Positioning, RAIM, linear programming, factor graph.

Contents

Chapter 1

Problem Formulation

1.1 Brief Introduction and Motivation

Due to the trend towards globalization and mobility in today's telecommunication industry, Location-aware technologies are utilized at many applications of commercial, public and military sectors. The application of evolving location-aware technologies is a revolution to the wireless communication industry. Highly accurate ubiquitous location-awareness is achieved by communication between nodes in wireless networks.

Actually, The Global Positioning System (GPS) has already provided a reliable positioning solution for users worldwide regardless of weather, darkness or time. It has been applied in many military and civilian applications, as GPS receivers are now integrated in many 3G mobile phones. However, equipping every node or sensor in a wireless network with a GPS receiver is impractical and unnecessary, as it leads to problems such as an increased cost, higher battery consumption and lack of robustness to jamming for military applications [1].

The consistency of the communication between nodes is an important factor, which affects the performance of the location-awareness a lot. We wish the network is working consistently all the time. However, the information transmitted between nodes is not reliable all the time, due to all kinds of reasons, such as the malfunctioning problems happened to the nodes, or information transmission errors. The lack of consistency will

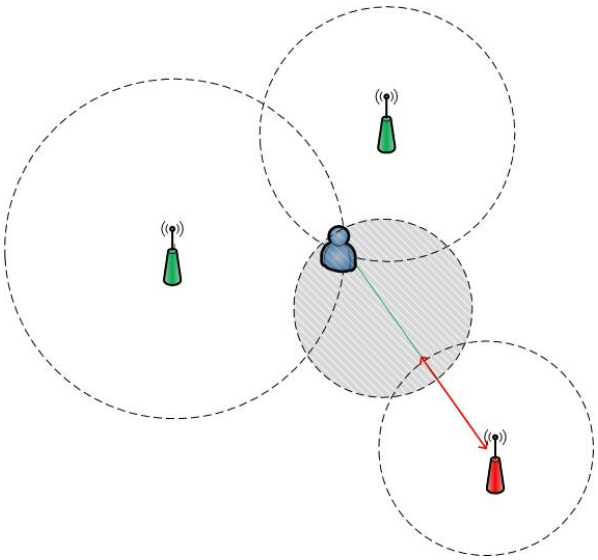


Figure 1.1: How malfunctioning problems affect positioning

give rise to a location error of the nodes, which will have detrimental effect to the whole network. Therefore, a proper consistency verification mechanism is necessary to bring in to the wireless network systems, thus the nodes which transmit incorrect location-aware information can be detected, and the location correctness can be ensured.

Here is an example about how malfunctioning problems give rise to localization failure. As figure 1.1 shows, there are one agent and three anchors in a certain wireless network. Two anchors are working well and provide correct distance estimate to the agent, while the third anchor is malfunctioning, and makes the agent get an incorrect distance estimate. The red line represents the incorrect distance. Due to the malfunctioning problem, based on these distance measurements, the agent can't localize it accurately, and the shadowed area includes all possible agent position. The larger distance measurement error is, the larger location error will be.

1.2 Basic Localization Model

1.2.1 Localization Model Construction

In this thesis, we define two kinds of nodes in the location-aware communication system, one is called "agent", which is the node who needs to locate itself, based on the location

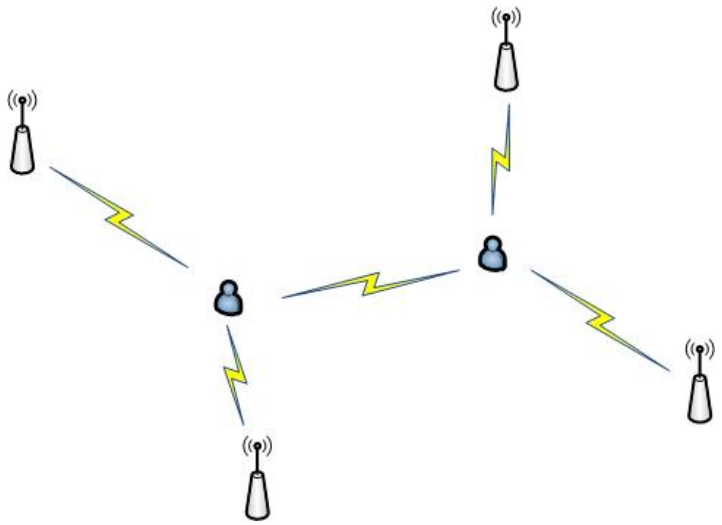


Figure 1.2: An example of a wireless network

information of other nodes. The other one kind of nodes is called "anchor", which provide the location information to the "agent". When the agent needs to locate itself, all the neighbor anchors will keep transmitting their location to the agent.

The organization example of the network model is drawn in Figure 5.1. We mark the agents as a and mark the anchors as $x_j, j = 1, 2, 3 \cdots 10$. The anchors can give the estimation of their positions: $\hat{x}_j, j = 1, 2, 3 \cdots 10$. The information transmitted between anchors and agents are the position of the anchor. Based on the transmission, the agent can estimate the distance.

$$d_j = \|x_j - a\|, j = 1, 2, 3 \cdots \tag{1.1}$$

There are many kinds of methods which can give a distance estimate. For instance, there are a series of distance related measurement systems, such as RSS (received signal strength), TOA (time of arrival) and TDOA (time difference of arrival). In RSS system, the power of transmitted signal and received signal is measured, and using Friis equation, distance can be estimated. TOA system estimates distance by measuring the propagation time between transmitter and receiver; TDOA system measures the propagation time between one transmitter and a number of receivers, and estimate the distance using the time differences [2].

As it mentioned before, due to the malfunction problems, some of the estimation values are not reliable. So we define two kinds of the distance estimates, one is for those anchors working well, and the other one is for other anchors not working well. In particular, we have:

$$H_0 : \hat{d}_j = \|x_j - a\| + n_{good} \tag{1.2}$$

$$H_1 : \hat{d}_j = n_{bad} \tag{1.3}$$

We define n_{good} is the measurement noise for those anchors which are working well, and the measurement noise applies the normal distribution $N(0, \sigma^2_{good})$, $\sigma_{good} = 0.1(m)$. We define n_{bad} represents the noise brought by the malfunction problems, which applies the uniform distribution $U(0, R_{max})$, where $R_{max} = 20m$, representing that the maximum communication range between the agent and the anchors are 20 m.

The basic principle of the localization is showed in the figure 1.3, in which, there exists 3 anchors and 1 agent. The circles around the 3 circles represent the distribution of the possible agent position, based on the distance measurement $\hat{d}$. The common interception of those 3 circles gave out the exact agent position. It is obvious that at least 3 anchors are needed to localize an agent correctly, because if there are only 2 anchors, then there will be 2 common interceptions which are the possible agent positions, therefore the localization can't be achieved.

In this thesis research, it will be divided into two major parts, according to two different models. One of them is the non-cooperative model. In a non-cooperative communication network, the only one agent is taken into account, even if there are more agents existing in the range. The only one communication type in a non-cooperative communication is the one between the agent and the anchors. The communication between agents is not considered in this model. Apparently, in the figure 1.2, there exists communication between two agents, so it is not a non-cooperative communication

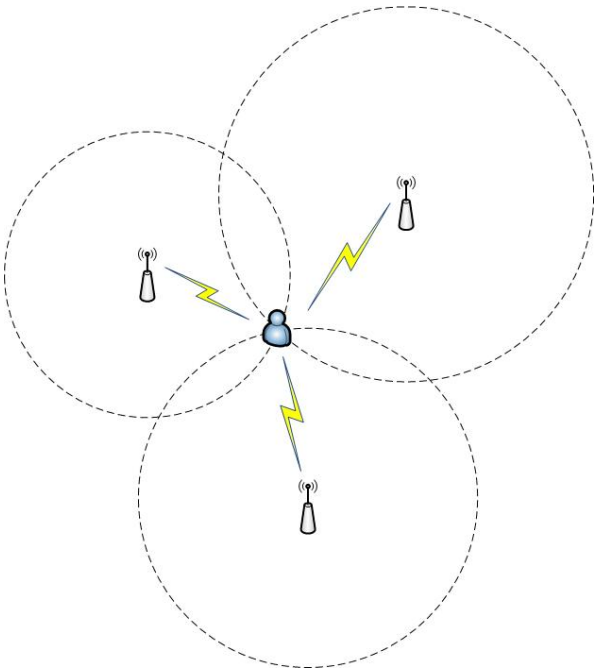


Figure 1.3: The basic principle of localization

type. We call it as cooperative communication model. In such model, more than one agent is taken into account. The agents communicate to each other and share the relative information so as to improve the localization accuracy.

In the first part, for non-cooperation, three different algorithms will be designed and implemented (GPS integrity monitoring, Linear programming method, Factor graph method). After that, one algorithm with best performance will be chosen to implement on the cooperation part. And the final conclusion is based on all of the results above.

1.2.2 Non-Bayesian Estimation

This thesis is based on the estimation theory, so it is necessary to give a brief overview of the estimation methods, which is the base of the localization method. There are a number of approaches for estimation, and they are categorized into two types: Non-Bayesian or Bayesian. All these methods can estimate a parameter x from an observation z. The difference between these two types of estimation is non-Bayesian estimation treat x as an unknown deterministic parameter, but the Bayesian estimation treat x as a realization of a random variable X. In the case of this thesis, x represents the status of the

anchors, i.e. the location estimation of the anchors, which are unknown deterministic parameter, so it is proper to use non-Bayesian estimation method.

There are two non-Bayesian estimation methods we will consider, one is least-squares (LS) estimation and the other one is maximum likelihood (ML) estimation [3].

$$\hat{x}_{LS} = \arg\min \|z - f(\cdot)\|^2 \tag{1.4}$$

$$\hat{x}_{ML} = \arg\max_{X} p_{Z|X}(z|x) \tag{1.5}$$

More specific to this thesis research, the LS and likelihood estimation can be written as:

$$\hat{a}_{LS} = \arg\min_{a} \sum_{j=1}^{N_a} \left\| \hat{d}_j - \|a - x_j\| \right\|^2 \tag{1.6}$$

$$\hat{a}_{ML} = \arg\max_{a} \sum_{j=1}^{N_a} \log p(\hat{d}_j | x_j, a) \tag{1.7}$$

In the following research, non-Bayesian LS estimation will be used a lot, firstly to implement on building performance benchmark, the upper bound and the lower bound. Secondly, it will be used in GPS integrity method and Linear programming method, and also referred in the Factor graph method.

1.3 Descriptions of the Research

1.3.1 The Connections with Existing Work

Nowadays, the technology of GPS has already used widely all over the world, but it is not possible to utilize GPS in any kinds of network, due to some issues such as power consumption. GPS standards have defined some algorithms to maintain consistency, there are some limitations with those algorithms which will be talked about later, in

this research, the algorithms in GPS will be modified and implemented [4].

Besides, people have done some research of detecting outliers in source localization using optimization methods, such as linear programming. It is found that the performance can be reliable only when the number of outliers is below some thresholds. In this thesis research, linear programming method will be used and we will also have a research to see what the performance will be when the malfunctioning problem is severe.

The algorithms above are already designed by people's researches, and will be discussed in the next chapters that these two algorithms will be both based on least-square method. Another kind of algorithm will be designed which is brand new in localization area. It uses the concept of "factor graph". It has been proved that factor graph can deal with the marginal probability problems very effectively.

Those three algorithms above are designed based on a non-cooperative network, but there is rarely research which is related to the consistency verification in cooperative networks. There are some publications which defined the cooperative localization algorithms, this thesis will have a research on consistency verification continually [3].

1.3.2 Simulation Environment Set Up

In order to compare the performances between different algorithms, it is necessary to set up a simulation environment which is used for all of the algorithms in this thesis research. The benchmark will be also built based on this same simulation environment.

For the non-cooperative part, consider there is one agent a and ten anchors $x_i, i = 1, 2 \cdots 10$, all of the anchors can directly communicate with the agent. Assume there is no communication between each anchors. As it has been mentioned, consider there are two types of measurement noises existed in the wireless network. One is when the anchors are working well, the measurement noise $n_{good} \sim N(0, \sigma^2_{good}), \sigma_{good} = 0.1(m)$. The other one type of noise is when the anchors are malfunctioning, the measurement noise $n_{bad} \sim U(0, R_{max})$, where R_{max} represents the maximum communication range in

this wireless network. Assume $R_{max} = 20$m.

For the algorithm simplicity consideration, assume the agent exists in a relatively smaller range, let's say the size of the range is $10m \times 10m$. That is to say, the theoretical maximum location error is $14.14m$. In least square method, the optimization is performed with a grid, where the searching step length is set as 0.01m. In other algorithms in this thesis research, the searching resolution is set as $0.05m$.

This research is on the localization performance when malfunctioning problems happen. In order to compare the performances under different levels of malfunctioning, we set the malfunctioning probability is $prob = [0 : 0.1 : 1]$. For each probability value, it will perform 1000 experiments, and compute the average location error based on the 1000 results. The expected performance curves will be based on these probabilities. The x-axis is the malfunctioning probability, and the y-axis is the average location error.

1.3.3 Structure of the Thesis

The structure of this thesis is designed as follows: The first chapter will describe the existing consistency problems and the research motivation, and current relative researches are introduced, as well as the simulation environment set up. Chapter 2 will set the performance benchmark, the upper bound and lower bound of the algorithm performance will be given out. Chapter 3 will introduce the GPS integrity monitoring method, and the new algorithm based on the idea of the GPS method will be given out. Chapter 4 introduced the linear programming, also a new algorithm using linear programming will be designed and implemented. Chapter 5 will demonstrate the concept of factor graph and the sum-product algorithm, and design the algorithm used for non-cooperative wireless network. Chapter 6 will modify the algorithm designed in Chapter 5, and use the algorithm in cooperative wireless network. Chapter 7 will have a wrap-up of all these results, and come to a conclusion. The future work based on this research will be talked about also.

Chapter 2

Performance Benchmark

2.1 Introduction

As it has been talked about in previous section, three kinds of algorithms will be designed and implemented in the following sections. In order to compare and analysis the performance between these three algorithms, it is necessary to build up a performance benchmark which includes the performance upper bound and lower bound.

Here defines how performance is evaluated in the following research. Since it focuses at evaluating the consistency of the localization when malfunctioning problems happen to the anchors. So the performance of the algorithms is based on the accuracy of the localization, which can be assessed by computing the average location error (between the true agent position and the estimated agent position) for one certain agent. We assume that the malfunctioning problems happen in a certain probability in this wireless network, and the probability is ranged from 0 to 1.

For all of the algorithms, including the non-cooperative and cooperative, we put the performance curve in such coordinator: the x-axis represents the malfunctioning probability, which is ranged from 0 to 1, with interval of 0.1; the y-axis represents the average location error. In the performance benchmark, there are two curves which are given by the upper bound and the lower bound of the algorithms. The upper bound represents the maximum average location error; the lower bound represents the

Algorithm 2.1 LS localization

For l=1 to N_{iter}($N_{iter}$is the iteration number)

$\hat{a}^{(l)} = \hat{a}^{(l-1)} + \delta \sum(\hat{d}_j^{(l-1)} - \tilde{d}_j^{(l-1)})e_j^{(l-1)}$

End for

theoretical minimum average location error.

2.2 Upper bound of the average location error

We assume the malfunction problem happened to the anchors with a probability $p = 0, 0.1, 0.2 \cdots 1$. For each bad anchor probability, we will determine the average location error. As it is mentioned before, the upper bound represents the maximum average location error, so it is reasonable to set the upper bound as the performance curve when the original Least Square method is used. GPS integrity monitoring method and Linear Programming method is based on the Least Square method, so theoretically, they will have better performance rather than the Upper bound. Factor graph is not based on the Least Square method, but it is still worth to put the result of factor graph together with all other algorithms.

Now let's demonstrate the algorithm 2.1 which is used to determine the performance upper bound. $\hat{a}$ is the estimated position of the agent, and δ is the step size for each updating. $\tilde{d}_j$ represents the estimated distance between the jth anchor and the estimated position of agent, i.e.

$$\tilde{d}_j = \|x_j - \hat{a}\| \tag{2.1}$$

and $e_j^{(l-1)}$ is a unit vector oriented along the line connecting $\hat{a}$ and $\hat{x}_j$:

$$e_j^{(l-1)} = \frac{(\hat{a}^{(l-1)} - \hat{x}_j^{(l-1)})}{\left\|\hat{a}^{(l-1)} - \hat{x}_j^{(l-1)}\right\|} \tag{2.2}$$

The updating equation can be demonstrated as follows: In each iteration, the difference between $\hat{d}_j$and $\tilde{d}_j$ is hoped to decrease. When they equal to each other, then the

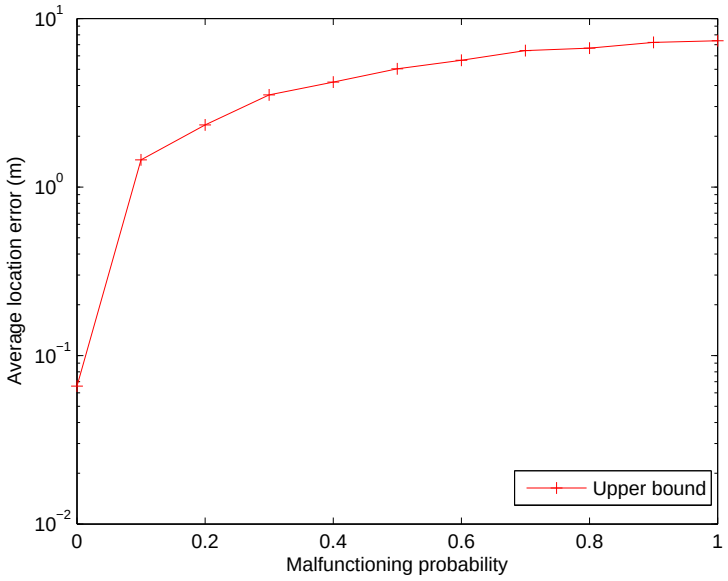


Figure 2.1: The upper bound of the average location error

summation is zero in the updating equation. When $\hat{d}_j$ is less than $\tilde{d}_j$, the LS algorithm corrects this by moving the estimated position $\hat{a}$ towards $\hat{x}_j$.On the other hand, when $\hat{d}_j$ is bigger than $\tilde{d}_j$, then LS algorithm will correct this by moving $\hat{a}$ towards x_j. After a number of iterations, the estimate position of the agent will move close to its true position, and the step size δ controls the converging speed.

Applying this algorithm into simulation, we consider 10 anchors neighbor to 1 agent, the anchors are generated randomly. After 1000 iterations, the estimated location of the agent is $\hat{a}$. Therefore, the location error is defined as the difference between the estimated location and the true location a, i.e.

$$n = \|a_i - \hat{a}_i\| \tag{2.3}$$

We do this experiment for 1000 times, and we can get the average location errors for each malfunctioning probability.

The result of the upper bound of the average location error is showed as Figure 2.1.

Algorithm 2.2 Algorithm for computing the lower bound

For l=1 to N_{iter}($N_{iter}$is the iteration number)

$$\hat{a}^{(t,l)} = \hat{a}_i^{(t,l-1)} + \delta_i^{(t,l)} \sum_{good\ anchors} (\hat{d}_j^{(t,l-1)} - \tilde{d}_j^{(t,l-1)}) e_{ij}^{(t,l-1)}$$

End for

2.3 Lower Bound for the Average Location Error

Computing the lower bound of the average location error is similar with its upper bound. In the upper bound computation, we consider all the neighbor anchors of the agent a, including the "good anchors" which working well and the "bad anchors" which are not working well. Clearly the "bad anchors" will cause the location error. In the lower bound computation, we assume that the agent knows exactly which anchors are malfunctioning ("bad anchors"), and which anchors are working well ("good anchors"). Furthermore, the agent ignores the measurements given by the "bad anchors", and only uses the measurements from "good anchors", under this assumption, we can achieve the lower bound. It is worth mentioning that this lower bound is only suitable in theory, not in reality, since the agent does not know which anchors are malfunctioning in reality.

Define the algorithm which is used for fixing the lower bound as Algorithm 2.2. The curve of the lower bound is as in Figure 2.2, given together with the upper bound curve.

2.4 Discussion

The benchmark gives the experimental upper bound and lower bound of the algorithm within the certain simulation environment. The upper bound is determined by implementing ordinary least square method, all of the anchors (including good anchors and bad anchors) are used for localization, while the lower bound is determined by only using good anchors for least square. This explains why there is a huge gain appeared between upper bound and lower bound. The lower bound only took good anchors into account, so even the malfunctioning probability is 0.1, there will be not so much effect on the localization performance. When malfunctioning probability increases, fewer an-

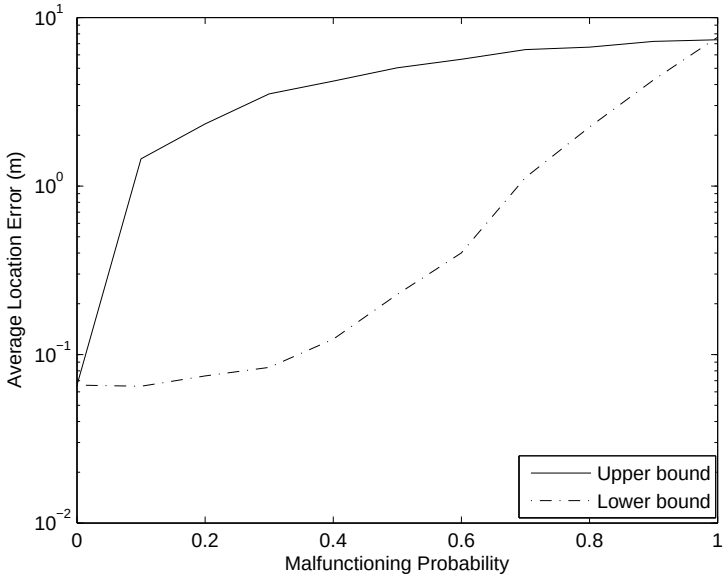


Figure 2.2: The upper and lower bound of average location error

chors are used for localization, so the location errors will increase accordingly. On the other hand, when we compute the upper bound, all of the anchors are considered, so the bad anchors give rise to huge location error. The bigger malfunctioning probability is, the bigger average location error will be.

Chapter 3

GPS Integrity Monitoring Method

3.1 Introduction

Global Positioning System has already been used widely around the world, as it is mentioned before, it is unnecessary or impossible to utilize GPS in each wireless network, especially in some specific environment, for many considerations. However, GPS has defined its own outliers detecting algorithms, which can be implemented in this research, even though it needs to have some modification on the algorithm [5]. In this chapter, the general idea of GPS integrity monitoring method will be introduced, and the specific algorithm for the thesis will be demonstrated and implemented. The simulation result will be compared with the benchmark in the end.

RAIM (Receiver Autonomous Integrity Monitoring) is a technology developed to assess the integrity of Global Positioning System (GPS) signals in a GPS receiver system [6]. It detect fault with the "pseudorange" measurement, which is approximation of distance between a satellite and a navigation satellite receiver. RAIM needs at least 5 visible satellites to finish the positioning. Navigation satellite receiver performs consistency checks within the visible satellites, and provides alert to the satellite if consistency fails to happen. It is worth noting that RAIM availability is an important issue needed to be considered when GPS integrity monitoring algorithm is used. Because of geometry and satellite service maintenance, there are not always 5 visible satellites around

and then RAIM availability is not guaranteed.

This thesis uses the idea of GPS RAIM algorithm to detect malfunctioning anchor. The algorithm used in this thesis focuses to minimize the location error. Whenever the target agent needed to locate itself, it builds the communication with its neighbor anchors, if the number of anchors is less than 3, then the algorithm availability is not ensured, and then the location error will increase.

3.2 Mathematical Deduction of the Algorithm

3.2.1 Computing Residuals

RAIM uses pseudorange measurement to detect fault. The pseudorange measurement can be expressed as:

$$Y = Hx + e \tag{3.1}$$

Where y is pseudorange measurement with size of $n \times 1$, n is the number of visible neighbor anchors. In RAIM, the generic element of y represents the difference between the raw pseudo-range from satellite and the corresponding geometric distance between the linearization point and the computed satellite position. In this thesis, y is the difference between $\hat{d}$ and $\tilde{d}$, i.e.

$$y = \hat{d} - \tilde{d} \tag{3.2}$$

x is 2×1vector representing the change of agent's position after each updating iteration. i.e.

$$x = \begin{bmatrix} x_a^{(t,l)} - x_a^{(t,l-1)} \\ y_a^{(t,l)} - y_a^{(t,l-1)} \end{bmatrix} \tag{3.3}$$

And H is $n \times 2$ linear transformation matrix connecting vector x and vector y. i.e.

$$H = \begin{bmatrix} a_{x_1} & a_{y_1} \\ a_{x_2} & a_{y_2} \\ \vdots & \vdots \\ a_{x_n} & a_{y_n} \end{bmatrix} \tag{3.4}$$

where,

$$a_{x_i} = \frac{x_i - x_a}{\bar{d}_i} \tag{3.5}$$

$$a_{y_i} = \frac{y_i - y_a}{\bar{d}_i} \tag{3.6}$$

The size of vector $\mathbf{y}$ is $n \times 1$, each row of $\boldsymbol{y}$ is related to the anchors respectively, and the malfunction problem can be detected using $\boldsymbol{y}$.

3.2.2 Threshold Value Settings

The algorithm using GPS integrity monitoring method can be divided into two parts. The first part focuses at setting threshold to detect the malfunctioning anchors. In this part, the initial estimated agent position will be set to its true position. Under this assumption, the distribution of elements in vector $\boldsymbol{y}$ can be clearly plot. Using the distribution, the threshold can be fixed. The second part uses the threshold to detect the malfunctioning anchors. As different as the first part, all of the agents and the anchors don't know agent's true position, and the initial estimated agent position is chose randomly in the area around the anchors.

The figures 3.1 show the distribution of the elements in vector $\boldsymbol{y}$. The first one shows the distribution of vecto $\boldsymbol{y}$ when all the anchors are "good anchors", the distribution of $\boldsymbol{y}$ in this figure is marked as y_{good}; the second figure is when all the anchors are "bad anchors", the distribution of y_{bad} in this figure is marked as y_{bad}. The mean value for y_{good} is 0.006m, the standard deviation is 0.1; the mean value for y_{bad} is 30m, and the

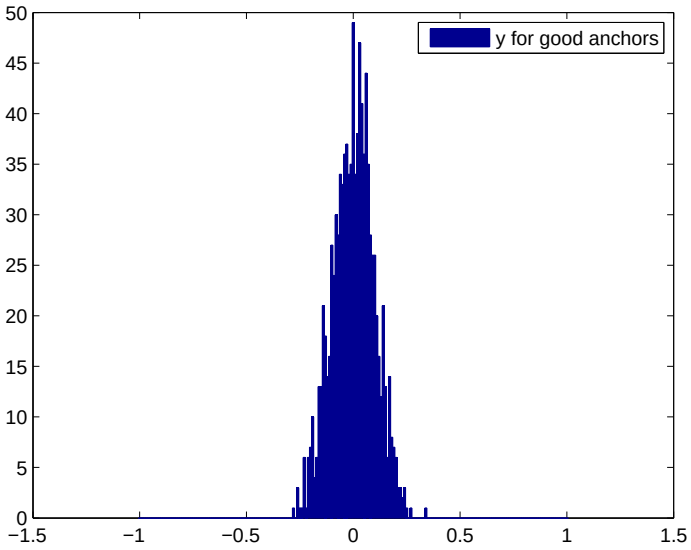


Figure 3.1: The distribution of the elements in vector y when all anchors are working well

standard deviation is 13.4m. Treat these distributions as Gaussian distribution, i.e. , $y_{good} \sim N(0, 0.1)$, $y_{bad} \sim N(30, 13.4)$

Thus the threshold can be set using these distributions, and the decision is made as follows: if the value of the element in y is higher than the threshold, then the corresponding anchor is judged as a bad anchor; otherwise, the corresponding anchor is judged as a good anchor. There are two important probabilities which should be considered. One is miss detection probability and the other one is fault alert probability.

- **Miss Detection Probability**: it is defined as the probability of bad anchors are not detected and treated as good anchors. Obviously, the miss detection probability is a conditional probability of random variable $\boldsymbol{y}$, which can be expressed as

$$P(x_i \, is \, judged \, as \, good \, anchor \, | x_i \in B) \quad = \quad P(y < \gamma | y \in y_{bad}) \tag{3.7}$$

- **Fault Alert Probability**: It is defined as the probability when good anchors are detected as bad anchors. Similarly with the miss detection probability, the

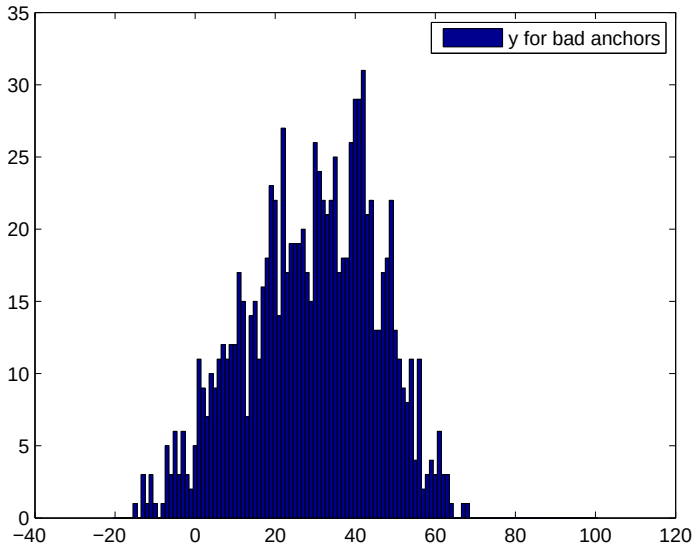


Figure 3.2: the distribution of elements in vector y when all anchors are malfunctioning

fault alert probability can be expressed as:

$$P(x_i\, is\, judged\, as\, bad\, anchor\, |x_i\, is\, a\, good\, anchor) = P(y > threshold|y \in y_{good}) \tag{3.8}$$

Based on the definition, the relationship between the threshold and the two probabilities can be showed as follows:

In this thesis, since the bad anchors will give rise to inaccurate positioning, so it is more essential to control the miss detection probability at a very low level. In this algorithm, the threshold is set as 0.1(m). Under this setting, the miss detection probability is 2%, and the fault alert probability is 10%.

3.3 Algorithm Implementation

As it is mentioned before, in previous step, the initial estimated agent position is assumed as its true position; in this part, the actual situation is simulated when all the anchors and agents don't have the information of agent's true position. The threshold fixed from previous step will be used in this part to detect the malfunctioning anchors.

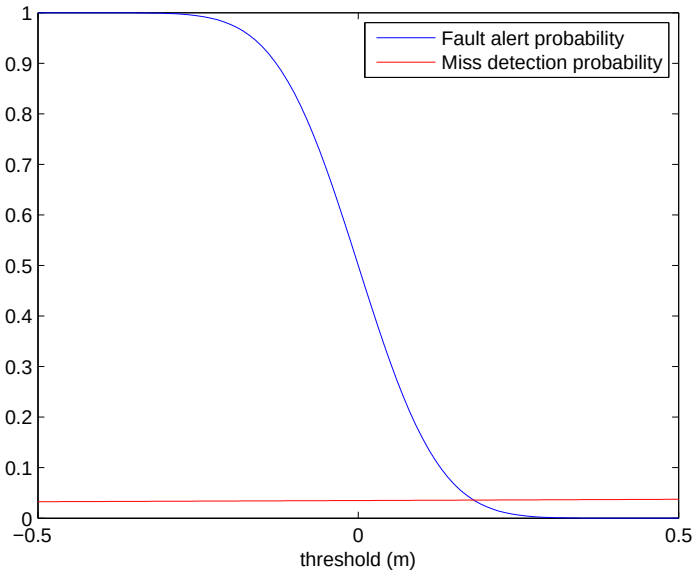


Figure 3.3: the relationship between threshold and the miss detection probability,the fault alert probability

Algorithm 3.1 Localization using GPS method

For l=1 to N_{iter} do {iteration index}
  Find the biggest value y_{big}in $abs(y)$
    If $y_{big} < threshold$ or $N \leq 3$
      go to (*)
    Else
      remove the anchor inaccordance with y_{big}
      N=N-1;
    End if
    * Execute Algorithm 1 to estimate the agent position
End for

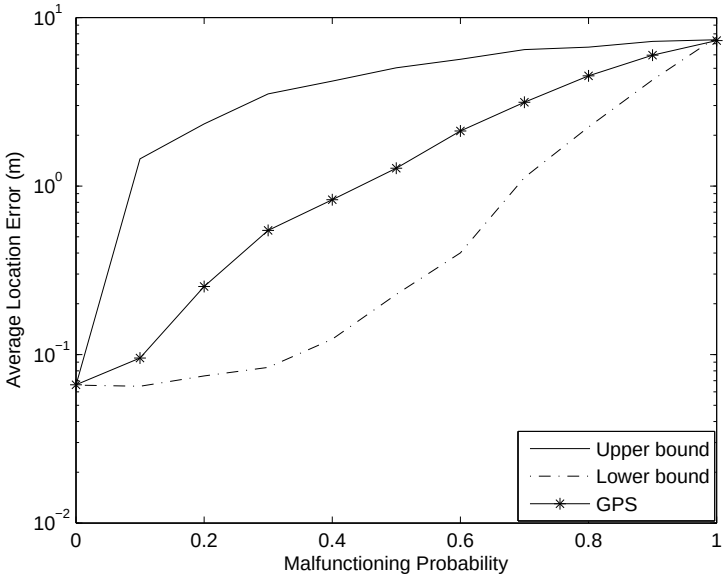


Figure 3.4: The Lower and upper bound of the average location error, the performance of GPS method

3.4 Simulations and Results

Maintaining the same settings as above, repeat the simulations 1000 times, for each malfunctioning probability respectively, we get the performance plots of GPS method as in figure 3.4. We can see from the figure, using GPS integrity monitoring method, we can achieve a good consistency verification performance when malfunctioning probability is 0.1, but when the probability is higher, the performance will be much worse.

Chapter 4

Linear Programming Method

4.1 Introduction

In this chapter, another malfunctioning anchor detection method will be talked about, by using linear programming. Linear programming is one type of optimization methods. There are some researches which proved that linear programming method can detect outliers in wireless network. Here we are going to introduce the concept of linear programming, and implement the linear programming in our simulation environment and have an analysis.

Linear programming is a widely known and used special subclass of convex optimization, which aims at providing the best outcome in a given mathematical models presented as linear equations [7]. More in details, linear programs are problems that can be expressed in canonical form.

$$\begin{aligned} &minimize: \ c^T x \\ &subject\, to: a_i^T x \ \ \leq b_i, \ i = 1, \cdots m \end{aligned} \tag{4.1}$$

where the vectors c, a_1, a_2, $\cdots a_m$ and coefficients $b_1, b_2, \cdots b_m$ are parameters formulated from the problems that specify the objective and constraint functions.

4.2 Mathematical Deduction of the Algorithm

4.2.1 Linearization of the LS Objective

As the introduction above mentioned, linear programming can solve problems presented as linear equations. Hence the first task to do with the algorithm is to formulate the linear representations.

Similar as the other localization-based problems, we can formulate as follows:

$$y = Ax + e \tag{4.2}$$

Where, the vector x is the unknown parameters which include the coordinates of the agent. The matrix $\boldsymbol{A}$ and the vector $\boldsymbol{y}$ are functions of the available measurements and the known parameters such as the coordinates of the anchors.

More in details, assume there are N anchors which are in the communication range of the target agent. According with the mathematical model built in previous chapter, the distance measurement is given by $\hat{d}_j = \|x_j - a\| - n_j$, $j = 1, 2 \cdots N$. Substituting the agent and anchors we get, $\hat{d_j}^2 = \|a\|^2 + \|x_j\|^2 - 2a^T x_j + \epsilon$. Collecting the measurements form N anchors, we get:

$$y = \left[\|x_1\|^2, \cdots \|x_N\|^2\right]^T - \hat{d_j}^2 \tag{4.3}$$

$$A = \left[2\left[x_1, x_2 \cdots x_N\right]^T, 1_N\right] \tag{4.4}$$

$$x = \left[a^T, -\|a\|^2\right]^T \tag{4.5}$$

Where 1_N is an $N \times 1$ vector of ones. With linear equations above, taking measurement noise into account, we get

$$\boldsymbol{y = Ax + e}.$$

4.2.2 Linear Programming

Based on the equations above, the consisting problem can be expressed in two ways:

$$\min_x \|\boldsymbol{Ax} - \boldsymbol{y}\|_0 \tag{4.6}$$

$$\min_x \|\boldsymbol{Ax} - \boldsymbol{y}\|_1 \tag{4.7}$$

where $\|\cdot\|_0$ stands for the $l_0 - norm$ (the number of vector elements that are not zero), and $\|\cdot\|_1$ stands for the $l_1 - norm$ (the sum of absolute values of the vector elements). However, computing minimization of $\|\cdot\|_0$ is NP hard. But fortunately, linear programming can solve the $\|\cdot\|_1$ problem, and it is found that the performance is very good, especially when the vector $\boldsymbol{e}$ is sparse (most entries are zero or "small", and only a few entries are "large"), which is exactly our scenario.

In [8], it has been shown that the maximum number of identical malfunctioning anchors can be found. When anchors are uniformly distributed, the upper bound of identical malfunctioning anchors are $B(N) = \frac{N}{10} + o(N)$; When the anchors are uniform circular distributed, $B(N) = \frac{N}{6}$. Hence, in our scenario, we expect that we can detect bad anchors a fraction of 10%.

4.3 Algorithm Implementation

4.3.1 Linear Programming Implementation

Matlab has its own defined function to solve the optimization problem using linear programming, such as "linprog" in the Matlab Optimization Toolbox.

For instance, if we want to solve $\min_x \|\boldsymbol{Ax} - \boldsymbol{y}\|_1$ problem, the Matlab codes can be written as:

Here we introduce an alternative toolbox "CVX" (Matlab Software for Disciplined Convex Programming), which turns Matlab into a modeling language, allowing con-

```
f = [ zeros(n,1); ones(m,1); ones(m,1) ];
Aeq = [ A, -eye(m), +eye(m) ];
ly = [ -Inf(n,1); zeros(m,1); zeros(m,1) ];
xzz = linprog(f,[],[],Aeq,y,ly,[]);
x_l1 = xzz(1:n,:);
```

Algorithm 4.1 Localization using linear programming

For l=1 to N_{iter} do {iteration index}
    Compute $\boldsymbol{e}$, using linear programming
    Find the entries in $\|\boldsymbol{e}\|$ which are bigger than threshold and remove the anchors accordingly
    Execute Algorithm 1 to estimate the agent position
End for

straints and objectives to be specified using standard Matlab expression syntax. Thus CVX copes with optimization problem fast, direct and clear. For example, the $\min_x \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_1$ problem can be written as follows, using CVX:

```
cvx_begin
variable x(n);
minimize( norm(A*x-y,1) );
cvx_end
```

4.3.2 Malfunctioning Anchors Detection Algorithm

Based on the linear programming method and least square algorithm, the malfunctioning anchors detection algorithm is expressed as algorithm 4.1

4.4 Simulations and Results

Similar as GPS integrity monitoring method, the thresholds should be chosen according to the distribution of entries in vector$\boldsymbol{e}$. However, there is one difference here, when the malfunctioning probability changed, the distribution of $\boldsymbol{e}$'s entries shifted, and the threshold also changed. i.e. the linear equations can be written as

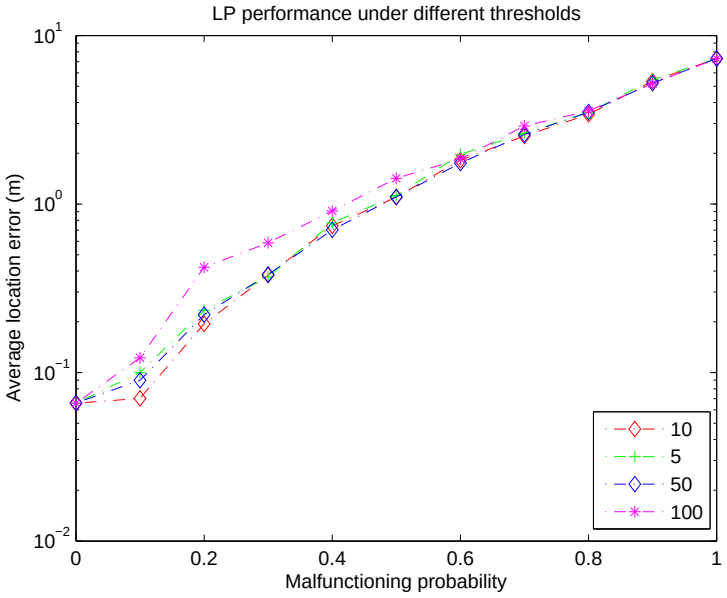


Figure 4.1: Linear programming method performance under different thresholds

$$y = Ax + e + n$$

where n is vector related to the malfunctioning probability.

We have chosen different thresholds to compare the performance, the results are as in figure 4.1.Clearly, when the threshold is 10, best performance can be achieved,especially when the malfunctioning probability is small. Put this result with the GPS method to compare as in figure 4.2. From figure 4.2, it is obvious seen that linear programming gives perfect performance when the malfunctioning probability is 0.1, which proves what has been talked about before that linear programming method can detect one malfunctioning anchor perfectly. When the probability increases, performance will decline.

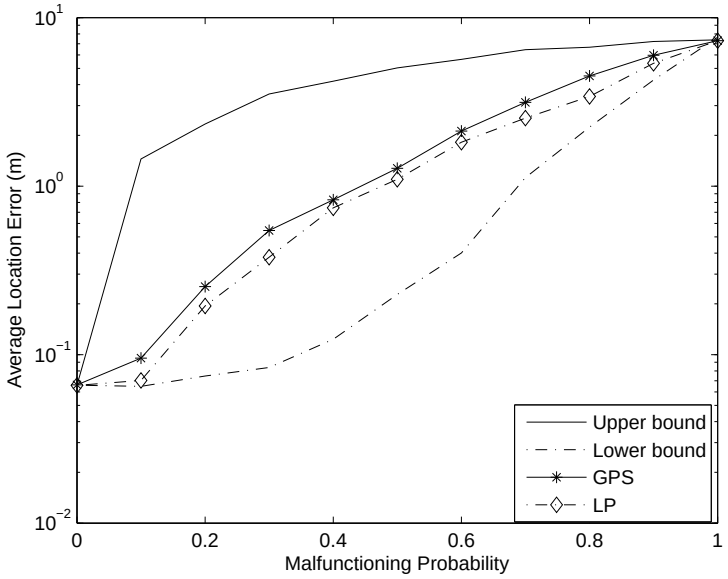


Figure 4.2: Performance comparasion between LP method and GPS method

Chapter 5

Factor Graph Method

5.1 Introduction

In this chapter, we will introduce the concept of the factor graph as well as the sum-product algorithm. Based on those concepts, we will draw the factor graph for this research, and design the algorithm. In the end of this chapter, we will give out the simulation results and have an analysis.

Factor graph is a kind of graphical model which can present algorithms in terms of message passing on a graph. Factor graph operates with passing messages between nodes (function nodes and variable nodes). The origins of factor graphs are used for solving problems in coding theory. In fact, combined with algorithms, factor graph can deal with a large amount of signal processing problems,

In this research, the factor graph will work with Sum-Product Algorithm (or belief propagation algorithm).

5.1.1 Description of the Factor Graph

As mentioned, factor graph operates by passing messages between functions and variables. When there are multiple variables appeared in functions, we need to factorize the function regarding as variables, this process is called "factorization´´ [9].

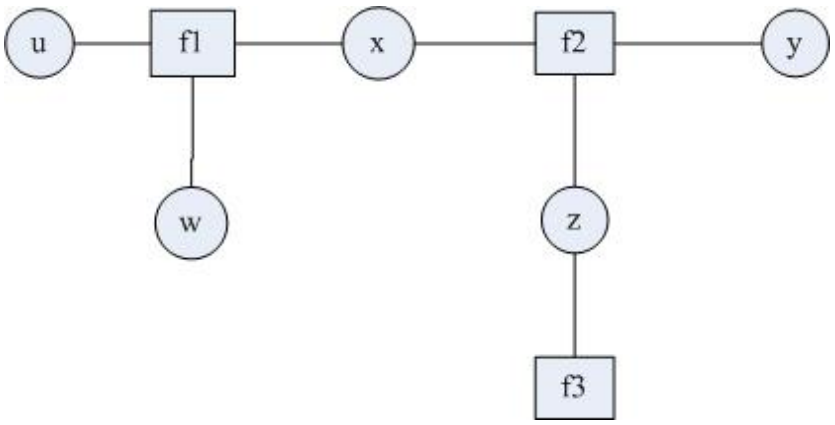


Figure 5.1: An example of factor graph

Take an example for instance. If there is a function $f(u, w, x, y, z) = f_1(u, w, x)f_2(x, y, z)f_3(z)$. Then it can be factorized in factor graph as in figure 5.1. The figure 5.1 shows a simple example of factor graph. In general, one factor graph contains function nodes, variable nodes, and edges connecting nodes. All messages are passed between these nodes through edges. The *factor graph* is a bipartite graph that expresses the structure of the factorization, which is defined under following rules: Each node represents specific function or variable, and unique to each other. Besides, the edge connecting function f and variable x if and only if f is a function of x. In the figure 5.1, f_1, f_2, f_3 are function nodes, and u, v, w, x, y are variable nodes. Sometimes, the functions are called as *local functions*, and their production is called *global function*.

5.1.2 Sum-Product Algorithm

Factor graph is used to compute the marginal distribution. For discrete random variables, the marginal probability mass function can be written as Pr(X = x). This is:

$$Pr(X = x) = \sum_{y} Pr(X = x, Y = y) \tag{5.1}$$

Similarly for continuous random variables, the marginal probability density function can be written as $p_X(x)$, as in 5.2

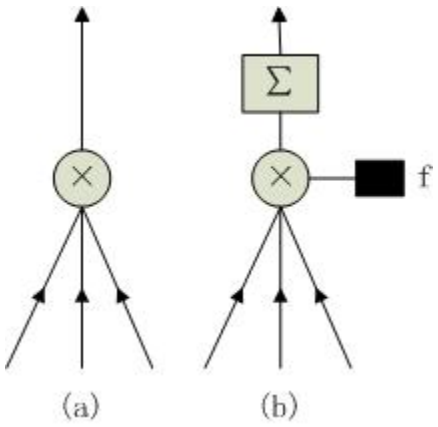


Figure 5.2: Local messages passing to express a marginal function at (a) a variable node and (b) a factor node

$$p_X(x) = \int_y p_{X,Y}(x, y)dy \tag{5.2}$$

Such computations can be finished efficiently by applying sum-product algorithm, which is shown in figure 5.2.

Let $\mu_{x\to f}(x)$ denote the message sent from node x to node f in the operation of the sum-product algorithm, let $\mu_{f\to x}(x)$ denote the message sent from node f to node x. Also, let $n(v)$ denote the set of neighbors of a given node v in a factor graph. Then, the message computations performed by the sum-product algorithm may be expressed as follows:

variable to function:

$$\mu_{x\to f} = \prod_{h\in n(x)\setminus f} \mu_{h\to x}(x) \tag{5.3}$$

function to variable:

$$\mu_{f\to x}(x) = \sum_{\sim\{x\}} \Big(f(\mathbf{X}) \prod_{h\in n(x)\setminus f} \mu_{y\to f}(y) \Big) \tag{5.4}$$

where $X = n(f)$ is the set of arguments of the function f [10].

5.2 Mathematical Deduction of the Algorithm

Based the same notations as in previous chapters, consider the posterior probability $p(\boldsymbol{\theta}|\boldsymbol{y})$, which presents the conditional probability of the position vecotr $\theta = [a, b_1, \cdots b_{10}]$ based on the measurement vector $y = [\hat{d}_1 \cdots \hat{d}_{10}]$, where b_i represents the state of the anchors, which include two state: malfunctioning (written as "good´´) or not malfunctioning (written as "bad´´). The locationing is based on the analysis of this posterior probability. Using factor graph, the posterior probability can be factorized so as to be analyzed.

According to Bayes' theorem, we have:

$$P(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{\theta}) \cdot p(\boldsymbol{\theta})}{p(\boldsymbol{y})} \propto p(\boldsymbol{y}|\boldsymbol{\theta}) \cdot p(\boldsymbol{\theta}) \tag{5.5}$$

That is to say, *posterior probability* $\propto$ *likelihood* $\times$ *prior probability*, where the prior probability stands for $p(\theta)$. Due to the positions of agent and all anchors are independent, so we have:

$$p(\theta) = p(a) \cdot \prod_{i=1}^{10} p(b_i) \tag{5.6}$$

Similar, due to the independence, The likelihood $p(y|\theta)$ can be written as:

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \prod_{i=1}^{10} p(\hat{d}_i|x_i, b_i, a_i) \tag{5.7}$$

Thus, the posterior probability can be "factorized´´ as follows:

$$\begin{aligned} P(\boldsymbol{\theta}|\boldsymbol{y}) &= [\prod_{i=1}^{10} p(\hat{d}_i|x_i, b_i, a_i)] \cdot [p(a) \cdot \prod_{i=1}^{10} p(b_i)] && (5.8) \\ &= \prod_{i=1}^{10} [p(\hat{d}_i|x_i, b_i, a_i) \cdot p(b_i)] \cdot p(a) && (5.9) \end{aligned}$$

When it comes to the details, the prior probability $p(b_i)$ includes two part infor-

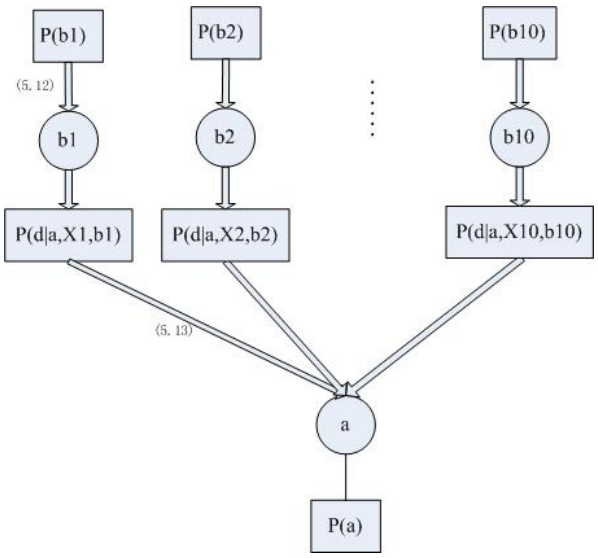


Figure 5.3: the factor graph for 1 agent and 10 anchors

mation, it can be presented as a matrix $\begin{bmatrix} prob \\ 1-prob \end{bmatrix}$, where $prob$ means the malfunctioning probability. "$1-prob$´´ means the probability where the anchors are working well. When the anchor is malfunctioning, as we mentioned in previous sections, the measurement errors apply a uniform distribution $U(0, R_{max})$, where R_{max} means the maximum communication range. Therefore, according to unifoirm distribution, the posterior probability can be written as:

$$p(\hat{d}_i|x_i, a_i, b_i = bad) = \frac{1}{R_{max}} \tag{5.10}$$

When the anchor is working well, then the measurement noise applies the gaussian distribution $N(\|a - x_i\|, \sigma^2_{good})$, where$\|a - x_i\|$ represents the true distance between the agent and anchors, σ_{good} presents deviation, which equals to 0.1m. Therefore, we have:

$$p(\hat{d}_i|x_i, a_i, b_i = good) = \frac{1}{\sqrt{2\pi\sigma^2_{good}}} e^{-\frac{(\hat{d}_i - \|a-x_i\|)^2}{2\sigma^2_{good}}}. \tag{5.11}$$

Based on the mathematical deduction of the problem, we can factorize the posterior probability into factors regarding as certain variables. Therefore, we can put all of these factors and variables into the factor graph, as figure 5.3 shows.

5.3 Algorithm Implementation

The sum-product algorithm starts from the leave nodes at the top, and the messages pass from top to down from anchor $x_1$to anchor x_{10}. Finally, all the messages converge at the node a.

First step, the message from $p(b)$ to b_i can be written as:

$$\mu_{p(b_i)\to b_i} = \begin{bmatrix} prob \\ 1 - prob \end{bmatrix} \tag{5.12}$$

where *prob* is the malfunctioning probability.

Second step, the message from node $p(\hat{d}_i|x_i, b_i, a)$ to node a is as follows:

$$\mu_{p(\hat{d}_i|x_i,b_i,a)\to a} = \sum_{b_i\in\{good,bad\}} \mu_{p(b_i)\to b_i} \times \Phi \tag{5.13}$$

where,

$$\Phi = p(\hat{d}_i|x_i, b_i, a) \tag{5.14}$$

$$= \left\{ \begin{array}{cc} \frac{1}{\sqrt{2\pi\sigma^2_{good}}} e^{-\frac{(\hat{d}_i - \|a-x_i\|)^2}{2\sigma^2_{good}}}, & when\ the\ anchor\ is\ working\ well \\ \frac{1}{R_{max}} & when\ the\ anchor\ is\ malfunctioning \end{array} \right\} \tag{5.15}$$

Third step, compute the message from node a to $p(a)$:

$$\mu_{a\to p(a)} = \prod_{i=1}^{10} \mu_{p(\hat{d}_i|x_i,a_i)\to a} \tag{5.16}$$

Finally, we can get that the product of forward messages (from a to $p(a)$) and backward messages (from $p(a)$ to a) is :

Algorithm 5.1 Factor graph method

Initialized measurements:

{measure the distance between a and x_j, $j = 1, 2 \cdots 10$}

From $j = 1 to 10$ in parellel

compute the forward information passed from x_j to a.

$$\mu_{x_j \to a}(a) = \sum_{b_i \in \{good, bad\}} \mu_{p(b_i) \to b_i} \times \Phi$$

where,

$$\Phi = \left\{ \begin{array}{cc} \frac{1}{\sqrt{2\pi\sigma^2_{good}}} e^{-\frac{(\hat{d}_i - \|a - x_i\|)^2}{2\sigma^2_{good}}}, & when\ the\ anchor\ is\ working\ well \\ \frac{1}{R_{max}} & when\ the\ anchor\ is\ malfunctioning \end{array} \right\}$$

compute the information passed from a to $p(a)$

$$\mu_{a \to p(a)} = \prod_{i=1}^{10} \mu_{x_j \to a}$$

estimate position of a

$$\hat{a} = argmax\{p(a_i) \times \mu_{a \to p(a)}\}$$

$$\begin{aligned} f &= \mu_{p(a) \to a} \times \mu_{a \to p(a)} && (5.17) \\ &= p(a) \times \mu_{a \to p(a)} && (5.18) \end{aligned}$$

In order to find the estimated agent position, we need to find the maximum value of f, thus the $\hat{x}$ accordingly will be the estimated agent position.i.e.

$$\begin{aligned} \hat{a} &= argmax\{f\} && (5.19) \\ &= argmax\{p(a) \times \mu_{a \to p(a)}\} && (5.20) \end{aligned}$$

Written as algorithm, it will be as Algorithm 5.1:

It is worth noting that there are multiple local maximum value of f, when implement

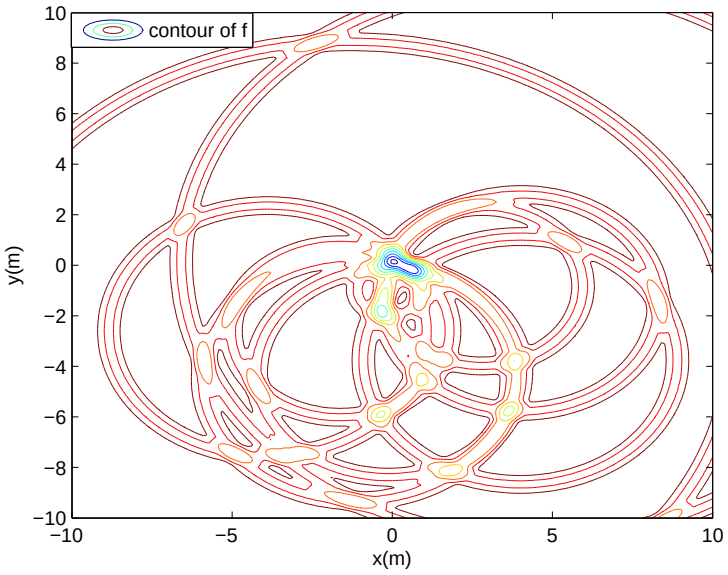


Figure 5.4: The contour of the value of f

the searching in MATLAB, it can't work out well using function $fminsearch$ directly, because $fminsearch$ only gives out local solutions. For instance, the contour figure of " $-f$ '' is showed as figure 5.4.

As the Figure 5.4 showed, every interception points represent the possible agent position. They are all local minimum, thus there are only one true agent position. The global minimum value of $-f$ decides the correct estimated agent position. In this figure, the malfunctioning probability is set as 0.2, after implementing the algorithm, the estimated position is [0.1, 0.1], which has only 0.14m estimation error.

5.4 Simulation and Results

We have set two simulation cases here, one is when the agent knows exactly the anchor malfunctioning probability, the other case is when the agent does not know the probability, then it assumes the malfunctioning probability as a certain value (as 0.1 for example). The performance curves related to these 2 cases will be put out together.

Maintaining the same settings as in previous chapters, repeat the simulations 1000 times, for each malfunctioning probability respectively, we get the performance plots

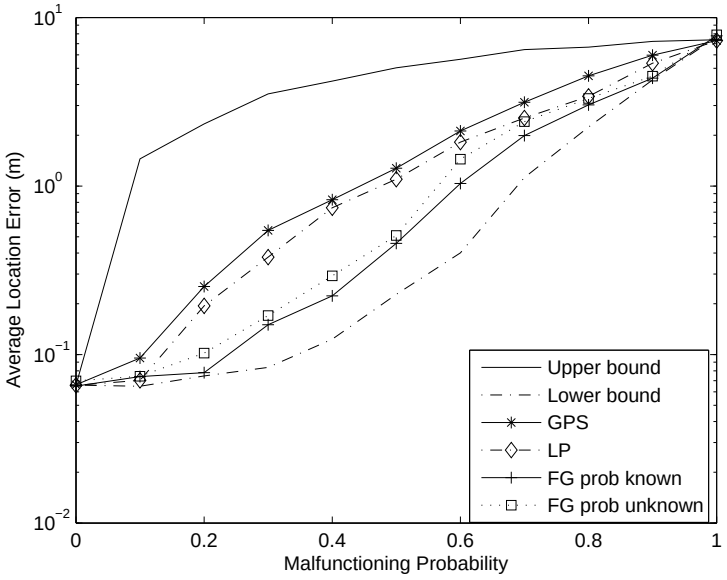


Figure 5.5: The performance of factor graph method and previous methods

of factor graph as figure 5.5. As to compare the performance with GPS integrity monitoring method and Linear Programming method, we put the performance curves together:

From figure 5.5, we can see that using factor graph method when the agent knows exactly the malfunctioning probability, it can achieve the best performance, when the agent does not know the malfunctioning probability, and assume it as 0.1, the performance will be slightly worse than when probability is known, but still is better than using GPS and LP method.

Chapter 6

Factor Graph Method with Cooperation

6.1 Introduction

In this chapter, we will continually have a research on the factor graph, but different as in chapter 5, we will introduce a new wireless communication scenario which is called cooperative communication. We will draw a new factor graph based on cooperative communication network and implement the algorithm to detect the malfunctioning anchors. Also in the end, the simulation results will be given out and compared with all others.

According with the development of wireless communication, a fairly new concept has been brought in, which is cooperative communication technology. Different as traditional non-cooperative communication, it relies on the communication between agents, rather than through a fixed infrastructure. Nowadays, cooperative communication technology has been applied a lot, such as Bluetooth and Zigbee. It can be expected that this new technology will be used over the next few years.

6.2 Algorithm Implementation

It can be seen from previous research results, factor graph method can achieve the best performance compared with GPS integrity monitoring and linear programming method. One of the most important reasons for this is that in factor graph, no anchors are removed during the localization. On the other hand, when GPS and Linear programming method are implemented, thresholds are used to remove the possible malfunctioning anchors. Due to the miss detection and fault alert problems, there are some "good anchors´´ treated as malfunctioning, while some "bad anchors´´ treated as working well. Factor graph method used all of the information without threshold, so it can achieve the best information.

In this section of cooperation, we will apply the factor graph method continually, based on the mathematical deduction of the last section. Only one thing different is that we change the non-cooperative network into the one with cooperation, so the new factor graph will be given out.

6.2.1 Cooperation Build-up and Factor Graph

The concept of cooperation is based on the direct communication between agents. Therefore, at least two agents are needed in the network to ensure the cooperation. For simplicity, we set this certain wireless network, which contains two agents and ten anchors. Similar as in the non-cooperative part, we mark the agents as $a_1$and a_2, and mark the agents as x_i,$i = 1, 2 \cdots 10$. Both of the two agents can communicate with those ten anchors, they give the measurements of the anchors separately and share the information to locate themselves.

Regarding as the factor graph, there is no much difference compared as the non-cooperative part, except one more agent is added. The factor graph of cooperative part is drawn as figure 6.1.

As the Figure 6.1 on page 44 shows, two agents a_1 and a_2 are connecting with 10 anchors, the shadowed boxes in the factor graph represent the prior probability

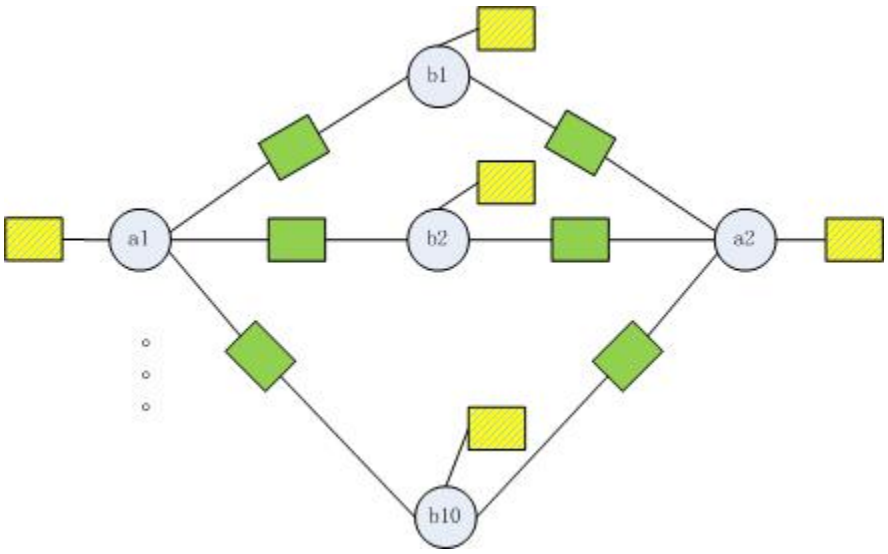


Figure 6.1: Factor graph for cooperative localization

$p(b_i)$,$i = 1, 2 \cdots 10$ and $p(a_1)$, $p(a_2)$. The other boxes represent the posterior probability $p(\hat{d}_{i,j}|a_i, x_j, b_j)$, where $\hat{d}_{i,j}$ means the measured distance between agent a_i and anchor x_j, estimated by agent a_i.

6.2.2 Algorithm Design

Based on the mathematical deduction of non-cooperative part, we design the algorithm as follows:

In Algorithm 6.1, the first iteration is exactly the same thing with non-cooperative localization. No information is shared between two agents. From the second iteration, agent x_1 update $\mu_{x_j \to a_1}$with $\mu_{a_2 \to x_j}$. That is to say, in the first iteration, the only message passed out from x_j is the prior probability $p(x_j)$, but from the second iteration, there added more information besides the prior probability, which agents to localize more accurately.

Algorithm 6.1 Cooperative localization

Initialization

{measure the distance between a_i and x_j,$i = 1, 2$, $j = 1, 2 \cdots 10$}

For Iter=1 to Niter (iteration index)

from $j = 1 to 10$ in parallel

compute the forward information passed from x_j to a_i.

$$\mu_{x_j \to a_i}(a_i) = \sum_{b_i \in \{good, bad\}} \mu_{p(b_i) \to b_i} \times \Phi$$

where,

$$\Phi = \left\{ \begin{array}{cc} \frac{1}{\sqrt{2\pi\sigma^2_{good}}} e^{-\frac{(\hat{d}_i - \|a - x_i\|)^2}{2\sigma^2_{good}}} & , when\ the\ anchor\ is\ working\ well \\ \frac{1}{R_{max}} & when\ the\ anchor\ is\ malfunctioning \end{array} \right\}$$

and

$$\mu_{p(b_i) \to b_i} = \left[\begin{array}{c} prob \\ 1 - prob \end{array} \right]$$

compute the information passed from a_i to $p(a_i)$

$$\mu_{a_i \to p(a_i)} = \prod_{i=1}^{10} \mu_{x_j \to a_i}$$

estimate the position of $a_i, i = 1, 2$

$$\hat{a}_i = argmax\{p(a_i) \times \mu_{a_i \to p(a_i)}\}$$

compute the backward information passed from a_i to x_j.

$$\mu_{a_i \to x_j}(x_j) = p(a_i) \times \prod_{k \neq j, k \in \{1,2,\cdots 10\}}^{9} \mu_{x_j \to a_i}(a_i)$$

updating by sharing information from the other agent:

for a_1, update the information from x_j to a_1 with information from a_2 to x_j

for a_2, update the information from x_j to a_2 with information from a_1 to x_j

end Iter

6.3 Simulation and Results

6.3.1 Simulation Environment Settings

The major part of the simulation environment for cooperation will be the same as the one for non-cooperation, except that there are 2 agents here and localization iteration is brought in.

agent distribution: uniform distribution in a $10(m) \times 10(m)$ grid

agent number: 2

anchors numbers: 10

The maximum communication range between the agents and the anchors:20m

measurement noise distribution when anchors are working well: $n \sim N(0, 0.01^2)$

measurement noise distribution when anchors are malfunctioning: $n \sim U(0, 20^2)$

agent localization iteration:10

Searching resolution: 0.05m

6.3.2 Simulation Results

Maintain the settings above and repeat the simulation for 1000 times, we get the result for cooperative factor graph method as in figure 6.2. From this figure, it can be seen that using factor graph in a cooperative network can achieve the best performance, it is very close to the optimal curve (the lower bound). However, it is worth noting that the implementation of factor graph in cooperative network takes a huge time. One reason for this is that there are 10 iterations here, in each iteration, the algorithm will search the whole map. Actually, from 3 or 4 iteration, the result has already converged and achieved its optimal value.

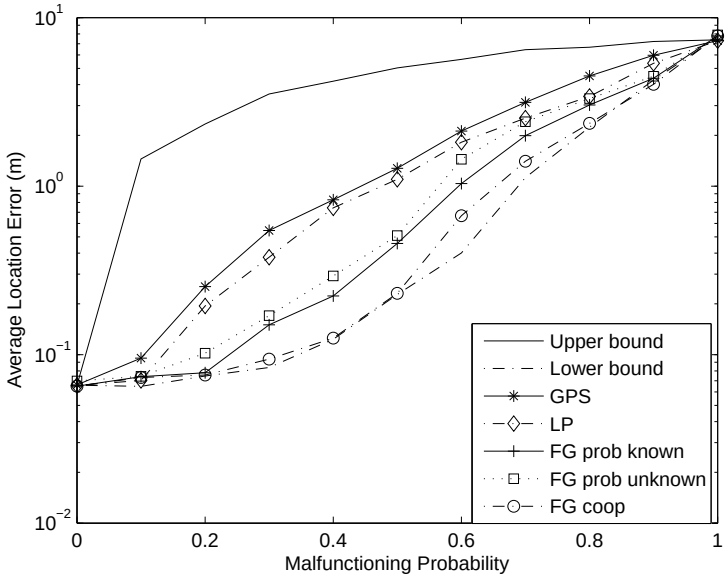


Figure 6.2: The results for cooperative factor graph method and all other methods

Chapter 7

Conclusions and Future Work

In this thesis, we have investigated consistency verification methods in wireless networks. We have designed and implemented three types of algorithms: GPS integrity monitoring, linear programming method and factor graph method.

Firstly, we implement these algorithms in a wireless network without cooperation. The first two algorithms are implemented with least square. From the results given by simulations, we found that GPS and LP method can achieve a profound performance with some limitations. In other words, when the malfunctioning problem is not severe and the malfunctioning probability is low, then GPS and LP can work well, especially for LP method. However, when the probability is equal or higher than 0.2, the performances of these two methods will get much worse. One of the reasons related to this is when we implement with least square algorithm, we always remove the possible bad anchors by setting a threshold. The thresholds are set according to the distribution of the residuals, the miss detection and fault alert problems will happen when using threshold, which will decline the performance. Besides, for linear programming method, there is an upper bound for the detectable malfunctioning anchors. In our research, if there are 10 anchors, the upper bound is 1, which proves why the LP performance is perfect when the malfunctioning probability is less than 0.1. The factor graph method is proved to have the best performance. When we implemented this algorithm using factor graph and sum-product algorithm, we did not set a threshold, all of the anchor information

is used, and thus there won't be miss detection and fault alert problems.

Secondly, we considered a wireless network with cooperative communication. Since the factor graph method can achieve the best performance in non-cooperative network, we use this algorithm with cooperation. The result shows that factor graph method using cooperation is better than the one with no cooperative communication. The reason for this is iterations are set within the algorithm in this part, in each iteration, the prior probabilities of anchors are updated, and two agents share the information of the anchors to localize themselves. It is also worth noting that iterations take a huge time, the simulation time in this part is much longer than in previous algorithms.

In the future, there is still much work to do with the consistency verification. There could be more practical methods which can be utilized to detect the outliers in a network, and more optimization algorithms could be applicable. Besides, there is still space to improve the performance when cooperation is used in practice. For instance, in the cooperation scenario of this thesis, we assume that the two agents can both communicate with ten anchors. In practice, different agents may communicate with different anchors, and the agents can share all these information to each other. In other words, we can improve the number of anchors used for localization.

Bibliography

[1] N. Patwari, J. Ash, S. Kyperountas, A. Hero Iii, R. Moses, and N. Correal, "Locating the nodes: cooperative localization in wireless sensor networks," *Signal Processing Magazine, IEEE*, vol. 22, no. 4, pp. 54–69, 2005.

[2] S. Lanzisera and K. Pister, "Burst Mode Two-Way Ranging with Cramer-Rao Bound Noise Performance," in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pp. 1–5, IEEE, 2008.

[3] H. Wymeersch, J. Lien, and M. Win, "Cooperative localization in wireless networks," *Proceedings of the IEEE*, vol. 97, no. 2, pp. 427–450, 2009.

[4] A. Srinivasan and J. Wu, "A survey on secure localization in wireless sensor networks," *Encyclopedia of Wireless and Mobile communications*, 2007.

[5] A. Brown and M. Sturza, "The effect of geometry on integrity monitoring performance," in *Proceedings of the ION 46th Annual Meeting June*, pp. 26–28, 1990.

[6] J. Liu, M. Lu, Z. Feng, and J. Wang, "GPS RAIM: Statistics Based Improvement on the Calculation of Threshold and Horizontal Protection Radius," in *International Symposium on GPS/GNSS*, pp. 8–10, Citeseer, 2005.

[7] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge Univ Pr, 2004.

[8] J. Picard and A. Weiss, "Accurate Geolocation in the Presence of Outliers using Linear Programming," in *17th European Signal Processing Conference (EUSIPCO, Glasgow, Scotland*, 2009.

[9] C. Bishop and S. O. service), *Pattern recognition and machine learning*, vol. 4. Springer New York, 2006.

[10] F. Kschischang, B. Frey, and H. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 498–519, 2001.