POLITECNICO DI MILANO
MSc Degree in Computer Science and Engineering
Dipartimento di Elettronica, Informazione e Bioingegneria

A SURVEY ON SHORT-RANGE RSSI-BASED GEOLOCALIZATION

Relatore: Prof. Lorenzo Mario Fagiano

Tesi di Laurea di:
Alberto Tiraboschi, matricola 920775

Academic Year 2020/2021

Contents

1 Introduction **1**
1.1 Environment settings 1

2 Mathematical models of RSSI propagation **3**
2.1 Path loss model 3

3 Disturbances of RSSI **5**
3.1 Multipath Fading 5
3.2 Shadow Fading 6

4 Noise removal **7**
4.0.1 IQR 7
4.0.2 Z-score 10

5 Algorithms for parameters estimation **12**
5.1 Estimation of path loss index α 12
5.2 Estimation of σ_i^2 14

6 Algorithms for position estimation **15**
6.1 Common Paradigm 15
6.2 Classification 15
6.3 Trilateration 18
6.4 Min Max 21
6.5 Linear estimation 23
6.6 Weighted linear estimation 27
6.7 Quadratic linearization 30
6.8 Lines intersection 33
6.9 Multilateration 37
6.10 Maximum Likelihood method 39
6.11 Centroid algorithm 41
6.12 DV-Hop Algorithm 44

6.13 Fingerprinting 46

7 Conclusions **48**

Bibliography **51**

Chapter 1

Introduction

Finding the location of unknown objects has been since a long time a topic under investigation. The applications are vast and span many different topics, from security, to discover unauthorized intrusions, to warfare, to discover nearby opponents, or detecting submarines, but also to localize someone who needs to be rescued, for example in disaster response. With the recent advances in technology it has been possible to locate objects using low-cost technology, available to almost every person in the world, the Wi-Fi. The main purpose of this thesis is to tidy up a little the actual state of the art of the RSSI-based localization algorithms. Here I will do a revisitation of the most important algorithms, trying to mediate from theory to application, without losing mathematical accuracy, by adapting the theoretical concepts in a form that is ready to be implemented algorithmically in whichever language it is preferred. I will also add at the top of the description of each algorithm which are the parameters that one needs to know beforehand.

1.1 Environment settings

The main setting that will be presented is the most commonly used up to now, which is a space filled with receiving/transmitting devices (anchor nodes), each placed in a regular position (usually a grid) with known positions, that can read the signal intensity of the other nodes. The activity starts when a device with an unknown position is put in the area and many (if not every) nodes are close enough to get the signal intensity of that unknown node. For commercial devices with average transmitting power and average amplification gain, this range of sensitivity can be seen as a circle of radius of about 50 meters in open environment. At this point, each anchor node can communicate a tuple like this (x_i, y_i, v_i^u), where x_i, y_i are the position of the node itself, and v_i^u is the signal strength intensity of the unknown node, measured by the anchor node i. Usually, the data is sent to a central processing device that makes the computations and outputs the estimated position.

This is not the only way however, the other way, which is lately becoming increasingly used is the employment of a mobile robot, e.g. a ground or aerial drone, that given some bounds on the area to scan, collects and outputs the samples, behaving as a "moving" anchor node. This thesis is organized as follows. Chapter 2 describes the most common mathematical models of RSSI propagation. In chapter 3 we will see which are the main disturbances that affects the RSSI measurement. In Chapter 4 instead we will take a look on how to remove the noise of the signal. Chapter 5 and 6 are the core of the thesis and consist of a review of the state of the art of the main parameter estimation and localization algorithms. In the final chapter there will be the conclusions of this work along with some personal opinions.

Chapter 2

Mathematical models of RSSI propagation

2.1 Path loss model

RSSI is an acronym for Received Signal Strength Indicator. It is measured in dbm (decibel-milliwatts). Given a transmitting device j and a receiving device i, placed at distance d_i^j to one another, the receiver device can measure the RSSI υ_i^j of the transmitting device. Those readings (according to the log-normal propagation model [Mun+09]) are of the form

$$\upsilon_i^j = \upsilon_0^j - 10\alpha \log_{10}\left(\frac{d_i^j}{d_0}\right) + \xi_i, \quad d_i^j > d_0 \tag{2.1.1}$$

$$d_i^j = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{2.1.2}$$

where ξ_i is a zero-mean gaussian variable ($\mathcal{N}(0, \sigma_i^2)$) that represents the noise, α is the path loss index (see Section 3.2), which can vary between 2 (open field) and 4 (environment fitted with obstacles) and υ_0^j is the RSSI read at d_0. Commonly d_0 is taken as 1 meter, so that we can simplify, the (2.1.1) as:

$$\upsilon_i = \upsilon_0 - 10\alpha \log_{10}(d_i) + \xi_i, \quad d_i > d_0 \tag{2.1.3}$$

where the letter j is usually dropped. Some algorithms don't require high precision, so they usually don't take into account the noise, whereas for some others it is necessary to rewrite the model as follows

$$\upsilon_i = \upsilon_0 - \beta \ln(d_i) + \xi_i \tag{2.1.4}$$

with

- $\beta = 10\alpha / \ln(10)$
- $\ln(\bullet)$ as the natural logarithm

and then deriving

$$\nu_i = \upsilon_i - \upsilon_0 = -\beta \ln(d_i) + \xi_i \tag{2.1.5}$$

Chapter 3

Disturbances of RSSI

The RSSI noise has a great influence over the measurements, thereby complicating the attempt to get a precise location. To give some numbers, the theoretical range within the RSSI can vary goes from -30 to -120 however the available range of signal intensity that a common WiFi device can measure varies in the interval $-30 \div -90$ dbm. This is because to establish a connection (and to retrieve at least the name of the transmitting device), some packets must be sent, and if the signal strength is too low, the quality of the packets degrades, therefore making impossible communication. The value measured at the receiving end includes also the noise, so the measurement deviates from the exact value generally by at most 6 dbm which is as much as 10% over the range of measurement. Due to its significant influence, this issue has been investigated many times to discover something more, for example about its probability distribution, but up to now, few to no answers have been found. To cite one, [Wu+08] showed that there are no significant patterns over the time domain neither in the frequency domain. The causes of the noise are the same that affect the propagation of any electromagnetic wave. We can summarize the main sources of noise in the following sections:

3.1 Multipath Fading

As outlined in [Rub], multipath fading occurs when there are multiple ways for the signal to reach the receiving device. This causes the signal waves to reach the receiving end at two different times, that is, there is a non-negligible delay between the time of arrival of the two (or more) waves. This means phase shifting, leading to constructive/destructive interference, therefore causing oscillations in the signal intensity read.

This effect is very prominent in closed environments when there are many surfaces able to reflect the signal. It is totally random [Rap01] which means unpredictable by definition. Due to its nature, the alterations of the signal caused are like spikes above or below the mean value. Therefore it can be removed to some extent by filtering the outliers with some of the methods presented later.

3.2 Shadow Fading

This effect is the attenuation of the signal caused by heavy and large (w.r.t. the wavelength) obstacles between the receiver and the sender that can adsorb the waves partially, or even totally. This is directly related to NLOS (Non-Line Of Sight) environments that is, when the receiver cannot "see directly" the sender. The shadow fading effect is taken into account in the model as the path loss index α (see Section 5.1). This parameter is easily obtained in an open field with no obstacles, whereas it is quite a problem when some obstacles are obstructing the way. In the latter case, the analytical way (by considering the absorption coefficients of the obstacles) is rarely used, instead, a somehow correct path loss index is usually estimated with some algorithms, one of them is presented later.

Shadow fading presents spatial correlation [LR92; ZG98], which is intuitive, all the measurements taken in a certain position, with a certain number of obstacles will be all affected by an almost equal shadow fading, whereas in another location it will not. The problematic effect of this noise is that it shows a systematic behavior, in other words, each measurement is affected by an almost equal amount, therefore trying to remove it by removing outliers is not an effective way. If the path loss index is taken wrong in the model for some reason, the estimated position will be altered with some significant errors, that may invalidate the improvements taken to reduce the other sources of noise. Even more complicated is when there are persons moving freely between the sender and the receiver. The body of each passing person adsorbs the signal, therefore altering the measure. The estimated position will bear some non-negligible errors.

Chapter 4

Noise removal

Here I will present some algorithms to remove outliers in a series of data. Those methods, given their independence from the RSSI can be used to remove outliers of any series of data.

4.0.1 IQR

This method, whose name is InterQuartile Range, is based, as the name suggests, on the quartiles of a dataset. First, let's start with an algorithm that can be used to obtain the quartiles. I will assume some facts:

- Arrays index starts from zero
- Float numbers are cast to integer with truncation of the decimal part, e.g.

$$1.2 \to 1$$

$$-1.8 \to -1$$

- Datasets
  - d1, d2, d3
- Size of each dataset
  - s1, size of d1
  - s2, size of d2
  - s3, size of d3

Algorithm 1: Quartiles computation

```
Result: Q1, Q2, Q3
d1.sort(ascending);
Q1 = Q2 = Q3 = 0;
s1 = d1.size();
if s1 is even then
    Q2 = (d1[s1/2 - 1] + d1[s1/2])/2;
    d2 = d1[0 : s1/2 - 1];
    d3 = d1[s1/2 : s1 - 1];
else
    Q2 = d1[s1/2];
    d2 = d1[0 : s1/2 - 1];
    d3 = d1[s1/2 + 1 : s1 - 1];
end
s2 = d2.size();
s3 = d3.size();
if s2 is even then
    Q1 = (d2[s2/2 - 1] + d2[s2/2])/2;
else
    Q1 = d2[s2/2];
end
if s3 is even then
    Q3 = (d3[s3/2 - 1] + d3[s3/2])/2;
else
    Q3 = d3[s3/2];
end
```

A little explanation here is required. At the beginning, the algorithm sorts the initial dataset $d1$ in ascending order. We will take the median ($Q2$) of $d1$. Then we have two possible paths. If $s1$, size of $d1$, is odd, then the median is at $1/2s$, the central position (remind the assumption about array indexing and decimal casting). If it is not, the median will be the average of the two elements at the center. Either the cases, we get two datasets, the upper half ($d2$) and the lower half ($d3$), separated by the median. Then the operation is repeated for each one of $d2, d3$, to obtain $Q1$ median of the upper half, and $Q3$ median of the lower half. As can be seen, 25% of data is below $Q1$, 50% of data is below $Q2$, which is the common median, and 75% of data is below $Q3$.

Going back to the IQR method the algorithm goes through the following steps:

```
Algorithm 2: Outliers removal with IQR
   Result: The filtered dataset
 1 Q1, Q2, Q3 known from the previous algorithm;
 2 IQR = Q3 - Q1;
 3 $I = [Q1 - 1.5 \cdot IQR, \quad Q2 + 1.5 \cdot IQR]$;
 4 k = 0;
 5 while $iter \in [0 : n - 1]$ do
 6     if ! $dataset[k] \in I$ then
 7         dataset.remove(k);
 8         k = k - 1;
 9     end
10     k = k + 1;
11     iter = iter + 1;
12 end
```

Here it is given a numeric example. Suppose you have the following dataset

$$\text{dataset} = \begin{bmatrix} -1.9 \\ 0.3 \\ 3.7 \\ 4.1 \\ 6 \\ 6.1 \\ 7.2 \\ 9 \\ 19 \end{bmatrix}$$

already sorted, where the first seven numbers are extracted from $-2+X$, where X follows the uniform distribution in $[0, 10]$. The latest two values are possible outliers. Since the size of the dataset is odd, $Q2$ can be found at the fifth position and is 6. We have then the first half and the second half. Since the first half has an even number of data points, the median of the first half ($Q1$) will be $(0.3+3.7)/2 = 2$, the same reasoning applies to $Q3$ for the second half, which is equal to 8.1. Therefore the IQR will be $8.1 - 2 = 6.1$ the interval I will be

$$I = [2 - 1.5 \cdot 6.1, 8.1 + 1.5 \cdot 6.1] = [-7.15, 17.25]$$

As you can see, the number 19 will be treated as an outlier and thus be removed. You can also choose how wide the interval you want it to be, for example, if you wanted to be more strict you could choose 1 instead of 1.5 as a multiplying factor, obtaining the following interval

$$I' = [2 - 1 \cdot 6.1, 8.1 + 1 \cdot 6.1] = [-4.1, 14.2]$$

4.0.2 Z-score

The Z-score is another method to find outliers. The procedure estimates the sample mean and sample standard deviation as follows

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{4.0.1}$$

$$\hat{\sigma} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \hat{\mu})^2} \tag{4.0.2}$$

Then it computes the z-score y_i of each value x_i

$$y_i = \frac{x_i - \hat{\mu}}{\hat{\sigma}} \tag{4.0.3}$$

The filtering process now removes every data whose z-score is outside the interval $[-3, 3]$.

Here an example, with the same dataset of the previous algorithm

$$\text{dataset} = \begin{bmatrix} -1.9 \\ 0.3 \\ 3.7 \\ 4.1 \\ 6 \\ 6.1 \\ 7.2 \\ 9 \\ 19 \end{bmatrix}$$

The estimated mean is $\hat{\mu} = 5.94$ and the sample standard deviation is $\hat{\sigma} = 5.95$. Then the corresponding set of z-scores is

$$\text{z-scores} = \begin{bmatrix} -1.32 \\ -0.95 \\ -0.38 \\ -0.31 \\ 0.01 \\ 0.03 \\ 0.21 \\ 0.51 \\ 2.19 \end{bmatrix}$$

note the last value, for the previous algorithm it should have been discarded, but according to this one, it is still kept. Clearly, you can choose a narrower interval than $[-3, 3]$ if you want to be more strict on the outliers.

Chapter 5

Algorithms for parameters estimation

5.1 Estimation of path loss index α

Assume to know:		
υ_0	$\xi_i, \forall i$	$d_i^j, \forall i$

Place a transmitting device $j, j > n$ at known position. For the anchor node $i, 1 \leq i \leq n$, the values of signal intensity of the node j are obtained according to (2.1.3). The referenced model is basically a constant term $[\upsilon_0 - 10\alpha \log_{10}(d_i)]$ added to a zero mean gaussian variable $[\xi_i]$, so from [Ros20], let $a \in \mathbb{R}$

$$a + \mathcal{N}(0, \sigma^2) \sim \mathcal{N}(a, \sigma^2) \tag{5.1.1}$$

therefore the (2.1.3) is gaussian as follows:

$$X_i \sim \mathcal{N}(\mu_i, \sigma_i^2) \tag{5.1.2}$$

where

$$\mu_i = \upsilon_0 - 10\alpha \log_{10}(d_i) \tag{5.1.3}$$

which has probability density function

$$p_i(x) = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{1}{2}\frac{(x-\mu_i)^2}{\sigma_i^2}} \tag{5.1.4}$$

Given a sample of n measurements

$$\upsilon_1, ..., \upsilon_n$$

one for each node, where each υ_i follows (5.1.2). Then we can employ the Maximum Likelihood to estimate the path loss index. It follows

$$L_n(\alpha, \sigma_1^2, ..., \sigma_n^2) = \prod_{i=1}^{n} P(X_i = \upsilon_i) = \frac{1}{(2\pi)^{\frac{n}{2}} (\prod_{i=1}^{n} \sigma_i^2)^{\frac{1}{2}}} e^{-\frac{1}{2}\sum_{i=1}^{n} \frac{(\upsilon_i - \mu_i)^2}{\sigma_i^2}} \tag{5.1.5}$$

by applying the logarithm to both members

$$\ell_m(\alpha, \sigma_1^2, ..., \sigma_n^2) = -\frac{n}{2}\ln(2\pi) - \frac{1}{2}\ln\left(\prod_{i=1}^{n} \sigma_i^2\right) - \frac{1}{2}\sum_{i=1}^{n} \frac{(\upsilon_i - \mu_i)^2}{\sigma_i^2} \tag{5.1.6}$$

Maximizing ℓ_n^i w.r.t to α is equal to finding the stationary point of the third term in the above equation since the first and second one are independent from α therefore [Mun+09]

$$\frac{\partial}{\partial\alpha}\frac{1}{2}\sum_{i=1}^{n} \frac{(\upsilon_i - \mu_i)^2}{\sigma_i^2} = 0 \tag{5.1.7}$$

$$\sum_{i=1}^{n} \frac{\upsilon_i \log_{10}(d_i)}{\sigma_i^2} - \upsilon_0 \sum_{i=1}^{n} \frac{\log_{10}(d_i)}{\sigma_i^2} + 10\alpha \sum_{i=1}^{n} \frac{(\log_{10}(d_i))^2}{\sigma_i^2} = 0 \tag{5.1.8}$$

$$\alpha = \frac{\upsilon_0 \sum_{i=1}^{n} \frac{\log_{10}(d_i)}{\sigma_i^2} - \sum_{i=1}^{n} \frac{\upsilon_i \log_{10}(d_i)}{\sigma_i^2}}{10 \sum_{i=1}^{n} \frac{(\log_{10}(d_i))^2}{\sigma_i^2}} \tag{5.1.9}$$

Sometimes to ease the computation, σ_i^2 is assumed constant

$$\sigma_1^2 = ... = \sigma_n^2 = \sigma^2$$

then the equation simplifies as

$$\alpha = \frac{\upsilon_0 \sum_{i=1}^{n} \log_{10}(d_i) - \sum_{i=1}^{n} \upsilon_i \log_{10}(d_i)}{10 \sum_{i=1}^{n} (\log_{10}(d_i))^2} \tag{5.1.10}$$

5.2 Estimation of σ_i^2

Assume to know:		
υ_0	α	d_i^j

In this case, instead, we consider only one node, since we are interested to estimate only σ_i^2 which is the variance of node i. So after placing a transmitting device j as before, we have m samples from node i

$$\upsilon_{i,1}, ..., \upsilon_{i,m}$$

each of them follows the distribution

$$X_i \sim \mathcal{N}(\mu_i, \sigma_i^2) \tag{5.2.1}$$

where

$$\mu_i = \upsilon_0 - 10\alpha \log_{10}(d_i) \tag{5.2.2}$$

Accordingly, the pdf is

$$p_i(x) = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{1}{2}\frac{(x-\mu_i)^2}{\sigma_i^2}} \tag{5.2.3}$$

The Maximum Likelihood function is

$$L_m(\alpha, \sigma_i^2) = \prod_{j=1}^{m} P(X_i = \upsilon_{i,j}) = \frac{1}{(2\pi\sigma_i^2)^{\frac{m}{2}}} e^{-\frac{1}{2}\sum_{j=1}^{m}\frac{(\upsilon_{i,j}-\mu_i)^2}{\sigma_i^2}} \tag{5.2.4}$$

by applying the logarithm to both members

$$\ell_m(\alpha, \sigma_i^2) = -\frac{m}{2}\ln(2\pi\sigma_i^2) - \frac{1}{2\sigma_i^2}\sum_{j=1}^{m}(\upsilon_{i,j} - \mu_i)^2 \tag{5.2.5}$$

Its derivative w.r.t σ_i^2 set to 0 has the following form

$$\frac{-m\sigma_i^2 + \sum_{j=1}^{m}(\upsilon_{i,j} - \mu_i)^2}{2\sigma_i^4} = 0 \tag{5.2.6}$$

Since $\sigma_i^2 > 0$ we have

$$\hat{\sigma}_i^2 = \frac{1}{m}\sum_{j=1}^{m}(\upsilon_{i,j} - \mu_i)^2 \tag{5.2.7}$$

Chapter 6

Algorithms for position estimation

6.1 Common Paradigm

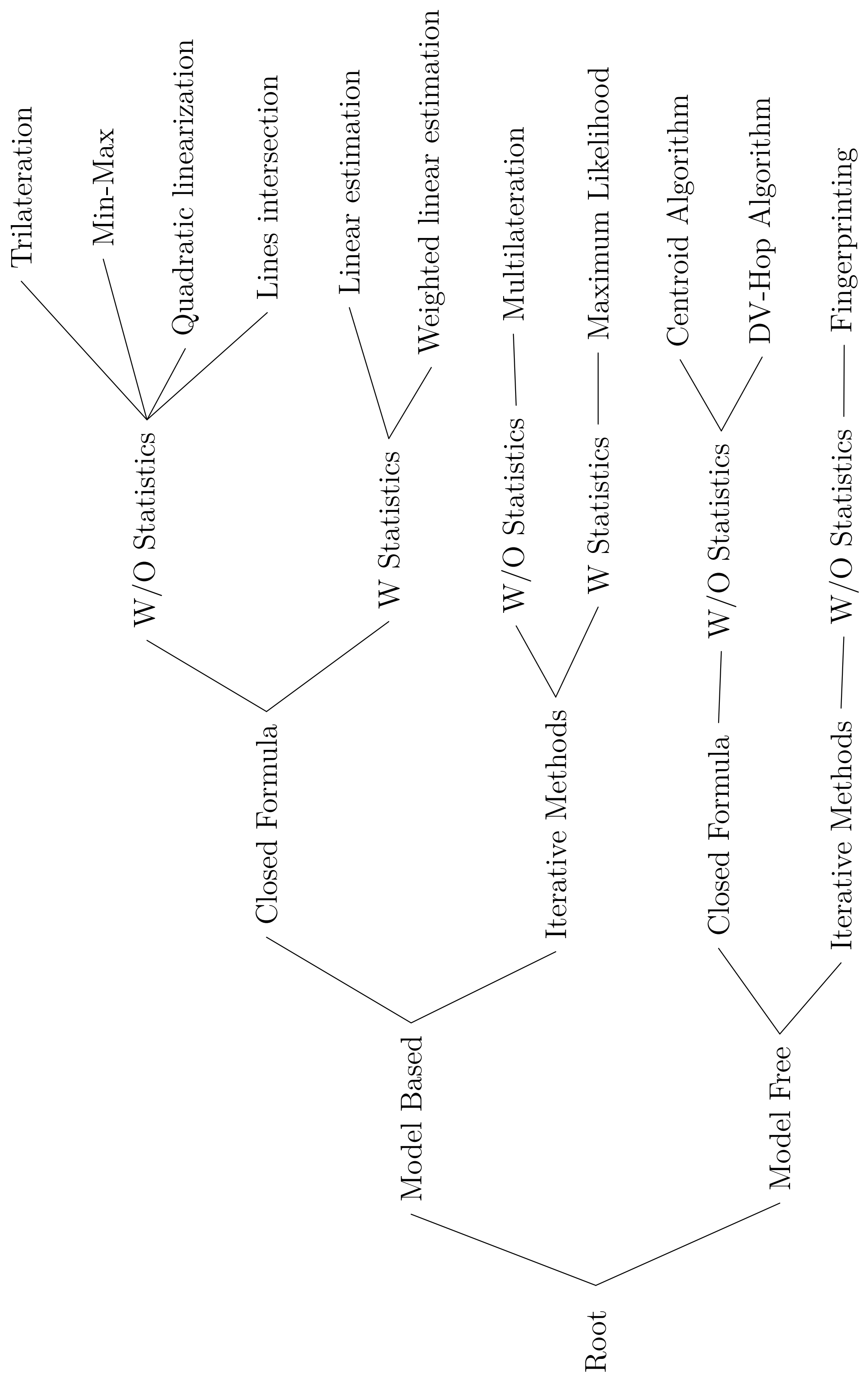
To avoid redundancy, given the fact that each algorithm differentiates from the others only because of the mathematic tools employed, it makes sense to focus mainly on it in the description of the algorithms and to present here the common instructions. When instead the steps are quite different, there will be a more detailed description of all the procedures.

Algorithm 3: Common steps of localization algorithms

```
1  Collect the signal strength;
2  Possibly pre-process the data;
3  if the method has a closed formula then
4      apply the formula and obtain $\hat{\mathbf{x}}$
5  else
6      while ! convergence OR ! max iteration reached do
7          update $\hat{\mathbf{x}}$;
8      end
9  end
10 return results;
```

6.2 Classification

Here follows a classification of the localization methods considered. Model-based means that the algorithms children of that node employ directly the model of log propagation seen in Chapter 2, whereas model-free don't use it. Closed formula/iterative methods are quite self-explaining. Regarding statistics, the sub-classification is based on the employment of probabilistic parameters, one above all, the variance σ_i^2 of the noise error ξ_i. Of course, model-free methods, since they do not use the model above, they do not consider either the variance of the noise or any other probabilistic parameters.


Root
Model Based
Model Free
Closed Formula
Iterative Methods
Closed Formula
Iterative Methods
W/O Statistics
W Statistics
W/O Statistics
W Statistics
W/O Statistics
W/O Statistics
Trilateration
Min-Max
Quadratic linearization
Lines intersection
Linear estimation
Weighted linear estimation
Multilateration
Maximum Likelihood
Centroid Algorithm
DV-Hop Algorithm
Fingerprinting

6.3 Trilateration

Assume to know:	
υ_0	α

Consider the trilateration method from a purely theoretical point of view. For this method, it is required to have only 3 anchor nodes. Each of them can output its position (x_i, y_i) along with the RSSI read in its position of the unknown node. By exploiting the model of the distance and the RSSI read, and assuming no noise or multipath fading at all, one can deduce the distance d_i between the unknown node $u, u > n$ and current node from the model (2.1.3) as follows:

$$d_i = \sqrt{(x_u - x_i)^2 + (y_u - y_i)^2} = 10^{\frac{\upsilon_0 - \upsilon_i}{10\alpha}} \tag{6.3.1}$$

As a consequence we can imagine to draw a circle centered in (x_i, y_i) with radius d_i. The unknown node u should lay on this circumference. Of course, this is not enough to locate u. Therefore, we need to get data also from the other 2 anchor nodes. In this way, we can draw 3 circles and obtain a unique point of intersection as shown in fig. 6.1. Note that 3 nodes are the minimum number of nodes to have an unambiguous result. In fact, if we were to have only 2 nodes, then there would be 2 possible outcomes (the intersection of two circumferences generally results in 2 points). When we step into reality however the disturbances of the signal inevitably lead to wrong estimation of the distance between the current node and the unknown node, causing the circles to intersect in wrong points or to not intersect at all, as in fig.6.2. In the latter case, provided the algorithm has no software-level exception-catching, it would cause a runtime error and give no results at all. However, some improvements can be applied. The most intuitive one is to consider from the RSSI read, not just a single value but taking the max and the min values recorded during the sampling phase, and draw an annulus of possible locations as follows:

Algorithm 4: Obtaining the derived values

```
  Result: min, max: radii of the annulus
1 Set samples = $\emptyset$;
2 while samples.size() $<$ 10 do
3     samples.add(new measurement);
4 end
5 min = samples.min();
6 max = samples.max();
```

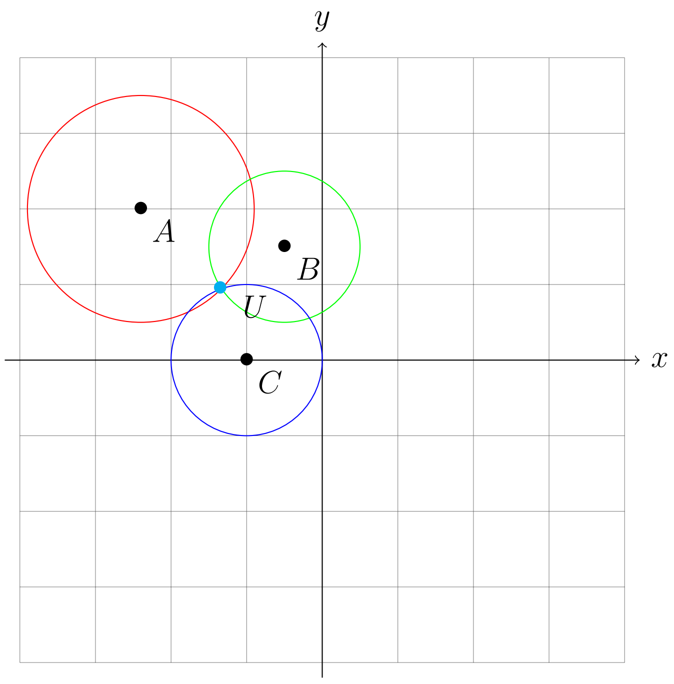


Figure 6.1: Trilateration with noise $= 0$

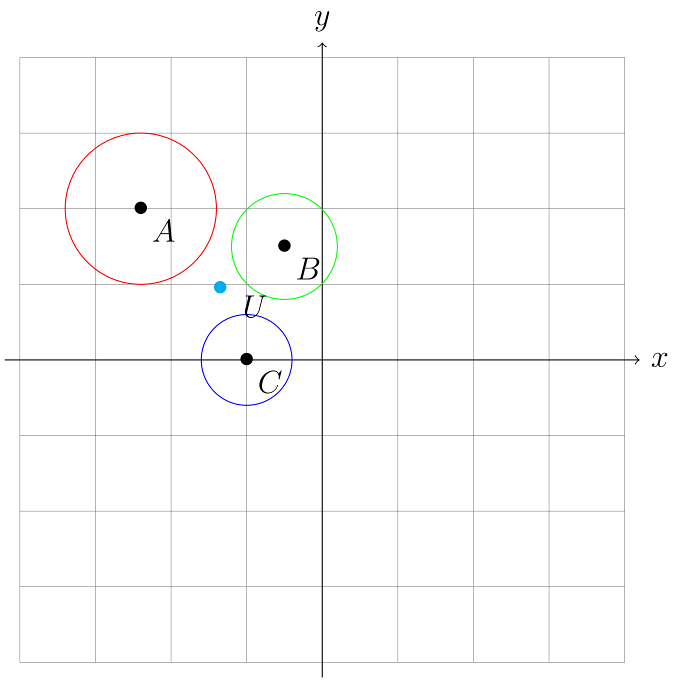


Figure 6.2: Trilateration with noise $\neq 0$

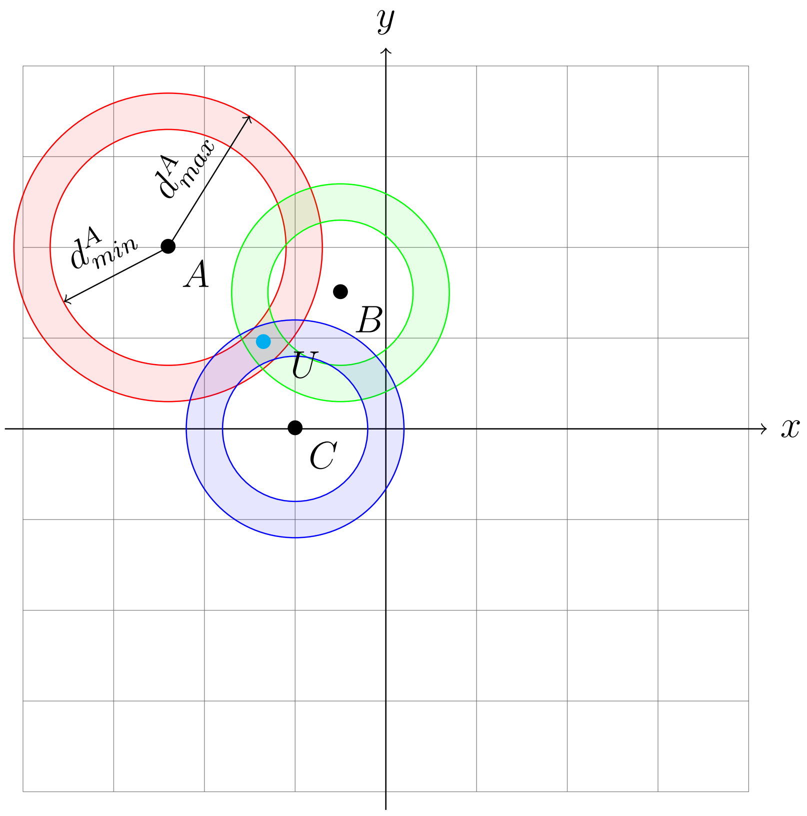


Figure 6.3: Trilateration with noise compensation

The new distances will be obtained as follows:

$$d_i^{min} = 10^{\frac{v_0 - min}{10\alpha}} \tag{6.3.2}$$

$$d_i^{max} = 10^{\frac{v_0 - max}{10\alpha}} \tag{6.3.3}$$

The result is now given as an area that corresponds to the intersection of all the three annuli, shown in fig 6.3. Although the results are quite promising, this algorithm is rarely used, due to the computation difficulties of treating annuli. One important drawback is the low number of measurements exploited. Clearly, a higher number of samples implies a lower influence of the noise on the final estimation.

6.4 Min Max

Assume to know:	
v_0	α

Consider the previous framework. As usual one can deduce the distance of the unknown node, by reverting the propagation model as done in the previous section. Then we can obtain a number of circumferences, one for each anchor node, that represent the possible locations of the unknown node. In this algorithm, however, you need to consider not circles but squares. The development of this algorithm follows [Rat+15]. Each anchor node i with estimated distance d_i from node $u, u > n$ (with (6.3.1)) is associated to a square centered in (x_i, y_i) with side equal to $2d_i$. At this point, one has a set of overlapping squares, and can easily find the intersection of all those squares, which results in a rectangles of vertices $(x_{min}, y_{min}), (x_{max}, y_{min}), (x_{max}, y_{max}), (x_{min}, y_{max})$ with the following equations:

$$x_{min} = \max(x_1 - d_1, ..., x_n - d_n) \tag{6.4.1}$$

$$x_{max} = \min(x_1 + d_1, ..., x_n + d_n) \tag{6.4.2}$$

$$y_{min} = \max(y_1 - d_1, ..., y_n - d_n) \tag{6.4.3}$$

$$y_{max} = \min(y_1 + d_1, ..., y_n + d_n) \tag{6.4.4}$$

The final location will be estimated as the center of this rectangle, as follows:

$$x_u = \frac{x_{min} + x_{max}}{2} \tag{6.4.5}$$

$$y_u = \frac{y_{min} + y_{max}}{2} \tag{6.4.6}$$

The gray area in 6.4 shows the intersection of all the squares.

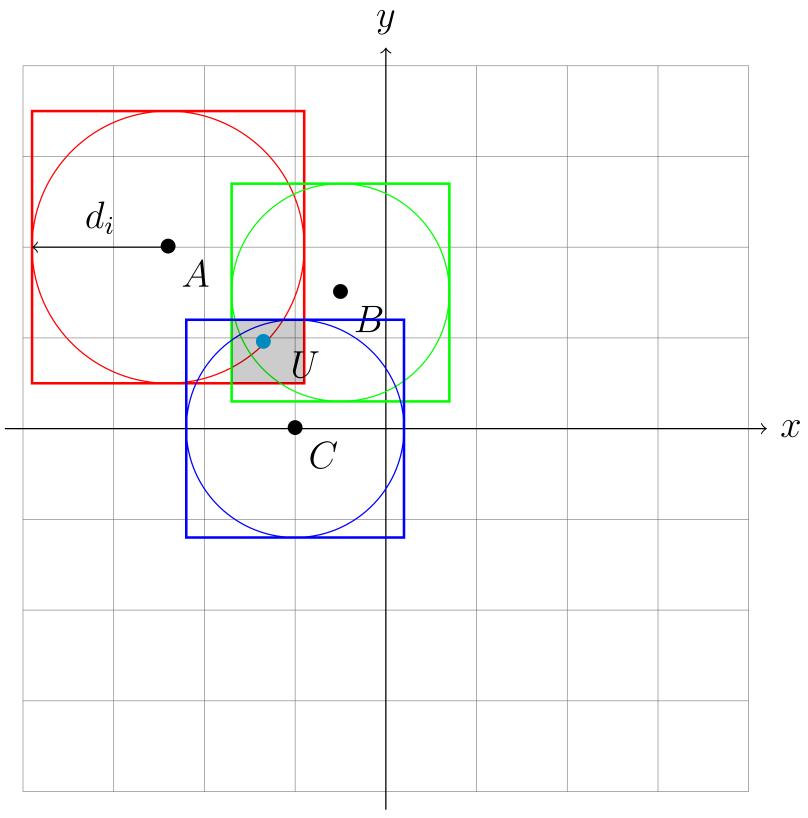


Figure 6.4: Min-Max estimation

6.5 Linear estimation

Assume to know:		
υ_0	α	$\xi_i, \forall i$

In [ZB19] you can find this version but using a slightly different model. Consider the model (2.1.5) below reported

$$\nu_i = -\beta \ln(d_i) + \xi_i \tag{6.5.1}$$

After some time, for each anchor node, we will have a list of measurements, drawn from the random variable

$$-\beta \ln(d_i) + \xi_i \sim \mathcal{N}(-\beta \ln(d_i), \sigma_i^2) \tag{6.5.2}$$

Rewrite the (6.5.1) as follows

$$\nu_i = -\ln\left(d_i^{2\frac{\beta}{2}}\right) + \xi_i \tag{6.5.3}$$

By exponentiating both members of (6.5.3) we have

$$e^{\nu_i} = e^{-\ln\left(d_i^{2\frac{\beta}{2}}\right)+\xi_i} \tag{6.5.4}$$

$$\left(e^{\nu_i}\right)^{-1} = \left(e^{-\ln\left(d_i^{2\frac{\beta}{2}}\right)+\xi_i}\right)^{-1} \tag{6.5.5}$$

$$e^{-\nu_i} = e^{\ln\left(d_i^{2\frac{\beta}{2}}\right)-\xi_i} \tag{6.5.6}$$

$$e^{-\nu_i} = d_i^{2\frac{\beta}{2}} e^{-\xi_i} \tag{6.5.7}$$

$$\left(e^{-\nu_i}\right)^{\frac{2}{\beta}} = \left(d_i^{2\frac{\beta}{2}} e^{-\xi_i}\right)^{\frac{2}{\beta}} \tag{6.5.8}$$

$$e^{-\frac{2}{\beta}\nu_i} = d_i^2 e^{-\frac{2}{\beta}\xi_i} \tag{6.5.9}$$

Remind that the exponentiation of a gaussian distribution is [BG11] a log-normal distribution. In other words, given $X \sim \mathcal{N}(\mu, \sigma^2)$, then $Y = e^X \sim \text{Lognormal}(\mu, \sigma^2)$ with

$$E[Y] = E\left[e^X\right] = e^{\mu+\frac{\sigma^2}{2}} \tag{6.5.10}$$

$$Var[Y] = Var\left[e^X\right] = (e^{\sigma^2} - 1)e^{2\mu+\sigma^2} \tag{6.5.11}$$

From [Ros20], it is clear also that if $a \in \mathbb{R}$ and $X \sim \mathcal{N}(\mu, \sigma^2)$ then

$$aX \sim \mathcal{N}(a\mu, a^2\sigma^2) \tag{6.5.12}$$

so in this case,

$$-\frac{2}{\beta}\xi_i \sim \mathcal{N}\left(0, \frac{4}{\beta^2}\sigma_i^2\right) \tag{6.5.13}$$

From equations (6.5.10) and (6.5.11)

$$E\left[e^{-\frac{2}{\beta}\nu_i}\right] \tag{6.5.14}$$

$$= E\left[d_i^2 e^{-\frac{2}{\beta}\xi_i}\right] \tag{6.5.15}$$

$$= d_i^2 E\left[e^{-\frac{2}{\beta}\xi_i}\right] \tag{6.5.16}$$

$$= d_i^2 e^{\frac{2}{\beta^2}\sigma_i^2} \tag{6.5.17}$$

$$Var\left[e^{-\frac{2}{\beta}\nu_i}\right] \tag{6.5.18}$$

$$= Var\left[d_i^2 e^{-\frac{2}{\beta}\xi_i}\right] \tag{6.5.19}$$

$$= d_i^4 Var\left[e^{-\frac{2}{\beta}\xi_i}\right] \tag{6.5.20}$$

$$= d_i^4 (e^{\frac{4}{\beta^2}\sigma_i^2} - 1) e^{\frac{4}{\beta^2}\sigma_i^2} \tag{6.5.21}$$

As we can see from (6.5.17), $e^{-\frac{2}{\beta}\nu_i}$ is a biased estimate of d_i^2, however we can remove its bias by dividing it by $e^{\frac{2}{\beta^2}\sigma_i^2}$ (which is the bias itself). Therefore, an unbiased estimate of d_i^2 is

$$\hat{d_i^2} = e^{-\frac{2}{\beta}\nu_i - \frac{2}{\beta^2}\sigma_i^2} \tag{6.5.22}$$

We want to use the above estimator obtained from the measurements to build a linear model in $\mathbf{x}$. Since the estimator is now unbiased by writing

$$(x_u - x_i)^2 + (y_u - y_i)^2 = \hat{d_i^2} \tag{6.5.23}$$

we can write

$$(x_u - x_i)^2 + (y_u - y_i)^2 = e^{-\frac{2}{\beta}\nu_i - \frac{2}{\beta^2}\sigma_i^2} \tag{6.5.24}$$

$$-2x_ix_u - 2y_iy_u + x_u^2 + y_u^2 + x_i^2 + y_i^2 = e^{-\frac{2}{\beta}\nu_i - \frac{2}{\beta^2}\sigma_i^2} \tag{6.5.25}$$

$$-2x_ix_u - 2y_iy_u + x_u^2 + y_u^2 = e^{-\frac{2}{\beta}\nu_i - \frac{2}{\beta^2}\sigma_i^2} - x_i^2 - y_i^2 \tag{6.5.26}$$

By setting the condition

$$R^2 = x_u^2 + y_u^2 \tag{6.5.27}$$

we can write it in matrix form

$$\begin{bmatrix} -2x_1 & -2y_1 & 1 \\ \vdots & \vdots & \vdots \\ -2x_n & -2y_n & 1 \end{bmatrix} \begin{bmatrix} x_u \\ y_u \\ R^2 \end{bmatrix} = \begin{bmatrix} e^{-\frac{2}{\beta}\nu_1 - \frac{2}{\beta^2}\sigma_1^2} & -x_1^2 & -y_1^2 \\ \vdots & \vdots & \vdots \\ e^{-\frac{2}{\beta}\nu_n - \frac{2}{\beta^2}\sigma_n^2} & -x_n^2 & -y_n^2 \end{bmatrix} \tag{6.5.28}$$

with

$$\mathbf{A} = \begin{bmatrix} -2x_1 & -2y_1 & 1 \\ \vdots & \vdots & \vdots \\ -2x_n & -2y_n & 1 \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} x_u \\ y_u \\ R^2 \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} e^{-\frac{2}{\beta}\nu_1 - \frac{2}{\beta^2}\sigma_1^2} & -x_1^2 & -y_1^2 \\ \vdots & \vdots & \vdots \\ e^{-\frac{2}{\beta}\nu_n - \frac{2}{\beta^2}\sigma_n^2} & -x_n^2 & -y_n^2 \end{bmatrix}$$

Now since $\mathbf{x}^*$ is unknown we would like to choose $\mathbf{x} \approx \mathbf{x}^*$ to have

$$\mathbf{Ax} - \mathbf{b} \approx 0 \tag{6.5.29}$$

One easy way is to seek the minimum of

$$J(\mathbf{x}) = (\mathbf{Ax} - \mathbf{b})^{\mathbf{T}}(\mathbf{Ax} - \mathbf{b}) \tag{6.5.30}$$

$$= \mathbf{x^TA^TAx - 2x^TA^Tb + b^Tb} \tag{6.5.31}$$

We can fit this model with the Least Square Method, that for a linear model (which is ours) has a known explicit solution [Rao+07] that can be easily verified. In fact since $J(\mathbf{x})$ is a quadratic function of $\mathbf{x}$, then there is a unique minimum [Ort87; ZB19]. The LLS estimate corresponds to:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} J(\mathbf{x}) \tag{6.5.32}$$

which can be found by setting to 0 its derivative

$$\begin{aligned}
&\frac{\partial J(\mathbf{x})}{\partial \mathbf{x}} = 0 \\
&2\mathbf{A^T A \hat{x}} - 2\mathbf{A^T b} = 0 \\
&\mathbf{A^T A \hat{x}} = \mathbf{A^T b} \\
&\mathbf{\hat{x}} = (\mathbf{A^T A})^{\mathbf{-1}} \mathbf{A^T b}
\end{aligned} \tag{6.5.33}$$

The target position can be finally obtained as

$$\begin{bmatrix} \hat{x}_u \\ \hat{y}_u \end{bmatrix} = \begin{bmatrix} [\hat{\mathbf{x}}]_1 \\ [\hat{\mathbf{x}}]_2 \end{bmatrix} \tag{6.5.34}$$

6.6 Weighted linear estimation

Assume to know:		
υ_0	α	$\xi_i, \forall i$

In the previous section we have seen how to use statistics to obtain an estimator of the distance from each anchor node to the unknown node, and then putting all this in a linear form, to deduce the location of the unknown node with a simple closed formula such as the linear least squares. However we can still improve it by givin to each anchor node a weight, bound to the variance of the estimator $\hat{d_i^2}$ obtained from the anchor node i. A good idea would be to give more importance to the nodes that produces an estimator with low variance, and to give less importance to those node that produces estimators with high variance. To ease the computation we consider each anchor node to be indepentendet from each other. One way to do that is to introduce a symmetric covariance matrix $\mathbf{W}'$, which is also diagonal (because of the independence of anchor nodes). Since we want the weight assigned to each node to be inversely proportional to the variance we consider $\mathbf{W} = (\mathbf{W}')^{-1}$ [ZB19; SZL08]. It is important to note that this improvement does not regard the generation of the vector of estimators, rather the minimization of the cost function. Now we compute the variance of the estimator $\hat{d_i^2}$

$$
\begin{aligned}
\sigma^2_{\hat{d_i}} =& Var\left[\hat{d_i^2}\right] = && \quad (6.6.1)\\
&= Var\left[e^{-\frac{2}{\beta}\nu_i - \frac{2}{\beta^2}\sigma_i^2}\right] && \text{from (6.5.22)} \quad (6.6.2)\\
&= \left(e^{-\frac{2}{\beta^2}\sigma_i^2}\right)^2 Var\left[e^{-\frac{2}{\beta}\nu_i}\right] && \text{because it is a constant} \quad (6.6.3)\\
&= \cancel{e^{-\frac{4}{\beta^2}\sigma_i^2}}\, d_i^4 \left(e^{\frac{4}{\beta^2}\sigma_i^2} - 1\right) \cancel{e^{\frac{4}{\beta^2}\sigma^2}} && \text{from (6.5.21)} \quad (6.6.4)\\
&= d_i^4 \left(e^{\frac{4}{\beta^2}\sigma_i^2} - 1\right) && \quad (6.6.5)\\
&\approx \left(\hat{d_i}^2\right)^2 \left(e^{\frac{4}{\beta^2}\sigma_i^2} - 1\right) && \quad (6.6.6)\\
&= \left(e^{-\frac{2}{\beta}\nu_i - \frac{2}{\beta^2}\sigma_i^2}\right)^2 \left(e^{\frac{4}{\beta^2}\sigma_i^2} - 1\right) && \text{from (6.5.22)} \quad (6.6.7)\\
&= e^{-\frac{4}{\beta}\nu_i - \frac{4}{\beta^2}\sigma_i^2} \left(e^{\frac{4}{\beta^2}\sigma_i^2} - 1\right) && \quad (6.6.8)\\
&= e^{-\frac{4}{\beta}\nu_i} \left(1 - e^{-\frac{4}{\beta^2}\sigma_i^2}\right) && \quad (6.6.9)\\
&= \frac{1 - e^{-\frac{4}{\beta^2}\sigma_i^2}}{e^{\frac{4}{\beta}\nu_i}} && \quad (6.6.10)
\end{aligned}
$$

The covariance matrix of the estimators is

$$\mathbf{W}' = \text{diag}\left(\sigma^2_{\hat{d_1}}, ..., \sigma^2_{\hat{d_n}}\right)$$

Now we get to the weighting matrix

$$
\begin{aligned}
\mathbf{W} = \left(\mathbf{W}'\right)^{-1} &= \text{diag}\left((\sigma^2_{\hat{d_1}})^{-1}, ..., (\sigma^2_{\hat{d_n}})^{-1}\right) && (6.6.11)\\
&= \text{diag}\left(\frac{e^{\frac{4}{\beta}\nu_1}}{1 - e^{-\frac{4}{\beta^2}\sigma_1^2}}, ..., \frac{e^{\frac{4}{\beta}\nu_n}}{1 - e^{-\frac{4}{\beta^2}\sigma_n^2}}\right) && (6.6.12)
\end{aligned}
$$

The improved cost function [ZB19] takes the following form

$$
\begin{aligned}
J(\mathbf{x}) &= (\mathbf{Ax} - \mathbf{b})^{\mathbf{T}}\mathbf{W}(\mathbf{Ax} - \mathbf{b}) && (6.6.13)\\
&= \mathbf{x^T A^T W A x} - \mathbf{2x^T A^T W b} + \mathbf{b^T W b} && (6.6.14)
\end{aligned}
$$

that can be minimized with the usual LLS

$$\hat{\mathbf{x}} = (\mathbf{A^T W A})^{-\mathbf{1}}\mathbf{A^T W b} \qquad (6.6.15)$$

The precision of this estimation can be further improved by imposing the constraint (6.5.27) on the minimization. This condition is ignored if we use LLS, but it can be successfully employed with the method of Lagrange multipliers [Che+04; Lop94; PS98].

6.7 Quadratic linearization

Assume to know:	
υ_0	α

From the canonical model extract d_i as follows

$$\begin{aligned}10^{\upsilon_i} &= 10^{\upsilon_0 - 10\alpha\log_{10}(d_i)}\\ 10^{\upsilon_i - \upsilon_0} &= 10^{-10\alpha\log_{10}(d_i)}\\ 10^{\upsilon_i-\upsilon_0} &= d_i^{-10\alpha}\\ d_i &= 10^{\frac{\upsilon_0-\upsilon_i}{10\alpha}} \end{aligned} \tag{6.7.1}$$

so that we have the following set of equations

$$\begin{aligned}(x_1-x)^2+(y_1-y)^2 &= d_1^2\\ (x_2-x)^2+(y_2-y)^2 &= d_2^2\\ &\vdots\\ (x_n-x)^2+(y_n-y)^2 &= d_n^2\end{aligned} \tag{6.7.2}$$

Consider

$$\frac{1}{n}\sum_{i=1}^{n} d_i^2 = \tag{6.7.3}$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left[(x_i-x)^2+(y_i-y)^2\right] \tag{6.7.4}$$

$$= \frac{1}{n}\sum_{i=1}^{n}(x_i-x)^2+\frac{1}{n}\sum_{i=1}^{n}(y_i-y)^2 \tag{6.7.5}$$

$$= \frac{1}{n}\sum_{i=1}^{n}[x_i^2-2x_ix+x^2]+\frac{1}{n}\sum_{i=1}^{n}[y_i^2-2y_iy+y^2] \tag{6.7.6}$$

$$= \frac{1}{n}\sum_{i=1}^{n}[x_i^2]-2x\frac{1}{n}\sum_{i=1}^{n}[x_i]+x^2+\frac{1}{n}\sum_{i=1}^{n}[y_i^2]-2y\frac{1}{n}\sum_{i=1}^{n}[y_i]+y^2 \tag{6.7.7}$$

which is finally

$$\frac{1}{n}\sum_{i=1}^{n}[x_i^2]-2x\frac{1}{n}\sum_{i=1}^{n}[x_i]+x^2+\frac{1}{n}\sum_{i=1}^{n}[y_i^2]-2y\frac{1}{n}\sum_{i=1}^{n}[y_i]+y^2=\frac{1}{n}\sum_{i=1}^{n}d_i^2 \tag{6.7.8}$$

Frone one of the (6.7.2) it follows

$$x_i^2 + x^2 - 2x_i x + y_i^2 + y^2 - 2y_i y = d_i^2 \quad \text{develop the square} \quad (6.7.9)$$
$$-x_i^2 - x^2 + 2x_i x - y_i^2 - y^2 + 2y_i y = -d_i^2 \quad \text{multiply by } -1 \quad (6.7.10)$$

Now by adding member to member (6.7.8) to (6.7.10) you have

$$\begin{aligned}&\Bigg(-x_i^2 + \frac{1}{n}\sum_{i=1}^{n}[x_i^2]\Bigg) + \Bigg(-y_i^2 + \frac{1}{n}\sum_{i=1}^{n}[y_i^2]\Bigg) + \\ &+\Bigg(2x_i x - 2x\frac{1}{n}\sum_{i=1}^{n}[x_i]\Bigg) + \Bigg(2y_i y - 2y\frac{1}{n}\sum_{i=1}^{n}[y_i]\Bigg) = \frac{1}{n}\sum_{i=1}^{n}[d_i^2] - d_i^2\end{aligned} \quad (6.7.11)$$

and by rearranging the equation

$$\begin{aligned}&+2x\Bigg(x_i - \frac{1}{n}\sum_{i=1}^{n}[x_i]\Bigg) + 2y\Bigg(y_i - \frac{1}{n}\sum_{i=1}^{n}[y_i]\Bigg) + \\ &+\Bigg(-x_i^2 + \frac{1}{n}\sum_{i=1}^{n}[x_i^2]\Bigg) + \Bigg(-y_i^2 + \frac{1}{n}\sum_{i=1}^{n}[y_i^2]\Bigg) = \frac{1}{n}\sum_{i=1}^{n}[d_i^2] - d_i^2\end{aligned} \quad (6.7.12)$$

Then the previous set of equations (6.7.2) becomes

$$\begin{aligned}&\Bigg(x_1 - \frac{1}{n}\sum_{i=1}^{n} x_i\Bigg)x + \Bigg(y_1 - \frac{1}{n}\sum_{i=1}^{n} y_i\Bigg)y \\ &\qquad = \frac{1}{2}\Bigg[\Bigg(x_1^2 - \frac{1}{n}\sum_{i=1}^{n} x_i^2\Bigg) + \Bigg(y_1^2 - \frac{1}{n}\sum_{i=1}^{n} y_i^2\Bigg) - \Bigg(d_1^2 - \frac{1}{n}\sum_{i=1}^{n} d_i^2\Bigg)\Bigg] \\ &\qquad \vdots \\ &\Bigg(x_n - \frac{1}{n}\sum_{i=1}^{n} x_i\Bigg)x + \Bigg(y_n - \frac{1}{n}\sum_{i=1}^{n} y_i\Bigg)y \\ &\qquad = \frac{1}{2}\Bigg[\Bigg(x_n^2 - \frac{1}{n}\sum_{i=1}^{n} x_i^2\Bigg) + \Bigg(y_n^2 - \frac{1}{n}\sum_{i=1}^{n} y_i^2\Bigg) - \Bigg(d_n^2 - \frac{1}{n}\sum_{i=1}^{n} d_i^2\Bigg)\Bigg]\end{aligned} \quad (6.7.13)$$

So we can write the above set of equations in matrix form $\mathbf{Ax} = \mathbf{b}$ by setting

$$\mathbf{A} = \begin{bmatrix} x_1 - \frac{1}{n}\sum_{i=1}^{n} x_i & y_1 - \frac{1}{n}\sum_{i=1}^{n} y_i \\ \vdots & \vdots \\ x_n - \frac{1}{n}\sum_{i=1}^{n} x_i & y_n - \frac{1}{n}\sum_{i=1}^{n} y_i \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{b} = \frac{1}{2} \begin{bmatrix} \left(x_1^2 - \frac{1}{n}\sum_{i=1}^n x_i^2\right) + \left(y_1^2 - \frac{1}{n}\sum_{i=1}^n y_i^2\right) - \left(d_1^2 - \frac{1}{n}\sum_{i=1}^n d_i^2\right) \\ \vdots \\ \left(x_n^2 - \frac{1}{n}\sum_{i=1}^n x_i^2\right) + \left(y_n^2 - \frac{1}{n}\sum_{i=1}^n y_i^2\right) - \left(d_n^2 - \frac{1}{n}\sum_{i=1}^n d_i^2\right) \end{bmatrix}$$

Both **A** and **b** are completely known, also d_i, due to (6.7.1). Then the solution can be computed with the usual Linear Least Square procedure obtaining

$$\mathbf{x} = (\mathbf{A}^{\mathbf{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathbf{T}}\mathbf{b} \tag{6.7.14}$$

6.8 Lines intersection

Assume to know:	
v_0	α

Let's start from a geometrical analysis. Given two intersecting circles at center x_1, y_1 and x_2, y_2 respectively and with radius r_1 and r_2 respectively, the line that passes by the two points of intersection has the following equation [ZB19]

$$(x_2 - x_1)x + (y_2 - y_1)y = \frac{1}{2}\big[(x_2^2 + y_2^2) - (x_1^2 + y_1^2) - (r_2^2 - r_1^2)\big] \quad (6.8.1)$$

or equivalently

$$y = \frac{1}{y_2 - y_1}\Bigg(- (x_2 - x_1)x + \frac{1}{2}\big(x_2^2 + y_2^2 - r_2^2 - (x_1^2 + y_1^2 - r_1^2)\big)\Bigg) \quad (6.8.2)$$

Note that the formula is still valid even if the two circles do not intersect each other. Given n measurements (so n circles), only $n - 1$ lines are needed to have an independent system of equations. For example, suppose to have three circles, it is needed to have only 2 lines, for example the line passing between the first and the second circle, and the line passing between the second and the third one, as in fig 6.5. Therefore, given n points we have a set of $n - 1$ equations like the above (6.8.1), that can be written in matrix form $\mathbf{Ax} = \mathbf{b}$ with

$$\mathbf{A} = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \\ \vdots & \vdots \\ x_n - x_1 & y_n - y_1 \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{b} = \frac{1}{2}\begin{bmatrix} (x_2^2 + y_2^2) - (x_1^2 + y_1^2) - (d_2^2 - d_1^2) \\ (x_3^2 + y_3^2) - (x_1^2 + y_1^2) - (d_3^2 - d_1^2) \\ \vdots \\ (x_n^2 + y_n^2) - (x_1^2 + y_1^2) - (d_n^2 - d_1^2) \end{bmatrix}$$

where the d_i are computed with the (6.7.1).

The solution is obtained as an approximated intersection of all these

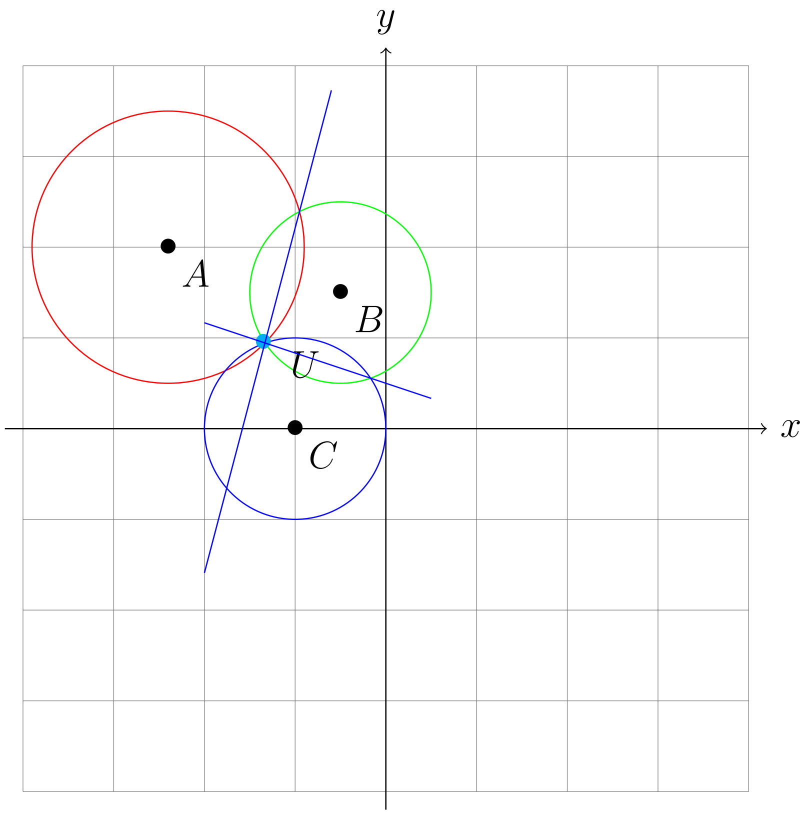


Figure 6.5: Application of the algorithm with 3 circles and 2 lines

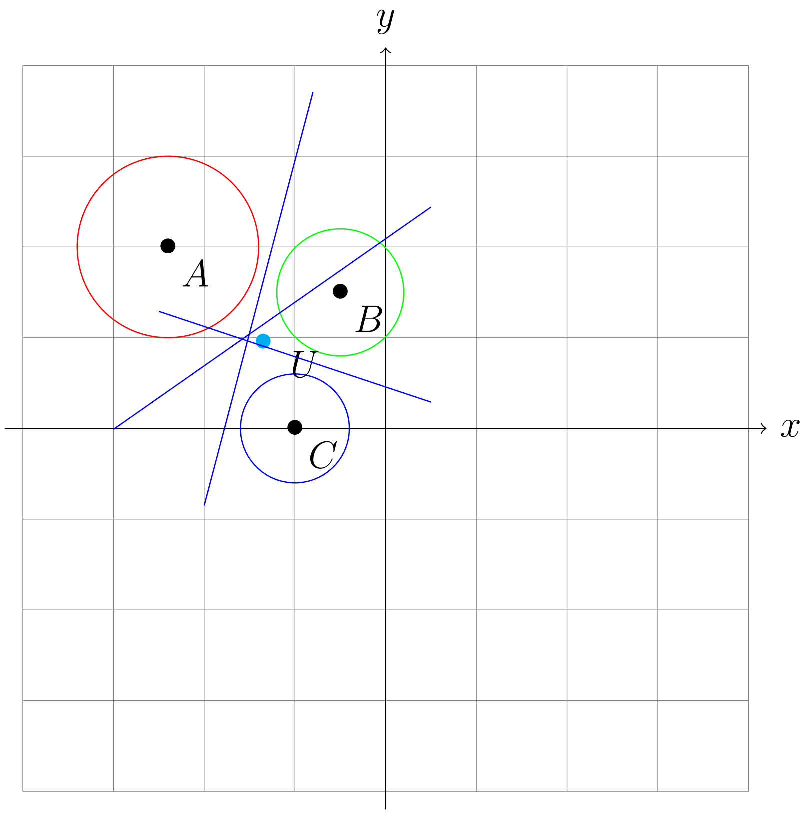


Figure 6.6: Application of the algorithm with noise compensation

lines with the LLS, as in fig. 6.6. You can see that they do not need to intersect in exactly one point, that's why the solution is computed using LLS. In cyan it is shown the real position of the unknown node (U).

Given the high influence in this formula of the first anchor node (present in all equations), one can try an optimization by rearranging the above formula in a way that it relays more evenly to all the nodes as follows

$$\mathbf{A} = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_2 & y_3 - y_2 \\ \vdots & \vdots \\ x_n - x_{n-1} & y_n - y_{n-1} \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\mathbf{b} = \frac{1}{2} \begin{bmatrix} (x_2^2 + y_2^2) - (x_1^2 + y_1^2) - (d_2^2 - d_1^2) \\ (x_3^2 + y_3^2) - (x_2^2 + y_2^2) - (d_3^2 - d_2^2) \\ \vdots \\ (x_n^2 + y_n^2) - (x_{n-1}^2 + y_{n-1}^2) - (d_n^2 - d_{n-1}^2) \end{bmatrix}$$

6.9 Multilateration

Assume to know:
$\emptyset$

This algorithm is the generalization of the trilateration method and is known as Multilateration. This method can provide two main benefits over the classic trilateration:

- Always exists a solution (the algorithm may not be able to find it though) (for non pathological forms)
- The result is made on more observation than the trilateration

So, from the measurement of each node, we try straight away to fit those data to the canonical model. To do so we write a cost function to be minimized

$$\begin{aligned} &J(\upsilon_0, \alpha, x_u, y_u) = \\ &\quad = \sum_{i=1}^{n} \left(\upsilon_i - \upsilon_0 + 10\alpha \log_{10} \left(\sqrt{(x_i - x_u)^2 + (y_i - y_u)^2} \right) \right)^2 \end{aligned} \tag{6.9.1}$$

$$\hat{x}_u, \hat{y}_u = \arg \min_{\upsilon_0, \alpha, x_u, y_u} J(\upsilon_0, \alpha, x_u, y_u) \tag{6.9.2}$$

The great advantage of this method is that it theoretically doesn't need to know any parameter of the model, however in reality to have good results at least υ_0 must be known. This is usually not a problem, since the signal intensity at a fixed distance can be obtained by manual measurement beforehand.

To increase the accuracy of this method, one can also redefine the cost function as

$$\begin{aligned} &J(\upsilon_0, \alpha_1, ..., \alpha_n, x_u, y_u) = \\ &\quad = \sum_{i=1}^{n} \left(\upsilon_i - \upsilon_0 + 10\alpha_i \log_{10} \left(\sqrt{(x_i - x_u)^2 + (y_i - y_u)^2} \right) \right)^2 \end{aligned} \tag{6.9.3}$$

One big drawback of this method is that convergence is not guaranteed, and it often depends from the starting guess. A “necessary condition” for convergence is that the starting guess must be quite close to the real location of the unknown device. For this reason, the usual choice for the initial guess is the estimation obtained by the linear methods.

The (6.9.1) and (6.9.3) are not linear systems, therefore the solution must be computed with some iterative methods, (GN, steepest descent...) [CZ13].

6.10 Maximum Likelihood method

Assume to know:	
υ_0	$\xi_i, \forall i$

We can start by considering the model (6.5.1) and (6.5.2). A look more closely related to the probability point of view is (with $p(\bullet)$ the probability density function)

$$p(\nu_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{1}{2}\frac{(\nu_i+\beta\ln(d_i))^2}{\sigma_i^2}} \tag{6.10.1}$$

The joint probability distribution of all the nodes (assuming independence between the nodes)

$$p(\nu_1, ..., \nu_n) = p(\nu_1)\cdot ...\cdot p(\nu_n) = \frac{1}{(2\pi)^{\frac{n}{2}}\Pi_{i=1}^n \sigma_i} e^{-\frac{1}{2}\sum_{i=1}^n \frac{(\nu_i+\beta\ln(d_i))^2}{\sigma_i^2}} \tag{6.10.2}$$

And in matrix form [ZB19], we can write the covariance matrix as

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_n^2 \end{bmatrix} \tag{6.10.3}$$

with

$$\mathbf{x} = \begin{bmatrix} x_u \\ y_u \end{bmatrix} \tag{6.10.4}$$

$$\mathbf{f}(\mathbf{x}) = -\beta \begin{bmatrix} \ln(\sqrt{(x_u - x_1)^2 + (y_u - y_1)^2}) \\ \vdots \\ \ln(\sqrt{(x_u - x_n)^2 + (y_u - y_n)^2}) \end{bmatrix} \tag{6.10.5}$$

To obtain

$$p(\boldsymbol{\nu}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\mathbf{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}\left(\boldsymbol{\nu}-\mathbf{f}(\mathbf{x})\right)\mathbf{\Sigma}^{-1}\left(\boldsymbol{\nu}-\mathbf{f}(\mathbf{x})\right)} \tag{6.10.6}$$

with

$$\boldsymbol{\nu} = \begin{bmatrix} \nu_1 \\ \vdots \\ \nu_n \end{bmatrix} \tag{6.10.7}$$

For the Maximum Likelihood method we need to find the maximum of (6.10.6) w.r.t $\mathbf{x}$. As usual to ease the computation we consider the log.

$$\ell_n(\mathbf{x}) = -\ln\left((2\pi)^{\frac{n}{2}}|\mathbf{\Sigma}|^{\frac{1}{2}}\right) - \frac{1}{2}\left(\boldsymbol{\nu} - \mathbf{f}(\mathbf{x})\right)^{\mathbf{T}}\mathbf{\Sigma}^{-1}\left(\boldsymbol{\nu} - \mathbf{f}(\mathbf{x})\right) \tag{6.10.8}$$

The first part is independent of $\mathbf{x}$, so we need only to find the stationary point of the second part, that is

$$\hat{\mathbf{x}} = \arg\min_{\beta,\mathbf{x}} \frac{1}{2}\big(\boldsymbol{\nu} - \mathbf{f}(\mathbf{x})\big)^{\mathbf{T}}\boldsymbol{\Sigma}^{-1}\big(\boldsymbol{\nu} - \mathbf{f}(\mathbf{x})\big) \tag{6.10.9}$$

As the Multilateration case, the minimization must be done with iterative algorithms.

6.11 Centroid algorithm

Assume to know:
$\emptyset$

Algorithm 5: Centroid algorithm

```
Result: Estimated position of the unknown node
1 Each node $i$ broadcasts its position $(x_i, y_i)$ to every other node;
2 Define the threshold;
3 Each anchor node retains the information of the nodes that are
    above the threshold;
4 The unknown node computes its position as:
    $$x_u = \frac{\sum_{i=1}^m x_i}{m}, \quad y_u = \frac{\sum_{i=1}^m y_i}{m} \qquad (6.11.1)$$
    which is the mean of the position of the retained nodes.
```

From [BHE00] the filtering based on the threshold can be done in two ways. The first method consists of a simple selection based on the rssi level, that is, every nearest node (whose level is above the rssi threshold) is kept. The second method employs a more sofisticated approach. Every node broadcasts multiple packets in the network, then each node replies with one packet per packet received. Therefore node j can draw a list for every node i as follows

$$g_i = \frac{\text{n. of packets received from node } i}{\text{n. of packets sent to node } i} \qquad (6.11.2)$$

Since the quality/quantity of packets degrades with the distance, closer nodes will send back many packets, therefore having a high grade, whereas farther nodes will reply back with fewer packets, therefore will have a lower grade. Then only the nodes with a grade greater or equal to a certain threshold (e.g. 0.9) will be considered. This is a quite simple algorithm since it generally assumes that the nodes with known positions are placed in a regular grid. It presents however low accuracy, and also it may be difficult to determine the threshold value. The main advantage however is the simultaneous localization of multiple unknown nodes. Below (fig. 6.7) a possible arrangement of anchor nodes.

There exists some improvements of the previous algorithm [KKG19], for

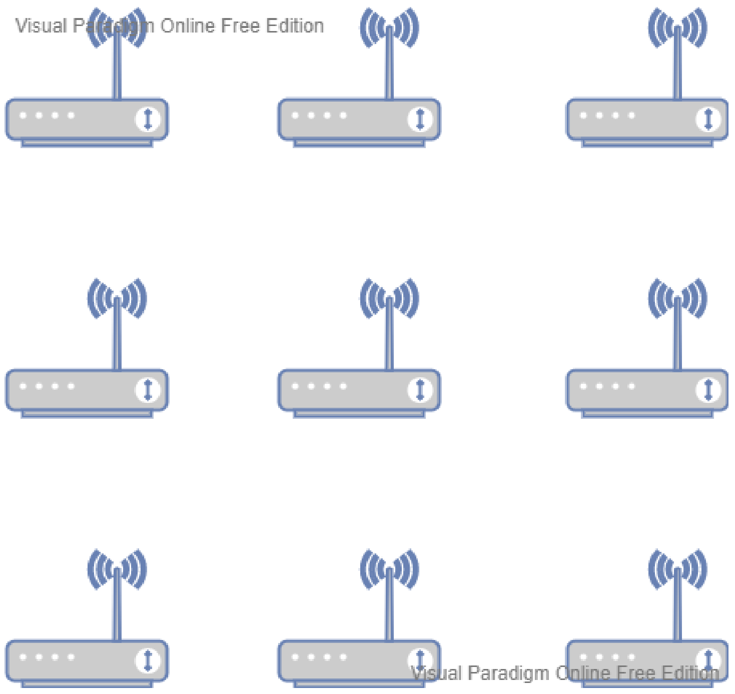

Visual Paradigm Online Free Edition
Visual Paradigm Online Free Edition

Figure 6.7: Possible arrangement of nodes

example the Weighted Centroid Algorithm. Before getting into the formula we need to remind the meaning of hop count. Given a set of connected nodes, the hop count is the number of nodes that the packet goes through before reaching the destination node. For example in the fig 6.8 node A wants to send a packet to node D, but to do so, the packet need to be relayed by B, then by C and finally it reaches D. The hop count therefore is 3.

Here we define a new metric as

$$w_{u,i} = \frac{1}{h_{u,i}} \tag{6.11.3}$$

where $h_{u,i}$ is the hop count from node i to the unknown node u. The line 4 of the previous algorithm then becomes

$$x_u = \frac{\sum_{i=1}^{m} w_{u,i} x_i}{\sum_{i=1}^{m} w_{u,i}}, \quad y_u = \frac{\sum_{i=1}^{m} w_{u,i} y_i}{\sum_{i=1}^{m} w_{u,i}} \tag{6.11.4}$$

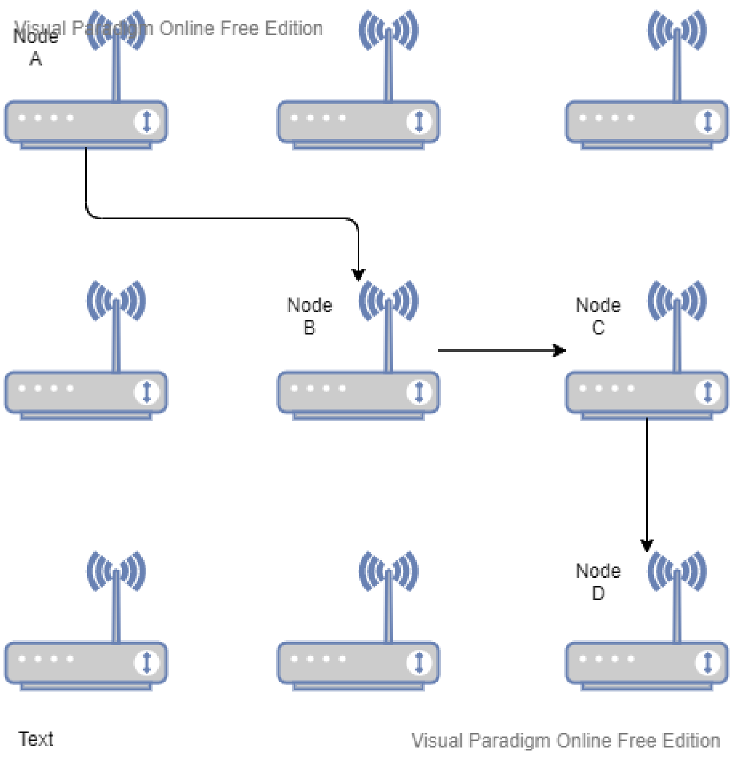

Visual Paradigm Online Free Edition
Node A
Node B
Node C
Node D
Text
Visual Paradigm Online Free Edition

Figure 6.8: Path from node A to node D, hop-count = 3

6.12 DV-Hop Algorithm

Assume to know:
$\emptyset$

The DV-Hop algorithm was first published in [NN01].

Algorithm 6: DV-Hop algorithm

```
Result: Estimated position of the unknown node
1 Each node has built its own database;
2 Each node compute the average distance of one hop
      $$AvgHopDistance_i = \frac{\sum_{j=1,j\neq i}^{n} \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}}{\sum_{j=1,j\neq i}^{n} h_{i,j}} \quad (6.12.1)$$
3 The unknown node $u$ computes the approximate distance from
    anchor node i using
      $$d_{u,i} = AvgHopDistance_i \cdot h_{u,i} \quad (6.12.2)$$
```

The algorithm begins with a broadcast of all nodes with a packet, that contains informations about its position and the hop count, which starts from zero and it is incremented by one at each node it passes through. Each node upgrade its metrics accordingly. After some time necessary to reach a stable point where no information update happen, we have a situation where every node has the minimum hop count from itself to each node and the positions of each node. For example the following table could be the database stored in node i

Node ID	position	hop count
1	(40,-35)	6
2	(-13,18)	4
...	...	...

As soon as the data is available each unknown node can compute its approximated distance from each node i as in line 3 of the previous algorithm. Therefore the unknown node $u, 1 \leq u \leq n$ is able to derive the following equations (with x_u, y_u) unknown

$$\begin{gathered}(x_1 - x_u)^2 + (y_1 - y_u)^2 = d_{u,1}^2 \\ (x_2 - x_u)^2 + (y_2 - y_u)^2 = d_{u,2}^2 \\ \vdots \\ (x_n - x_u)^2 + (y_n - y_u)^2 = d_{u,n}^2\end{gathered} \tag{6.12.3}$$

where $d_{u,i}$ is the distance (in meters) from node u to node i unknown exactly but estimated as in (6.12.2).

By subtracting the n-th equation to all the others, those can be transformed to

$$\begin{aligned}x_1^2 - x_n^2 + y_1^2 - y_n^2 - d_{u,1}^2 - d_{u,n}^2 &= 2x_u(x_1 - x_n) + 2y_u(y_1 - y_u) \\ x_2^2 - x_n^2 + y_2^2 - y_n^2 - d_{u,2}^2 - d_{u,n}^2 &= 2x_u(x_2 - x_n) + 2y_u(y_2 - y_u) \\ &\vdots \\ x_{n-1}^2 - x_n^2 + y_{n-1}^2 - y_n^2 - d_{u,n-1}^2 - d_{u,n}^2 &= 2x_u(x_{n-1} - x_n) + 2y_u(y_{n-1} - y_u)\end{aligned} \tag{6.12.4}$$

Can be written in matrix form $\mathbf{Ax} = \mathbf{b}$ where

$$\mathbf{A} = 2\begin{bmatrix} x_1 - x_m & y_1 - y_m \\ x_2 - x_m & y_2 - y_m \\ \vdots & \vdots \\ x_{m-1} - x_m & y_{m-1} - y_m \end{bmatrix} \tag{6.12.5}$$

$$\mathbf{x} = \begin{bmatrix} x_u \\ y_u \end{bmatrix} \tag{6.12.6}$$

$$\mathbf{b} = \begin{bmatrix} x_1^2 - x_m^2 + y_1^2 - y_m^2 - d_{u,1}^2 - d_{u,m}^2 \\ x_2^2 - x_m^2 + y_2^2 - y_m^2 - d_{u,2}^2 - d_{u,m}^2 \\ \vdots \\ x_{m-1}^2 - x_m^2 + y_{m-1}^2 - y_m^2 - d_{u,n-1}^2 - d_{u,m}^2 \end{bmatrix} \tag{6.12.7}$$

The result can be computed with the closed formula of the LLS

$$\mathbf{x} = (\mathbf{A^T A})^{\mathbf{-1}} \mathbf{A^T b} \tag{6.12.8}$$

6.13 Fingerprinting

Assume to know:
previous $v_i^j, \forall i, j$

Fingerprinting [Yiu+17] is a quite general technique, that can be used for many scenarios. Its main drawback is the offline training phase, which needs to be done before the localization takes place, therefore its use is mostly reserved for example for the location of devices in a known area, that can be for example factories, shopping center, where the nodes have all the time to acquire a precise and accurate representation of the normal environment. This comes with a positive side, it is an efficient model-free algorithm, therefore its use is not restricted to signal intensity localization only. As soon as this phase is done, it can start the localization phase. The main principle underlying this method is the analysis of the similarity of signal intensity observed by the unknown node ($u, u > n$) w.r.t. the other nodes. The main principle underlying this method is to compute the squared difference between the signal intensity of all the anchor nodes measured by the unknown node and the signal intensity of all the other nodes measured by each node. The algorithm employed by this method is known as K-NN or K nearest neighbors [Bis06]. Here is presented first the version working with 1-NN. In the learning (offline) phase, each node builds an internal database with records of every node along with the received signal strength as in the the following table.

Node ID	RSSI
1	-54
2	-68
...	...

As soon as the localization phase starts (computed remotely, since we must have the signal intensity of all the nodes), the following formula is used

$$\hat{i} = \arg\min_i \left\{ \sum_{\substack{j=1 \\ j \neq i}}^{n} (v_i^j - v_u^j)^2 \right\} \tag{6.13.1}$$

This formula just iterates over all the nodes, and by each node i it computes the difference squared between the intensity of the node j read by the node i (v_i^j) and by the node u (v_u^j) for every node. Clearly, the summation skips the values of equal i and j, because it doesn't make sense to measure the intensity of the signal produced by the node itself.

Therefore, the node $\hat{i}$ with the minimum sum of square differences, according to the signal intensity read, should be closest to the unknown node. The estimated position is the position of that node. This is a quite raw estimation, since can take only one out of the n known positions, but it can be efficiently improved for higher values of k. In those cases, the selected nodes are k instead of one, and the final estimate is computed as the mean of the position of those k nodes, as the name says, the k nearest neighbors.

Chapter 7

Conclusions

In this work, I have discussed the main algorithms nowadays available to locate a device with an unknown position through the received signal strength. It is very important to note, however, that those algorithms can be used for the whole class of problems that are based on some path decreasing propagation, regardless of the nature of the signal itself (e.g. sound intensity). Let's take the Quadratic linearization algorithm. Provided you have the mathematical description of the propagation model, and you can derive the distance as in (6.7.1), then the final matrices only depend on the position of the sensing devices and the estimated distances. This reasoning applies also to the other algorithms, clearly the Multilateration algorithm and also the model-free method (possibly with some little adjustments).

It seems appealing that some algorithms don't require the knowledge of any parameter beforehand. Those are some of the model-free methods and the Multilateration. Regarding the model-free methods, it should be said however that they work well if the anchor nodes are many and regularly positioned (in a grid for example), otherwise the precision downgrades. Good results can be obtained with the Fingerprinting method, however, its downside is that it requires a learning phase, where each device measures the intensity read of all the other devices. This reduces the range of applicability of this algorithm. For example, it could not be used generally for disaster response, since there is no physical time to learn the signal intensity of the other nodes. Regarding instead the Multilateration, it could actually work without any previous learning phase nor any parameter but the quality of the estimation degrades. Usually, at least v_0 is required. Additionally, one important drawback is the starting guess. It must be close enough to the real location, otherwise, the method could diverge. A good choice for the starting guess is to use the estimated position from other methods. A good increase in the performance is given by employing an array of custom path loss indexes, different for each anchor node, particularly useful if the location process happens in an environment fitted with obstacles. This improvement comes however with a decrease in performance.

Optimum results can be obtained when the field of application is well known and the algorithm has all the time required to estimate parameters $\alpha, \sigma_i^2 \, \forall i$ with the algorithms presented earlier. In my opinion, the most accurate one would be the Weighted linear estimation, which considers carefully the effect of the variance of the measurements, which is in fact a concise representation of the noise. A particular mention needs to be given also to the line intersection algorithm.

The use of UAV is spreading more and more in many areas. As a moving anchor node, it could be effectively employed in Multilateration for example, by moving in the area while recording samples as

$$(x_i, y_i, v_i)$$

and then proceed with the usual computation. However it could not be employed in some range free methods, for example the Fingerprinting, since we cannot compare simultaneous measurements in different places.

As a consequence, it seems that the perfect algorithm doesn't exist, not even a perfect sequence of algorithms. It is a task for the user to choose what fits best its need, based also on his known parameters and his expectations on the accuracy.

Below it is presented a table that briefly summarizes pros and cons of the algorithms.

Approach	**Ref**	**Pros**	**Cons**	**Exp. testing**
Trilateration	[ZB19]	Easiness of computation, Few samples required	Low accuracy	[Rat+15]
Min-Max	[Rat+15]	Easiness of computation, Few samples required	Low accuracy	[Rat+15]
Lines intersection	[ZB19]	Easy to compute, robust to noise	Medium accuracy	
Weighted linear estimation	[ZB19]	High precision, sensible to variance, easy to compute	Required probabilistic parameters	
Multi-lateration	[ZB19]	Hard computation, starting guess	Up to no parameters needed	[ZB19]
Centroid algorithm	[BHE00]	No model neither parameters needed	Low accuracy, regular placement of nodes	[WZ14]
DV-Hop algorithm	[NN01]	No model neither parameters needed	Low accuracy	[Yin19]
Fingerprint.	[Yiu+17]	No model neither parameters needed	Low accuracy, regular placement of nodes	

Bibliography

[Ort87] James M. Ortega. "Quadratic Forms and Optimization". In: *Matrix Theory: A Second Course*. Boston, MA: Springer US, 1987, pp. 143–173. ISBN: 978-1-4899-0471-3. DOI: `10.1007/978-1-4899-0471-3_4`. URL: `https://doi.org/10.1007/978-1-4899-0471-3_4`.

[LR92] J.C. Liberti and T.S. Rappaport. "Statistics of shadowing in indoor radio channels at 900 and 1900 MHz". In: *MILCOM 92 Conference Record*. 1992, 1066–1070 vol.3. DOI: `10.1109/MILCOM.1992.244122`.

[Lop94] Robert J. Lopez. "The Lagrange Multiplier, Part One". In: *Maple via Calculus: A Tutorial Approach*. Boston, MA: Birkhäuser Boston, 1994, pp. 133–139. ISBN: 978-1-4612-0267-7. DOI: `10.1007/978-1-4612-0267-7_26`. URL: `https://doi.org/10.1007/978-1-4612-0267-7_26`.

[PS98] Carlo Domenico Pagani and Sandro Salsa. In: *Analisi Matematica Volume 2*. Masson, 1998. ISBN: 9788808018755.

[ZG98] K. Zayana and B. Guisnet. "Measurements and modelisation of shadowing cross-correlations between two base-stations". In: *ICUPC '98. IEEE 1998 International Conference on Universal Personal Communications. Conference Proceedings (Cat. No.98TH8384)*. Vol. 1. 1998, 101–105 vol.1. DOI: `10.1109/ICUPC.1998.732812`.

[BHE00] N. Bulusu, J. Heidemann, and D. Estrin. "GPS-less low-cost outdoor localization for very small devices". In: *IEEE Personal Communications* 7.5 (2000), pp. 28–34. DOI: `10.1109/98.878533`.

[NN01] D. Niculescu and B. Nath. "Ad hoc positioning system (APS)". In: *GLOBECOM'01. IEEE Global Telecommunications Con-*

ference (Cat. No.01CH37270). Vol. 5. 2001, 2926–2931 vol.5. DOI: 10.1109/GLOCOM.2001.965964.

[Rap01] Theodore Rappaport. *Wireless Communications: Principles and Practice*. 2nd. USA: Prentice Hall PTR, 2001. ISBN: 0130422320.

[Che+04] K.W. Cheung et al. "Least squares algorithms for time-of-arrival-based mobile location". In: *IEEE Transactions on Signal Processing* 52.4 (2004), pp. 1121–1130. DOI: 10.1109/TSP.2004.823465.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[Rao+07] C. Radhakrishna Rao et al. *Linear Models and Generalizations: Least Squares and Alternatives*. 3rd. Springer Publishing Company, Incorporated, 2007. ISBN: 3540742263.

[SZL08] H.C. So, Jun Zheng, and Wing Kin Lui. "Best linear unbiased estimator approach for time-of-arrival based localisation". In: *Signal Processing, IET* 2 (July 2008), pp. 156–162. DOI: 10.1049/iet-spr:20070190.

[Wu+08] Rong-Hou Wu et al. "Study of characteristics of RSSI signal". In: *2008 IEEE International Conference on Industrial Technology*. 2008, pp. 1–3. DOI: 10.1109/ICIT.2008.4608603.

[Mun+09] David Munoz et al. "CHAPTER 2 - Signal Parameter Estimation for the Localization Problem". In: *Position Location Techniques and Applications*. Ed. by David Munoz et al. Oxford: Academic Press, 2009, pp. 23–65. ISBN: 978-0-12-374353-4. DOI: https://doi.org/10.1016/B978-0-12-374353-4.00008-9. URL: https://www.sciencedirect.com/science/article/pii/B9780123743534000089.

[BG11] Jan Beran and Sucharita Ghosh. "Moment Generating Function". In: *International Encyclopedia of Statistical Science*. Ed. by Miodrag Lovric. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 852–854. ISBN: 978-3-642-04898-2. DOI: 10.1007/978-3-642-04898-2_375. URL: https://doi.org/10.1007/978-3-642-04898-2_375.

[CZ13] E.K.P. Chong and S.H. Zak. *An Introduction to Optimization*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 2013. ISBN: 9781118515150. URL: `https://books.google.it/books?id=iD5s0iKXHP8C`.

[WZ14] Zu-Min Wang and Yi Zheng. "The Study of the Weighted Centroid Localization Algorithm Based on RSSI". In: *2014 International Conference on Wireless Communication and Sensor Network*. 2014, pp. 276–279. DOI: `10.1109/WCSN.2014.63`.

[Rat+15] Banjerd Rattanalert et al. "Problem Investigation of Min-max Method for RSSI Based Indoor Localization". In: June 2015. DOI: `10.1109/ECTICon.2015.7207057`.

[Yiu+17] Simon Yiu et al. "Wireless RSSI fingerprinting localization". In: *Signal Processing* 131 (2017), pp. 235–244. ISSN: 0165-1684. DOI: `https://doi.org/10.1016/j.sigpro.2016.07.005`. URL: `https://www.sciencedirect.com/science/article/pii/S0165168416301566`.

[KKG19] Amanpreet Kaur, Padam Kumar, and Govind P. Gupta. "A weighted centroid localization algorithm for randomly deployed wireless sensor networks". In: *Journal of King Saud University - Computer and Information Sciences* 31.1 (2019), pp. 82–91. ISSN: 1319-1578. DOI: `https://doi.org/10.1016/j.jksuci.2017.01.007`. URL: `https://www.sciencedirect.com/science/article/pii/S1319157816300842`.

[Yin19] Lu Yin. "A New Distance Vector-Hop Localization Algorithm Based on Half-Measure Weighted Centroid". In: *Mobile Information Systems* 2019 (Jan. 2019), pp. 1–9. DOI: `10.1155/2019/9892512`.

[ZB19] Reza Zekavat and R. Michael Buehrer. *Handbook of Position Location: Theory, Practice and Advances*. 2st. Wiley-IEEE Press, 2019. ISBN: 978-1-119-43458-0.

[Ros20] Sheldon M. Ross. *A first course in probability / Sheldon Ross*. eng. Tenth edition, global edition. Harlow: Pearson Education Limited, 2020. ISBN: 9781292269238.

[Rub] Julian T. Rubin. URL: `https://juliantrubin.com/encyclopedia/electronics/multipath.html`.