Indoor positioning with Probabilistic WKNN WiFi fingerprinting

Submitted to the Faculty of the
Escola Tècnica d’Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya

by

Santiago de Nadal

In partial fulfilment
of the requirements for the degree in
Telecommunications Systems ENGINEERING

Advisor: Núria Duffo

Barcelona, June 2016
Abstract

We focus on a probabilistic approach to implement Wi-Fi fingerprinting Weighted k-Nearest Neighbors (WkNN) algorithms in indoor position estimation. Still, deterministic approach is compared with the probabilistic methods tested, to see if there is an improvement on the results. This is the main goal of this experiment. In this work, several tests are carried out in an indoor scenario. Many parameters and distribution functions are used, and the results show that the probabilistic option with the lowest mean error in the estimation is the exponential non weighted, but is not lower, by little, than in the deterministic approach. In a future work, the same experiment should be done in better conditions, doing a better testbed and considering more factors when implementing the algorithms.
Index

1 Introduction .................................................. 1

2 Theoretical approach ......................................... 2
  2.1 Wi-Fi fingerprinting ......................................... 2
     2.1.1 Offline phase ........................................... 2
     2.1.2 Online phase ........................................... 3
  2.2 Different approaches ........................................ 4
     2.2.1 Deterministic approach ................................ 4
  2.3 Probabilistic approach ...................................... 5

3 Project development ........................................... 8
  3.1 Measurements ............................................... 8
  3.2 Practical application ....................................... 10
  3.3 Code explanation ........................................... 13
     3.3.1 Implementing the algorithms ......................... 14
         3.3.1.1 Gaussian ........................................... 14
         3.3.1.2 Exponential ........................................ 15
         3.3.1.3 Kernel ............................................. 15
         3.3.1.4 Deterministic ..................................... 17

4 Experimental results ......................................... 20
  4.1 TPs in detail ............................................... 22
  4.2 Mean error graphics ....................................... 25

5 Conclusions and future development ......................... 32

6 Bibliography .................................................. 33

7 Annexes ....................................................... 34
  7.1 Annex I: Map of DIET 2nd floor .......................... 34
  7.2 Annex II: Matlab code .................................... 34
  7.3 Annex III: Histograms ..................................... 47
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Unit circles for Manhattan and Euclidean p values</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>WiFi Compass app logo</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>Loading the map</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td><em>Scan Here!</em> button</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>Detecting RSSI from APs. Outline</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>Process of creating RPs</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>Map with all the RPs</td>
<td>11</td>
</tr>
<tr>
<td>8</td>
<td>Measurements orientation - North</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>Map with all the TPs</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>Summary of main variables used</td>
<td>13</td>
</tr>
<tr>
<td>11</td>
<td>Example of TP 2 with the K=5 closest RPs for Exponential distribution. (See Table 4)</td>
<td>17</td>
</tr>
<tr>
<td>12</td>
<td>Histogram examples of RSSI from a pair RP-AP</td>
<td>20</td>
</tr>
<tr>
<td>13</td>
<td>Different distributions</td>
<td>20</td>
</tr>
<tr>
<td>14</td>
<td>Histograms received from every AP in RP1</td>
<td>21</td>
</tr>
<tr>
<td>15</td>
<td>Example: TP9 using Gaussian distribution Non Weighted with K=2</td>
<td>22</td>
</tr>
<tr>
<td>16</td>
<td>Example: TP2 using Kernel distribution (with Exponential) Weighted with K=3 and a smoothing parameter h=75</td>
<td>22</td>
</tr>
<tr>
<td>17</td>
<td>Example: TP11 using Exponential distribution Non Weighted with K=3</td>
<td>23</td>
</tr>
<tr>
<td>18</td>
<td>Example: TP11 using Exponential distribution Non Weighted with K=5</td>
<td>23</td>
</tr>
<tr>
<td>19</td>
<td>Example: TP1 using deterministic approach Non Weighted with K=7</td>
<td>24</td>
</tr>
<tr>
<td>20</td>
<td>Gaussian - Non Weighted graphic</td>
<td>25</td>
</tr>
<tr>
<td>21</td>
<td>Gaussian - Weighted graphic</td>
<td>25</td>
</tr>
<tr>
<td>22</td>
<td>Exponential - Non Weighted graphic</td>
<td>26</td>
</tr>
<tr>
<td>23</td>
<td>Exponential - Weighted graphic</td>
<td>26</td>
</tr>
<tr>
<td>24</td>
<td>Graphics to find the optimum value of h.</td>
<td>27</td>
</tr>
<tr>
<td>25</td>
<td>Gaussian Kernel (h=23) - Non Weighted graphic</td>
<td>28</td>
</tr>
<tr>
<td>26</td>
<td>Gaussian Kernel (h=14) - Weighted graphic</td>
<td>28</td>
</tr>
<tr>
<td>27</td>
<td>Exponential Kernel (h=33) - Non Weighted graphic</td>
<td>29</td>
</tr>
<tr>
<td>28</td>
<td>Exponential Kernel (h=75) - Weighted graphic</td>
<td>29</td>
</tr>
<tr>
<td>29</td>
<td>deterministic - Non Weighted graphic</td>
<td>30</td>
</tr>
<tr>
<td>30</td>
<td>deterministic - Weighted graphic</td>
<td>30</td>
</tr>
<tr>
<td>31</td>
<td>Graphic of the best options for deterministic and probabilistic approaches</td>
<td>31</td>
</tr>
<tr>
<td>32</td>
<td>All TPs (o) and RPs (x).</td>
<td>34</td>
</tr>
<tr>
<td>33</td>
<td>Histograms in RP1</td>
<td>47</td>
</tr>
<tr>
<td>34</td>
<td>Histograms in RP2</td>
<td>47</td>
</tr>
<tr>
<td>35</td>
<td>Histograms in RP3</td>
<td>48</td>
</tr>
<tr>
<td>36</td>
<td>Histograms in RP4</td>
<td>48</td>
</tr>
<tr>
<td>37</td>
<td>Histograms in RP5</td>
<td>49</td>
</tr>
</tbody>
</table>
List of Tables

1. Different pdf’s $f_{v_{ij}}$ for the measurement error.  
2. Ex: RP25. RSSI values detected from each AP in a certain sample.  
3. Probability values sorted. Example for Exponential distribution.  
4. Closest RPs in order for every TP. Example for Exponential distribution.  
6. Closest RPs in order for every TP. Deterministic.  
7. Gaussian - Non Weighted results  
8. Gaussian - Weighted results  
9. Exponential - Non Weighted results  
10. Exponential - Weighted results  
11. Gaussian Kernel (h=23) - Non Weighted results  
12. Gaussian Kernel (h=14) - Weighted results  
13. Exponential Kernel (h=33) - Non Weighted results  
14. Exponential Kernel (h=75) - Weighted results  
15. deterministic - Non Weighted results  
16. deterministic - Weighted results

List of abbreviations

RP: Reference Point  
AP: Access Point  
TP: Test Point  
MU: Mobile Unit  
RSSI: Received Signal Strength Indicator  
WKNN: Weighted K Nearest Neighbours  
WLAN: Wireless Local Area Network  
pdf: probability density function  
GPS: Global Positioning System
1 Introduction

Location estimation is an interesting area related with modern communication technology. Methods like GPS have been very extended and become very popular, but the poor performance of this one in indoor environments caused that in the last years indoor positioning systems have been developed and researched.

Location estimation can make use of the WLAN in the area considering the RSSI values detected by a MU. This way it is easier to implement the system, due to the fact that you can take advantage of the already existing infrastructure. In addition, RSSI has a strong correlation with the position. For that reason fingerprinting, which uses RSSI comparisons, is one of the best options.

This research could be very useful to improve the reliability of indoor localization. The applications can be many, from tracking kids in kindergartens, elderly in nursing homes, or medical devices in hospitals, to life-saving situations of firefighters, for example.

My thesis is an experimental project with a theoretical basis about fingerprinting. It is especially based on the research of other departments in other universities.

I am going to focus on an specific approach related with Wi-Fi fingerprinting, the probabilistic WKNN, so the main objective is to test if using this approach we can obtain better results than using others, specifically the one that has been used until the moment in the department work, that is, the deterministic WKNN.

Since we are trying to estimate the position of a MU, the problem to solve would be to decrease the mean error (many tests will be carried out) of this estimation depending on the algorithm and the parameters used.

The scope of the thesis include to observe which are the final results of the tests applying the different algorithms proposed, and explain the improvements or possible solutions to obtain better results.

Throughout the whole project there are several stages: First it was necessary to read all the literature related with the topic to deal with. There are other papers from different researchers talking about this specific work. Then, continue with the testbed, making all the measurements to create a database as a starting point. Once it was done, it was time to program all the algorithms in Matlab to test them using the data obtained in the previous phase of the project. The last part was to make some graphics to show the results with the most relevant information for us.
All of this work has a very technical component that justifies the fact that it should be done by an engineer. Especially in the programming stage, where it is essential, besides the mere fact of knowing programming languages, the understanding of some concepts related with electronics and signal propagations.

The work is organized as follows: First we have a theoretical basis on Wi-Fi fingerprinting (Section 2.1). Then, in Section 2.2, the different options to approach fingerprinting are explained, focusing on the probabilistic method. The next step (Section 3) will be explaining everything related with the project development itself, that is, how it was carried out and how the theory was applied. Finally, in Section 4, the results will be shown and discussed, and from them we will draw the conclusions (Section 5).

2 Theoretical approach

2.1 Wi-Fi fingerprinting

Wi-Fi fingerprinting is a location estimation method based on the signal power strength received from each internet (WLAN) connection spot (AP). In this case it would be used for indoor localization.

There are two stages in fingerprinting, the offline stage (calibration phase) and the online stage (working phase).

2.1.1 Offline phase

In the offline stage fingerprints are collected at each reference point (RP) and then stored to form our database.

The calibration points are obtained collecting RSSI (received signal strength indicator) values (in dBm) from the radio signals transmitted by the access points (APs) during a certain period of time, getting several samples. At the end, what we call a radio map is created, containing many reference points (calibration points).

The radio map is in fact the database that we use to store all the necessary (e.g. RSSI) to estimate the mobile unit (MU) position.

Each of the elements in the radio map has this form:

\[ \Omega_i = (B_i, R_i) \quad i = 1, ..., M \]

- \( B_i \) is the cell where it is located.
− $R_i$ is the fingerprint, which contains:
  
  A reference point (RP) representing each cell.
  
  Vectors $\vec{a}_{ij}$ holding RSSI values measured in $RP_i$ from every $AP_j$.
  
  A parameter indicating the orientation of the MU, $\theta_i$.

Each fingerprint would have this form:

$$R_i = \{RP_i, \vec{a}_{ij}, \theta_i\}_{j=1}^{L}$$

An example of $\vec{a}_{ij}$ would be represented as follows:

$$\vec{a}_{12} = \begin{bmatrix} a_{11}^{12} & a_{12}^{12} & \cdots & a_{K}^{12} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1L}^{12} & a_{2L}^{12} & \cdots & a_{K}^{1L} \end{bmatrix}$$

These would be the RSSI values measured from $AP_2$ in $RP_1$ taking $K$ samples.

Extrapolating to the other APs, a matrix could be created to represent, for a single RP, all the RSSI values detected from every AP in every sample taken. As an example:

$$A_1 = \begin{bmatrix} a_{11}^{11} & a_{12}^{11} & \cdots & a_{K}^{11} \\ a_{12}^{11} & a_{12}^{12} & \cdots & a_{K}^{12} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1L}^{1L} & a_{2L}^{1L} & \cdots & a_{K}^{1L} \end{bmatrix}$$

Each of the elements of this matrix has this form: $a_{ij}^k$. As we said before,

− $i$ indicates the number of the RP with a maximum of $M$.

− $j$ indicates the number of each AP detected with a maximum of $L$:

− $k$ indicates the number of the sample with a maximum of $K$.

We can also take the mean of each of the rows of matrix $A$, for a certain RP, and have a vector $\vec{a}_i$ containing the mean RSSI values received from every AP. Subindex $i$ indicates the RP ($\vec{a}_{RP}$).

### 2.1.2 Online phase

This would be the part of the work to test our system.

In the online stage RSSI values are also collected, in this case in $\vec{y}$, to create every test point (TP), and from those values location is estimated as a function of the positions of the closest RPs. But this depends on the algorithm used for location estimation.
The main disadvantage of Wi-Fi fingerprinting is that depends a lot on the environment. If this changes, the previous fingerprints are not useful anymore and the database should be updated again.

Furthermore, we have to face some problems when taking the measurements to be the most accurate as possible, especially when we are creating the radio map. Setting the RPs can be tough and sometimes quite inefficient in sense of time and efforts dedicated. The same happens with the number of samples taken in each RP or TP. For that reason an optimal way to do it must be found.

Once the radio map is defined we have to propose a location estimation method.

2.2 Different approaches

To estimate the location of the mobile unit (MU) we have two main possibilities:

One is considering just the mean of the measurements vector in each RP that includes the RSSI samples from the different APs detected (Deterministic approach). The other one is to take every measurement into account. That is, its probability distribution (Probabilistic approach).

Both the deterministic and the probabilistic approaches are typically based on Weighted k-Nearest-Neighbor (WKNN), which keeps K biggest weights and sets the others to zero. With equal weights is called KNN, and with K=1 (the simplest method), NN method. In NN method the closest RP is estimated.

2.2.1 Deterministic approach

So the first case would be the deterministic one, where we consider the location vector as already known (that is deterministic). WKNN deterministic solutions have the positive point that they are relatively easy to implement.

The main work here would be to estimate the mean of that vector. In the case of WkNN, a convex combination of the calibration points is used, with different weights for each of them.

\[
\hat{x} = \frac{1}{\sum_{j=1}^{k} w_j} \sum_{i=1}^{k} RP_i w_i
\]

Here, K indicates the number of RPs taken into account when estimating the location.

The weight of each calibration point can be the inverse of the norm of the RSSI innovation, and there are many ways to calculate the norm.
\[ w_i = \frac{1}{||\vec{y} - \vec{a}_{RP}||} \]  

(2)

For the norm, the Minkovski distance is commonly used.

\[ dist.Minkovski = \left( \sum_{i=1}^{L} |y^i - a_{RP}^i|^p \right)^\frac{1}{p} \]  

(3)

With \( p = 1 \) it is the Manhattan distance, and with \( p = 2 \) it is the Euclidean distance.

![Figure 1: Unit circles for Manhattan and Euclidean p values.](image)

2.3 Probabilistic approach

The performance of the deterministic solutions, in theory, could be improved by adopting a probabilistic approach. However, it is more complex to implement. In my thesis I am going to focus on the probabilistic framework. In this case, the location (or state) of the mobile is a random vector represented by \( \vec{x} \).

\[ \vec{x} = [x_1, x_2] = [x, y] \]

Starting from a set of measurements \( \vec{y} \) it consists in computing the conditional pdf (probability density function) of \( \vec{x} \). For that, the Bayes rule is used:

\[ f_{\vec{x}|\vec{y}}(x|y) = \frac{f_{\vec{y}|\vec{x}}(y|x)f_{\vec{x}}(x)}{f_{\vec{y}}(y)} \]  

(4)

That would be the posterior distribution of the location, where

- \( f(x|y) \) is the likelihood. The distribution of the position \( \vec{x} \) given the measurements \( \vec{y} \).
- \( f(y|x) \) is the distribution of the reception of the set of RSSI values \( \vec{y} \) if we already know we are in a certain position \( \vec{x} \).
- \( f(x) \) is the prior. The a priori probability density function of being in \( \vec{x} \).
- \( f(y) \) is the probability density function of receiving those measurements. It is a normalizing constant.
At this level it is commonly used a uniform prior:

\[
f(x) = \frac{\sum_{i=1}^{M} \chi B_i(x)}{\sum_{j=1}^{M} |B_j|}
\]  

(5)

Where \(|B_i|\) is the area of \(B_i\) on the plane (2D) and

\[
\chi B_i(x) = \begin{cases} 
1 & x \in B_i \\
0 & x \notin B_i 
\end{cases}
\]  

(6)

We use this last function to determine which region or cell do the coordinates of point \(\vec{x}\) correspond to.

Besides, the distribution of the RSSI in each cell \(f(y|x)\) we assume it the same as the set of RSSI measurements in each RP. That means that we have a constant likelihood inside each cell \(B_i\), so

\[
f(y|x) = \frac{\sum_{i=1}^{M} f(y|i)\chi B_i(x)}{\sum_{j=1}^{M} |B_j|}
\]  

(7)

where \(f(y|i) = f_{\vec{v}_i}(y - \bar{a}_i)\) and \(\vec{v}_i = y - \bar{a}_i\).

E.g. If \(\vec{x}\) belongs to the 4th cell:

\[
f(y|x) = f(y|4)
\]

We also assume as independent the components of the random vector \(\vec{v}_i\) (Independence between measurements from different APs). Thus, the likelihood function is the product of the marginal probabilities:

\[
f(y|i) = \prod_{j=1}^{L} f_{\vec{v}_{ij}}(y_j - \bar{a}_{ij}) \quad y \in \mathbb{R}^L
\]  

(8)

\(f_{\vec{v}_{ij}}\) is the pdf of the measurement error of the RSSI signal from the jth station (AP\(j\)) in the ith cell (RP\(i\)).

To finally compute this \(f(y|i)\) there are many possible implementations using different approximation methods for \(f_{\vec{v}_{ij}}\). Some of them are shown in this table:
Depending on the characteristics of the fingerprint histogram, we will use one or another.

The estimation of the position in the probabilistic approach is very similar to the deterministic in the final step.

For KNN:

$$\hat{x} = \frac{\sum_{i=1}^{K} RP_i}{K}$$  \hspace{1cm} (9)

And for WKNN:

$$\hat{x} = \frac{1}{\sum_{j=1}^{K} f(x|y)_j} \sum_{i=1}^{K} RP_i f(x|y)_i$$  \hspace{1cm} (10)

Where K is the number of RPs considered when estimating, i indicates the ith most probable RP estimated, and RP_i are the coordinates of the mentioned RP.

<table>
<thead>
<tr>
<th>Name</th>
<th>(f_{vij})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Histogram</td>
<td>(f_{vij}(x) = H_{vij}(x))</td>
</tr>
<tr>
<td>Gaussian</td>
<td>(f_{vij}(x) = \frac{1}{\sqrt{2\pi\hat{\sigma}<em>{ij}^2}} \exp\left(-\frac{x^2}{\hat{\sigma}</em>{ij}^2}\right))</td>
</tr>
<tr>
<td>Exponential</td>
<td>(f_{vij}(x) = \frac{1}{2} e^{-x})</td>
</tr>
<tr>
<td>Kernel</td>
<td>(f_{vij}(x) = \frac{\sum_{k=1}^{</td>
</tr>
</tbody>
</table>

Table 1: Different pdf’s \(f_{vij}\) for the measurement error.
3 Project development

The first thing that we may take into account is that we should start from a plan containing the area in which we want to estimate the indoor location, and this area should be divided into small cells. These divisions will help us when setting the calibration points.

This cell division is actually impossible to draw in the map, but since each cell is represented by a RP, we will consider that the coordinates are part of that cell if in that point the same RP is estimated as the closest one.

On the other hand we have Access Points (APs) distributed in the whole area, so that the user can access the WLAN and, from that, be able to be localized. The location of these APs is irrelevant when using fingerprinting method, since we only consider the signal strength received from each one.

3.1 Measurements

The project tests are carried out in one of the buildings in our faculty of Telecommunications Engineering, specifically in the 1st and 2nd floor (DIET department). In my thesis I work only in part of the 2nd floor.

For both the offline and online phase, to take all the measurements, a Samsung tablet GT-N8000 was used. The OS was Android 4.1.2.

For this purpose an app called 'WiFi Compass' allow us to simulate a platform where our floor is represented.

![WiFi Compass app logo.](image)

Figure 2: WiFi Compass app logo.

So the steps are the following:

1. Open the WiFi Compass app.
2. Create a New Project.
3. Set the Title and the Project Site title.
4. Load the map of the floor where you want to work on. Options → Map Settings → Set Map Image → Choose Image.
The image used is a bitmap image file, so that we can identify points of the map as coordinates.

Once the map is loaded we can start with the measurements. In each of the spots where we decide to set a RP we must be standing totally still and holding the tablet, looking always towards the same direction for every RP. We have to place the little man of the app in the same place where we are situated, trying to be the most accurate as possible in the position.

Then we click the option *Scan here!* to start detecting all the WLAN APs.

From each of the APs we will perceive an RSSI.

---

(a) Map still not loaded.  
(b) Map loaded.

Figure 3: Loading the map.

Figure 4: *Scan Here!* button.

Figure 5: Detecting RSSI from APs. Outline.
<table>
<thead>
<tr>
<th>AP</th>
<th>RSSI (dBm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>−74</td>
</tr>
<tr>
<td>2</td>
<td>−69</td>
</tr>
<tr>
<td>3</td>
<td>−81</td>
</tr>
<tr>
<td>4</td>
<td>−93</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>−102</td>
</tr>
</tbody>
</table>

Table 2: Ex: RP25. RSSI values detected from each AP in a certain sample.

We take several samples in the same place, each one will leave a fingerprint.

Figure 6: Process of creating RPs.

The data obtained (all the fingerprints) is saved within the project created. By connecting the tablet to a computer we can get this data and organize it in a bigger database. This way, we can create matrices and have this data in a tidier way.

3.2 Practical application

In this section we are going to particularize for our case all the explained in section 2.3.

The number of elements M of our database or radio map is actually the number of RPs (look section 2.1.1). I focused on just one side of the floor, to avoid unnecessary dedicated time, so for my thesis that was enough. We have defined M=27 at a whole.
Several APs are distributed along the floor. There are L=19, so that is the maximum number that can be detected.

We have focused on just one orientation ($\theta_i = \text{North} \; \forall i$) when measuring RSSI values. We take as North as the direction looking lengthwise towards the part of the floor where our lab is located (See Fig.8).

The number of samples taken at each RP in the offline phase is $K=30$.

And the number of samples taken for the TP at random positions in the online phase is 10.

To test the system in the online phase, $N=20$ test points were used. They are randomly chosen, so they are in different positions than the RPs.
Talking about the probabilistic compute of the location, we explained in section 2.3 how to calculate the prior (equation 5), but we are going to simplify it more. In stead of taking into account the area of each cell or give a higher weight to some of them, every cell is considered equal. In addition, due to the fact that the position is absolutely random and that we don’t give higher probability for being in a particular room or another, being in a certain point on the plane would mean being in one of the 27 cells defined by the RPs. So, the prior probability density function is constant and it would be:

\[
f_x(x) = f_i(i) = \frac{1}{N_{RPs}} = \frac{1}{M} = \frac{1}{27}
\]

The probability density function of getting a certain set of measurements \( f(y) \), as we also said in section 2.3, is a normalizing constant, but since we are going to use the MAP decisor to estimate the closest RPs and it is just to normalize we will not consider it.

\[
\hat{x}_{MAP}(y) = \arg \max_x \frac{f(y|x)f(x)}{f(y)} = \arg \max_x f(y|x)f(x)
\]

Since the prior, we assume, is uniform, the likelihood function determines the posterior distribution of the location.

The normalization with \( f(y) \) is not necessary to find just the closest RP (NN method), but in the case of averaging the K closest RPs to estimate the coordinates of the position (KNN or WKNN), this normalization may change a bit the results. So it will be calculated as follows:

\[
\hat{f}_y(y) = f_x(x) \sum_{i=1}^M f(y|x)
\]  \hspace{1cm} (11)

Regarding \( f_{\vec{v}_{ij}} \), we are going to compare the different distribution methods represented in table 1.
3.3 Code explanation

We use MATLAB to write the code to implement the algorithm. Starting from the data extracted from our database, some files .mat can be created and used for being accessed as matrices.

One is an \( \text{M} \times \text{L} \) (27x19) matrix with the RSS mean values (already averaged from the 30 samples) received in each RP (rows) from the different APs (columns). In our code it will be called \text{RSS\_RP}.

Another is an \( \text{M} \times \text{L} \times \text{K} \) (27x19x30) matrix with the RSS values obtained for each sample, in a certain RP and from an specific AP. In the code, named \text{RSS\_RP\_Kern}.

The same with the RSS mean values received in each TP. An \( \text{N} \times \text{L} \) (20x19) matrix called \text{RSS\_TP}.

Then we have again an \( \text{M} \times \text{L} \) (27x19) matrix with the variances of the distributions in each pair RP-AP. That will be called \text{varRP}.

From the database we can also get the x and y coordinates from the RPs and TPs. \( \text{RP}_x, \text{RP}_y, \text{TP}_x, \text{TP}_y \) are \( \text{M} \times 1 \) (27x1) and \( \text{N} \times 1 \) (20x1) vectors holding them.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>N_APs</td>
<td>19</td>
</tr>
<tr>
<td>N_RPs</td>
<td>27</td>
</tr>
<tr>
<td>N_TPs</td>
<td>20</td>
</tr>
<tr>
<td>num</td>
<td>30</td>
</tr>
<tr>
<td>RSS_RP</td>
<td>27x19 double</td>
</tr>
<tr>
<td>RSS_RP_Kern</td>
<td>27x19x30 double</td>
</tr>
<tr>
<td>RSS_TP</td>
<td>20x19 double</td>
</tr>
<tr>
<td>varRP</td>
<td>27x19 double</td>
</tr>
<tr>
<td>RP_x</td>
<td>27x1 double</td>
</tr>
<tr>
<td>RP_y</td>
<td>27x1 double</td>
</tr>
<tr>
<td>TP_x</td>
<td>20x1 double</td>
</tr>
<tr>
<td>TP_y</td>
<td>20x1 double</td>
</tr>
</tbody>
</table>

Figure 10: Summary of main variables used.
3.3.1 Implementing the algorithms

First of all, the .mat files must be read to access the matrices with all the information needed.

Previously, some constants are established indicating the type of distribution used (DISTR), if weighted or non-weighted is used (boolean WEIGHTED) and, in the case of Kernel, if it is done with Gaussian or Exponential distribution.

The options for DISTR are the following:
- 'd' → Deterministic method (Non Probabilistic).
- 'G' → Gaussian distribution.
- 'E' → Exponential distribution.
- 'K' → Kernel method.

The options for KERN:
- 'G' → Using Gaussian.
- 'E' → Using Exponential.

For each probabilistic approach option, \( f(y|i) \) and \( f(y) \) are calculated.

\[
f(y|i) = \prod_{j=1}^{L} f_{ij}(y_j - \bar{a}_{ij})
\]

3.3.1.1 Gaussian

\%GAUSSIAN
for k=1:N_TPs
  for i=1:N_RPs
    for j=1:N_APs
      prob_cond_y_x(i,k)=prob_cond_y_x(i,k)*pre_calc_gauss(i,j,k); %f(y|i)
    end
  end
  prob_y(k)=sum(prob_cond_y_x(:,k));
  prob_cond_y_x(:,k) = prob_cond_y_x(:,k)/prob_y(k); %normalized
end

Where pre_calc_gauss(i,j,k) is a matrix with all the values calculated for gaussian distribution and cancelling possible resulting 0s.

\[
f_{ij}(x) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} \exp\left(-\frac{x^2}{2\sigma_{ij}^2}\right)
\]
\begin{verbatim}
pre_calc_gauss(i,j,k)={1/sqrt(2*pi*varRP(i,j))}*
exp((-(RSS_TP(k,j)-RSS_RP(i,j))^2)/varRP(i,j));

Zeros are cancelled right after in the code, before executing the prob_cond_y_x(i,k) lines.

3.3.1.2 Exponential

\[ f_{\vec{v}_{ij}}(x) = \frac{1}{2}e^{(-x)} \]

Loops are left out

%EXponential
%f(y|i)
prob_cond_y_x(i,k)=prob_cond_y_x(i,k)*0.5*exp(-abs(RSS_TP(k,j)-RSS_RP(i,j)));

3.3.1.3 Kernel

\[
K = \sum_{k=1}^{\mid a_{ij}\mid} K\left( \frac{x + \hat{a}_{ij} - a_{kj}}{h} \right)
\]

Where K can be G or E.

%KERNEL
for l=1:num
  if KERNEL=='G'
    G(i,k)=G(i,k)+(1/sqrt(2*pi*var2(i,j)))*
    exp((-(RSS_TP(k,j)-RSS_RP_Kern(i,j,l))/h)^2)/var2(i,j));
  elseif KERNEL=='E'
    E(i,k)=E(i,k)+0.5*exp(-abs((RSS_TP(k,j)-RSS_RP_Kern(i,j,l))/h));
  end
end

\[ f_{\vec{v}_{ij}}(x) = \frac{K}{\mid a_{ij}\mid h} \]

Loops are left out

if KERNEL=='G'
  prob_cond_y_x(i,k)=prob_cond_y_x(i,k)*G(i,k)/(num*h); %f(y|i)
endif
else if KERNEL=='E'
  prob_cond_y_x(i,k)=prob_cond_y_x(i,k)*E(i,k)/(num*h); %f(y|i)
endif
end

Parameter h of Kernel function can be changed in the code whenever.
\end{verbatim}
Parameter num is the number of samples used for measuring RPs in the offline phase.

Finally, var2 parameter is the general variance used for Kernel.

After computing \( f(y|i) \rightarrow f(x|y) \) is calculated:

\[
f(x|y) = \frac{f(y|x)f(x)}{f(y)}
\]

\[
f(y|x) = f(y|i)
\]

\[
f(y|i)_{Norm} = \frac{f(y|i)}{f(y)}
\]

\[
f(x|y) = f(y|i)_{Norm}f(x)
\]

\[\text{prob}_\text{cond}_x_y=\text{prob}_\text{cond}_y_x*\text{prob}_x;\]

\[\text{prob}_\text{cond}_x_y\] is a 27x20 matrix with all the evaluated conditional probabilities for every pair RP-TP. Each column contains the probabilities of a TP for the different RPs. These probabilities should be put in order.

\[
\text{[values pos_sorted]} = \text{sort}(\text{prob}_\text{cond}_x_y, \text{'descend'});
\]

<table>
<thead>
<tr>
<th>TP</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.4086e–18</td>
<td>1.0000</td>
<td>0.0000</td>
<td>···</td>
<td>0.7870</td>
<td>1.0000</td>
<td>···</td>
<td>0.0000</td>
<td>1.0000</td>
<td>0.4378</td>
<td>0.0000</td>
</tr>
<tr>
<td>3</td>
<td>2.2336e–19</td>
<td>1.2983e–18</td>
<td>1.5749e–18</td>
<td>1.5749e–18</td>
<td>1.5749e–18</td>
<td>1.5749e–18</td>
<td>···</td>
<td>1.5749e–18</td>
<td>1.5749e–18</td>
<td>1.5749e–18</td>
<td>1.5749e–18</td>
</tr>
<tr>
<td>5</td>
<td>···</td>
<td>7.7594e–18</td>
<td>4.1286e–52</td>
<td>4.7851e–16</td>
<td>2.0068e–11</td>
<td>···</td>
<td>1.1311e–12</td>
<td>1.5875e–18</td>
<td>3.9786e–05</td>
<td>1.1039e–45</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>···</td>
<td>···</td>
<td>1.4000e–14</td>
<td>6.1375e–09</td>
<td>2.2516e–12</td>
<td>3.5000e–11</td>
<td>···</td>
<td>1.1646e–10</td>
<td>4.0843e–17</td>
<td>5.9963e–14</td>
<td>3.5000e–05</td>
</tr>
</tbody>
</table>

Table 3: Probability values sorted. Example for Exponential distribution.
Table 4: Closest RPs in order for every TP. Example for Exponential distribution.

<table>
<thead>
<tr>
<th>TP</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>1</td>
<td>15</td>
<td>13</td>
<td>2</td>
<td>9</td>
<td>23</td>
<td>...</td>
<td>27</td>
<td>19</td>
<td>25</td>
<td>3</td>
</tr>
<tr>
<td>2nd</td>
<td>3</td>
<td>19</td>
<td>12</td>
<td>1</td>
<td>21</td>
<td>22</td>
<td>...</td>
<td>26</td>
<td>15</td>
<td>19</td>
<td>4</td>
</tr>
<tr>
<td>3rd</td>
<td>2</td>
<td>13</td>
<td>25</td>
<td>3</td>
<td>10</td>
<td>20</td>
<td>...</td>
<td>23</td>
<td>13</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>4th</td>
<td>18</td>
<td>12</td>
<td>9</td>
<td>11</td>
<td>12</td>
<td>9</td>
<td>...</td>
<td>17</td>
<td>24</td>
<td>24</td>
<td>1</td>
</tr>
<tr>
<td>5th</td>
<td>14</td>
<td>25</td>
<td>21</td>
<td>4</td>
<td>25</td>
<td>21</td>
<td>...</td>
<td>20</td>
<td>14</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>6th</td>
<td>24</td>
<td>24</td>
<td>20</td>
<td>18</td>
<td>24</td>
<td>12</td>
<td>...</td>
<td>11</td>
<td>25</td>
<td>14</td>
<td>11</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>27th</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>...</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>15</td>
</tr>
</tbody>
</table>

Figure 11: Example of TP 2 with the K=5 closest RPs for Exponential distribution. (See Table 4)

3.3.1.4 Deterministic

Since we are focusing in this work on probabilistic approaches, the deterministic one is only computed to make a comparison between all the methods.

In the deterministic method, in stead of using the conditional pdf’s, we use the Minkovski distance (We will use the Euclidean distance p=2) to know which are the closest RPs:

\[
\text{dist. Minkovski} = \left( \sum_{i=1}^{L} \left| y^i - a^i_{RP} \right|^2 \right)^{\frac{1}{2}}
\]
% Euclidean distance
for i=1:N_RPs
  for k=1:N_TPs
    D_Minkovski(i,k) = norm(RSS_TP(k,:) - RSS_RP(i,:), 2);
  end
end

We sort them, the same that in the probabilistic method.

[dist_order, index] = sort(D_Minkovski);

Table 5: Minkovski distances sorted. Deterministic.

<table>
<thead>
<tr>
<th>TP</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>41.7992</td>
<td>21.9177</td>
<td>17.0912</td>
<td>49.9894</td>
<td>21.1550</td>
<td>22.4770</td>
<td>...</td>
<td>23.0294</td>
<td>23.9389</td>
<td>26.6545</td>
<td>46.6766</td>
</tr>
<tr>
<td>5</td>
<td>43.1241</td>
<td>22.6350</td>
<td>19.9885</td>
<td>51.5081</td>
<td>21.3084</td>
<td>24.0004</td>
<td>...</td>
<td>23.1066</td>
<td>27.5195</td>
<td>28.8648</td>
<td>42.0650</td>
</tr>
<tr>
<td>6</td>
<td>46.2914</td>
<td>23.0119</td>
<td>25.2432</td>
<td>53.1485</td>
<td>23.0066</td>
<td>27.7105</td>
<td>...</td>
<td>23.2429</td>
<td>27.5682</td>
<td>22.2371</td>
<td>50.1884</td>
</tr>
</tbody>
</table>

... ... ... ... ... ... ... ... ... ...

81.5488 | 102.3939 | 105.7155 | 86.4840 | 95.3207 | 92.3613 | ... | 80.7927 | 104.7751 | 101.7823 | 85.1873 |

Table 6: Closest RPs in order for every TP. Deterministic.

<table>
<thead>
<tr>
<th>closest RP</th>
<th>TP</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>1</td>
<td>15</td>
<td>12</td>
<td>2</td>
<td>9</td>
<td>23</td>
<td>...</td>
<td>27</td>
<td>19</td>
<td>13</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2nd</td>
<td>18</td>
<td>19</td>
<td>13</td>
<td>1</td>
<td>10</td>
<td>20</td>
<td>...</td>
<td>26</td>
<td>15</td>
<td>24</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3rd</td>
<td>14</td>
<td>25</td>
<td>25</td>
<td>3</td>
<td>21</td>
<td>22</td>
<td>...</td>
<td>23</td>
<td>13</td>
<td>14</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4th</td>
<td>2</td>
<td>13</td>
<td>21</td>
<td>4</td>
<td>22</td>
<td>21</td>
<td>...</td>
<td>20</td>
<td>18</td>
<td>19</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>5th</td>
<td>3</td>
<td>12</td>
<td>9</td>
<td>14</td>
<td>24</td>
<td>9</td>
<td>...</td>
<td>17</td>
<td>14</td>
<td>25</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6th</td>
<td>24</td>
<td>18</td>
<td>20</td>
<td>24</td>
<td>12</td>
<td>12</td>
<td>...</td>
<td>22</td>
<td>25</td>
<td>15</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>...</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>27th</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>...</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>19</td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Closest RPs in order for every TP. Deterministic.

This way we can determine which are the closest RPs and that helps us to estimate the coordinates of the position.

For probabilistic methods we use this function:

\[
\text{estimated_coords}(...);
\]
And for deterministic method we use this other function:

```c
#define ESTIMATED_COORDINATES
estimated_coords_det(...);
```

Inside these functions is differentiated whether we want to estimate the coordinates weighted or not, depending on the value of the parameter WEIGHTED previously mentioned. Also the number of RPs considered \( K \) is taken into account to average.

Both of the functions return \([\text{coord\_est, error\_TP\_def}]\);

In the same function, the error in the estimation for every TP is calculated too, but what actually interests us is knowing the mean error:

```c
mean_error_def = \text{sum(error\_TP\_def)/N\_TPs;}
mean_error\_meters\_def(K) = mean_error\_def \times SF;
```

Where the SF parameter is the Scaling Factor used to pass from pixels (in our map) to meters (in reality).

\( SF = 0.075862069; \)

In the code there will be two main options:

1. Show a map with a certain TP and the coordinates estimated taking \( K \) RPs.
2. Display a graphic to observe the mean error in the estimation depending on the number of RPs averaged \( K \).

In the case of the graphic \( E(m)/K \), the distribution can be chosen. The same happens for the option of weighted/non weighted.

For the map option, the distribution, weighted/non weighted, \( K \), and a TP must be specified. In this option some extra information will also be displayed, such as the distance between the TP and the estimated point (error), or the closest RPs found for that specific TP.

*(All the details of the functions are in the Annex 7.2)*
4 Experimental results

Here we have a couple of examples of histograms representing RSSI in a certain RP receiving from a certain AP:

![Histogram examples of RSSI from a pair RP-AP.](image1)

(a) Exponential shape histogram.  
(b) Gaussian shape histogram.

Figure 12: Histogram examples of RSSI from a pair RP-AP.

The functions that best approximate the histograms above are Exponential and Gaussian:

![Different distributions.](image2)

(a) Exponential distribution.  
(b) Gaussian distribution with variances from 0.1 to 10.

Figure 13: Different distributions.

It must be said that those examples are quite definite, but if we take them as a general rule, we can make our tests with one of these two functions.

In fact that is not the best option, since the histograms change a lot from one to another and one function cannot fit with all of them. To show it, here we have an example of the histograms received from APs in RP1 (The rest are shown in the Annexes 7.3):
However, to simplify it, we will make it with just one distribution per test.

Once we executed all the possibilities of the programmed code, we obtained some results. From the map option we could see the estimated coordinates on the plane of the floor, with an specific TP and also the closest RPs.

Nevertheless, it was on the graphics were we could take the information that actually showed which was the best method to estimate the position.
4.1 TPs in detail

Here we have some examples of how the programme estimates the coordinates choosing a certain TP and the parameter K.

**Figure 15:** Example: TP9 using Gaussian distribution Non Weighted with K=2.

Closest RPs: 11, 16

Error with $K=1$ (Distance to closestRP): 7.2420 m
Error with $K=2$ (Distance to estimated coords): 5.3533 m

**Figure 16:** Example: TP2 using Kernel distribution (with Exponential) Weighted with $K=3$ and a smoothing parameter $h=75$.

Closest RPs: 19, 24, 15

Error with $K=1$ (Distance to closestRP): 4.4966 m
Error with $K=3$ (Distance to estimated coords): 3.9811 m
Closest RPs: 17, 27, 16
Error with K=1 (Distance to closestRP): 2.5750 m
Error with K=3 (Distance to estimated coords): 2.8415 m

Closest RPs: 17, 27, 16, 26, 6
Error with K=1 (Distance to closestRP): 2.5750 m
Error with K=5 (Distance to estimated coords): 4.0017 m
Closest RPs: 1, 18, 14, 2, 3, 24, 15
Error with K=1 (Distance to closestRP): 2.4567 m
Error with K=7 (Distance to estimated coords): 1.7417 m
4.2 Mean error graphics

Table 7: Gaussian - Non Weighted results

<table>
<thead>
<tr>
<th>K</th>
<th>Mean Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6.1387m</td>
</tr>
<tr>
<td>2</td>
<td>4.3637m</td>
</tr>
<tr>
<td>3</td>
<td>5.0945m</td>
</tr>
<tr>
<td>4</td>
<td>6.0137m</td>
</tr>
<tr>
<td>5</td>
<td>6.5409m</td>
</tr>
<tr>
<td>6</td>
<td>6.7123m</td>
</tr>
<tr>
<td>7</td>
<td>7.0976m</td>
</tr>
<tr>
<td>8</td>
<td>7.5793m</td>
</tr>
<tr>
<td>9</td>
<td>7.5075m</td>
</tr>
<tr>
<td>10</td>
<td>7.5723m</td>
</tr>
</tbody>
</table>

Table 8: Gaussian - Weighted results

<table>
<thead>
<tr>
<th>K</th>
<th>Mean Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6.1387m</td>
</tr>
<tr>
<td>2</td>
<td>6.1244m</td>
</tr>
<tr>
<td>3</td>
<td>6.1244m</td>
</tr>
<tr>
<td>4</td>
<td>6.1244m</td>
</tr>
<tr>
<td>5</td>
<td>6.1244m</td>
</tr>
<tr>
<td>6</td>
<td>6.1244m</td>
</tr>
<tr>
<td>7</td>
<td>6.1244m</td>
</tr>
<tr>
<td>8</td>
<td>6.1244m</td>
</tr>
<tr>
<td>9</td>
<td>6.1244m</td>
</tr>
<tr>
<td>10</td>
<td>6.1244m</td>
</tr>
</tbody>
</table>
To determine the Kernel distribution graphic, first it was necessary to find which was the best option (the one that gives the minimum error) changing the parameters of the function \((h, \text{var}2)\).

The \text{var}2 parameter does not affect Exponential Kernel, but it does to Gaussian Kernel, although it is not reflected in the mean error.

The most important parameter here is \(h\), which affects to both the Exponential and Gaussian function, and it can change a lot the results.
So, to find the optimum value of $h$, a graphic was used, representing the minimum mean errors resulted from testing the different values of $K$ for every $h$ value (see the Matlab function used: `KERNEL_Graphic_h.m` in 7.2 Annex II).

(a) Exponential Kernel minimum errors changing $h$.

(b) Gaussian Kernel minimum errors changing $h$.

Figure 24: Graphics to find the optimum value of $h$. 
For those values, these are the Kernel graphics depending on K:

Figure 25: Gaussian Kernel (h=23) - Non Weighted graphic

<table>
<thead>
<tr>
<th>K</th>
<th>Mean Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.6734m</td>
</tr>
<tr>
<td>2</td>
<td>4.2268m</td>
</tr>
<tr>
<td>3</td>
<td>4.3447m</td>
</tr>
<tr>
<td>4</td>
<td>4.3932m</td>
</tr>
<tr>
<td>5</td>
<td>4.6123m</td>
</tr>
<tr>
<td>6</td>
<td>4.5485m</td>
</tr>
<tr>
<td>7</td>
<td>4.5881m</td>
</tr>
<tr>
<td>8</td>
<td>4.7595m</td>
</tr>
<tr>
<td>9</td>
<td>4.8934m</td>
</tr>
<tr>
<td>10</td>
<td>5.0315m</td>
</tr>
</tbody>
</table>

Table 11: Gaussian Kernel (h=23) - Non Weighted results

Figure 26: Gaussian Kernel (h=14) - Weighted graphic

<table>
<thead>
<tr>
<th>K</th>
<th>Mean Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.4126m</td>
</tr>
<tr>
<td>2</td>
<td>4.3673m</td>
</tr>
<tr>
<td>3</td>
<td>4.1391m</td>
</tr>
<tr>
<td>4</td>
<td>4.2030m</td>
</tr>
<tr>
<td>5</td>
<td>4.3520m</td>
</tr>
<tr>
<td>6</td>
<td>4.2038m</td>
</tr>
<tr>
<td>7</td>
<td>4.2654m</td>
</tr>
<tr>
<td>8</td>
<td>4.2316m</td>
</tr>
<tr>
<td>9</td>
<td>4.3051m</td>
</tr>
<tr>
<td>10</td>
<td>4.4000m</td>
</tr>
</tbody>
</table>

Table 12: Gaussian Kernel (h=14) - Weighted results
Figure 27: Exponential Kernel (h=33) - Non Weighted graphic

<table>
<thead>
<tr>
<th>K</th>
<th>Mean Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.4966m</td>
</tr>
<tr>
<td>2</td>
<td>4.4000m</td>
</tr>
<tr>
<td>3</td>
<td>4.0584m</td>
</tr>
<tr>
<td>4</td>
<td>4.4703m</td>
</tr>
<tr>
<td>5</td>
<td>4.7903m</td>
</tr>
<tr>
<td>6</td>
<td>4.6789m</td>
</tr>
<tr>
<td>7</td>
<td>4.7946m</td>
</tr>
<tr>
<td>8</td>
<td>4.7086m</td>
</tr>
<tr>
<td>9</td>
<td>4.9258m</td>
</tr>
<tr>
<td>10</td>
<td>4.0318m</td>
</tr>
</tbody>
</table>

Table 13: Exponential Kernel (h=33) - Non Weighted results

Figure 28: Exponential Kernel (h=75) - Weighted graphic

<table>
<thead>
<tr>
<th>K</th>
<th>Mean Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.4966m</td>
</tr>
<tr>
<td>2</td>
<td>4.2793m</td>
</tr>
<tr>
<td>3</td>
<td>3.9811m</td>
</tr>
<tr>
<td>4</td>
<td>4.1045m</td>
</tr>
<tr>
<td>5</td>
<td>4.2971m</td>
</tr>
<tr>
<td>6</td>
<td>4.1988m</td>
</tr>
<tr>
<td>7</td>
<td>4.3338m</td>
</tr>
<tr>
<td>8</td>
<td>4.2940m</td>
</tr>
<tr>
<td>9</td>
<td>4.3961m</td>
</tr>
<tr>
<td>10</td>
<td>4.4487m</td>
</tr>
</tbody>
</table>

Table 14: Exponential Kernel (h=75) - Weighted results
And finally, the deterministic results:

**Figure 29: deterministic - Non Weighted graphic**

<table>
<thead>
<tr>
<th>K (number of RPs taken)</th>
<th>Mean Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>d - NW</td>
</tr>
<tr>
<td>1</td>
<td>4.5569m</td>
</tr>
<tr>
<td>2</td>
<td><strong>3.6796m</strong></td>
</tr>
<tr>
<td>3</td>
<td>4.1864m</td>
</tr>
<tr>
<td>4</td>
<td>4.6938m</td>
</tr>
<tr>
<td>5</td>
<td>4.3904m</td>
</tr>
<tr>
<td>6</td>
<td>4.6385m</td>
</tr>
<tr>
<td>7</td>
<td>4.4285m</td>
</tr>
<tr>
<td>8</td>
<td>4.6469m</td>
</tr>
<tr>
<td>9</td>
<td>4.9372m</td>
</tr>
<tr>
<td>10</td>
<td>5.3544m</td>
</tr>
</tbody>
</table>

Table 15: deterministic - Non Weighted results

**Figure 30: deterministic - Weighted graphic**

<table>
<thead>
<tr>
<th>K (number of RPs taken)</th>
<th>Mean Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>d - W</td>
</tr>
<tr>
<td>1</td>
<td>4.5569m</td>
</tr>
<tr>
<td>2</td>
<td><strong>3.7186m</strong></td>
</tr>
<tr>
<td>3</td>
<td>4.0577m</td>
</tr>
<tr>
<td>4</td>
<td>4.3895m</td>
</tr>
<tr>
<td>5</td>
<td>4.1500m</td>
</tr>
<tr>
<td>6</td>
<td>4.3429m</td>
</tr>
<tr>
<td>7</td>
<td>4.2171m</td>
</tr>
<tr>
<td>8</td>
<td>4.3210m</td>
</tr>
<tr>
<td>9</td>
<td>4.5226m</td>
</tr>
<tr>
<td>10</td>
<td>4.7827m</td>
</tr>
</tbody>
</table>

Table 16: deterministic - Weighted results
The methods, sorted from the one that gives the highest to the lowest mean error in the estimation, are:

GW > GNW > EW > KGNW > KGW > KENW > KEW > ENW > dW > dNW

So the best option is the deterministic Non Weighted method, and for the probabilistic, the best one is the Exponential Non Weighted.

![Mean error in the estimation (Averaging K RPs)](image)

Figure 31: Graphic of the best options for deterministic and probabilistic approaches

We can see how the mean errors in the estimation are very similar.

In fact, the difference between those different options is even measurable, since the minimum mean error of each method varies in cm from one to another, and on the map you cannot appreciate it. So that is not enough to determine which is the best.
5 Conclusions and future development

We expected from most of the literature written, the probabilistic method to be better than the deterministic. True is that it is by centimeters that the mean error in the estimation is lower in the deterministic than in the best case of probabilistic. For that reason we think that we could still make some improvements in the probabilistic method, some of which are maybe beyond the scope of this project.

The first action would be to take more samples in the offline phase to define more the distributions. In my project I took 30 samples, and the next step could be to take 50 in stead.

About the normalization of $f_y(y)$, this could change a bit the results. If a good normalization was taken we may get a lower error for each case.

Looking at all the histograms for each pair RP-AP (7.3 Annex III) we can observe how different they are between them. So, one possibility to improve the results of probabilistic approach could be using different distribution functions for each pair when computing $f(y|i)$, according to the shape of each histogram. In fact, it would make a lot of sense, since the histograms are something you can see from the very beginning once you create the database with all the RPs, and they don’t change. Moreover, it is very difficult to approximate all the histograms to a single distribution function. They are all too different.

In a more realistic situation, $f_x(x)$ does not have to be necessarily constant. It can be calculated as a pdf considering the frequency of people walking through the different areas by estadistics, for example. In our case, the corridor could have a higher passing frequency than the offices, and therefore, a higher probability of being there.

If we also take into account that we can create more RPs or consider more APs detected, not just the ones that we put in the ceiling of the DIET 1st and 2nd floor for our tests, the results should be more accurate.

And finally, say that this has been an initial and quite basic experiment to test the probabilistic approach, but there are many factors involved in the measurements such as the orientation (that we only took one, North), or the impact of human body presence, or the device used for measuring (tablet, mobile phone, laptop...), that can alter considerably the power received.

For all of this, we think that in such a non-ideal situation, where you have to consider many aspects that make the estimation harder to implement, it might be better to apply the deterministic method.
6 Bibliography


[2] Giuseppe Caso, Luca De Nardis, Maria-Gabriella Di Benedetto, "Frequentist Inference for WiFi Fingerprinting 3D Indoor Positioning", DIET Department, Sapienza University of Rome, Italy.


[5] Azadeh Kushki (Student Member, IEEE), Konstantinos N. Plataniotis (Senior Member, IEEE) and Anastasios N. Venetsanopoulos (Fellow, IEEE), "Kernel-Based Positioning in Wireless Local Area Networks", IEEE TRANSACTIONS ON MOBILE COMPUTING, VOL. 6, NO. 6, JUNE 2007.
7 Annexes

7.1 Annex I: Map of DIET 2nd floor.

Figure 32: All TPs (o) and RPs (x).

7.2 Annex II: Matlab code

`prob_approach_Graphic.m`

```matlab
%SHOWS THE MEAN ERROR (K) GRAPHICS DEPENDING ON THE DISTRIBUTION USED

clear all
clc

%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
DISTR='d'; % Gaussian-->'G' | Exponential-->'E' | Kernel-->'K' | deterministic-->'d'
KERNEL='E';
h=75; %Kernel smoothing parameter
sigma=7; %Kernel variance
WEIGHTED=false;
K_MAX=10; %To represent in the graphic (Top=27)
colour='k-'; %r -> red | b -> blue | g -> green | m -> magenta
| k -> black | y -> yellow ....
extra_color=false;
color=[0.8 0.5 0]; %For extra colours
%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%

RP=load('Diet_Partial_IIF_side_LEFT_March2016_RPs.mat');
RP_Kern=load('Santi_samples.mat');
TP=load('Diet_Partial_IIF_side_LEFT_March2016_TPs.mat');

%Table RSS values are already the mean of the different samples taken at the RPs and TPs
RSS_RP = RP.RSS_RPs_SPinV;
RSS_RP_Kern = RP_Kern.RSS_WiFiC; %3D - RPs|APs|samples
```
RSS_TP = TP.RSS_TPs_SPinV;
varRP = RP.RSS_var_RPs_SPinV;
%coordinates
RPx = RP.RPs_X;
RPy = RP.RPs_Y;
TPx = TP.TPs_X;
TPy = TP.TPs_Y;

[N_RPs, N_APs, num] = size(RSS_RP_Kern);
[N_TPs, N_APs] = size(RSS_TP);

SF = 0.075862069; %To pass from pixels to meters
var2=sigma*ones(N_RPs,N_APs);

prob_cond_y_x = ones(N_RPs,N_TPs);
prob_y = zeros(1,N_TPs);
error_TP=zeros(1,N_TPs); %Error with K=1
error_TP_def=zeros(1,N_TPs); %Error with the chosen value of K
mean_error_meters_def=zeros(1,K_MAX);
coord_est=zeros(N_TPs,2); %Coordinates [(x,y) in columns] estimated for each TP

if DISTR=='G'
    pre_calc_gauss=zeros(N_RPs,N_APs,N_TPs);

    %Looking for the values resulting 0
    for k=1:N_TPs
        for i=1:N_RPs
            for j=1:N_APs
                pre_calc_gauss(i,j,k)=(1/sqrt(2*pi*varRP(i,j)))*exp((-RSS_TP(k,j)-RSS_RP(i,j))^2)/varRP(i,j));
            end
        end
    end

    %Cancelling 0s
    for k=1:N_TPs
        for i=1:N_RPs
            for j=1:N_APs
                if pre_calc_gauss(i,j,k)==0; %when the values of the exponential function are too high...
                    A=pre_calc_gauss(i,:,k);
                    pre_calc_gauss(i,j,k)=min(A(A~=0)); %Substitute 0 for the minimum of the row ~0
                elseif RSS_TP(k,j)==RSS_RP(i,j) %when the values of the exponential function cancels...
                    A=pre_calc_gauss(i,:,k);
                    %Substitute the high values for the next maximum of the row
                    pre_calc_gauss(i,j,k)=max(A(~=(1/sqrt(2*pi*varRP(i,j)))));
                end
            end
        end
    end

    %GAUSSIAN
    for k=1:N_TPs
        for i=1:N_RPs
            for j=1:N_APs
                prob_cond_y_x(i,k)=prob_cond_y_x(i,k)*pre_calc_gauss(i,j,k); %p(y|i)
            end
        end
    end
end
end
prob_y(k)=sum(prob_cond_y_x(:,k));
prob_cond_y_x(:,k) = prob_cond_y_x(:,k)/prob_y(k);
end

elseif DISTR=='E'
%EXPONENTIAL
for k=1:N_TPs
    for i=1:N_RPs
        for j=1:N_APs
            %p(y|i)
            prob_cond_y_x(i,k)=prob_cond_y_x(i,k)*
            0.5*exp(-abs(RSS_TP(k,j)-RSS_RP(i,j)));
        end
    end
    prob_y(k)=sum(prob_cond_y_x(:,k));
    prob_cond_y_x(:,k) = prob_cond_y_x(:,k)/prob_y(k);
end

elseif DISTR=='K'
%KERNEL
G=zeros(N_RPs,N_TPs);
E=zeros(N_RPs,N_TPs);
for k=1:N_TPs
    for i=1:N_RPs
        for j=1:N_APs
            for l=1:num
                if KERNEL=='G'
                    G(i,k)=G(i,k)+(1/sqrt(2*pi*var2(i,j)))*
                    exp((-((RSS_TP(k,j)-RSS_RP_Kern(i,j,l))/h)^2)/var2(i,j));
                elseif KERNEL=='E'
                    E(i,k)=E(i,k)+0.5*exp(-abs((RSS_TP(k,j)-RSS_RP_Kern(i,j,l))/h));
                end
            end
            if KERNEL=='G'
                prob_cond_y_x(i,k)=prob_cond_y_x(i,k)*G(i,k)/(num*h); %p(y|i)
            elseif KERNEL=='E'
                prob_cond_y_x(i,k)=prob_cond_y_x(i,k)*E(i,k)/(num*h); %p(y|i)
            end
        end
    end
    prob_y(k)=sum(prob_cond_y_x(:,k));
    prob_cond_y_x(:,k) = prob_cond_y_x(:,k)/prob_y(k);
end

if DISTR=='d'
%Deterministic
D_Minkovski=zeros(N_RPs,N_TPs);
for i=1:N_RPs
    for k=1:N_TPs
        D_Minkovski(i,k)=norm(RSS_TP(k,:)-RSS_RP(i,:),2); %Euclidean distance
end

end

[i=1:N_TPs
  %Error with K=1
  error_TP(i)=sqrt(((RPx(index(i,i))−TPx(i))^2)+(RPy(index(i,i))−TPy(i))^2));

for K=1:K_MAX
  format short
  %ESTIMATED COORDINATES
  [coord_est,error_TP_def]=estimated_coords_det(K,WEIGHTED,RPx,RPy,TPx,TPy,
  N_TPs, dist_order, index, error_TP_def, coord_est);
  mean_error_def=sum(error_TP_def)/N_TPs;
  mean_error_meters_def(K)=mean_error_def*SF;
  fprintf('ACT. MEAN ERROR (in meters): %2.4fm\n\n', mean_error_meters_def(K))
end

else %PROBABILISTIC
  \( p(x) \)
  prob_x=1/N_RPs;
  \( p(x|y)=p(y|x)p(x) \)
  prob_cond_y_x=prob_cond_y_x*prob_x;
  [values, pos_sorted] = sort(prob_cond_y_x, 'descend');

  for K=1:K_MAX
    %Compute of the K closest RPs (regarding signal strength)
    closestRPs=zeros(K,N_TPs); %Matrix with the K closest RPs per each TP
    closestRPs, error_TP)=K_closest_RPs(K,RPx,RPy,TPx,TPy,N_RPs,N_TPs,
    pos_sorted(1,:),closestRPs,error_TP,prob_cond_y_x);
    format short
    %ESTIMATED COORDINATES
    [coord_est,error_TP_def]=estimated_coords(K,WEIGHTED,RPx,RPy,TPx,TPy,
    N_TPs, prob_cond_y_x, pos_sorted, error_TP_def, coord_est);
    mean_error=sum(error_TP_def)/N_TPs;
    mean_error_meters=mean_error*SF;
    mean_error_def=sum(error_TP_def)/N_TPs;
    mean_error_meters_def(K)=mean_error_def*SF;
    K %Shows the value of K in each iteration
    fprintf('ACT. MEAN ERROR (in meters): %2.4fm\n\n', mean_error_meters_def(K))
  end
end

closestRPs

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

37
hold all
axis_x=1:K_MAX;
figure(2)
p=plot(axis_x,mean_error_meters_def,colour);
if xtra_color
    set(p,'Color',color)
end
title('Mean error in the estimation (Averaging K RPs)')
xlabel('K (number of RPs taken)') % x-axis label
ylabel('Error (m)') % y-axis label

prob_approach_MAP_TPdetails.m

%DISPLAYS THE FLOOR MAP WITH THE TP, THE K CLOSEST RPs AND THE ESTIMATED COORDS

clear all
clc

%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
DISTR='K' % Gaussian-->'G' | Exponential-->'E' | Kernel-->'K' | deterministic-->'d'
KERNEL='E';
h=75; %Kernel smoothing parameter
WEIGHTED=true;
K=3;
TP_detail=2;
%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%
RP=load('Diet_Partial_IIF_side_LEFT_March2016_RPs.mat');
RP_Kern=load('Santi_samples.mat');
TP=load('Diet_Partial_IIF_side_LEFT_March2016_TPs.mat');
%Table RSS values are already the mean of the different samples taken at the RPs and TPs
RSS_RP = RP.RSS_RPs_SPinV;
RSS_RP_Kern = RP_Kern.RSS_WiFiC; %3D - RPs|APs|samples
RSS_TP = TP.RSS_TPs_SPinV;
varRP = RP.RSS_var_RPs_SPinV;
%coordinates
RPx = RP.RPs_X;
RPy = RP.RPs_Y;
TPx = TP.TPs_X;
TPy = TP.TPs_Y;
[N_RPs, N_APs] = size(RSS_RP);
[N_TPs, N_APs] = size(RSS_TP);
SF = 0.075862069; %To pass from pixels to meters
prob_cond_y_x = ones(N_RPs,N_TPs);
error_TP=zeros(1,N_TPs); %Error with K=1
error_TP_def=zeros(1,N_TPs); %Error with the chosen value of K
coord_est=zeros(N_TPs,2); %Coordinates \((x,y)\) in columns\ estimated for each TP

if DISTR=='G'
    pre_calc_gauss=zeros(N_RPs,N_APs,N_TPs);
    for k=1:N_TPs
        for i=1:N_RPs
            for j=1:N_APs
                pre_calc_gauss(i,j,k)=(1/sqrt(2*pi*varRP(i,j)))*
                exp((-((RSS_TP(k,j)-RSS_RP(i,j))^2)/varRP(i,j)));
            end
        end
    end
    for k=1:N_TPs
        for i=1:N_RPs
            for j=1:N_APs
                if pre_calc_gauss(i,j,k)==0;
                    A=pre_calc_gauss(i,:,k);
                    pre_calc_gauss(i,j,k)=min(A(A~=0));
                elseif RSS_TP(k,j)==RSS_RP(i,j)
                    A=pre_calc_gauss(i,:,k);
                    pre_calc_gauss(i,j,k)=max(A(A~=(1/sqrt(2*pi*varRP(i,j)))));
                end
            end
        end
    end
else
    elseif DISTR=='E'
        %EXPONENTIAL
        for k=1:N_TPs
            for i=1:N_RPs
                for j=1:N_APs
                    prob_cond_y_x(i,k)=prob_cond_y_x(i,k)*pre_calc_gauss(i,j,k); %p(y|i)
                end
            end
        end
    end
else
    elseif DISTR=='K'
        %KERNEL
        G=zeros(N_RPs,N_TPs);
        E=zeros(N_RPs,N_TPs);
        for k=1:N_TPs
            for i=1:N_RPs
                for j=1:N_APs
                    var2(i,j)=7;
                end
            end
        end
num=30;
for l=1:num
    if KERNEL=='G'
        G(i,k)=G(i,k)+(1/sqrt(2*pi*var2(i,j)))*
        exp((-((RSS_TP(k,j)-RSS_RP_Kern(i,j,l))/h)^2)/var2(i,j));
    elseif KERNEL=='E'
        E(i,k)=E(i,k)+0.5*exp(-abs((RSS_TP(k,j)-RSS_RP_Kern(i,j,l))/h));
    end
end
if KERNEL=='G'
    prob_cond_y_x(i,k)=prob_cond_y_x(i,k)*G(i,k)/(num*h); %p(y|i)
elseif KERNEL=='E'
    prob_cond_y_x(i,k)=prob_cond_y_x(i,k)*E(i,k)/(num*h); %p(y|i)
end
end
prob_y(k)=sum(prob_cond_y_x(:,k));
prob_cond_y_x(:,k) = prob_cond_y_x(:,k)/prob_y(k);
end
if DISTR=='d'
%Deterministic
D_Minkovski=zeros(N_RPs,N_TPs);
for i=1:N_RPs
    for k=1:N_TPs
        D_Minkovski(i,k)=norm(RSS_TP(k,:)-RSS_RP(i,:),2); %Euclidean distance
    end
end
[dist_order, index]=sort(D_Minkovski);
for i=1:N_TPs
    error_TP(i)=sqrt(((RPx(index(1,i))-TPx(i))^2)+
    ((RPy(index(1,i))-TPy(i))^2)); %Error with K=1
end
format short
%ESTIMATED COORDINATES
[coord_est, error_TP_def]=estimated_coords_det(K,WEIGHTED,RPx,RPy,TPx,TPy,
N_TPs,dist_order,index,error_TP_def,coord_est);
mean_error_def=sum(error_TP_def)/N_TPs;
mean_error_meters_def=mean_error_def*SF;
else %PROBABILISTIC
    %p(x)
    prob_x=1/N_RPs;
    %p(x|y)=p(y|x)p(x)
    prob_cond_x_y=prob_cond_y_x*prob_x;
    [values, pos_sorted] = sort(prob_cond_x_y, 'descend');

    %Compute of the K closest RPs (regarding signal strength)
    closestRPs=zeros(K,N_TPs); %Matrix with the K closest RPs per each TP
    closestRPs, error_TP)=K_closest_RPs(K,RPx,RPy,TPx,TPy,N_RPs,N_TPs,
    pos_sorted(1,:),closestRPs, error_TP, prob_cond_x_y);
end
format short

% ESTIMATED COORDINATES
[coord_est, error_TP_def] = estimated_coords(K, WEIGHTED, RPx, RPy, TPx, TPy, N_TPs, prob_cond_x_y, pos_sorted, error_TP_def, coord_est);

mean_error = sum(error_TP)/N_TPs;
mean_error_meters = mean_error*SF;
mean_error_def = sum(error_TP_def)/N_TPs;
mean_error_meters_def = mean_error_def*SF;

fprintf('

PREV. MEAN ERROR (in pixels): %3.4fpx
', mean_error)
fprintf('ACT. MEAN ERROR (in pixels): %3.4fpx

', mean_error_def)
end

if DISTR~='d'
    fprintf('PREV. MEAN ERROR (in meters): %2.4fm', mean_error_meters)
    fprintf('
ACT. MEAN ERROR (in meters): %2.4fm

', mean_error_meters_def)
else
    fprintf('PREV. MEAN ERROR (in pixels): %3.4fpx
', mean_error)
fprintf('ACT. MEAN ERROR (in pixels): %3.4fpx

', mean_error_def)
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
−−
TEST OF SPECIFIC VALUES
−−
%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure(2)
if DISTR=='d'
    details_TP(TP_detail, K, RPx, RPy, TPx, TPy, coord_est, index(1,:),
        index, error_TP, error_TP_def, SF);
else
    details_TP(TP_detail, K, RPx, RPy, TPx, TPy, coord_est,
        pos_sorted(1,:), pos_sorted, error_TP, error_TP_def, SF);
end

K_closest_RPs.m

function [ closestRPs, error_TP ] = K_closest_RPs(K, RPx, RPy, TPx, TPy,
    N_RPs, N_TPs, pos_max_RP, closestRPs, error_TP, prob_cond_x_y )
% Compute of the K closest RPs (regarding signal strength)
for i=1:N_TPs
    cont=2;
    % Error with K=1
    error_TP(i)=sqrt(((RPx(pos_max_RP(i))-TPx(i))^2)+
        ((RPy(pos_max_RP(i))-TPy(i))^2));
    % The 1st closest RP is for the maximum value of prob.
    closestRPs(1,i)=pos_max_RP(i);
    MAX=0;
    exit=false;
    while cont<K
        for j=1:N_RPs
            for l=1:K
                if j==closestRPs(l,i)

6
exit=true; %Not consider this RPj position.
%Already taken as one of the maximums.
end
end
if ~exit && prob_cond_x_y(j,i)>=MAX
MAX=prob_cond_x_y(j,i);
closestRPs(cont,i)=j;
end
exit=false;
end
MAX=0;
cont=cont+1;
end
end

estimated_coords.m

function [ coord_est, error_TP_def ] = estimated_coords( K, WEIGHTED, RPx, RPy, 
TPx, TPy, N_TPs, prob_cond_x_y, closestRPs, error_TP_def, coord_est )
%Compute of estimated coordinates averaging the closest K RPs
x=zeros(K,1);
y=zeros(K,1);
probs=zeros(K,1);
for i=1:N_TPs
  for j=1:K
    x(j)=RPx(closestRPs(j,i)); %All the x coordinates of the K closest RPs
    y(j)=RPy(closestRPs(j,i));  %All the y coordinates of the K closest RPs
    %Probabilities of the K closest RPs
    probs(j,i)=prob_cond_x_y(closestRPs(j,i),i);
  end
  %WEIGHTED
  if WEIGHTED
    coord_est(i,1)= x'*probs(:,i)/(sum(probs(:,i))); %x component
    coord_est(i,2)= y'*probs(:,i)/(sum(probs(:,i))); %y component
  end
  %NOT weighted
  if ~WEIGHTED
    coord_est(i,1)=sum(x)/K; %x component
    coord_est(i,2)=sum(y)/K; %y component
  end
  error_TP_def(i)=sqrt(((coord_est(i,1)-TPx(i))^2)+
  ((coord_est(i,2)-TPy(i))^2));
end
end

estimated_coords_det.m

function [ coord_est, error_TP_def ] = estimated_coords_det( K, WEIGHTED, RPx, RPy, 
TPx, TPy, N_TPs, prob_cond_x_y, closestRPs, error_TP_def, coord_est )
%Compute of estimated coordinates averaging the closest K RPs
x=zeros(K,1);
y=zeros(K,1);
probs=zeros(K,1);
for i=1:N_TPs
  for j=1:K
    x(j)=RPx(closestRPs(j,i)); %All the x coordinates of the K closest RPs
    y(j)=RPy(closestRPs(j,i));  %All the y coordinates of the K closest RPs
    %Probabilities of the K closest RPs
    probs(j,i)=prob_cond_x_y(closestRPs(j,i),i);
  end
  %WEIGHTED
  if WEIGHTED
    coord_est(i,1)= x'*probs(:,i)/(sum(probs(:,i))); %x component
    coord_est(i,2)= y'*probs(:,i)/(sum(probs(:,i))); %y component
  end
  %NOT weighted
  if ~WEIGHTED
    coord_est(i,1)=sum(x)/K; %x component
    coord_est(i,2)=sum(y)/K; %y component
  end
  error_TP_def(i)=sqrt(((coord_est(i,1)-TPx(i))^2)+
  ((coord_est(i,2)-TPy(i))^2));
end
end
TPx, TPy, N_TPs, dist_order, index, error_TP_def, coord_est )

%Compute of estimated coordinates averaging the closest K RPs
x=zeros(K,1);
y=zeros(K,1);
weights=zeros(K,1);

for i=1:N_TPs
    for j=1:K
        x(j)=RPx(index(j,i)); %All the x coordinates of the K closest RPs
        y(j)=RPy(index(j,i)); %All the y coordinates of the K closest RPs
        weights(j,i)=1/dist_order(j,i); %weights in order
    end

    %WEIGHTED
    if WEIGHTED
        coord_est(i,1)= x'*weights(:,i)/(sum(weights(:,i))); %x component
        coord_est(i,2)= y'*weights(:,i)/(sum(weights(:,i))); %y component
    end

    %NOT weighted
    if ~WEIGHTED
        coord_est(i,1)=sum(x)/K; %x component
        coord_est(i,2)=sum(y)/K; %y component
    end

%Error averaging K RPs
error_TP_def(i)=sqrt(((coord_est(i,1)-TPx(i))^2)+((coord_est(i,2)-TPy(i))^2));
end
end

details_TP.m

function [] = details_TP(i, K, RPx, RPy, TPx, TPy, coord_est, pos_max_RP, closestRPs, error_TP, error_TP_def, SF)

error_meters=error_TP(i)*SF;
error_meters_def=error_TP_def(i)*SF;

fprintf('
−−TP: %i−−

', i) %prints the TPi chosen, in detail
fprintf('PREVIOUS POSITION ESTIMATED: RP_%i	[%3.4f %3.4f]
', pos_max_RP(i), [RPx(pos_max_RP(i)) RPy(pos_max_RP(i))])
fprintf('ACTUAL POSITION ESTIMATED: [%3.4f %3.4f]
', coord_est(i,:))
fprintf('PREV. ERROR IN THE ESTIMATION (in pixels): %3.4f px
', error_TP(i))
fprintf('ACT. ERROR IN THE ESTIMATION (in pixels): %3.4f px
', error_TP_def(i))
fprintf('PREV. ERROR IN THE ESTIMATION (in meters): %2.4f m
', error_meters)
fprintf('ACT. ERROR IN THE ESTIMATION (in meters): %2.4f m
', error_meters_def)

%IMAGE OF TPs / RPs / Location estimation
floor2=imread('DIET_Floor_2_Ultraport.bmp');
imshow(floor2);
hold all
s1=scatter(TPx(i),TPy(i), 'd');
KERNEL_Graphic_h.m

%SHOWS IN A GRAPHIC THE MINIMUM ERROR(h) FOR KERNEL DISTRIBUTION.

clear all
clc

KERNEL='E'; %Gaussian-->'G' | Exponential--> 'E'
sigma=7; %Varianza Kernel
WEIGHED=false;
K_MAX=10; %Top=27
colour='m.-'; %r -> red  |  b -> blue  |  g -> green  |  m -> magenta  |  k -> black  |  y -> yellow ...

RP=load('Diet_Partial_IIF_side_LEFT_March2016_RPs.mat');
RP_Kern=load('Santi_samples.mat');
TP=load('Diet_Partial_IIF_side_LEFT_March2016_TPs.mat');

s1.LineWidth = 1.2;
s1.MarkerEdgeColor = 'r';
s1.MarkerFaceColor = 'r';

s2=scatter(RPx(pos_max_RP(i)),RPy(pos_max_RP(i)), 'filled');
s2.LineWidth = 1;
s2.MarkerEdgeColor = 'k';
s2.MarkerFaceColor = 'k';

s3=scatter(coord_est(i,1), coord_est(i,2), 'x');
s3.LineWidth = 2.5;
s3.MarkerEdgeColor = 'k';

plot([TPx(i) RPx(pos_max_RP(i))],[TPy(i) RPy(pos_max_RP(i))],'r')
plot([TPx(i) coord_est(i,1)],[TPy(i) coord_est(i,2)],'r')
quiver(TPx(i),TPy(i),coord_est(i,1)−TPx(i),coord_est(i,2)−TPy(i),
    'linewidth',1,'MaxHeadSize',0.5, 'Color_I', 'r')
quiver(TPx(i),TPy(i),RPx(pos_max_RP(i))−TPx(i),RPy(pos_max_RP(i))−TPy(i),
    'linewidth',1,'MaxHeadSize',0.5, 'Color_I', 'r')

if K>1
    for k=2:K
        scatter(RPx(closestRPs(k,i)),RPy(closestRPs(k,i)), 'c',
            'LineWidth', 0.05, 'MarkerEdgeColor', 'k')
        plot([coord_est(i,1) RPx(closestRPs(k,i))],
             [coord_est(i,2) RPy(closestRPs(k,i))], 'g')
    end
end
end

KERNEL_Graphic_h.m
% Table RSS values are already the mean of the different samples taken at the RPs and TPs

\[
\text{RSS\_RP} = \text{RP\_RSS\_RPs\_SPinV};
\]

\[
\text{RSS\_RP\_Kern} = \text{RP\_Kern\_RSS\_WiFiC};
\]

\[
\text{RSS\_TP} = \text{TP\_RSS\_TPs\_SPinV};
\]

\[
\text{var\_RP} = \text{RP\_RSS\_var\_RPs\_SPinV};
\]

% coordinates

\[
\text{RP\_x} = \text{RP\_RPs\_X};
\]

\[
\text{RP\_y} = \text{RP\_RPs\_Y};
\]

\[
\text{TP\_x} = \text{TP\_TPs\_X};
\]

\[
\text{TP\_y} = \text{TP\_TPs\_Y};
\]

\[
[N\_RPs, N\_APs, \text{num}] = \text{size}($\text{RSS\_RP\_Kern}$);
\]

\[
[N\_TPs, N\_APs] = \text{size}($\text{RSS\_TP}$);
\]

\[
\text{SF} = 0.075862069; % To pass from pixels to meters
\]

\[
\text{min\_errors}=\text{zeros}(1,150);
\]

\[
\text{for } h=1:\text{length}(\text{min\_errors})
\]

\[
\text{prob\_cond\_y\_x} = \text{ones}(N\_RPs,N\_TPs);
\]

\[
\text{prob\_y} = \text{zeros}(1,N\_TPs);
\]

\[
\text{var2}=\text{sigma}^2\times\text{ones}(N\_RPs,N\_APs);
\]

\[
\text{error\_TP}\_\text{def}=\text{zeros}(1,N\_TPs); % Error with the chosen value of K
\]

\[
\text{mean\_error\_meters\_def}=\text{zeros}(1,K\_MAX);
\]

\[
\text{coord\_est}=\text{zeros}(N\_TPs,2); % Coordinates [(x,y) in columns] estimated for each TP
\]

% KERNEL

\[
\text{G}=\text{zeros}(N\_RPs,N\_TPs);
\]

\[
\text{E}=\text{zeros}(N\_RPs,N\_TPs);
\]

\[
\text{for } k=1:N\_TPs
\]

\[
\text{for } i=1:N\_RPs
\]

\[
\text{for } j=1:N\_APs
\]

\[
\text{if } \text{KERNEL}=='G'\]

\[
\text{G}(i,k)=G(i,k)+(1/sqrt(2*\text{pi} \times \text{var2}(i,j)))\times
\]

\[
\exp((-((\text{RSS\_TP}(k,j)-\text{RSS\_RP\_Kern}(i,j,l))/h)^2)/\text{var2}(i,j));
\]

\[
\text{elseif } \text{KERNEL}=='E'\]

\[
\text{E}(i,k)=E(i,k)+0.5\times\exp(-abs((\text{RSS\_TP}(k,j)-\text{RSS\_RP\_Kern}(i,j,l))/h));
\]

\[
\text{end}\]

\[
\text{end}\]

\[
\text{if } \text{KERNEL}=='G'\]

\[
\text{prob\_cond\_y\_x}(i,k)=\text{prob\_cond\_y\_x}(i,k)\times\text{G}(i,k)/(\text{num}\times h); % p(y|i)
\]

\[
\text{elseif } \text{KERNEL}=='E'\]

\[
\text{prob\_cond\_y\_x}(i,k)=\text{prob\_cond\_y\_x}(i,k)\times\text{E}(i,k)/(\text{num}\times h); % p(y|i)
\]

\[
\text{end}\]

\[
\text{end}\]

\[
\text{prob\_y}(k)=\text{sum}(\text{prob\_cond\_y\_x}(i,k));
\]

\[
\text{prob\_cond\_y\_x}(i,k) = \text{prob\_cond\_y\_x}(i,k)/\text{prob\_y}(k);
\]

\[
\text{% p(x)}
\]

\[
\text{prob\_x}=1;%/N\_RPs;
\]

\[
\text{prob\_x}\_y(p(x) = \text{prob\_y}(k)\times\text{prob\_x};
\]

45
for K=1:K_MAX
    closestRPs=zeros(K,N_TPs); %Matrix with the K closest RPs per each TP
    [closestRPs,error_TP]=K_closest_RPs(K,RPx,RPy,TPx,TPy,N_RPs,N_TPs,pos_sorted(1,:),closestRPs,error_TP,prob_cond_x_y);

    format short
    %ESTIMATED COORDINATES
    [coord_est,error_TP_def]=estimated_coords(K,WEIGHTED,RPx,RPy,TPx,TPy,N_TPs,prob_cond_x_y,pos_sorted,error_TP_def,coord_est);

    mean_error=sum(error_TP)/N_TPs;
    mean_error_meters=mean_error*SF;
    mean_error_def=sum(error_TP_def)/N_TPs;
    mean_error_meters_def(K)=mean_error_def*SF;
end

min_errors(h)=min(mean_error_meters_def);
end

[minimum, h_value]=sort(min_errors);
fprintf('min. ERROR (in meters): %2.4fm

', minimum(1))
fprintf('Value of h that gives the minimum: %i

', h_value(1))

hold all
axis_x=1:length(min_errors);
figure(1)
plot(axis_x,min_errors, 'colour')
title('min. errors in the estimation (Varying h)')
xlabel('h (Kernel smoothing parameter)') % x-axis label
ylabel('Min. Errors (m)') % y-axis label

hist_subplots.m

%CREATES ONE FIGURE FOR EACH RP WITH ALL THE HISTOGRAMS OF THE PAIRS RP–AP.
clear all
clc
RP_Kern=load('Santi_samples.mat');

RSS_RP_Kern = RP_Kern.RSS_WiFiC; %3D – RPs|APs|samples
[N_RPs, N_APs, num] = size(RSS_RP_Kern);
for i=1:N_RPs
    figure('Name',strcat('RP ',num2str(i)),'NumberTitle','off')
    for j=1:N_APs
        subplot(4,5,j); hist(squeeze(RSS_RP_Kern(i,j,:)),43)
        title(strcat('AP ',num2str(j)))
    end
end
7.3 Annex III: Histograms

Figure 33: Histograms in RP1

Figure 34: Histograms in RP2
Figure 35: Histograms in RP3

Figure 36: Histograms in RP4
Figure 37: Histograms in RP5

Figure 38: Histograms in RP6
Figure 39: Histograms in RP7

Figure 40: Histograms in RP8
Figure 41: Histograms in RP9

Figure 42: Histograms in RP10
Figure 43: Histograms in RP11

Figure 44: Histograms in RP12
Figure 45: Histograms in RP13

Figure 46: Histograms in RP14
Figure 47: Histograms in RP15

Figure 48: Histograms in RP16
Figure 49: Histograms in RP17

Figure 50: Histograms in RP18
Figure 51: Histograms in RP19

Figure 52: Histograms in RP20
Figure 53: Histograms in RP21

Figure 54: Histograms in RP23
Figure 55: Histograms in RP23

Figure 56: Histograms in RP24
Figure 57: Histograms in RP25

Figure 58: Histograms in RP26
Figure 59: Histograms in RP27