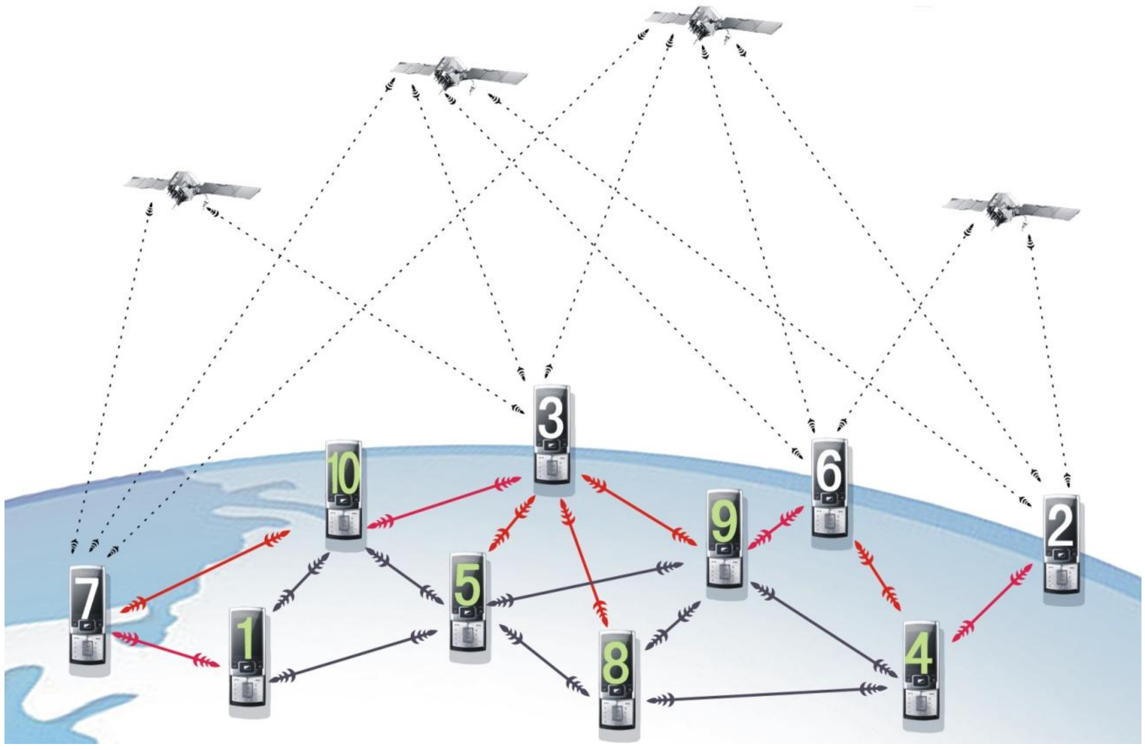


# Message Representation and Updates for Cooperative Positioning

*Master of Science Thesis [in the Master Degree Programme, Communication Engineering]*

WENJING LI

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2010
Report No.

REPORT NO.

**English Title**

# Message Representation and Updates for Cooperative Positioning

**Author, Program and Year**
Wenjing 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

This report is the result of a Master Thesis work done at Department of Signals and systems, Chalmers University of Technology.

## Acknowledgements

My deepest gratitude goes first and foremost to my supervisor Henk Wymeersch, who has been offering me constant guidance through all the stages of this thesis work. Not only his illuminating instructions have many times helped me overcome many difficulties, but also his rigorous academic attitude has greatly taught me how to do research. I could not have finished this thesis work to its present form without his supervision.

My thanks would also go to my beloved parents and my boyfriend for their supporting, caring, and trusting in me all through these years. I would also show my thanks to my classmates who have spent their time in listening to me and offered me with many valuable suggestions.

Last but not least, my thanksgiving shall be given to the Lord who provides me with his strength and ever-present help.

## Abstract

The work of the project was to implement an algorithm of message representation and updates for cooperative positioning which is an emerging technique for localization in wireless networks. The algorithm indicates in what form and manner the messages are exchanged between devices as well as how the nodes update their locations using the received information to. The thesis focuses on a parametric message representation, that is, each message is represented as a set of parameters. In this algorithm, the set of parameters for each message is in the form of a six-dimensional vector. The algorithm is implemented in MATLAB, and the results have shown that our parametric message representation is more computational efficient compared to non-parametric message representations. Also, by applying cooperative positioning, we achieve better performance in terms of accuracy of location estimation, more coverage, and flexibility in different environment.

Keywords: Cooperative Positioning, Message Representation, Parametric.

# Table of Contents

# 1. Introduction

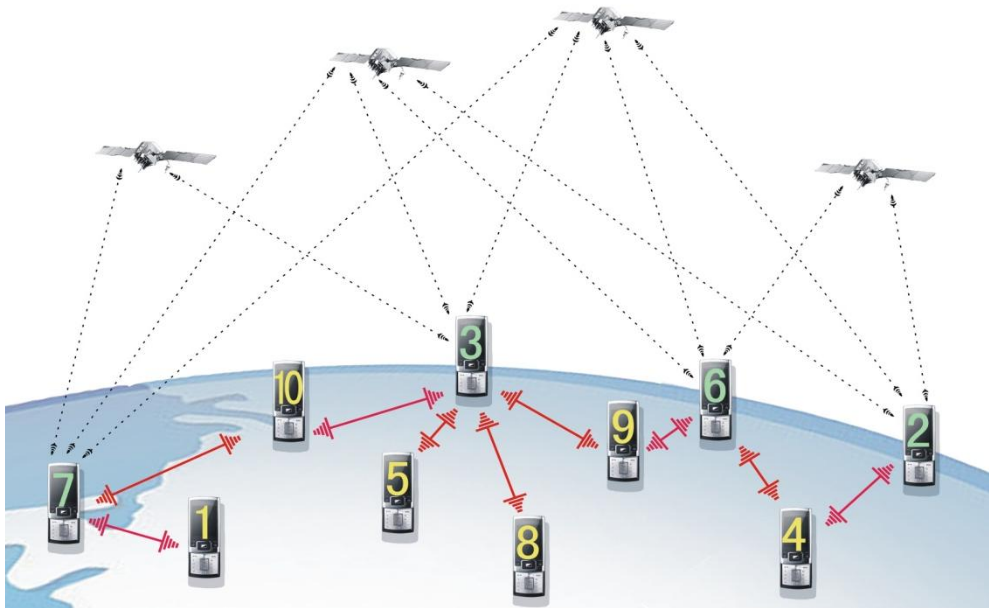
## 1.1 Research background and significance

Automatic localization has been a critical demand for the effective use of wireless networks due to the trend towards globalization and mobility in today's telecommunication industry. New forms of services based on location-awareness are considered as the potential market drivers. Some promising applications are: vehicle navigation, military target tracking, health care monitoring, etc.

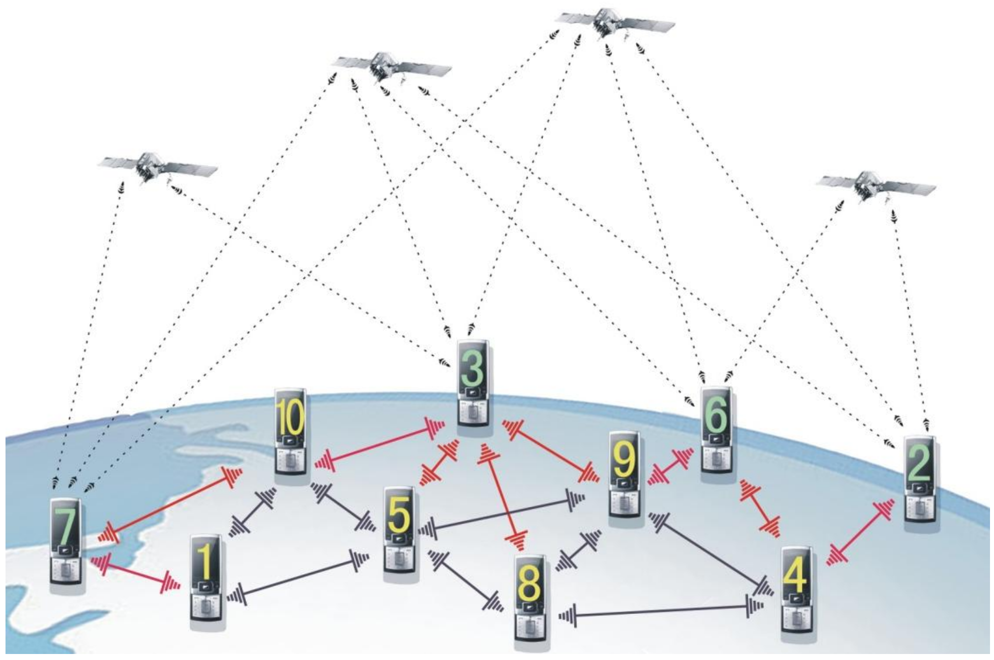
The Global Positioning System (GPS) has already provided a reliable positioning solution for users worldwide regardless of weather, darkness or time. It has been applied in many military and civilian applications, as GPS receivers are now integrated in many 3G mobile phones. However, equipping every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

Therefore, in many conventional localization systems, there are a limited number of reference nodes, called anchors, which have prior knowledge about their positions (e.g., equipped with GPS receivers). The devices with their positions unknown, called agents, establish communication links to their surrounding reference nodes, exploiting information (e.g., reference node's coordinates and distance measurements to them) to compute and determine their own positions. Generally, to infer an agent's own position in a two-dimensional scenario, at least three anchors are needed within its communication range. Hence, the performance of positioning depends to a large extent on the network connectivity, which is ensured by either long-range communication or a sufficient number of reference nodes distributed throughout the network.

Cooperative positioning technique is emerging to overcome this reliance on the coverage of reference nodes by introducing a peer-to-peer (P2P) network model on top of the cellular network model [2]. That is, additionally allowing connectivity between pairs of agents within communication range to improve performance in both accuracy and coverage of the localization system. Fig.1.1 (a) shows an example of the non-cooperative positioning network and (b) is the same network after apply cooperative positioning.



(a) Non-Cooperative



(b) Cooperative

Figure 1.1 Comparison between Non-Cooperative and Cooperative Localization

The nodes marked in green (Node 2, 3, 6, 7) are anchors and the ones marked in yellow (Node 1, 4, 5, 8, 9, 10) are agents. Observe that by allowing cooperation between agents, the network connectivity has vastly improved.

One key research topic in cooperative positioning is to indicate what type of information should be exchanged between devices (point estimates, distributions, regions, etc) and how this information should be represented (e.g., quantization, parametric vs non-parametric representation, curves describing regions). Usually, a good message representation needs to accurately describe the characteristics of the real information, but also to facilitate the computational complexity during processing. This thesis deals with message representation of SPAWN (sum-product algorithm over a wireless network) [7], where nodes exchange distributions of their positions. A detailed description of SPAWN will be introduced in Chapter 2.

## 1.2 State of the art

Before going straight to the point of message representations, it is useful to state the classification of location algorithms based on their output formats. There are mainly two types, namely point-based and area-based localization.

Point-based estimation has been applied as a traditional method in localization for previous Wireless Local Area Networks (WLAN) [3]. This type of localization returns result as a single point, which can be obtained through probabilistic approaches. One example is applying a maximum likelihood estimator to get the single point with highest probability [4].

Area-based localization represents the possible position of agent as an area or distribution. This brings in a critical advantage over the point-based estimation, which is, enabling users of the knowledge about positioning uncertainties and providing them with meaningful alternatives. In our framework SPAWN, the messages passing between devices are described as certain distributions. SPAWN also describes how messages are computed.

## 1.3 Problem Statement

Our task is to approximate the distributions of agents' locations based on the received information containing other node's location distributions and distance measurements. Example: as shown in Fig.1.2, the agent received messages from three anchors. Given an anchor's position and estimated distance to it, the agent's possible location can be narrowed down to a ring region, where the width of the ring is proportional to the ranging uncertainty.

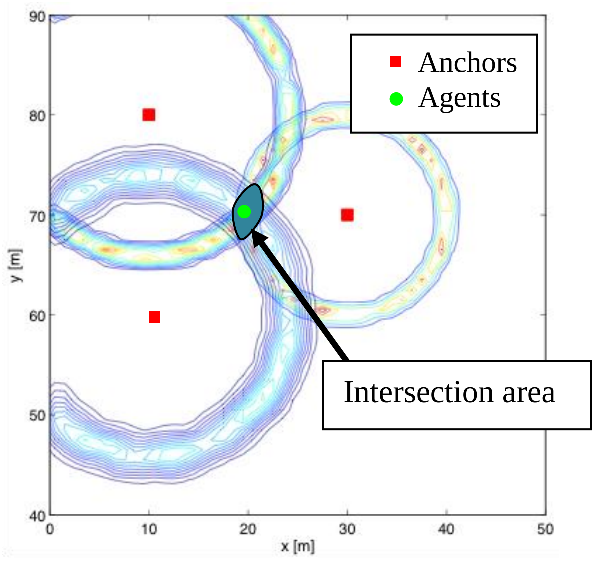


Figure 1.2 An Example of the Localization in SPAWN

Observe that by communicating to these three anchors, the agent can be localized. Moreover, we return its distribution over the intersection area as shown in Fig.1.2.

## 1.4 System model

In this thesis work, the system model is as follows: There are two types of nodes in a wireless network: anchors, which are the reference nodes with known positions, and agents, which have unknown positions and iteratively update their estimated positions. In non-cooperative positioning, agents receive messages only from anchors within their communication range. In cooperative networks, agents get updating information from other agents as well as from anchors.

The localization algorithm is range-based. There are several popular ranging methods such as received signal strength (RSS), time-of-arrival (TOA), time difference of arrival

(TDOA), round-trip time of arrival (RTOA), etc. In particular, we consider round-trip time of arrival (RTOA) range measurement as it can effectively reduce the effect of clock synchronization error between the devices [9]. In this method, the transmitter measures the time of signal travels to the receiver and come back based on its local time, the time offset between transmitter and receiver can thus be ignored.

We denote by $x_i$ the position of node $i$ in the network and by $\hat{x}_i$ its estimated position. $S_{\rightarrow i}$ is the set of nodes that send messages to node $i$. Based on the signal received from node $j \in S_{\rightarrow i}$, node $i$ can then estimate its distance to node $j$ as (see Fig.1.3)

$$\hat{d}_{j\rightarrow i} = \|x_i - x_j\| + n_{j\rightarrow i}, \tag{1.1}$$

where $n_{j\rightarrow i}$ is the measurement noise and we assume $n_{j\rightarrow i} \sim N(0, \sigma^2_{j\rightarrow i})$.

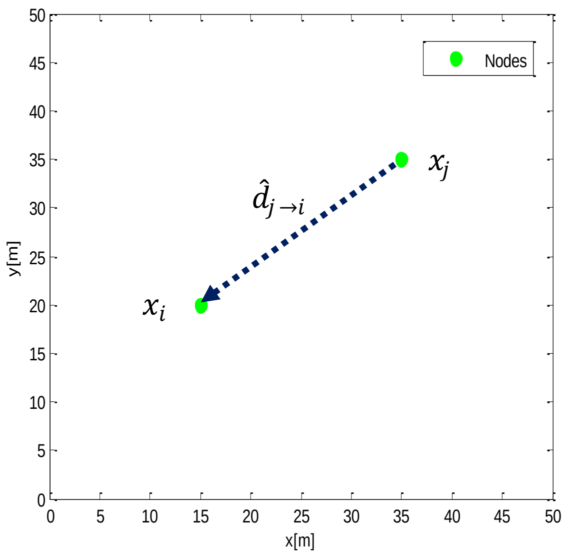


Figure 1.3 Range Measurement Model

## 1.5 Outline of the thesis report

The rest of this thesis report is organized as follows. We elaborate SPAWN in the next chapter and further describe the objectives of this thesis work. In Chapter 3, we apply parametric message representations to non-cooperative positioning, focusing on the initialization and optimization problems of the estimated distributions. After finding a suitable way to define the set of parameters and operations by which information is processed, we then extend the solution to cooperative positioning model in Chapter 4.

Simulation work is done in Chapter 5 and results are analyzed. Chapter 6 summarizes the whole thesis, presents our conclusion and gives some suggestion for future work.

# 2. Localization Framework (SPAWN)

The thesis work is built in a basic localization algorithm called SPAWN (sum-product algorithm over a wireless network) introduced in [7]. This algorithm is fully distributed, which means in contrast to centralized localization where a central processor collects and computes information for all agents, each agent infers its own position only based on the local information. The sum-product algorithm (or belief propagation) is a popular message passing algorithm operating on factor graphs. Therefore, we will first introduce the basic concepts of factor graphs and the sum-product algorithm in this chapter.

## 2.1 Factor Graphs

A factor graph is a diagram that represents a factorization of a function or a distribution. Consider a function $f(X_1,X_2,X_3,X_4)$ that can be factorized as follows:

$$f(X_1,X_2,X_3,X_4) = f_A(X_1)f_B(X_1,X_2)f_C(X_1,X_2)f_D(X_1,X_3,X_4), \qquad (2.1)$$

where $X = \{X_1,X_2,X_3,X_4\}$ are variables and $f = \{f_A,f_B,f_C,f_D\}$ are factors. We can create a factor graph as shown in Fig.2.1 (a). If we merge $f_B(X_1,X_2)$ and $f_C(X_1,X_2)$ into a new factor $f_{BC'}(X_1,X_2)$, the corresponding factor graph is shown in Fig.2.1 (b) .

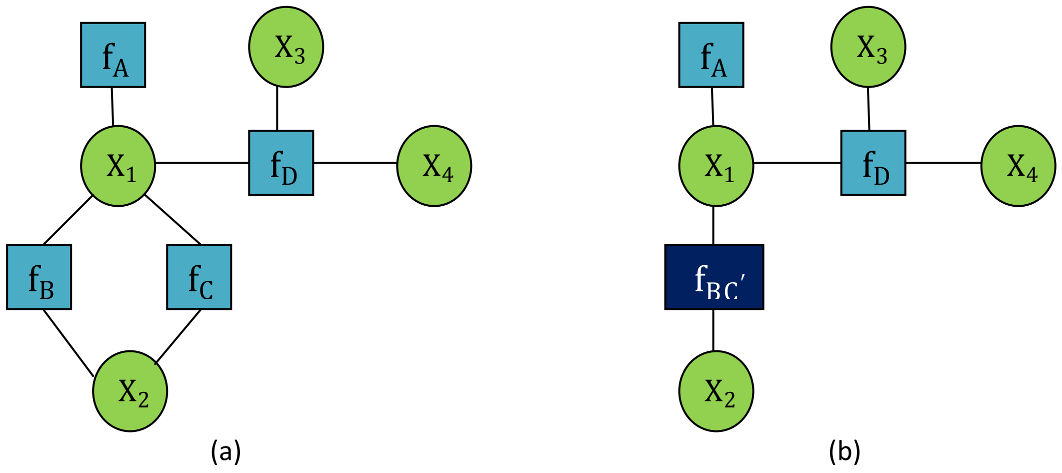


Figure 2.1 Factor Graphs of $f(X_1,X_2,X_3,X_4)$

Observe that factor graph (a) has a cycle while (b) does not. This is an important distinction, as when there are cycles in the graph, messages on any edge of the cycle will be iteratively transmitted round the same cycle without natural termination [8]. Therefore, message passing algorithms are usually exact for cycle-free graphs.

## 2.2 Sum-product algorithm

We now describe the computational rules of the sum-product algorithm [8]. Fig.2.2 shows a fragment of factor graph containing variable nodes drawn in circles and factor nodes drawn in squares. Messages are passing along the edges between variables and factors in both directions.

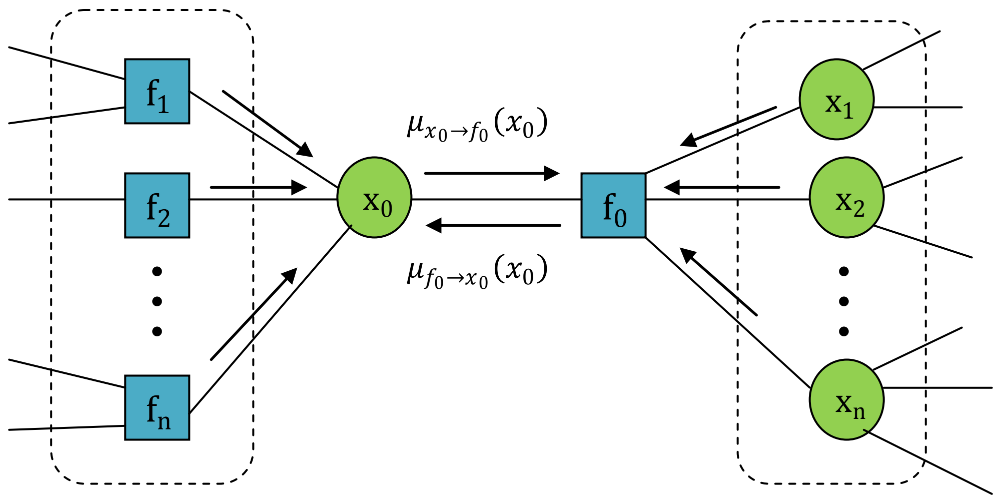


Figure 2.2 A fragment of Factor Graph showing the Sum-Product Algorithm

Let us denote by $\mu_{i \to j}(x_{\{i,j\}})$ the message sent from node $i$ to node $j$. Rules of sum-product algorithm that indicate message computation can be expressed as follows:

A message from a factor node to a variable node:

$$\mu_{f_o \to x_0}(x_0) = \sum_{x_1} \dots \sum_{x_n} f(x_0, x_1, \dots, x_n) \cdot \prod_{i=1}^{n} \mu_{x_i \to f_0}(x_i) . \tag{2.2}$$

A message from a variable node to a factor node:

$$\mu_{x_0 \to f_0}(x_0) = \prod_{i=1}^{n} \mu_{f_i \to x_0}(x_0). \tag{2.3}$$

The marginal distribution of node $x_0$ is given by

$$g(x_0) = \mu_{x_0 \to f_0}(x_0) \mu_{f_o \to x_0}(x_0) = \sum_{x_1} \dots \sum_{x_n} f(x_0, x_1, \dots, x_n). \tag{2.4}$$

We name the operation in (2.2) as message filtering and operation in (2.3) as message multiplication.

## 2.3 SPAWN

Our framework SPAWN maps a factor graph onto the network topology and develops the message passing scheme over the network factor graph [7]. To create the factor graph, we need first to formulate the relationship between all variables in the network and factorize it. Then we create a net factor graph based on this factor graph by associating each node with its local information. We implement the sum-product algorithm over the net factor graph, therefore, for each agent in the localization algorithm SPAWN, the updating of its own belief is based on the above mentioned rules in Eq. (2.2)(2.3).

For a demonstration, let us consider the example of network given in Fig.1.2 where an agent ( $x_1$ ) talks to three anchors ( $x_A, x_B, x_C$ ). Having observed the distance measurements ( $\hat{d}_{A\to 1}, \hat{d}_{B\to 1}, \hat{d}_{C\to 1}$ ), we can write the posteriori distribution function and factorize it as:

$$p\left(x_1, x_A, x_B, x_C \middle| \hat{d}_{A\to 1}, \hat{d}_{B\to 1}, \hat{d}_{C\to 1}\right) \propto p(x_1)p(x_A)p(x_B)p(x_C)$$
$$\cdot\, p\left(\hat{d}_{A\to 1}\middle|x_1, x_A\right)p\left(\hat{d}_{B\to 1}\middle|x_1, x_B\right)p\left(\hat{d}_{C\to 1}\middle|x_1, x_C\right). \qquad (2.5)$$

In the corresponding factor graph of this factorization, we associate each variable with the factors that contains its local information (shown in the colored block), resulting a net factor graph depicted in Fig.2.3 where arrows indicate the flow of messages.

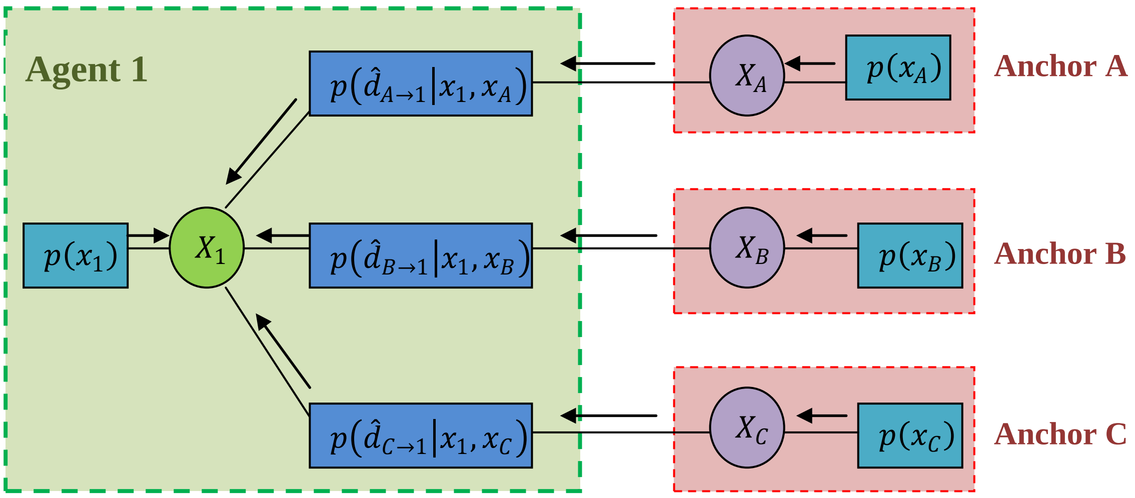

Agent 1
$p(\hat{d}_{A\to 1}|x_1, x_A)$
$p(x_1)$
$X_1$
$p(\hat{d}_{B\to 1}|x_1, x_B)$
$p(\hat{d}_{C\to 1}|x_1, x_C)$
$X_A$
$p(x_A)$
Anchor A
$X_B$
$p(x_B)$
Anchor B
$X_C$
$p(x_C)$
Anchor C

Figure 2.3 A Net Factor Graph and Its Message Passing

Observe that there are two types of messages: the ones within the blocks that are computed within the nodes and the ones between the blocks that are passing over the link. The former are from factor to variable therefore we apply Eq. (2.2) to compute them, while the latter are from variable to factor that need to obey the computation rule in Eq. (2.3).

Therefore, any representation of messages must enable efficient computation of these two operations. More specifically, given initialization of node $i$ 's prior as $p_i(x_i)$ and incoming messages $\mu_{j\to i}(x_i)$ from the set of neighboring nodes $S_{\to i}$, the belief is

$$b_{X_i}(x_i) \propto p_i(x_i) \prod_{j \in S_{\to i}} \mu_{j\to i}(x_i), \tag{2.6}$$

which is simply multiplying every incoming messages with its previous prior. Thus one significant objective of this master thesis is to solve the problem formulation of message multiplication to capture the true distribution.

## 2.4 Three types of Message Representations

There are different types of message representations, among which the simplest way is discretized message representation. Given the node's estimated distribution $\mathrm{p_X(x)}$, discretized means that rather than considering continuous x, we pre-define a finite grid of possible values for x. Hence, Eq. (2.6) only needs to be evaluated for a finite set. For example, a continuous distribution $\mathrm{p_X(x)}$ can be captured by a finite set of N values generated by $\{\mathrm{p_X(x_i)}\}_{\mathrm{i=1}}^{\mathrm{N}}$ with $\{\mathrm{x_i}\}_{\mathrm{i=1}}^{\mathrm{N}}$ being the grid points over a finite region X [5]. One of the main drawbacks of this solution is that the length of messages N is proportional to the number of grid points over X. Once there is a large uncertainty in node position which means the estimated distribution $p_X(x)$ is over a large domain $X$, the length of the messages can be very long when ensuring an accurate estimation. That is why the discretized message representation is not an efficient or even realistic solution to SPAWN.

Another type of message representation is sample-based message representation in which each message is represented by a number of samples that reside with high probabilities in the region. For example, a distribution $p_X(x)$ can be represented by a set of $N$ weighted samples$\{x_i, w_i\}_{i=1}^{N}$ , where $x_i$ are samples and $w_i$are their corresponding probabilities [6]. Since we only draw samples from the estimated distribution $p_X(x)$ , the number of samples used in sample-based method is much smaller compared to the one in the discrete method. However, an inadequacy of this solution is its computational complexity, especially when computing the multiplication of several incoming messages. Due to the fact that different messages may use different sets of the samples, thus the operation can not be simply multiplying the probabilities of the samples from individual messages as

$$b_X(x) \propto \prod_{m=1}^{M} p_X^{(m)}(x)\,, \tag{2.7}$$

but requiring more complex and larger amount of calculation. Therefore, the sample-based message representation has higher computational complexity when applying message multiplication operations.

A third way is so-called parametric message representation where message is represented as a set of parameters. We assume the distributions of true messages can be characterized from a family of some particular parameterized distributions (e.g., Gaussian distribution). Comparing with the former two representations, the parametric method has the lowest computational complexity but can only represent massages of some certain shapes. The thesis will focus on a parametric way of message representation in the positioning algorithm SPAWN.

# 3. Non-Cooperative Positioning

In this chapter, we will focus on the message representation and updates for non-cooperative positioning. Having described in Chapter 2 that we apply a parametric representation for messages, we will elaborate now how these messages are computed in our framework SPAWN, specifically the message multiplication step.

## 3.1 Message multiplication

As we have introduced in the previous chapter, message multiplication is a key step in SPAWN. Given $M$ independent incoming messages with corresponding distributions $p_X^{(m)}(x)\ \ (m = 1, \dots, M)$, the operation is to compute the joint probability distribution

$$b_X(x) \propto \prod_{m=1}^{M} p_X^{(m)}(x)\,, \tag{3.1}$$

usually this $b_X(x)$ can not be exactly represented parametrically but can be approximated.

Let us consider a simply case according to our system model. Suppose that some agent can communicate with 3 anchors, and their true positions are shown in Fig.3.1 (a). Each anchor sends a message containing information about its coordinates $x_j, (j \in S_{\to x_i})$ and distance measurements to the agent

$$p(\hat{d}_{j\to i}|x_i) = N\left(\|x_i - x_j\|,\ \sigma_{j\to i}^2\right). \tag{3.2}$$

According to Bayes' rule, agent can then get a posterior distribution based on this single message:

$$p(x_i|\hat{d}_{j\to i}) = \frac{p(\hat{d}_{j\to i}|x_i)p(x_i)}{p(\hat{d}_{j\to i})} \propto p(\hat{d}_{j\to i}|x_i)$$
$$= N\left(\|x_i - x_j\|, \sigma_{j\to i}^2\right). \tag{3.3}$$

Fig.3.1 (b) gives an example of such posterior distribution of the agent based on the message received from anchor 1. Since the distribution has a shape similar to a donut, we will name such distribution "donut" for the rest of our report.

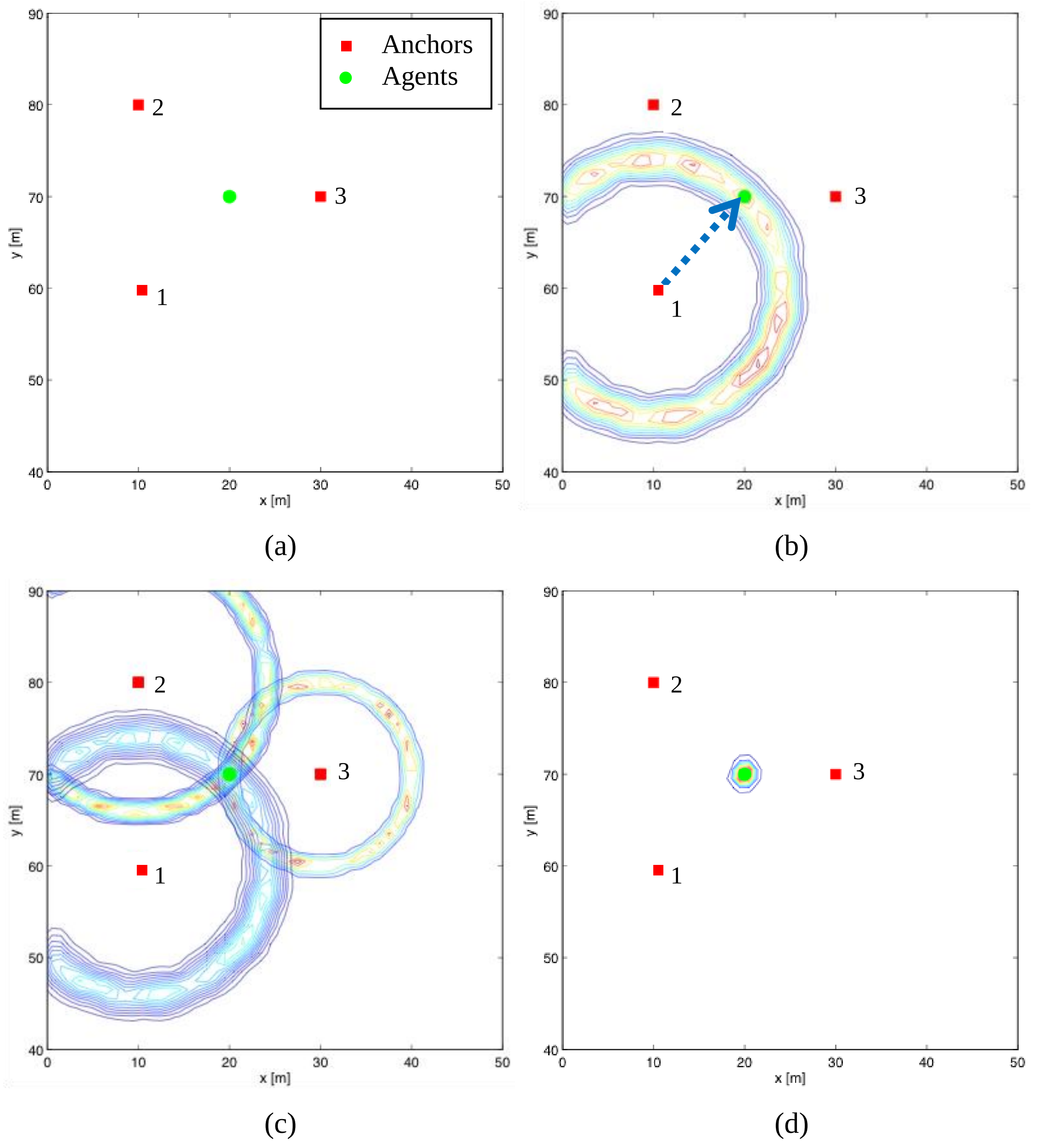


Figure 3.1 Example of Message Multiplication for Localization

Fig.3.1 (c) plots all the distribution $p(x_i|\hat{d}_{j\to i}),(j=1,2,3)$ from the three anchors. Since the distance measurements are independent, we can get

$$p(\hat{d}_{1\to i},\hat{d}_{2\to i},\ldots,\hat{d}_{n\to i}|x_i)=\prod_{j=1}^{n}p(\hat{d}_{j\to i}|x_i). \tag{3.4}$$

According to Bayes' rule,

$$p(x_i|\hat{d}_{1\to i},\hat{d}_{2\to i},\ldots,\hat{d}_{n\to i})\propto\prod_{j=1}^{n}p(\hat{d}_{j\to i}|x_i)=C\cdot\prod_{j=1}^{n}N\left(\|x_i-x_j\|,\sigma_{j\to i}^2\right)$$

$$= C \cdot \exp\left(\sum_{j=1}^{n}\left(-\frac{1}{2\sigma_{j\to i}^2}(\hat{d}_{j\to i} - \|x_i - x_j\|)^2\right)\right), \qquad (3.5)$$

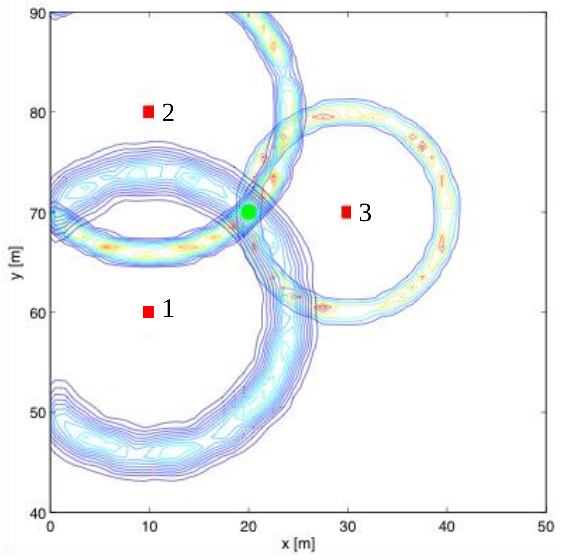
where $C$ is a constant. This operation is called message multiplication as it takes the product of all incoming messages. Fig.3.1 (d) plots the resulting distribution of the message multiplication.

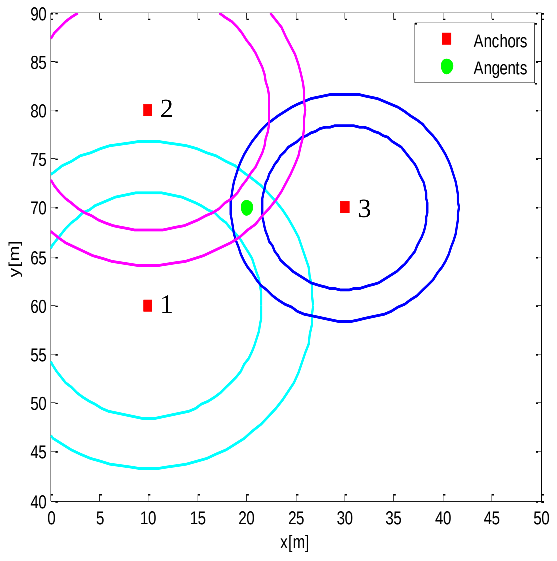
## 3.2 Initial Estimation of Agent's distribution

Observe from Fig.3.1 (d) that the agent fall into the area given by posterior distribution. However, it is always not realistic to represent the exact distribution in Eq. (3.5) by message, so we project it onto a family of distributions that we are able to represent. For example, the distribution in Fig.3.1 (d) can be approximated by a two-dimensional Gaussian distribution. More cases are taken into consideration in this section.

### 3.2.1 Overlapping Region of Incoming Messages

A first estimate of the posterior distribution can be determined geometrically by taking the overlapping area of all donuts given by the incoming messages. Given a message $[x_j, \hat{d}_{j\to i}, \sigma_{j\to i}^2]$, one way to capture the donut shape is to plot two circles around the contour lines. While the midpoints are easy to decide as $x_j$, radius can be chose from $(\hat{d}_{j\to i} \pm t \cdot \sigma_{j\to i})$, $t$ is a constant (e.g., $t = 2$).

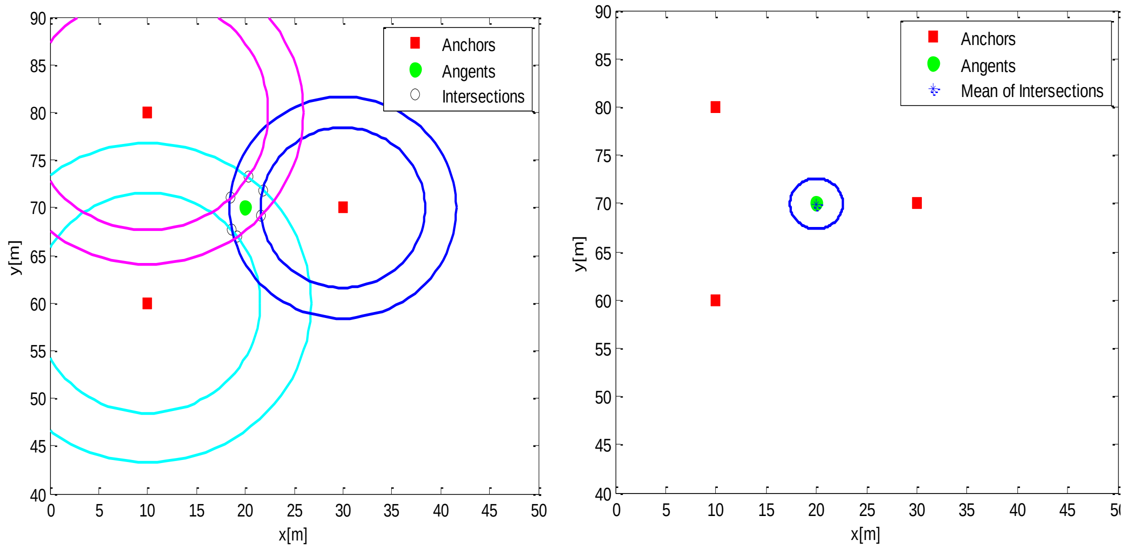




(a) (b)

Figure 3.2 Capture the Donut Shape Using Two Circles

When we have captured the donut shape as shown in fig.3.2 (b), the next step is to capture the overlapping region. This can be done by calculating the intersections of any two circles and selecting those within all the donuts. For instance, in the above network, if an intersection from the Anchor 1's circle and Anchor 2's circle falls also between Anchor 3's circles, then we consider it as useful, otherwise, we discard it.



(a) (b)

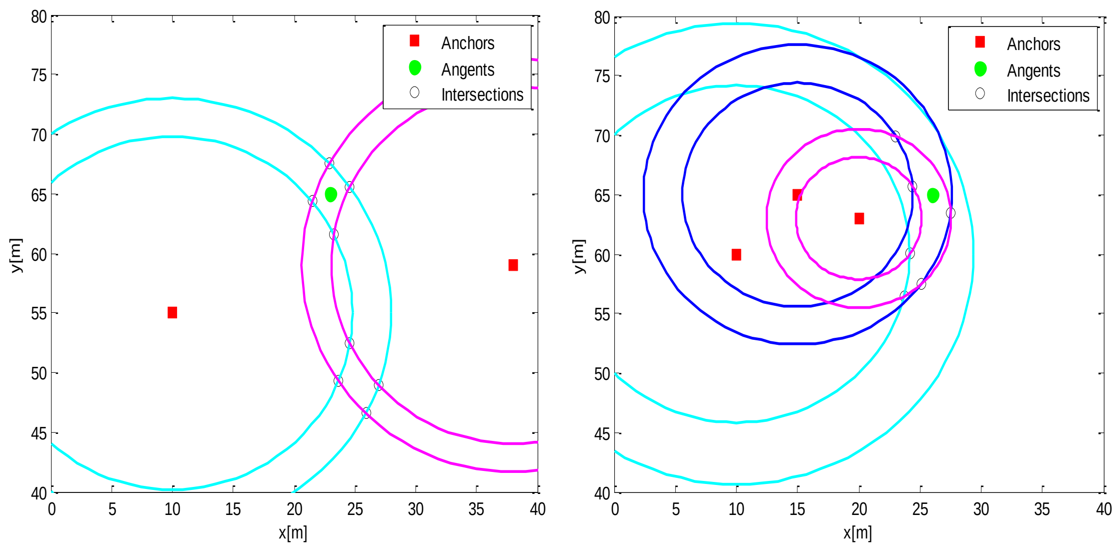
Figure 3.3 Select Useful Intersections for the Initialization

Once all the meaningful intersections are found out as in Fig.3.3 (a), we can obtain the knowledge of the overlapping area. One way to represent this area is to use a Gaussian distribution $N\left([m_x, m_y], \sigma^2\right)$, with $[m_x, m_y]$ being the mean of the intersections' coordinates and $\sigma^2$ being the corresponding variance. We plot the estimated mean and radius of 2* $\sigma$ in Fig.3.3 (b).

### 3.2.2 Decide the Type of Distribution

#### 3.2.2.1 A Single Gaussian Distribution

A single Gaussian distribution can well model such distribution as shown in above example. However, it can not properly describe every location distribution. There are mainly two typical unfit cases: one often occurs when agent only talks to two anchors and results in two possible locations as shown in Fig.3.4 (a), the other is when the distribution looks like a "banana" as shown in Fig.3.4 (b).



(a) (b)

Figure 3.4 Two Typical Cases Unfit for Single Gaussian Representation

To distinguish whether the posterior distribution can be captured by a single Gaussian distribution, we need to judge whether all the useful intersections are concentrated in a small area. That is, if intersections are scattered over a wide area (e.g., Fig 3.4 (b)) or an area with more than one center (e.g., Fig 3.4 (a)), the distribution can not be projected to a single Gaussian distribution. An efficient way to judge the closeness of intersections is to compute the ratio of standard deviation of the intersections $\sigma$ to the average standard deviation of input donuts $\frac{1}{n}\sum_{j=1}^{n}\sigma_{j\to i}$. The smaller this ratio, the "tighter" the distribution tends to show. A reasonable threshold can be set as 4 (which is used in our work) and any distribution fall below this threshold shall be considered as a single Gaussian distribution.

#### 3.2.2.2 A Mixture of Two Gaussians Distribution

Such distribution as in Fig3.4 (a) can be represented by a mixture of two Gaussians distribution

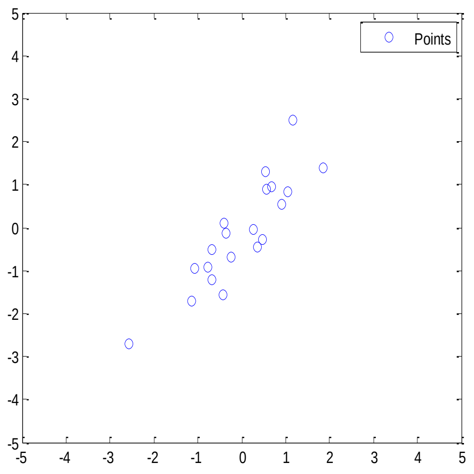
$$p_{GM}(x,y) = w_1 \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{1}{2\sigma_1^2}\left\|\begin{pmatrix}x\\y\end{pmatrix}-\begin{pmatrix}m_{x1}\\m_{y1}\end{pmatrix}\right\|^2\right)$$
$$+w_2 \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{1}{2\sigma_2^2}\left\|\begin{pmatrix}x\\y\end{pmatrix}-\begin{pmatrix}m_{x2}\\m_{y2}\end{pmatrix}\right\|^2\right), \qquad (3.6)$$

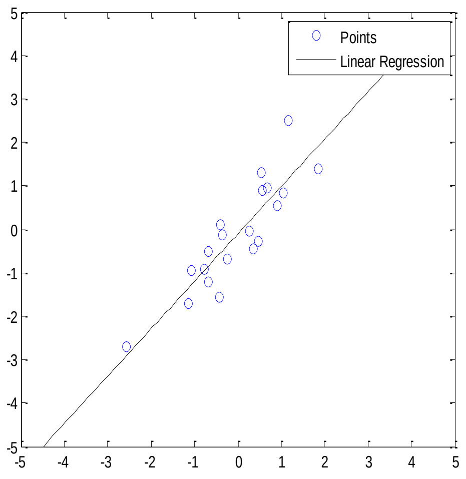
where $w_1$ and $w_2$ are the weights with constraints $w_1 \in [0,1]$ and $w_2 = 1 - w_1$. We assume that the two locations have equal probabilities, that is, $w_1 = w_2 = 0.5$.

To decide whether a distribution looks like a mixture of two Gaussians, an intuitive method is to separate the intersections into two groups, if the distance between the groups are fairly large and each group of intersections satisfies the condition of single Gaussian in Section 3.2.2.1, then the distribution can be captured by a mixture of two Gaussians.

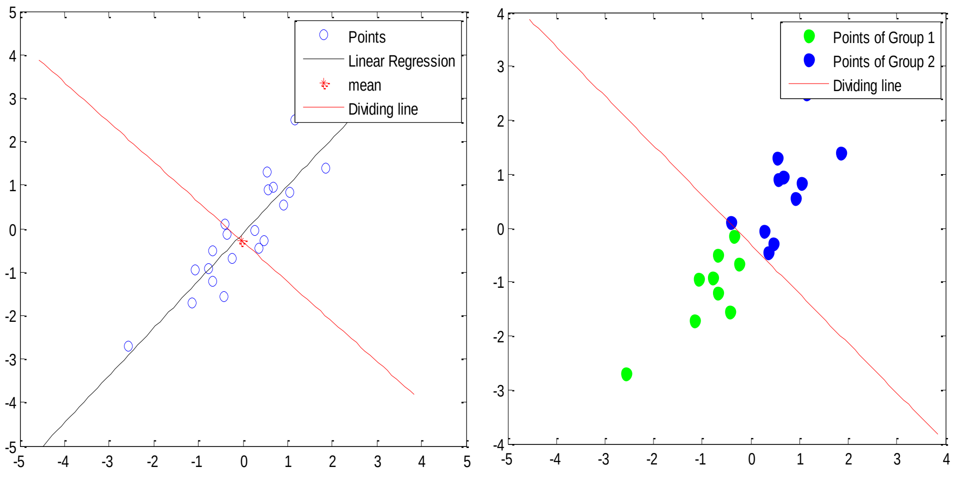
An approach using linear regression can divide intersections into two groups. Linear regression is a model of the linear relationship between different dependent variables. In our case, the x and y coordinates of intersections are the variables.

We use the following diagram to explain this approach. Suppose we have 20 points as shown in Fig.3.5 (a), the linear regression returns the line in (b) modeling the linear relationship of the $x$ and $y$ coordinates of points. Knowing the slope of this line and the mean of the 20 points, we can plot an orthogonal line passing the mean value of points as the dividing line in (c). The last step is to grouping the points on each side of the orthogonal line into a new group and (d) shows the final two groups marked in different colors.





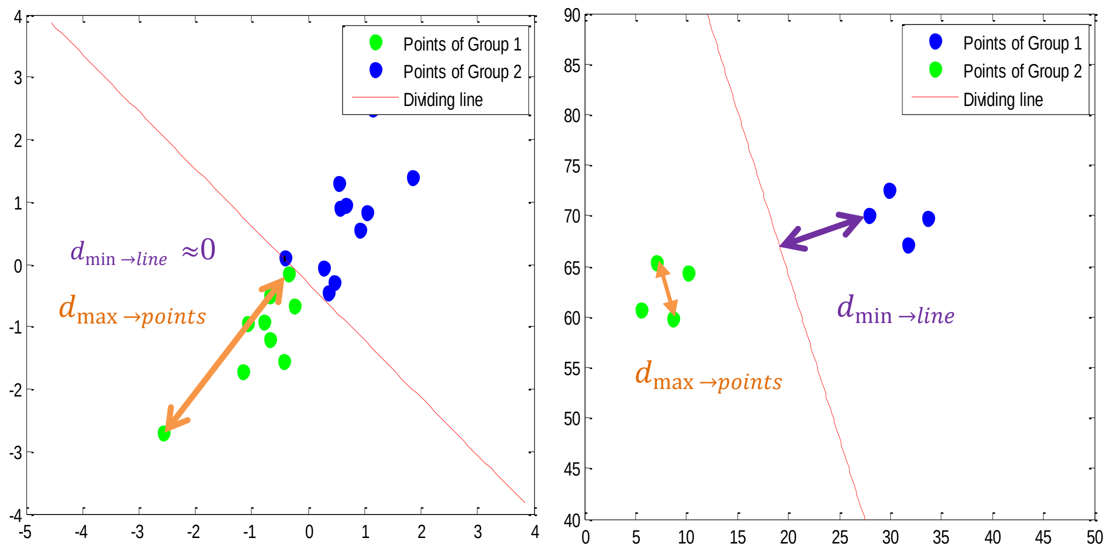
(a) (b)



(c) (d)

Figure 3.5 Linear Regression Approach to Divide Points into Two Groups

It is of importance to distinguish whether there are actually two possible locations or there is only one possible location with a large uncertainty (e.g., the “banana” shape distribution from Fig.3.4 (b)). A fairly good criterion of judging how far these two groups are departed from each other is the ratio $d_{\max\rightarrow points}/d_{\min\rightarrow line}$. Here $d_{\max\rightarrow points}$ denotes the maximum distance between any two intersections belong to a same group and $d_{\min\rightarrow line}$ is the minimum distance between any points to the dividing line. We set a threshold of 6 and distributions have ratios below this can be considered as two groups.



(a) (b)

Figure 3.6 A Criteria to Distinguish Whether There are Two Possible Locations

After applying the above criteria, only (b) in Fig.3.6 can be viewed as having two possible locations. We take the same step in 3.2.2.1 for each group of (b), if both satisfy the condition of being single Gaussian, then the distribution can be captured by a mixture of two Gaussians.

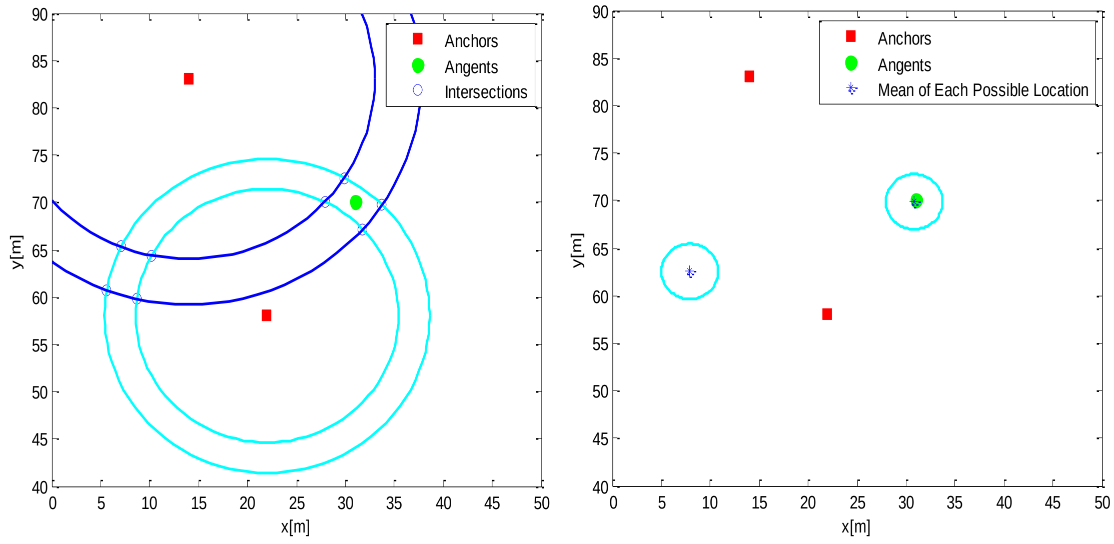


Figure 3.7 Initialization of a Mixture of Two Gaussians distribution

### 3.2.2.3 A Single Donut Distribution

Another common case is when agent only talks to one anchor, it will return with a single donut distribution (e.g., Fig.3.1 (b)). In order to mathematically represent such distribution, we introduce a new type of distribution:

$$\mathcal{D}\left(x,y;\ m_{x_1},m_{y_1},\sigma^2,\rho\right)=\frac{1}{\mathrm{C}(\sigma^2,\rho)}\exp\left(-\frac{1}{2\sigma^2}\left(\left\|\begin{pmatrix}x\\y\end{pmatrix}-\begin{pmatrix}m_{x_1}\\m_{y_1}\end{pmatrix}\right\|-\rho\right)^2\right),\quad (3.7)$$

where$\left[\,m_{x_1},m_{y_1}\right]$ indicates the midpoint of the distribution, $\sigma^2$ is the variance and $\rho$ is the radius, $\mathrm{C}(\sigma^2,\rho)$ is a normalization constant:

$$\mathrm{C}(\sigma^2,\rho)=2\pi\sigma^2\left(\exp\left(-\frac{\rho^2}{2\sigma^2}\right)+\frac{1}{2}\sqrt{\frac{2\pi\rho^2}{\sigma^2}}+(1+\mathrm{erf}\sqrt{\frac{\rho^2}{2\sigma^2}})\right).\qquad (3.8)$$

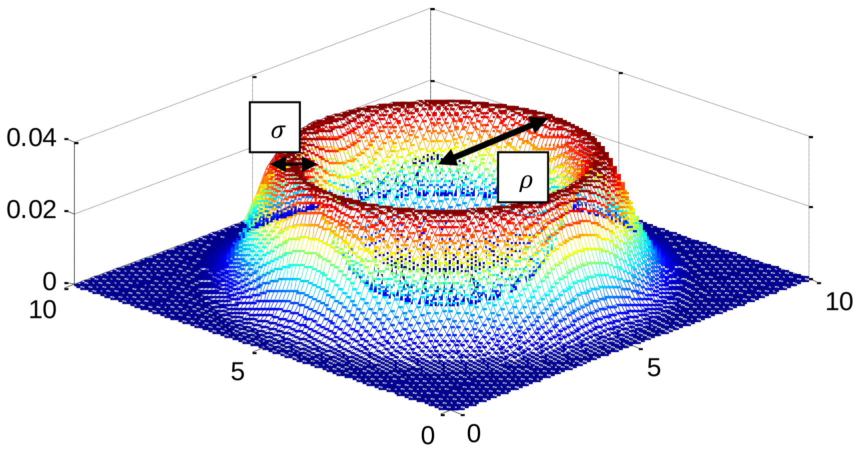


Figure 3.8 One example of the $\mathcal{D}$ distribution

As shown in Fig.3.8, the distribution looks like a donut. Note that one important feature of $\mathcal{D}$ distribution is it can revert to a two-dimensional Gaussian distribution by setting the value of $\rho$ to be 0.

**3.2.3 Message Representation**

We intend to determine a family of distributions that enables a large proportion of agents to represent their outgoing messages to neighboring agents. Therefore, we need to analyse which are the cases that frequently occur in order to find a general message representation to cover those cases.

- Case 1: When an agent talks to three or more anchors, it will have either a Gaussian or a "banana" shape distribution. The latter cases are not considered. We can represent a Gaussian distribution $\mathrm{N}\left([\mathrm{m_x}, \mathrm{m_y}], \sigma^2\right)$ using three parameters. As we mentioned in Section 3.2.1, $[\mathrm{m_x}, \mathrm{m_y}]$ can be estimated as the mean of the donuts' intersections and $\sigma^2$ is the corresponding variance.

- Case 2: When an agent talks to two anchors, it will have either a mixture of two Gaussians or a "banana" shape distribution. The mixture of two Gaussians $\mathrm{p_{GM}}(\mathrm{x}, \mathrm{y}) = 0.5 \cdot \mathrm{N}\left(\left[\mathrm{m_{x_1}}, \mathrm{m_{y_1}}\right], \sigma^2\right) + 0.5 \cdot \mathrm{N}\left(\left[\mathrm{m_{x_2}}, \mathrm{m_{y_2}}\right], \sigma^2\right)$ is an extension of Case 1. The initial estimates of each Gaussian distribution $\left(\left[\mathrm{m_{x_1}}, \mathrm{m_{y_1}}\right], \sigma_1\right)$

and $([m_{x_2}, m_{y_2}], \sigma_2)$can be get respectively from each group of intersections as Case 1. Then we initialize σ as $0.5 \cdot (\sigma_1 + \sigma_2)$.

- Case 3: When an agent talks to one anchor, it will have a donut distribution. $\mathcal{D}(m_{x_1}, m_{y_1}, \sigma^2, \rho)$, these four parameter can be directly get from the incoming message, as $[m_{x_1}, m_{y_1}]$ is anchor's position, ρ is the distance measurement and $\sigma^2$ is the variance.

Notice that a single Gaussian distribution can be treated as a special case of a mixture of two Gaussian distribution when $[m_{x_1}, m_{y_1}] = [m_{x_2}, m_{y_2}]$, also both single Gaussian distribution and mixture of two Gaussians distribution can be easily represented by setting $\rho$ to be 0. Therefore, to capture all cases (1-3), we represent all exchanging messages in the form of a six-dimensional vector $\boldsymbol{\alpha} = [m_{x_1}, m_{y_1}, m_{x_2}, m_{y_{2,}}, \sigma^2, \rho]$ as a mixture of two $\mathcal{D}$ distributions:

$$p_{\mathcal{D}_2}(x, y) = 0.5 \cdot \mathcal{D}(x, y;\ m_{x_1}, m_{y_1}, \sigma^2, \rho) + 0.5 \cdot \mathcal{D}(x, y; m_{x_2}, m_{y_{2,}}, \sigma^2, \rho). \qquad (3.9)$$

We name this family of distributions $\mathcal{D}_2$ in the rest of this report. This family of distributions will turn out to be sufficient for the cooperative case as well. A table of message representation for cooperative positioning is given as follows:

| **Representation** / **Type** | $m_{x_1}$ | $m_{y_1}$ | $m_{x_2}$ | $m_{y_2}$ | $\sigma$ | $\rho$ |
|---|---|---|---|---|---|---|
| Single Gaussian Distribution | $m_x$ | $m_y$ | $m_x$ | $m_y$ | $\sigma$ | 0 |
| Mixture of two Gaussians Distribution | $m_{x_1}$ | $m_{y_1}$ | $m_{x_2}$ | $m_{y_2}$ | $\frac{(\sigma_1 + \sigma_2)}{2}$ | 0 |
| Other distribution (incoming message exists) | $m_{x_1}$ | $m_{y_1}$ | $m_{x_2}$ | $m_{y_2}$ | $\sigma$ | $\rho$ |

Table 3.1 Message Representation for Non-Cooperative Positioning

Given a message $[m_{x_1}, m_{y_1}, m_{x_2}, m_{y_2}, \sigma^2, \rho]$, the represented distribution is

$$p_{\mathcal{D}_2}(x,y) = \frac{1}{2\cdot \mathrm{C}(\sigma^2,\rho)} \exp\left(-\frac{1}{2\sigma^2}\left(\left\|\binom{x}{y} - \binom{m_{x_1}}{m_{y_1}}\right\| - \rho\right)^2\right) + \frac{1}{2\cdot \mathrm{C}(\sigma^2,\rho)} \exp\left(-\frac{1}{2\sigma^2}\left(\left\|\binom{x}{y} - \binom{m_{x_2}}{m_{y_2}}\right\| - \rho\right)^2\right) \tag{3.10}$$

## 3.3 Optimization of Agent's Distribution

The optimization step is to find the best match of the posterior distribution out of the family of mixture Donuts distributions. Before the specific description of our approach, we first introduce some theoretical background.

### 3.3.1 Kullback-Leibler Divergence

Kullback-Leibler divergence is a measure of the difference between two probability distributions, defined as

$$D_{KL}(p_x \| q_x) = \int p_x(x) \log \frac{p_x(x)}{q_x(x)} dx. \tag{3.11}$$

The closer $p_x$ is to $q_x$, the smaller $D_{KL}(p_x \| q_x)$ will be. $D_{KL}(p_x \| q_x)$=0 when the two distribution are the same. In our case, given posterior distribution $p_{PD}$, the optimal $p_{\mathcal{D}2}$ to it can be found by minimizing $D_{KL}(p_{\mathcal{D}_2} \| q_{PD})$, with respect to the parameters of $p_{\mathcal{D}_2}$.

It is complicated to directly compute the definite integral in Eq. (3.11). We thus use Monte Carlo integration to get an approximate value of the integral. Suppose we wish to integrate a function

$$I = \int_a^b p_x(x) f(x)\, dx. \tag{3.12}$$

Monte Carlo integration scheme picks $N$ distributed points $x_1, x_2, \dots, x_N$ at random, traditionally these points can be uniformly distributed over the interval $[a, b]$, but to use adaptive sampling over the region can get a better estimate. The integral is then approximated to:

$$I \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i), \qquad x_i \sim p_x(x). \tag{3.13}$$

Based on this technique, we can determine our Kullback-Leibler divergence by

$$D_{KL}(p_{\mathcal{D}_2} \| q_{PD}) \approx \frac{1}{N} \sum_{i=1}^{N} (\log p_{\mathcal{D}_2}(x_{i,}y_i) - \log q_{PD}(x_i, y_i)),$$

$$[x_i, y_i] \sim p_{\mathcal{D}_2}(x, y). \quad (3.14)$$

Since $p_{\mathcal{D}_2}(x, y)$ is mixture of two $\mathcal{D}$ distributions given in Eq. (3.9), we can draw samples $\{[x_i, y_i]\}_{i=1}^{N}$ by taking $\frac{N}{2}$ samples from each $\mathcal{D}$ distribution. We can apply the importance sampling technique to sample from the donut distribution.

Note that Monte Carlo integrations are based on samples which are random, so that the results will come up with small difference each time. Larger amount of samples $N$ can provide with more reliable approximations.

### 3.3.2 Gradient Descent

To find the optimal distribution represented by $[m_{x_1}, m_{y_1}, m_{x_2}, m_{y_2}, \sigma^2, \rho]$ that minimize the $D_{KL}(p_{\mathcal{D}_2} \| q_{PD})$. We use a method named gradient descent which takes steps proportional to the negative gradient of $D_{KL}(p_{\mathcal{D}_2} \| q_{PD})$ at the current point. This is because the negative gradient is the direction in which the error decreases the fastest [10].

Let us denote the six-dimensional vector $[m_{x_1}, m_{y_1}, m_{x_2}, m_{y_2}, \sigma^2, \rho]$ by $\boldsymbol{\alpha}$**,**

$$D_{KL}(p_{\mathcal{D}_2}{}^{(\boldsymbol{\alpha})} \| q_{PD}) \approx \frac{1}{N} \sum_{i=1}^{N} (\log p_{\mathcal{D}_2}{}^{(\boldsymbol{\alpha})}(x_{i,}y_i) - \log\ q_{PD}(x_i, y_i)),$$

$$[x_i, y_i] \sim p_{\mathcal{D}_2}{}^{(\boldsymbol{\alpha})}(x, y). \quad (3.15)$$

The gradient of $D_{KL}(p_{\mathcal{D}_2} \| q_{PD})$ at the point $\boldsymbol{\alpha}$ is then given by

$$\nabla_{\boldsymbol{\alpha}} D_{KL}(p_{\mathcal{D}_2}{}^{(\boldsymbol{\alpha})} \| q_{PD}) = \begin{bmatrix} \frac{\partial}{\partial m_{x_1}} \\ \frac{\partial}{\partial m_{y_1}} \\ \frac{\partial}{\partial m_{x_2}} \\ \frac{\partial}{\partial m_{y_2}} \\ \frac{\partial}{\partial \sigma^2} \\ \frac{\partial}{\partial \rho} \end{bmatrix} D_{KL}(p_{\mathcal{D}_2}{}^{(\boldsymbol{\alpha})} \| q_{PD})$$

$$= \frac{1}{\mu} \begin{bmatrix} D_{KL}\left( p_{\mathcal{D}_2}{}^{\left(\boldsymbol{\alpha}+\begin{bmatrix}\boldsymbol{\mu}\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix}\right)} \| q_{PD} \right) - D_{KL}\left(p_{\mathcal{D}_2}{}^{(\boldsymbol{\alpha})} \| q_{PD}\right) \\ D_{KL}\left( p_{\mathcal{D}_2}{}^{\left(\boldsymbol{\alpha}+\begin{bmatrix}\mathbf{0}\\ \boldsymbol{\mu}\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix}\right)} \| q_{PD} \right) - D_{KL}\left(p_{\mathcal{D}_2}{}^{(\boldsymbol{\alpha})} \| q_{PD}\right) \\ D_{KL}\left( p_{\mathcal{D}_2}{}^{\left(\boldsymbol{\alpha}+\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ \boldsymbol{\mu}\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix}\right)} \| q_{PD} \right) - D_{KL}\left(p_{\mathcal{D}_2}{}^{(\boldsymbol{\alpha})} \| q_{PD}\right) \\ D_{KL}\left( p_{\mathcal{D}_2}{}^{\left(\boldsymbol{\alpha}+\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \boldsymbol{\mu}\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix}\right)} \| q_{PD} \right) - D_{KL}\left(p_{\mathcal{D}_2}{}^{(\boldsymbol{\alpha})} \| q_{PD}\right) \\ D_{KL}\left( p_{\mathcal{D}_2}{}^{\left(\boldsymbol{\alpha}+\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \boldsymbol{\mu}\\ \mathbf{0}\end{bmatrix}\right)} \| q_{PD} \right) - D_{KL}\left(p_{\mathcal{D}_2}{}^{(\boldsymbol{\alpha})} \| q_{PD}\right) \\ D_{KL}\left( p_{\mathcal{D}_2}{}^{\left(\boldsymbol{\alpha}+\begin{bmatrix}\mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \boldsymbol{\mu}\end{bmatrix}\right)} \| q_{PD} \right) - D_{KL}\left(p_{\mathcal{D}_2}{}^{(\boldsymbol{\alpha})} \| q_{PD}\right) \end{bmatrix}. \tag{3.16}$$

After obtaining the gradient $\nabla_{\boldsymbol{\alpha}} D_{KL}\left(p_{\mathcal{D}_2}{}^{(\boldsymbol{\alpha})} \| q_{PD}\right)$, we can move $\boldsymbol{\alpha}$ to a new position

$$\boldsymbol{\alpha}' = \boldsymbol{\alpha} + \varepsilon \cdot \left(-\nabla_{\boldsymbol{\alpha}} D_{KL}\left(p_{\mathcal{D}_2}{}^{(\alpha)} \| q_{PD}\right)\right), \tag{3.17}$$

where $\varepsilon$ is a small positive step that $0 < \varepsilon \ll 1$.

## 3.4 Flowcharts

A flowchart of the initialization step is shown as follows:

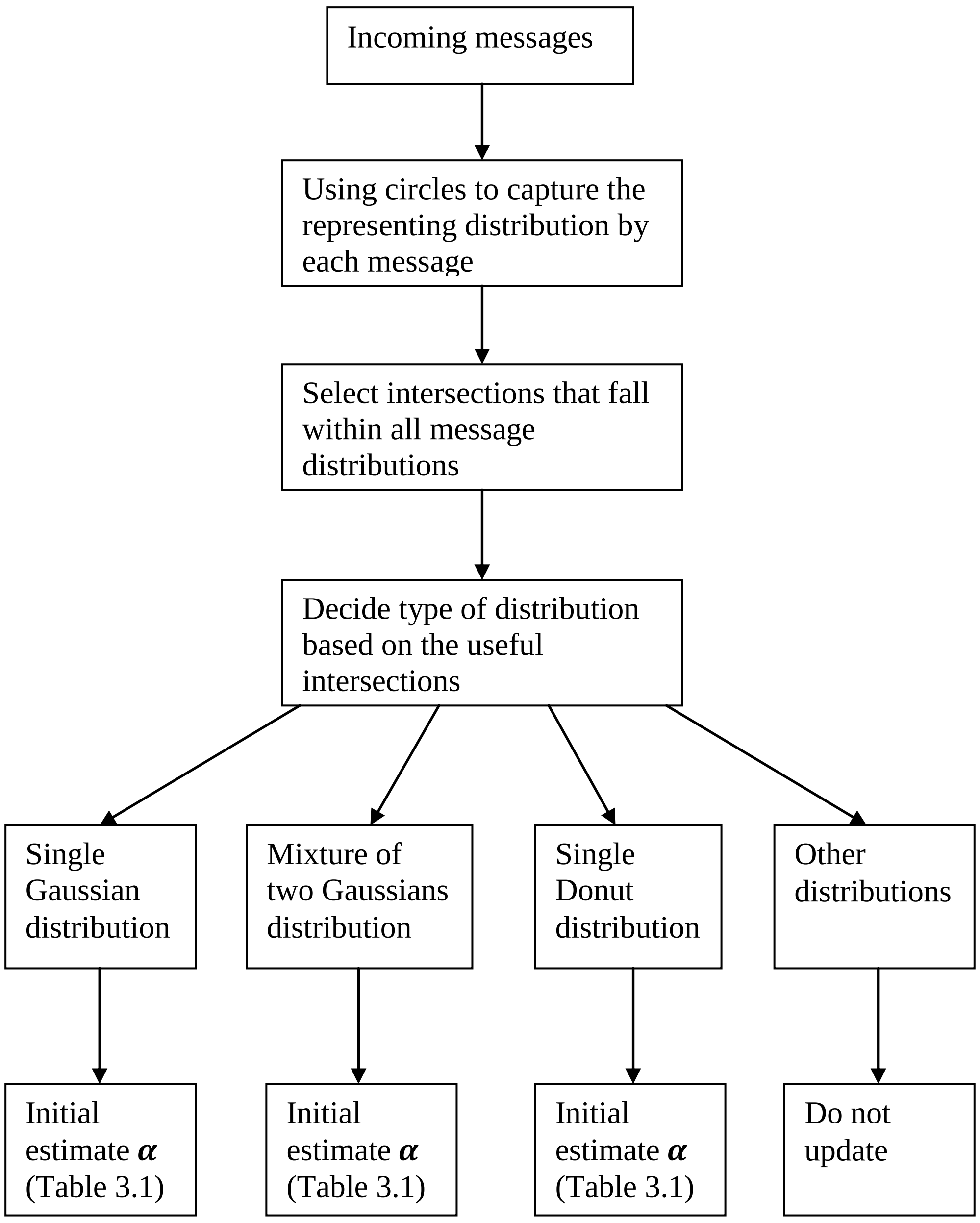

Incoming messages
Using circles to capture the representing distribution by each message
Select intersections that fall within all message distributions
Decide type of distribution based on the useful intersections
Single Gaussian distribution
Mixture of two Gaussians distribution
Single Donut distribution
Other distributions
Initial estimate $\boldsymbol{\alpha}$ (Table 3.1)
Initial estimate $\boldsymbol{\alpha}$ (Table 3.1)
Initial estimate $\boldsymbol{\alpha}$ (Table 3.1)
Do not update

Figure 3.9 Flowchart of the Initialization Step

A flowchart of the optimization step is shown as follows:

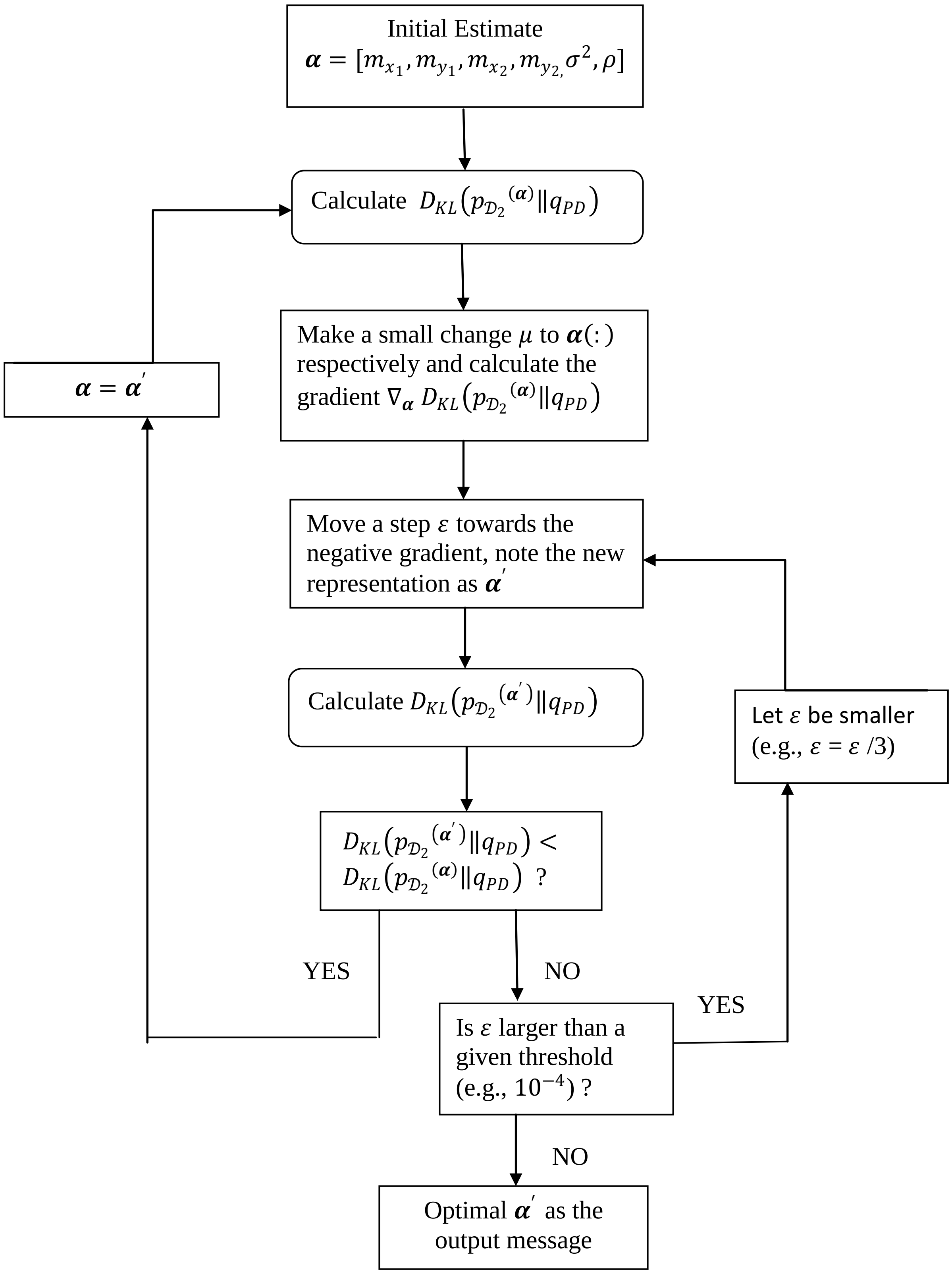

Initial Estimate
$\boldsymbol{\alpha} = [m_{x_1}, m_{y_1}, m_{x_2}, m_{y_2,} \sigma^2, \rho]$
Calculate $D_{KL}(p_{\mathcal{D}_2}{}^{(\boldsymbol{\alpha})} \| q_{PD})$
Make a small change $\mu$ to $\boldsymbol{\alpha}(:)$ respectively and calculate the gradient $\nabla_{\boldsymbol{\alpha}} D_{KL}(p_{\mathcal{D}_2}{}^{(\boldsymbol{\alpha})} \| q_{PD})$
$\boldsymbol{\alpha} = \boldsymbol{\alpha}'$
Move a step $\varepsilon$ towards the negative gradient, note the new representation as $\boldsymbol{\alpha}'$
Calculate $D_{KL}(p_{\mathcal{D}_2}{}^{(\boldsymbol{\alpha}')} \| q_{PD})$
Let $\varepsilon$ be smaller (e.g., $\varepsilon = \varepsilon /3$)
$D_{KL}(p_{\mathcal{D}_2}{}^{(\boldsymbol{\alpha}')} \| q_{PD}) <$ $D_{KL}(p_{\mathcal{D}_2}{}^{(\boldsymbol{\alpha})} \| q_{PD})$ ?
YES
NO
YES
Is $\varepsilon$ larger than a given threshold (e.g., $10^{-4}$) ?
NO
Optimal $\boldsymbol{\alpha}'$ as the output message

Figure 3.10 Flowchart of the Optimization Step

# 4. Cooperative Positioning

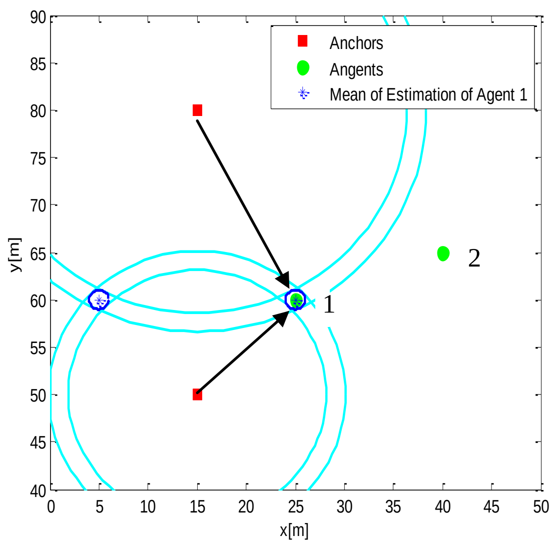
Having defined the general concept of cooperative positioning in Chapter 1, we are now interested in the message representation and updates for such system. As we have illustrated through Fig.1.1, the most essential difference between non-cooperative and cooperative positioning is whether the communication between agents are allowed. From the message point of view, sequentially two problems arise compared to non-cooperative positioning:

1. The format of incoming messages received from other agents.
2. What happens if an agent received message from other agent who has a mixture of two Gaussians distribution or a single donut distribution.

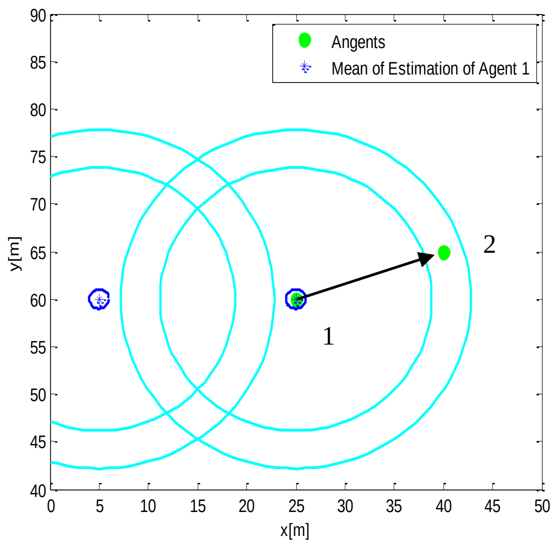
To cope with the first question, we represent all incoming messages as a distribution of the $\mathcal{D}_2$ family, with a six-dimensional vector $\boldsymbol{\alpha} = [m_{x_1}, m_{y_1}, m_{x_2}, m_{y_{2,}}, \sigma^2, \rho]$ (to be discussed in Section 4.1). About the second question, we would elaborate our solution in later sections of this chapter.

## 4.1 The Format of Incoming Messages

Let us consider several examples of network to illustrate the format of incoming message. A first example is shown in Fig.3.1 where Agent 1 can communicate with two anchors while Agent 2 has no communicate links to anchors.



(a)

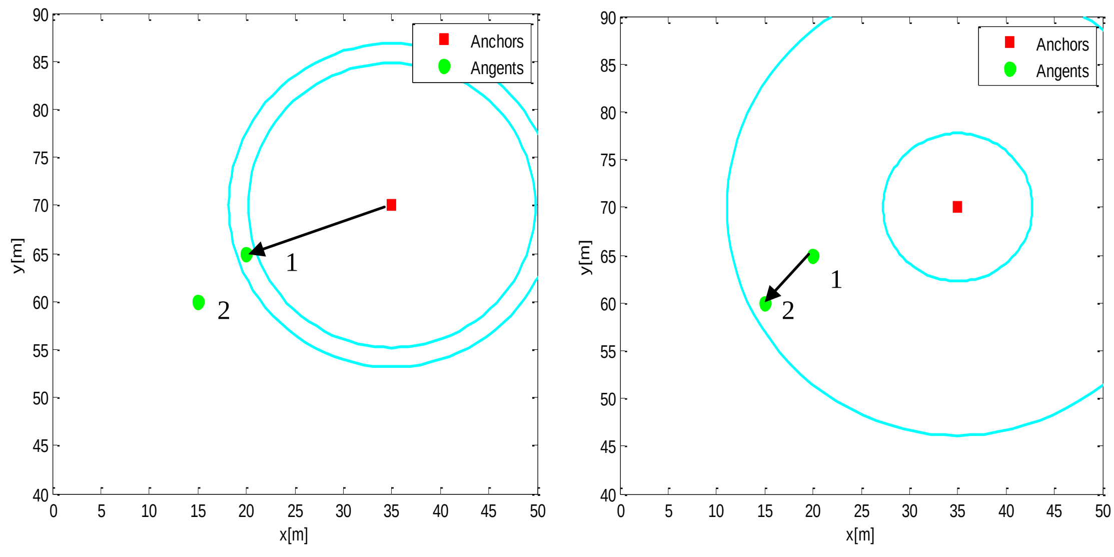


(b)

Figure 4.1 Incoming Message from Agent with a Mixture of Gaussians Distribution

In the first iteration, Agent 1 obtains its location as a mixture of two Gaussians distribution as shown in Fig.4.1 (a). In the second iteration, Agent 1 send a message to Agent 2 regarding the possible location of Agent 2 based on the distribution of Agent 1 and the distance measurement between these two agents. The representing distribution of this message is depicted in Fig.4.1 (b). The standard deviation of this message $\sigma_{1\to2} \approx 2\sigma_1 + \sigma_n$, where $\sigma_1$ given in the estimation of Agent 1 and $\sigma_n$ is from the range measurement.

Let us consider another case (see Fig.4.1) where Agent 1 talks to only one anchor and Agent 2 talks to no anchor. After the first iteration, Agent 1 obtains a Donut distribution as shown in Fig.4.1 (a). Later on Agent 1 sends a message to Agent 2 based on the distribution of Agent 1 and the distance measurement between these two agents. The corresponding distribution is described in Fig.4.2 (b). We can see that the donut is very "fat" which is due to $\sigma_{1\to2} \approx 2\sigma_1 + \hat{d}_{1\to2}$.



(a) (b)

Figure 4.2 Incoming Message from Agent with a Donut Distribution

Likewise, if an agent has a single Gaussian distribution, it will send to others a message with distribution like one of the donuts in Fig.4.2 (b). If an agent has a mixture of two donuts distribution as Agent 2 in Fig.4.2, it will in the next iteration send to its neighboring agents messages of a pair of fatter donuts. In order to include all the cases, we represent all incoming messages as a distribution of the $\mathcal{D}_2$ family which also brings about a beneficial effect that both incoming and outgoing messages are of the same format.

## 4.2 Message multiplication

Given $M$ independent incoming messages $\boldsymbol{\alpha}_m$ $(m = 1, \dots, M)$ with corresponding distributions $p^{(m)}(x, y) \in \mathcal{D}_2$ , the operation is to compute the product of these distributions as follows:

$$p(x, y|\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \dots, \boldsymbol{\alpha}_M) = \prod_{m=1}^{M} p^{(m)}(x, y|\boldsymbol{\alpha}_m)$$

$$= \prod_{i=1}^{M} \left( \frac{1}{2 \cdot \mathrm{C}(\sigma_m^2, \rho_m)} \exp\left( -\frac{1}{2\sigma_m^2} \left( \left\| \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} m_{x1}^{(m)} \\ m_{y1}^{(m)} \end{pmatrix} \right\| - \rho_m \right)^2 \right) + \frac{1}{2 \cdot \mathrm{C}(\sigma_m^2, \rho_m)} \exp\left( -\frac{1}{2\sigma_m^2} \left( \left\| \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} m_{x2}^{(m)} \\ m_{y2}^{(m)} \end{pmatrix} \right\| - \rho_m \right)^2 \right) \right) \quad (4.4)$$

As we mentioned in previous chapter, usually this $p(x, y|\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \dots, \boldsymbol{\alpha}_m)$ can not be exactly represented by $\mathcal{D}_2$ family but can be considered as a posterior distribution which we approximated our resulting distribution to. When $p(x, y|\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \dots, \boldsymbol{\alpha}_m) \notin \mathcal{D}_2$ (e.g., "banana" distribution), we do not approximate it and keep one original incoming message as its output.

## 4.3 Initial Estimation of Agent's distribution

For the initialization, an agent need not distinguish whether its received messages are from anchors or agents, but just to capture the area given by posterior distribution. Thus this step is similar to the non-cooperative positioning, only that there are some places that we need to pay attention.

### 4.3.1 Overlapping Region of Incoming Messages

As we described before, different from non-cooperative positioning where all incoming messages are of the shapes as single donuts, we consider each message in cooperative positioning as mixture of two donuts shape. The motivation is to capture the information from the agents who have two possible locations (e.g., a mixture of two Gaussians distribution). For messages from nodes who have good knowledge about their locations such as anchors, the two component donuts are actually overlapping so that $[m_{x_1}, m_{y_1}] = [m_{x_2}, m_{y_{2,}}]$.

Given a distribution $\boldsymbol{\alpha} = [m_{x_1}, m_{y_1}, m_{x_2}, m_{y_{2,}}, \sigma^2, \rho]$, we can capture its shape of two donut by plotting each with two circles (e.g., Fig.4.1 (b)). The midpoints of the two donuts are easy to decide as$[m_{x_1}, m_{y_1}]$ and $[m_{x_2}, m_{y_{2,}}]$, and each donut's two circles take radius from $(\rho \pm t \cdot \sigma)$, $t$ is a constant (e.g., $t = 2$).

To taking the overlapping area of all donuts given by the incoming messages, we calculate the intersections of any two circles from different messages and select those that fall within all messages distributions. There are two noteworthy points here:

1. Intersections of two donuts from a same message are not taken into account (see Fig.4.3)
2. Only when an intersection falls in neither of the two donuts from a message distribution can we say it is out of that distribution.

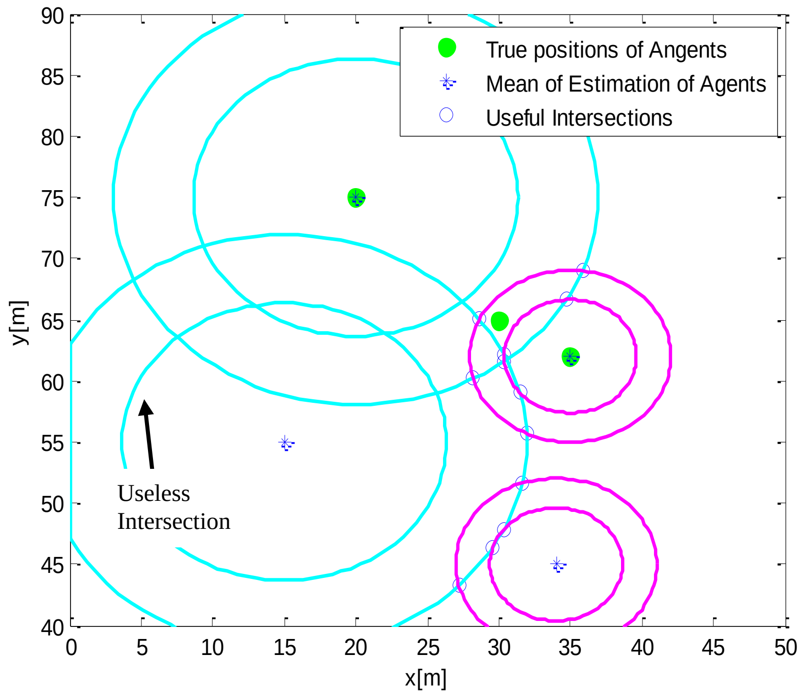


Figure 4.3 Useful Intersections of Two $\mathcal{D}_2$ Distributions

### 4.3.2 Decide the Type of Distribution

Once all the meaningful intersections are found out, we apply the same method and criteria as in Section 3.2.2 to decide whether the distribution can be projected onto a $\mathcal{D}_2$ distribution.

#### 4.3.3 Message Representation

As we mentioned earlier, all exchanged messages in our cooperative positioning algorithm are of the $\mathcal{D}_2$ family of distributions.

## 4.4 Optimization of Agent's Distribution

The same method as stated in Section 3.3 for optimization is applied here. In addition, for any agent who receives only one message based on which we can not get more accurate estimation than itself, we just return this original message as the optimal estimation for such cases.

# 5. Simulation

## 5.1 Simulation Setup

In the simulation part, we apply the message representation and updates algorithm to existing cooperative positioning framework SPAWN (refer to [7]). The simulation is performed on MATLAB.

We consider networks with 13 fixed anchors and 100 randomly distributed agents, within a 100m*100m area and a communication range of 20 m. Both anchors and agents are static. We run 20 iterations for location updating for cooperative positioning. Estimates are given as the means of the beliefs at any iteration.

## 5.2 Results from One Network

Here we consider one network to provide us with an intuitive view of results from non-cooperative positioning and the cooperative positioning.

### 5.2.1 Non-Cooperative Positioning

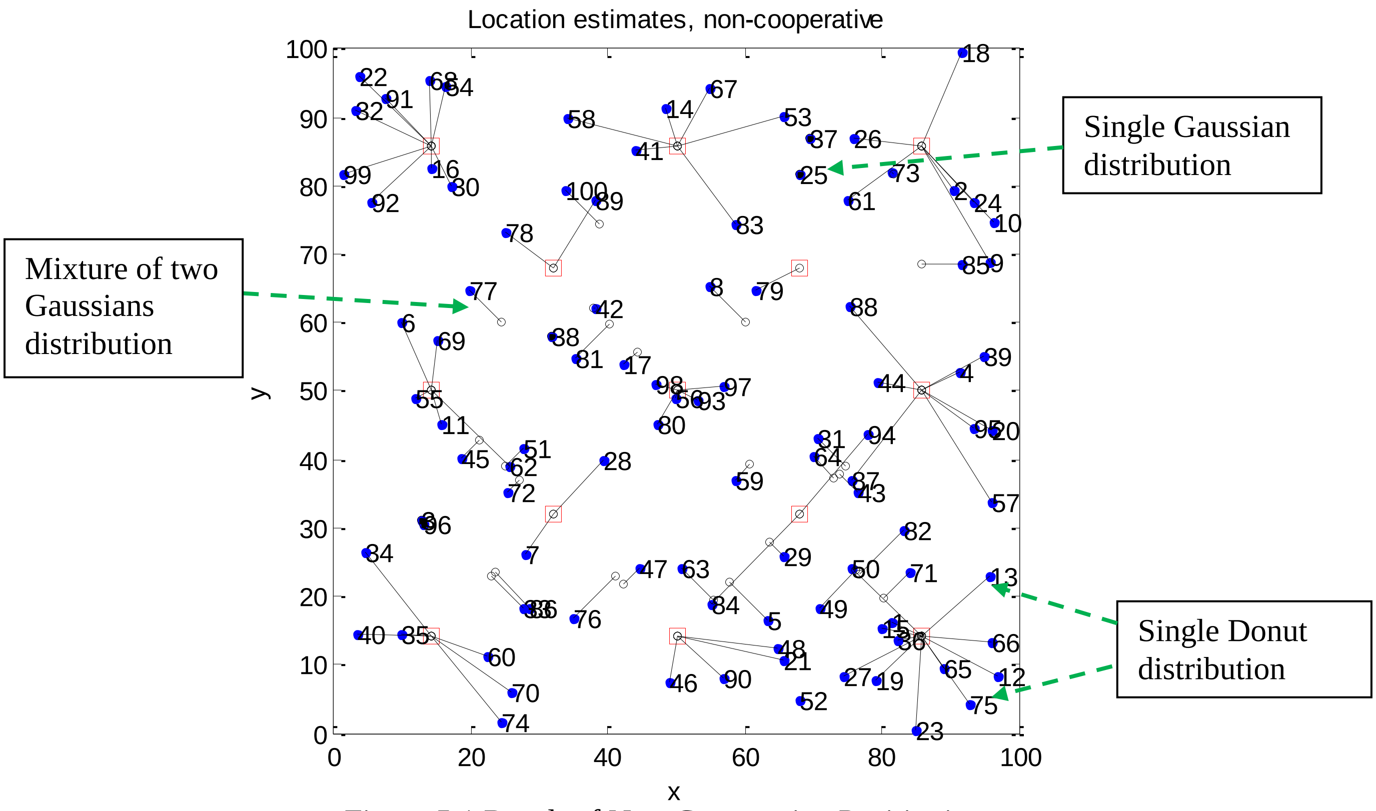
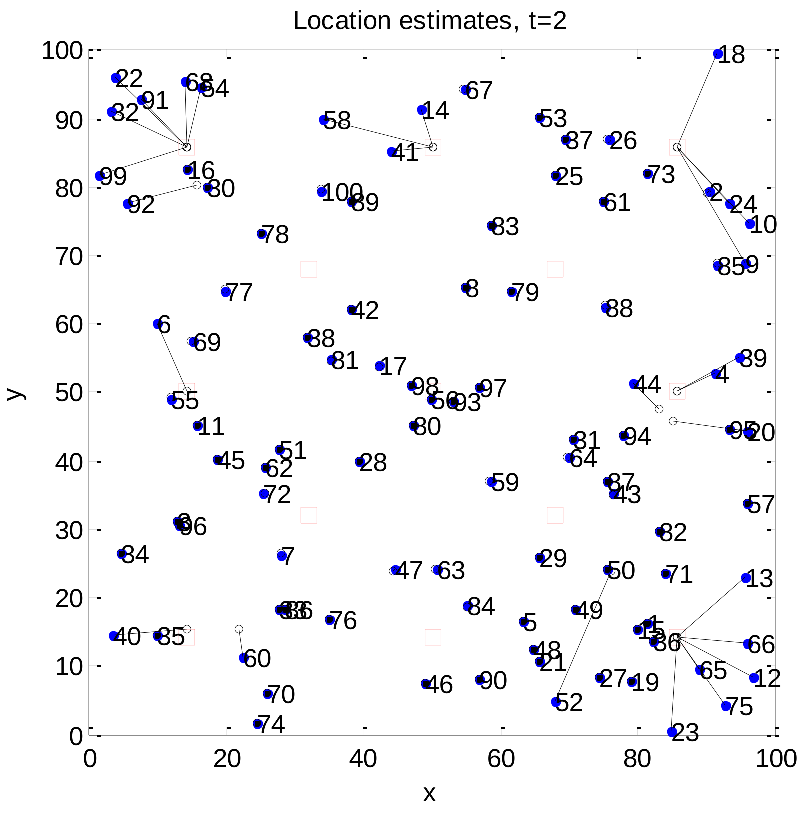


Figure 5.1 Result of Non-Cooperative Positioning

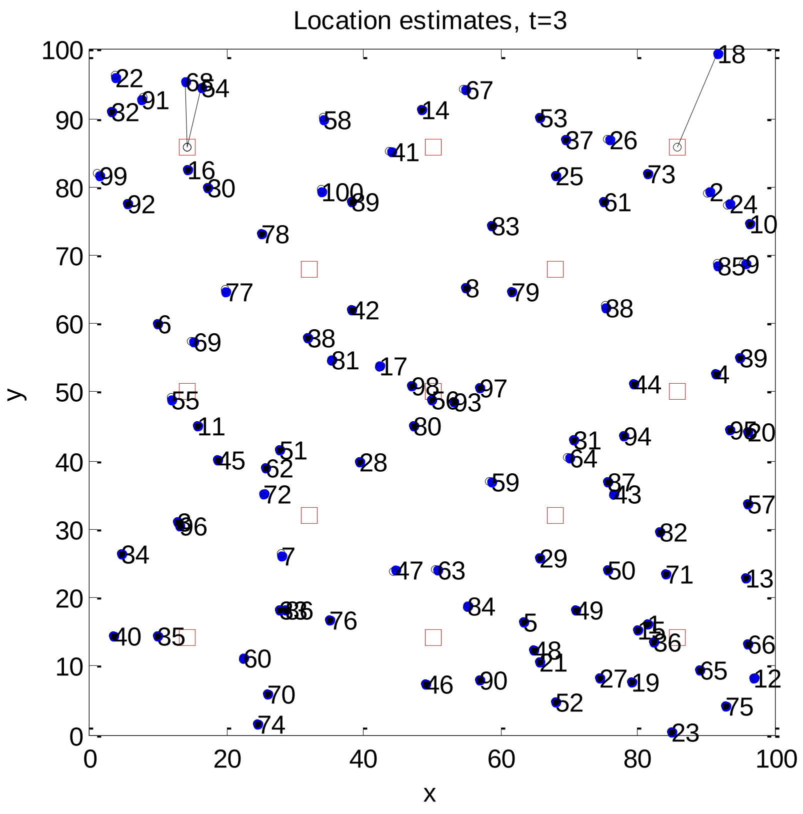
Result of non-cooperative positioning is shown in Fig.5.1. We mark the anchors with the red squares and the true positions of agents with blue dots. The black circles that each links to an agent are the mean values of estimated distributions of the corresponding agents. For example, most agents in the lower right corner (e.g., 13, 66, 75) link to a same anchor, as those agents can only talk to that anchor and return single donut distributions with means being the anchor's position. We also notice that when an agent talks to two anchors, usually it will have a mixture of two Gaussian distribution and the mean will be on the line between these two anchors (e.g., 77). Through each line between we can easily understand the location error. Observe that most agents can not communicate with three or more anchors, and thus have poor positioning estimates.

### 5.2.2 Cooperative Positioning

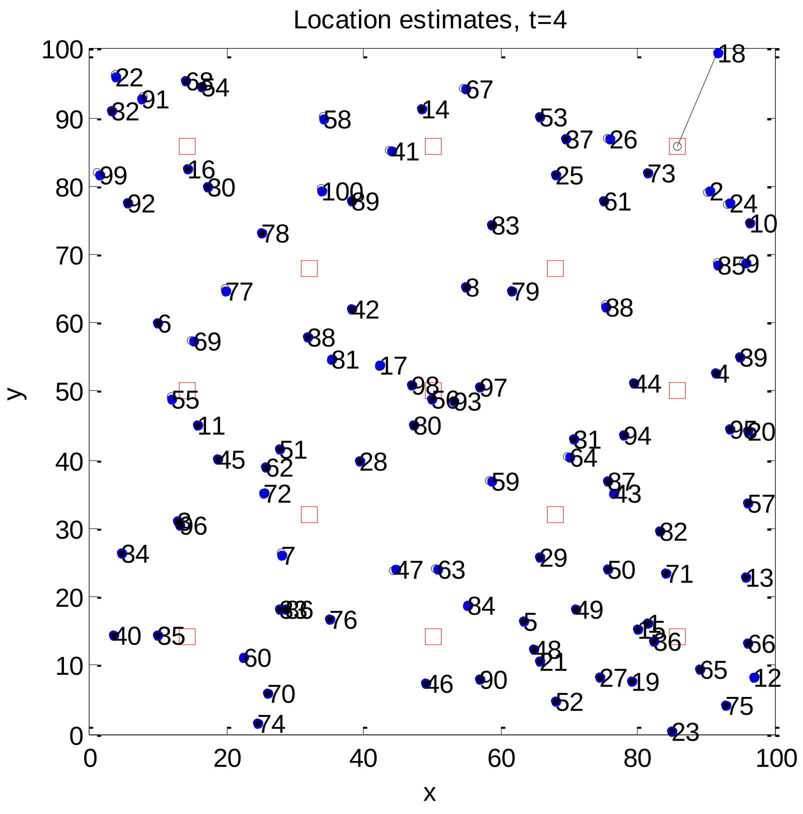
Since the agents have no knowledge about their location at the beginning, in the first iteration ($t = 1$) of cooperative positioning, agents can only receive messages from anchors which leads a same result as the non-cooperative positioning (Fig.5.1). Hence, we only show results here from later iterations.



(a)



(b)



(c)

Figure 5.2 Result of Cooperative Positioning

The result from the second iteration (Fig.5.2 (a)) shows an obvious improvement on the first iteration due to the cooperation between agents. To be more specific, most agents in the middle area can be well localized while agents on the side achieve less improvement. This is because agents in the middle area are more likely to communicate to two or more anchors so that after the first iteration, they can be localized as a mixture of two Gaussian distribution or a single Gaussian distribution which provide useful information for its neighboring agents in later iterations. But for those agents in the corner who only talk to one anchor, they can only send each other information as single donut distributions with means being that anchor's location. Observe that in the lower right corner those agents are communicating to a same anchor, so that all the messages that they send to each other are donut distributions with the same midpoint. Therefore, those agents still can not be localized.

After the third iteration, most agents on the side of the network can be localized with the help of earlier localized agents. For instance, agents in the lower right corner can be localized refer to 19, 27, 71, etc.

Observe that all agents except 18 (18 is not going to be localized as it only talks to one anchor and no agents) are localized with high accuracy after the fourth iteration which proves that the performance of localization can be significantly improved by introducing cooperation between agents.

## 5.3 Results from 25 Networks

### 5.3.1 Accuracy and Coverage

In order to evaluate the performance statistically, we employ a criteria of the outage probability [7]: for a given allowable error $e_{th}$, any agent whose error $\|x_i - \hat{x}_i\|$ exceeds this threshold $e_{th}$ is considered to be in outage. The outage probability is then to examine the percentage of those agents, given by

$$P_{outage}\,(e_{th}) = \mathbb{E}\{\mathbb{I}\{\|x_i - \hat{x}_i\| > e_{th}\}\}, \tag{5.1}$$

where $\mathbb{I}\{P\}$ is the indicator function, being 1 when $P$ is true and 0 when $P$ is false. Estimate $\hat{x}_i$ is taken as the mean of belief as we have mentioned before. The figure below plots with different curves the outage probabilities of all iterations based on 25 networks.

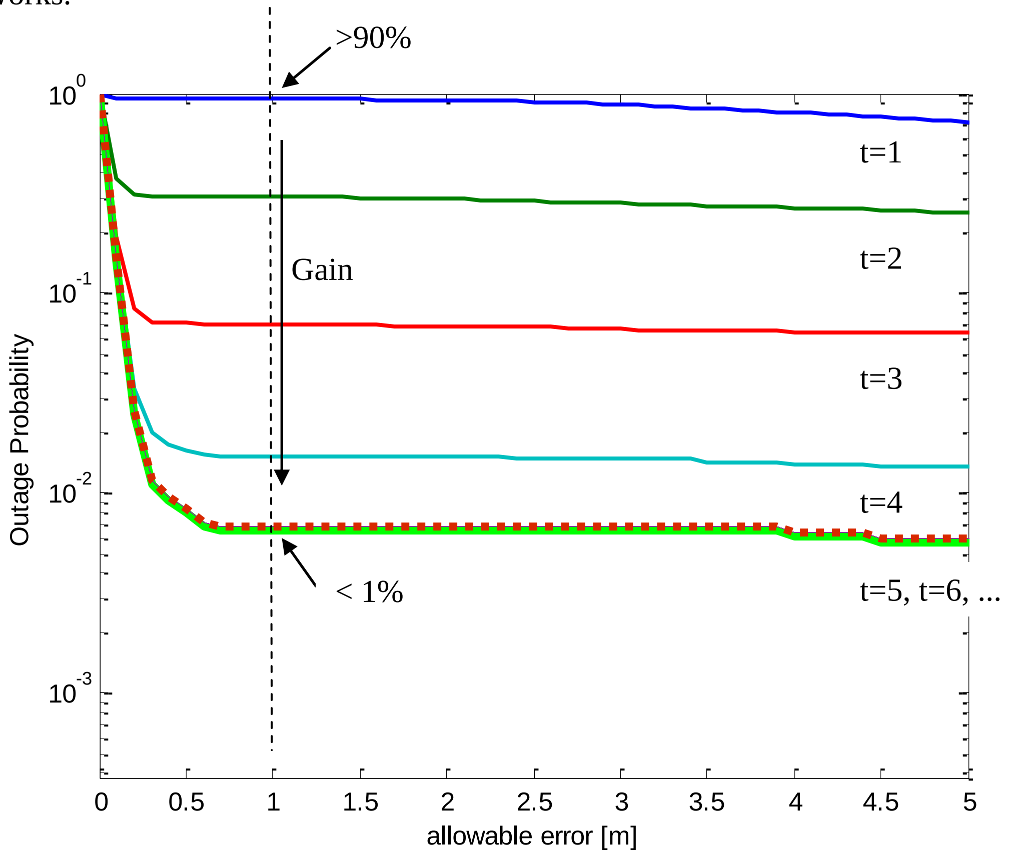


Figure 5.3 Outage Probability Estimated from 25 Networks

We can get the result of non-cooperative positioning through the top line (1st iteration) that over 90% of agents have errors larger than 1 m, while this rate drops below 1% after applying cooperative positioning (see Fig.5.3). Observe that convergence is generally achieved after 5 iterations, and over 99% of agents acquire accuracy estimations with errors less than 0.5m.

### 5.3.2 Complexity

Table 5.1 shows the comparison between the discretized, sample-based, and parametric message representations in terms of complexity. As we have discussed in Section 2.4, M is the number of messages, $N$ is the length of message in discretized and sample-based methods, and in parametric method $N$ refers to the number of samples used for approximating the $\mathcal{D}_2$ distribution to calculate the Kullback-Leibler divergence (Eq. (3.14)). We denote by $\mathcal{O}$ the number of required operations and get:

| **Approach** | **Operation** | **Complexity** | **Size of $N$** |
|---|---|---|---|
| Discretized | Filtering | $\mathcal{O}(N^2)$ | Large |
| Discretized | Multiplication | $\mathcal{O}(NM)$ | Large |
| Sample-based | Filtering | $\mathcal{O}(N)$ | Small |
| Sample-based | Multiplication | $\mathcal{O}(N^2M)$ | Small |
| Parametric | Filtering | $\mathcal{O}(N)$ | Small |
| Parametric | Multiplication | $\mathcal{O}(NM)$ | Small |

Table 5.1 Comparison of Complexity of Three Types of Representations

Note that in discretized representations, $N$ is much larger than for the other two (e.g., $N = 10^8$ compared to $N = 10^3$ for sample-based and parametric representations), which leads to the highest complexity. Observe that for message multiplication, the complexity of the sample-based method scales as $\mathcal{O}(N^2M)$ which is worse than the parametric approach ($\mathcal{O}(NM)$). For example, if we draw 1000 samples ($N = 10^3$) from a distribution, the sample-based approach requires$10^6$ operations while the parametric approach needs only$10^3$. We can thus come to a conclusion that the parametric message representation is more computational efficient compared to non-parametric message representations.

Another way to understand the complexity is to look into the real time cost in our simulation. We have set in our algorithm that if an agent has a single Gaussian distribution, it will not receive messages from other devices in later iteration so that requires little time for updating. Fig.5.3 shows the average time cost by a node from the 1st to the 20th iterations. Observe that the curve reaches its peak at 10 sec per node in the 2nd iteration followed by a consistent falling till the 6th iteration. The reason is that a largest proportion of agents can be localized in the 2nd iteration, leaving less and less nodes to update their beliefs in later iterations. After the 5 iterations, the convergence is achieved so that from the 6th to 20th iterations, the average time per node is only about 0.5 sec.

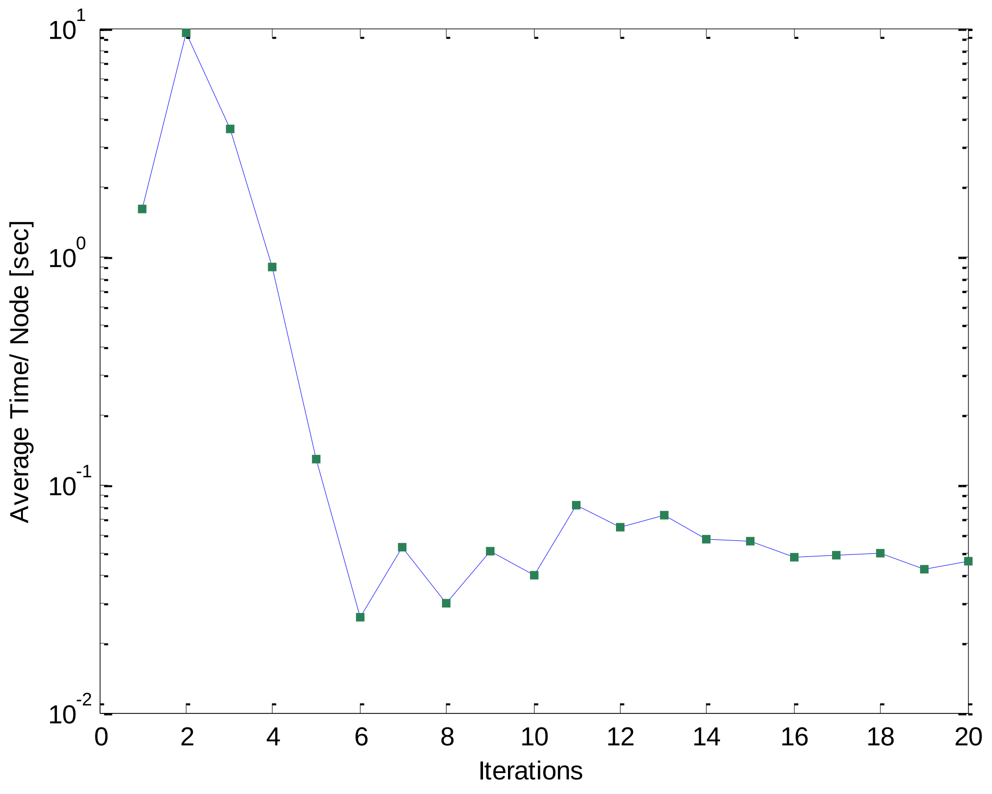


Figure 5.3 Average Time per Node from the 1st to 20th Iterations

# 6. Conclusion and Future Work

Cooperative positioning is an outstanding and potential solution to many wireless network applications that require location-awareness. In our thesis work, we apply framework SPAWN for cooperative localization and design the message representation for this algorithm using a parametric method. The major advantage of parametric representation is the computational efficiency. Besides, we have shown that our representation can well capture the information in most cases and achieve a standout performance in terms of accuracy of location estimation. We have also proved that cooperative positioning can release the dependence on equipping anchors, therefore gains more coverage and flexibility in different environment.

Here in our work, we take the covariance matrix of two dimensional Gaussian distribution as a scalar matrix ( $\sum = \begin{bmatrix} \sigma_{xy}^2 & 0 \\ 0 & \sigma_{xy}^2 \end{bmatrix}$ ), this may be extended to a more general case where the variance of the distribution may have different value in different axis so that distributions such as the "banana" shape can also be well represented. Another research topic that can improve the localization algorithm is known as censoring which removes the unnecessary communication links or the poorly estimated information. This can result in more computationally efficient and accurate positioning for our algorithm.

## Reference

[1] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero III, R. L. Moses, and N. S. Correal, “Locating the nodes: cooperative localization in wireless sensor networks,” IEEE Signal Process. Mag., vol. 22, no. 4, pp. 54–69, Jul. 2005.
[2] C.L.F. Mayorga, F.D. Rosa, S.A. Wardana, “Cooperative Positioning Techniques for Mobile Localization in 4G Cellular Networks”.
[3] E. Elnahrawy, Xiaoyan Li, Richard P. Martin, "Using Area-Based Presentations and Metrics for Localization Systems in Wireless LANs," lcn, pp.650-657, 29th Annual IEEE International Conference on Local Computer Networks (LCN'04), 2004
[4]Guoqiang Mao, Baris Fidan, Localization Algorithm and Strategies for wireless sensor networks, Information Science Reference, 2009
[5] D. Mirza and C. Schurgers, “Motion-aware self-localization for underwater networks,” in Proceedings of the third ACM international workshop on wireless network testbeds, experimental evaluation and characterization. New York, NY, USA: ACM, pp. 51–58, 2008.
[6] A. T. Ihler, J. W. Fisher III, R. L. Moses, and A. S. Willsky, “Nonparametric belief propagation for self-localization of sensor networks,” IEEE J. Sel. Areas Commun., vol. 23, no. 4, pp. 809–819, Apr. 2005.
[7] H. Wymeersch, J. Lien, and M. Z. Win, “Cooperative localization in wireless networks,” Proceedings of the IEEE, vol. 97, no. 2, pp. 427–450, Feb 2009.
[8] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, “Factor graphs and the sum-product algorithm,” IEEE Trans. Inf. Theory, vol. 47, no. 2, pp. 498–519, Feb. 2001.
[9] S. Lanzisera and K. S. J. Pister, "Burst Mode Two-Way Ranging with Cramer-Rao Bound Noise Performance," in Global Telecommunications Conference, pp. 1-5, 2008.
[10] Richard S. Sutton, Andrew G. Barto, Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 1998.