Robust TDOA Estimation in a Continuous Stereo Audio Track

Author: Alejandro Colomer Puig
Supervisor: Prof. Dr. Christian Schindelhauer

Technisches Facultät
Institut für Informatik
Lehrstuhl für Rechnernetze und Telematik

November 2013
Declaration of Authorship

I, Alejandro COLOMER PUIG, declare that this thesis titled, Robust TDOA Estimation in a Continuous Stereo Audio Track and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 

Date: 

i
The main goal of this Master Thesis is to determine the Time Difference Of Arrivals of two audio signals recorded with two microphones. The use of phase correlation is a vital point to take care of since the maximum of the phase correlation is, in fact, the TDOA. In this thesis, the entry point is the use of the FFT to compute the phase correlation instead of using the DFT which incurs more computational charge. The large noise of the observations taken obligates us to attempt to improve the performance of the algorithm. It is possible either to use a low pass filter to reduce the noise or simply changing some design parameters.

Still taking sake of the phase correlation of the two audio records, it is also possible to change the point of view into the use of the particle filter, which provides more accurate results although its implementation needs more time to be completed. The parameters of the particle filter may be changed and the performance in each case assessed for a better performance.

The computational charge translated into terms of time is also important. This is the reason why more algorithms are studied in this thesis. With no such accuracy as particle filter but with faster performance, the obtained results from the phase correlation are used in a heuristic algorithm, a so-called back-track tree algorithm and an interpolation algorithm. With these algorithms it is possible to obtain a percentage of error smaller than with only TDOA algorithm but at the same time a faster computation than the offered by the particle filter.

So, it is a good idea to compare those algorithms to decide in each situation which one is more suitable and appropriate in terms of time charge and accuracy.
# Contents

Declaration of Authorship i

Abstract ii

Acknowledgements v

1 Introduction 1
  1.1 Background and motivation ........................................... 1
  1.2 Particular focus ..................................................... 2

2 Related Work 4
  2.1 Work Related with Comparison of Algorithms ......................... 4
    2.1.1 Different Coordinates System .................................. 4
    2.1.2 Comparison of TDOA against TOA ............................... 5
    2.1.3 Work Related to Different Environments ......................... 11
    2.1.4 Comparison of Particle Filter Schemes ......................... 13
  2.2 Work Related to Other TDOA Schemes ................................ 17
  2.3 Other Algorithms to Locate .......................................... 22

3 Mathematical Tools 24
  3.1 DFT ................................................................. 24
  3.2 FFT ................................................................. 25
  3.3 Phase Correlation ................................................... 27
    3.3.1 Hadamard Product .............................................. 27
    3.3.2 TDOA ............................................................ 28
  3.4 Particle Filter Basics .............................................. 29
    3.4.1 Particle ....................................................... 29
    3.4.2 Extensive Basics ............................................... 30
  3.5 Mean and median ..................................................... 32
  3.6 Percentile .......................................................... 33

4 Algorithms 34
  4.1 TDOA Algorithm .................................................... 34
    4.1.1 Fixing Edge Effects ............................................ 34
    4.1.2 Fixing Noise Problems ......................................... 35
    4.1.3 Low Pass Filter ............................................... 36
4.2 Particle Filter Algorithm: Functional Principle .......................... 37
  4.2.1 Motion Step ......................................................... 37
  4.2.2 Weight Step ....................................................... 38
  4.2.3 Resampling Step ................................................... 39
  4.2.4 Final sampling ..................................................... 40
4.3 Heuristic Algorithm ..................................................... 40
4.4 Back-track tree Algorithm .............................................. 42
4.5 Interpolation Algorithm ............................................... 44

5 Experiments ................................................................. 45
  5.1 First Results ............................................................. 46
  5.2 Particle Filter Experiments ........................................... 52
    5.2.1 Experiments with Number of Particles ......................... 55
    5.2.2 Experiments with Variance Level at Taking Importance Sampling 57 
    5.2.3 Experiments with Resampling Level ............................. 57
    5.2.4 Experiments with Variance in Motion Model ................... 59
  5.3 Experiments with Heuristic Algorithm ................................. 61
  5.4 Experiments with back-track tree algorithm ........................ 62
  5.5 Experiments with Interpolation Algorithm ......................... 64
  5.6 Combination of Algorithms and Methods and Final Comparison .... 65

6 Conclusions and Future Work ............................................ 67
  6.1 Conclusions ............................................................. 67
  6.2 Future Work ............................................................ 68
Acknowledgements

On one hand, I would like to thank all that people that helped me during the implementation of every algorithm made in this Master Thesis and during the writing down as well. Specially I would like to thank my project advisor, Christian Schindelhauer, who gave me this project. Without him, this project would have never existed. I would also specially thank Johannes Wendeberg for his enormous patience with all my stuff during all the Master Thesis, for solving all my doubts and for making my stage here easier.

On the other hand, I would like to thank my home tutor from UPC (Universitat Politècnica de Catalunya, Barcelona), Francisco Javier Hernando Pericas, for his helpful and useful ideas even from far away, which have clarified some experimental conclusions by his experience in a field such as the localization of a speaker by using two microphones.

Aside from professors and supervisors, I also would like to thank my friends and university colleagues, especially to Diego Lendoiro, who helped me by creating for me an account on Matlab servers in UPC in order to allow me the chance of continuing my project from my home in Barcelona, provided that these servers would be faster than my laptop. To finish, special mention to Juan Alvarez, who also helped me by configuring some stuff once I had arrived in Barcelona; and to Noelia Sardinero, a close friend, who gave me some useful ideas that allowed me to continue with the project.
Chapter 1

Introduction

1.1 Background and motivation

The motivation for this Master Thesis comes from the following situation: a single speaker is moving in the surroundings of the receivers while talking and moving right and left as he/she wishes. In order to compute their position, the first tool provided by the chair is the Time Difference Of Arrivals (TDOA), whose computation is governed, in a general case between two receivers, for the next equation:

\[
TDOA = \frac{\|M_1 - S\| - \|M_2 - S\|}{c}
\]  

(1.1)

where \(M_1\) is the position of the first microphone, \(M_2\) is the position of the second one, \(S\) is the position of the speaker and \(c\) is the sound speed in the air. This equation represents a hyperbola that has the center in the point where the closest microphone is placed. For each pair of receivers there is a single hyperbola.

The situation which actually represents this equation in a general case is the represented in Fig. 1.1, where \(M_1\), \(M_2\) and \(M_3\) are the receivers and \(S\) is the position of the mobile.

Once we know all these values, the TDOA is easy to be computed. Anyway, in our situation, we do not know the position of the speaker, so we have to be able to compute the TDOA from another way in order to be capable of finding out the certain position of the speaker.

In our case, we will focus in an easier situation: only using two microphones in an indoor environment. The use of two microphones only allows us to compute the position of the speaker in 1 single dimension, i.e. right or left, due to the existence of only one hyperbola.
between the two receivers. Furthermore, the distance between these two microphones is completely unknown.

1.2 Particular focus

There is no only one method that allows us to reach our target. So the key point for this Thesis is not only the capacity for tracking a position but also to find out which is the most appropriated method to achieve it. That will be the part which involves more work charge, comparing some good (or not so good) algorithms and deciding which is the best one to be used in each certain application (establishing a compromise between some quality parameters).

We study thoroughly some different available options to compute the position of the speakers. We start explaining exhaustively how to compute the position of the speaker only by computing the Time Difference of Arrival (TDOA) between the two receivers, whose distance and position in the space is completely ignored by us.

After that, we start to talk about the use of the Particle Filter, which is a more accurate method used to follow speakers. However, due to its accuracy, its implementation is also more complicated than in the solution adopted using TDOA computation.

Moreover, we work changing the different design parameters of either TDOA method or Particle Filter method, whichever it is the used one. By changing this parameters, we evaluate the different reached performances of the methods, so we are capable of discerning the best option among all of them.
Finally, we finish talking about some other algorithms: Heuristic Algorithm, Back-track Tree Algorithm and Interpolation Algorithm, based on different basics from TDOA algorithm but related with it, though.

It is also important, after doing the experiments with all algorithms, to extract some conclusions that can be useful in a future work for either other people or even ourselves in order to carry on with topics related with the one addressed in this Master Thesis.
Chapter 2

Related Work

2.1 Work Related with Comparison of Algorithms

There are much related work related with this Master Thesis owing to the large number of algorithms we use here. Some work related with the comparison of different algorithms based on TDOA are especially focused on TDOA compared with TOA [2] and also comparing the different solutions achieved with TDOA based algorithms with direct solution method [3].

2.1.1 Different Coordinates System

In the last one, they work on TDOA passive location by using two different coordinates systems: rectangular coordinates and polar coordinates. They use a multi-station location due to its easily implementation and its highly precision.

Firstly, two of the observation stations receive the electromagnetic wave coming from the emitter and set a Range Difference(RD) curve by measuring the TDOA of emitting signals. To get two independent RD curves they use three observations instead of using two of them. They estimate the position of the target by adding the independent observation stations. For the TDOA location system they use a host station and three complementary stations. The equations system used lead into the same expression used in this Master Thesis in Eq. 1.1.

Next, they figure out that the key point of the TDOA location using rectangular coordinates is to calculate the distance between the target and the host station and so it will be computed the distance of the target to the other three complementary stations. For
the case of polar coordinates, only in 2-D, they compute the radial distance between the
target and the reference station \( r \) and then the angle \( \theta \).

Afterwards they compare both algorithms by observing the equation required to be solved.

Finally, as a conclusion, they extract that the difficult to solve the final equation is the
same owing to the ambiguity of the solution, but with a priori knowledge the accurate position location can be achieved and with an extra observation the ambiguity can be even deleted.

2.1.2 Comparison of TDOA against TOA

In [2] the authors work with Ultra-Wideband (UWB) sensor networks, where firstly are
necessary some measurements in order to obtain the impulse response of the channel by
using radio signals. In a general sight, they compare the use of some algorithms based
in TOA location estimation and some based in TDOA location estimation.

For both techniques they use the following algorithms: Analytical Method, Least Squares Method, Taylor Series Method, Approximate Maximum Likelihood Method, Two-Stages Maximum Likelihood Method and Generic Algorithm.

Unlike as in TDOA, TOA (Time of Arrival) algorithms use spheres instead of hyperbolas.
The intersection point of these spheres is the target location. The spheres are governed
by the following equation:

\[
(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = m_i
\]

(2.1)

being \( x, y, z \) the 3-D coordinates to locate, \( x_i, y_i, z_i \) and \( m_i \) the known coordinates of the reference nodes and the range estimations respectively; where \( i = 1, ..., n \) and \( n \) is the number of reference nodes.

In Analytical Method based on TOA, they simply substitute:

\[
x' = x - x_1
\]

(2.2)

\[
y' = y - y_1
\]

(2.3)

\[
z' = z - z_1
\]

(2.4)

and
and simply taking one of the unknowns as a parameter and the other two as a function of this parameter and they obtain:

\[ x_i' = x_i - x_1 \]  

(2.5)

There are three possibilities. If there is only one solution, the position is taken directly; if there are two solutions, it is necessary another more node and if there is no solution, it is caused by the range estimation error.

The main idea of this method is to average the three results obtained by considering each one of the coordinates as a parameter and then making the estimation.

On the other hand, in the Analytical Method based in TDOA, they have to cope with one more unknown than in TOA, which are the true distances between the reference nodes and the target node \( r_i \). All the TDOA are measured with respect to the first node of reference, so that, they subtract \( r_i \) from the range estimation of each nodes. So, to keep it valid:

\[
(x - x_j)^2 + (y - y_j)^2 + (z - z_j)^2 = (r_1 + m_{j,i})^2
\]  

(2.7)

They make again one of the unknowns to be a parameter and the other three a function of the parameter and they solve the quadratic solution to get the coordinates of the position.

Right after, the authors use the Least Squares Method, in which they subtract the first position (i=1) from the other equations such as a system of matrices remains:

\[
2At = b
\]  

(2.8)

where:

\[
t^T = [x \ y \ z].
\]  

(2.9)
\( \mathbf{A} = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ & \ddots & \vdots \\ x_n - x_1 & y_n - y_1 & z_n - z_1 \end{bmatrix} \) \hspace{1cm} (2.10)

\( \mathbf{b} = \begin{bmatrix} m_1^2 - m_2^2 + k_2 - k_1 \\ \vdots \\ m_1^2 - m_n^2 + k_n - k_1 \end{bmatrix} \) \hspace{1cm} (2.11)

where \( k_i = x_i^2 + y_i^2 + z_i^2 \). They isolate the matrix \( \mathbf{t} \) and finally find the position by the method of the Least Square.

For the case of TDOA, the authors cope with a similar system of equations but with different parameters:

\[ 2 \begin{bmatrix} x'_2 & y'_2 & z'_2 \\ \vdots \\ x'_n & y'_n & z'_n \end{bmatrix} \begin{bmatrix} x' \\ \vdots \\ z' \end{bmatrix} = \begin{bmatrix} k'_2 - m_{2,1}^2 \\ \vdots \\ k'_n - m_{n,1}^2 \end{bmatrix} + r_1 \begin{bmatrix} -m_{21} \\ \vdots \\ -m_{n1} \end{bmatrix} \] \hspace{1cm} (2.12)

where the variables \( x', y', z', x'_i, y'_i \) and \( z'_i \) are the defined above and the variables \( k'_i \) are analogue to the \( k_i \).

Finally, they isolate the vector of coordinates to obtain the final result, deriving from the parameter \( r_1 \).

For the Taylor Series Method, they separate the possible error at range estimation from the range estimation itself, so a new equation is defined as:

\[ \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} = m_i + \epsilon_i \] \hspace{1cm} (2.13)

where differently from the original equation, \( \epsilon_i \) is the mentioned range estimation error at reference node \( i \), and they call this equation as \( f_i(x, y, z) \). Whether some initial conditions such as \( x_v, y_v, z_v \) and location errors to be determined as \( \delta_x, \delta_y \) and \( \delta_z \) are given, the Taylor Series are defined as:

\[ f_{i,v} + a_{i,1} \delta_x + a_{i,2} \delta_y + a_{i,3} \delta_z \simeq m_i + \epsilon_i \] \hspace{1cm} (2.14)
where the coefficients $a_{i,j}$ are the partial derivatives of $f_i$ respect to the $j-th$ coordinate\textsuperscript{1}, and each one of them is equal to (for example for $j=1$):

$$a_{i,1} = \frac{x_v - x_i}{r_i} \quad (2.15)$$

and where:

$$r_i = \sqrt{(x_v - x_i)^2 + (y_v - y_i)^2 + (z_v - z_i)^2} \quad (2.16)$$

They use the matrix-wise form to define a new system with only one matrix-wise equation. From the initial position they guess the values of the initial conditions and then they figure out the values of the array formed as $\delta$ using the matrices equation. The initial conditions are continuously updated in order to compute the following positions.

In case of TDOA, the function is defined slightly different:

$$\sqrt{(x - x_{i+1})^2 + (y - y_{i+1})^2 + (z - z_{i+1})^2} - \sqrt{(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2} = m_{i+1,1} + \epsilon_{i+1,1} \quad (2.17)$$

The Taylor Series are applied equal to the case of TOA but the coefficients result, analogously, as:

$$a_{i,1} = \frac{x_1 - x_v}{r_1} - \frac{x_{i+1} - x_v}{r_{i+1}} \quad (2.19)$$

and the result is obtained following the same method than in case of TOA.

The next algorithm to be compared is the Approximate Maximum Likelihood Method, in which the authors assume that the probability density function of $m$ given $\theta$ (where $m$ is an array composed by the range estimations of each reference node and $\theta$ is an array formed by the coordinates to be encountered) is:

$$f(m | \theta) = (2\pi)^{-n/2}(detQ)^{-1/2}e^{-J/2} \quad (2.20)$$

where $Q$ is the covariance matrix with independent values and zero-mean corresponding to the range estimation error and $J$ is:

\textsuperscript{1}Where $j=1$ corresponds to $x$, $j=2$ corresponds to $y$ and $j=3$ corresponds to $z$.\n

\[ J = [\mathbf{m} - \mathbf{r}(\theta)]^T \mathbf{Q}^{-1} [\mathbf{m} - \mathbf{r}(\theta)] \]  
\[ (2.21) \]

So, they apply the condition for an ML estimator, which is:

\[ \frac{\partial J(\theta)}{\partial \theta} = 0 \]  
\[ (2.22) \]

and using the initial conditions and the least square estimation of the coordinates they solve a quadratic equation and keep repeating the procedure with the updated values of the coordinates in order to achieve a more accurate result.

In the case of TDOA, the matrix-wise equation to be solved is rather more complicated:

\[ 2\mathbf{H} \mathbf{D} \theta = \mathbf{H} (\mathbf{v} + 2\mathbf{r}_1 \mathbf{m}) \]  
\[ (2.23) \]

where they let the matrix \( \mathbf{H} \) to be the identity matrix,

\[ D = - \begin{bmatrix} x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ & \cdots \\ x_n - x_1 & y_n - y_1 & z_n - z_1 \end{bmatrix} \]  
\[ (2.24) \]

and

\[ \mathbf{v} = [m_{2,1}^2 + k_1 - k_2, \ldots, m_{n,1}^2 + k_1 - k_n]^T \]  
\[ (2.25) \]

Again, they repeat the procedure in order to get more results and take only that one which makes the smallest \( J \).

In the case of Two-Stage Maximum Likelihood Algorithm, both in TOA case and TDOA case, they rewrite the initial equation. In TOA, the chosen equation is:

\[ m_i^2 - k_i - [-2(x_i x + y_i y + z_i z) + s] = 0 \]  
\[ (2.26) \]

They are required to solve:

\[ \phi = 2\mathbf{B} \epsilon + \epsilon \cdot \epsilon \]  
\[ (2.27) \]
where $\mathbf{B} = \text{diag}(r_1 \cdots r_n)$. Then, the authors solve the ML estimation of the vector $\mathbf{z}(\mathbf{z} = [xyzs]^T)$ as:

$$\mathbf{z} = \arg\min((\mathbf{h} - \mathbf{Gz})^T\Phi^{-1}(\mathbf{h} - \mathbf{Gz}))$$

(2.28)

but, as they have $\Phi$ unknown, they have to define another cost function that causes:

$$\mathbf{z}' = (\mathbf{G'}^T\Phi'^{-1}\mathbf{G'})^{-1}\mathbf{G'}^T\Phi'^{-1}\mathbf{h}'$$

(2.29)

where $\mathbf{z}' = [x^2 y^2 z^2]^T$, 

$$\mathbf{G}' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix},$$

(2.30)

and $\mathbf{h}' = [z_1^2 z_2^2 z_3^2 z_4]^T$.

By working with these equations they obtain that the final estimation of the coordinates is $[xyz]^T = \pm \sqrt{z}$.

On the other hand, they rewrite the equation solution from the Two-stage Maximum Likelihood Algorithm in case of TOA as:

$$m_{i,1}^2 + 2m_{i,1}r_1 = -2(x_i x + y_i y + z_i z) + k_i - k_1$$

(2.31)

Similarly as in TOA case, the vector $\mathbf{z} = [x, y, z, s]^T$ is taken as an unknown vector. Here the values for vector $\mathbf{h}$ and the matrix $\mathbf{G}$ are different from TOA case:

$$\mathbf{h} = \begin{bmatrix} m_{2,1} - k_2 + k_1 \\ \cdots \\ m_{n,1} - k_n + k_1 \end{bmatrix}$$

(2.32)

$$\mathbf{G} = -2 \begin{bmatrix} x_{21} & y_{21} & z_{21} & m_{2,1} \\ \cdots & \cdots & \cdots & \cdots \\ x_{n1} & y_{n1} & z_{n1} & m_{2,1} \end{bmatrix}$$

(2.33)

The same as in TDOA, the value of $\phi$ is found but here $\mathbf{B} = \text{diag}\{r_2 \cdots r_n\}$. 
The first ML estimate for $z$ is the same than in TOA case and the cost function is actually the same but here:

\[
    h' = \begin{bmatrix}
        (z_1 - x_1)^2 \\
        (z_2 - y_1)^2 \\
        (z_3 - z_1^2) \\
        z_4
    \end{bmatrix}, \quad G' = \begin{bmatrix}
        1 & 0 & 0 \\
        0 & 1 & 0 \\
        0 & 0 & 1 \\
        1 & 1 & 1
    \end{bmatrix}, \quad z' = \begin{bmatrix}
        (x - x_1)^2 \\
        (y - y_1)^2 \\
        (z - z_1)^2
    \end{bmatrix}
\]

and then, the final ML estimate is encountered by: $[x\ y\ z]^T = [x_1\ y_1\ z_1]^T \pm \sqrt{z'}$.

For the Generic Algorithm, they mention in both cases the fact that this algorithm consist of the law of natural selection and genetics and three operations: reproduction, crossover and mutation.

Finally they base their comparisons in the computation of average errors, failure rate and distribution of positioning errors in both cases. As a general conclusion, they state that the performance of the algorithms depends on the network topology and the position of the target node. Their future work will be focused on the adaptive selection of the location estimation algorithms.

### 2.1.3 Work Related to Different Environments

Another point of view that can be adopted is to do such comparisons in a certain conditions given by the environment like either [4], in which the authors cope with sensor networks and assume a LOS environment, or [5] (NLOS environment).

In [4] the problem is formulated as a source localization scenario where a network of $n$ collaborate to locate a stationary emitter source whose position is unknown. Recursively, the sensors are continuously moving in each step in order to determine the emitter position.

The true positions of the sensors are, strictly talking, unknown, so it is required to define their noisy versions (which are actually the summation of the true positions plus the error addable component. The error is zero-mean Gaussian and has a covariance matrix $Q^k_z$. Equally, they model the TDOA as the true TDOA value plus the noise.

However, they work not only with TDOA but also with RDOA (Range Difference Of Arrivals), due to the equivalence easily applicable by multiplying the TDOA times the sound speed $c$ and obtaining RDOA. As it is done similarly to [2], they use a recursive algorithm that needs initial conditions but now using a Constrained Weight Least Squares (CWLS) method. This algorithm consists precisely on: at first, computing the
initial location with this method. At the next time step, the current location of the source is computed as well as the covariance of the error by using the WLS method. Taking the recently computed current localization $\hat{u}^*$ as a measurement of the unknown source position $u$ as follows:

$$\hat{u}^* = u + \Delta u^k$$ \hspace{1cm} (2.35)

where $\Delta u^k$ is the covariance of measurement noise, the location is now updated as:

$$\hat{u}^k = \hat{u}^{k-1} + K^k(y^k - \hat{u}^{k-1})$$ \hspace{1cm} (2.36)

where $K^k = \hat{P}^{k-1}(\hat{P}^{k-1} + P^k)^{-1}$ and $\hat{P}^k = (I - K^k)\hat{P}^{k-1}$. $P^k$ corresponds to the covariance of the noise.

Finally, they go back to second step of the algorithm.

In the improved version of the Recursive Algorithm, the vital difference is the use of the WLS method to compute the estimate of the current auxiliary variable $r_1^k$ and its error covariance $Q_{r,k}$ with current TDOA measurements. The auxiliary variable $r_1^k$ is used to define a new variable in the set of space coordinates: $u_1^k = [x, y, z, r_1^k]^T$.

Right after, they rearrange the non-linear TDOA measurement equations into a set of linear equations only with respect to the unknown source position $u$.

Now, the localization update is done by using a Kalman Filter and it is required to go back to the second step of the algorithm again.

To finish, they compare the performance of both algorithms and obtain the results depicted in Fig. 2.1, 2.2, 2.3, 2.4.

The situation in [5] is slightly modified from the previous one and only applied in a 2-D case only for simplicity. The equations used in this NLOS environment are quite equal to those ones already seen but the TOA-based LOS location algorithm is only accurate when the NLOS propagation is not heavy. When the NLOS propagation becomes heavy, it is necessary to apply a constraint: $G_aZ_a \leq h$, where here:

$$G_a = \begin{bmatrix} -2x_1 & -2y_1 & 1 \\ \vdots \\ -2x_M & -2y_M & 1 \end{bmatrix}$$ \hspace{1cm} (2.37)
Chapter 2. Related Work

Figure 2.1: Average Range Error over time when $\sigma_t = 100$ ns and $\sigma_x = \sigma_y = \sigma_z = \frac{10}{\sqrt{3}}$ m.

Figure 2.2: Standard Deviation over time when $\sigma_t = 100$ ns and $\sigma_x = \sigma_y = \sigma_z = \frac{10}{\sqrt{3}}$ m.

and the other two parameters are defined equally to the previous documents explained. The procedure applied is the same than in the case of LOS environment, but taking into account the existence of the constraint.

2.1.4 Comparison of Particle Filter Schemes

Always continuing with the comparisons topic, there is more work focused on the comparison of certain parts of a more global topic: for example, here some different resampling schemes for Particle Filter are compared: [6]. Among this different algorithms we can find Multinomial Resampling, Residual Resampling, Stratified Resampling and Systematic Resampling. Here they explain the basic properties of each one of the algorithms, with a chapter focused on the study of the large-sample behavior when using these algorithms.
Chapter 2. Related Work

The Multinomial Resampling is based on drawing the new positions independently from the common point mass distribution \( \sum_{j=1}^{m} \omega_j \delta_{\xi_j} \). They achieve it practically by repeating the inversion method: firstly, they draw \( n \) independent uniforms\(^2 \) \( \{U_i\}_{1 \leq i \leq n} \) in the interval \((0,1]\). Then is set \( I^i = D_{\omega}^{\text{inv}}(U^i) \) and \( \hat{\xi}^i = \xi^{I^i} \), where \( D_{\omega}^{\text{inv}} \) is the inverse of the cumulative distribution function associated with the normalized weights of the particles and \( \xi^i \) is a concrete particle position. This resampling is called this way since the duplication counts are distributed according to a multinomial distribution by definition.

The conditional variance of the Multinomial Resampling is given by:

\[
Var \left[ \frac{1}{n} \sum_{i=1}^{n} f(\hat{\xi}^i) \mid G^n \right] = \frac{1}{n} \left( \sum_{i=1}^{m} \omega_i f^2(\xi^i) - \left[ \sum_{i=1}^{m} \omega_i f(\xi^i) \right]^2 \right) \tag{2.38}
\]

\(^2\)Uniform distributions of probability.
where \( \omega_i \) are the weights associated to each particle, \( f \) is all integrable function applied to particles position and \( G^n \) is the \( \sigma \)-field generated by the generations of particles and the weight until the current time, this one included.

An efficient manner to reduce variance is the Residual Resampling. In this approach, for \( i = 1, \cdots, m \):

\[
N^i = \lfloor n\omega^i \rfloor + \mathcal{N}^i
\]

(2.39)

where \( \mathcal{N}^i \) are distributed according to the multinomial distribution \( \text{Mult}(n-R; \varpi^1, \cdots, \varpi^n) \) with \( R = \sum_{i=1}^{m} \lfloor n\omega^i \rfloor \); \( N^i \) is, by definition: \( \# \{ j, 1 \leq j \leq n : \hat{\xi}_j = \xi^i \} \), and:

\[
\bar{\omega}^i = \frac{n\omega^i - \lfloor n\omega^i \rfloor}{n - R}, \ i = 1, \cdots, m.
\]

(2.40)

Its variance can be computed as:

\[
\frac{1}{n^2} \text{Var} \left[ \sum_{i=1}^{n-R} f(\xi^i) | G^n \right] = \frac{1}{n} \sum_{i=1}^{m} \omega^i f^2(\xi^i) - \frac{1}{n} \sum_{i=1}^{m} \frac{\lfloor n\omega^i \rfloor}{n^2} f^2(\xi^i)
\]

\[
- \frac{n-R}{n^2} \left( \sum_{i=1}^{m} \bar{\omega}^i f(\xi^i) \right)^2
\]

(2.41)

and due to the Jensen’s inequality, it is demonstrated that the conditional variance of the Residual Resampling is smaller than the variance shown in Eq. 2.38. The Jensen’s inequality states, in its finite form, that:

\[
\phi \left( \frac{\sum a_i x_i}{\sum a_j} \right) \leq \frac{\sum a_i \phi(x_i)}{\sum a_j}
\]

(2.42)

so, in this case:

\[
\frac{1}{n} \sum_{i=1}^{n} \left[ \int_{(i-1)/n}^{i/n} f \circ \xi \circ D_{\omega}^{\text{inv}}(u) du \right]^2
\]

\[
\geq \left[ \sum_{i=1}^{m} \int_{(i-1)/n}^{i/n} f \circ \xi \circ D_{\omega}^{\text{inv}}(u) du \right]^2
\]

(2.43)

\[
= \left[ \sum_{i=1}^{m} \omega^i f(\xi^i) \right]^2.
\]
Stratified Resampling is based on ideas used in survey sampling and consists in pre-partitioning the \((0,1]\) interval into \(n\) disjoint sets \((0, 1/n) \cup \cdots \cup \left( \frac{n-1}{n}, 1 \right] \). The \(U^i\)'s are then drawn independently in each of these sub-intervals. Then the inversion method is used as in multinomial resampling. The difference between the duplication count \(N^i\) and its target value \(n\omega^i\) is less than one in absolute value and:

\[
E \left[ \sum_{i=1}^{n} f(\hat{\xi}^i) \mid G^n \right] = E \left[ \sum_{i=1}^{n} f \circ \xi \circ D_{\omega}^{\text{inv}}(U^i) \mid G^n \right] = n \sum_{i=1}^{n} \int_{(i-1)/n}^{i/n} f \circ \xi \circ D_{\omega}^{\text{inv}}(u) du = n \sum_{i=1}^{m} \omega^i f(\xi^i)
\]

for all integrable functions \(f\).

Due to the conditional independence between the uniform distributions \(U^1, \cdots, U^n\) given \(G^n\) for this method:

\[
\text{Var} \left[ \frac{1}{n} \sum_{i=1}^{n} f(\xi^i) \mid G^n \right] = \frac{1}{n} \sum_{i=1}^{m} \omega^i f^2(\xi^i) - \frac{1}{n} \sum_{i=1}^{n} \left[ n \int_{(i-1)/n}^{i/n} f \circ \xi \circ D_{\omega}^{\text{inv}}(u) du \right]
\]

and the Jensen’s inequality demonstrates that the variance of the Stratified Resampling is lower than the variance of Residual Resampling.

The last method is the Systematic Resampling, which is actually explained in Sec. 4.2.3.

They finally work on the topic of large-sample behavior of all those types of resampling. For reasons of space and simplicity they only consider in this point the case of the bootstrap filter (i.e., when the transition kernel \(q\) of the hidden chain is used as proposal) where resampling is performed at each time index. In this case, each iteration of the particle filtering may be decomposed into two successive steps: the prediction and the filtering.

**Prediction:** given the population of unweighted particles at time index \(k\), \(\{\hat{\xi}_k^i\}_{1 \leq i \leq m}\), each trajectory conditionally independently according to \(\xi_{k+1}^i q(\hat{\omega}_k^i, .)\) is extended.

**Filtering:** after computing the weights as:

\[
\omega_{k+1}^i = \frac{g_{k+1}(\xi_{k+1}^i)}{\sum_{j=1}^{m} g_{k+1}(\xi_{k+1}^j)}
\]

(2.46)
the resampling to obtain the new unweighted population of particles $\{\hat{\xi}_i^{k+1}\}_{1 \leq i \leq n}$ is performed. The choice of a particular resampling scheme impacts only on the second step.

They finish stating some assumptions and theorems explained in [6]. To sum up, the authors determine that the most common scheme on being used is the systematic resampling due to its easy implementation. A central limit theorem has been established for both residual sampling and stratified sampling approaches. As they tell, the situation is more involved in the latter case due to the fact that the new resampled particles, although still conditionally independent, have a distribution which depend on the order in which the particles are initially labelled.

### 2.2 Work Related to Other TDOA Schemes

On the other hand, the background of this Master Thesis is the localization of a sound source (in our case, a speaker), so it is also appropriate to mention some work related with this topic. In [7], the authors focus their efforts on creating a system called Whistle, from which they want to obtain accurate measurements and try to release the problem of synchronization between the receivers by changing the scheme of the TDOA, i.e. changing the geometric configuration of the group of receivers.

The authors describe how accurate are the measurements made in Whistle. In Fig. 2.5 is it shown a typical time sequence of two nodes. In Whistle, the clocks are maintained independent.

Although the receivers get the signal from S at a certain time, due to software and hardware delays, the detection of the emitter is done at a same time later. The same happens in the other node. Such latencies are unpredictable, so it is not possible to compute the estimate of TDOA as the difference between the times of arrival of each node. They introduce an extra sound system in order not to be obligated to synchronize the system. This system is placed in one of the receivers and, some time after have received the first signal, emits another signal according to its own clock. Now this node is called base node.

They define a time $t_{ab}$ as a time span lasting from the first signal is emitting and the second signal is received at node $b$. Let the times $t_a$ and $t_b$ be the propagation times between from the signal emitter to the nodes $a$ and $b$. Due to the triangle inequality they have that: $t_b \leq t_a + t_c$, where $t_c$ is the propagation time of the second emission from a (a is considered as the base node, and which actually emits the second signal) to b. Since the base node finds a delay between receiving the first emitter and sending
Chapter 2. Related Work

Figure 2.5: Time lines of two nodes A and B in the Whistle system [7].

the second one: \( t_{ab} > t_a + t_c \). So, \( t_{ab} > t_b \), which means that any non-base node firstly receives the first emitted signal and then the second one.

To compute the TDOA from a to b (\( T_{AB} \)), they only have to apply that:

\[
T_{AB} = t_{B1} - t_{A1} = \cdots = k_1 - T_{B2S} + T_{A2S} - k_2
\]  

(2.47)

where the two components of the same expression are extracted from Fig. 2.5, \( k_1 = d_{AB} / v \), \( k_2 = d_{AA} / v \), and the other two time components are the difference between the receiving of the two signals at each node (\( T_{D2S} \)).

Since the use of timestamps are not promising. Instead of doing this, they take into account that every \( 1/f \) (being \( f \) the sampling rate) seconds a node senses and translates signals into real or complex number by its A/D converter. So, it is possible to assure that \( T_{X2S} = j \frac{i-2}{f} \), and the higher is the sampling rate, the higher is the accuracy.

The most important factors that affect the accuracy of the system are the Signal to Noise Ratio (SNR), the Multi-path effects and the equation solving. The errors caused can contribute to a wrong TDOA computation, so they use the method of majority decision to get a more accurate result.

The framework design is based as follows: in order to detect in 3 dimensions it is necessary to have 5 nodes (4 of them non-base nodes and 1 base node).

In the first step, all nodes receive the first sound signal as explained before. In the second step, the base node emits the second signal after have received the first one and the non-base nodes stay receiving the second signal. Every node maintains a recording \( R_i \) which includes the both received signals.

They send the second signal with the same waveform than the first one, so that they have a good auto-correlation property, which implies two easily distinguishable peaks that represent the time of arrival of the two emitted signals.
In practice, the highest peak can not always represent the arrival of signal because of the multi-path effects and other uncertainties. They choose the earliest sharp peak in a shadow window of $\omega_0$ point right before the maximal peak instead. They also use two parameters: height and average slope. The last one is calculated as:

$$P = \frac{Y_{\text{peak}} - Y_{\text{valley}}}{X_{\text{peak}} - X_{\text{valley}}} \tag{2.48}$$

where $X_{\text{valley}}$ is the nearest valley point before the peak.

With that, they correlate now $R_i$ with the reference signal using this method.

For each node, the $T_{D2S}$ value depends only on its own clock, provided that the clock of each node is accurate in term longer that the period of the sound occurrence.

Finally, they test some geometrical configurations to determine how reliable is this system. To sum up, they summarize that Whistle system achieves low cost, rapid deployment and widespread use simultaneously.

Also in [8], the authors design a particular geometry in order to achieve their goal of localizing an object. In this case, the studies are centred on a 2-D location. In Fig. 2.6 and Fig. 2.7 the geometries used in such publications works are shown.

In [8], they improve the basic original TDOA algorithm in order to improve their performance and results.

As it is also said in [7], it is necessary to dispose of at least 5 sensors to determine the position of a mobile in a 3-D space. In case not to dispose of such number of receivers, to handle this case it is possible to use an iterative multilateral positioning method. The key point is that the localized sensor can be regarded as a new anchor to locate the next sensor.

Here, the authors assess only the situation where the sensors are upgraded if they have high precision. If not, the error begins to propagate since the point where the lower precision sensor is used until the last one.

The location algorithm is based on the following ideas: firstly, they search for, at least, four anchors around the current undetermined sensor node and use their distances with this sensor node and the TDOA method to locate this unknown sensor node.

Secondly, they use the geometric distance measurement error to measure the location accuracy. The geometric distance measurement error (GDME) is defined as:
Figure 2.6: Geometry established in [8].

\[
GDME = \max \left\{ \frac{|d_{ij} - \hat{d}_{ij}|}{d_{ij}}, j = 1 : N \right\}
\]

(2.49)

where \(d_{ij}\) is the measurement distance between the i-th unknown sensor and the j-th anchor around this sensor and \(\hat{d}_{ij}\) is their distance computed according to the estimated coordinate of this sensor and the coordinate of the j-th anchor by Euclidean distance formula. When GDME is smaller than the prescribed threshold, this current determined sensor node is upgraded to the anchor node, that is, the lower precision localization sensor is not involved in the later sensor location. The anchor set is updated as well as the unknown sensor set and this step is repeated until every unknown sensor is scanned for localization one time.

Finally, for those lower precision sensors which are not upgraded to the anchor nodes, they use the coordinate information of determined sensors and anchors and their distances to recalculate the coordinates of those lower precision sensors.

As a final conclusion, they summarize that the improved TDOA, according to their final results, has a probability of localize the sensors equal to the unity, whereas with the original TDOA algorithm, as the noise factor is increased, the probability of location is reduced considerably.

Continuing with location topic, although there is a large comparison component of work, in [9], authors discuss how good is the Particle Filter in order to the values of several state variables in the context of an arbitrary user. They analyse the application of the Particle Filter when the sensors are quite varied: GPS, a Palm Navigator electronic compass, RFID (Radio Frequency Identification), wireless LAN access points/Bluetooth and floor plan specially for indoor scenarios. Some results achieved when using a GPS system and different variables are plotted in Fig. 2.8.
Besides, they state the variables which take place in the movement model: position, direction and velocity. They assume Gaussian distribution for position and velocity. However, they add $N_s$ additional movement influencing state variables that affect these physical parameters and which are categorized in three groups:

1) the first group is directly dependent on the current position and direction of the pedestrian.

2) the second one are static values since they do not depend on the previous step, and they are assumed to be known by the system.

3) the last group is formed for those variables which can not be measured directly or stored in a database because they vary according to the human behavior.

For their model, the authors consider 11 additional variables. Thus, these 11 state variables result in 11 means and standard deviations for speed and direction.
Right after, they combine all these variables to a single new mean and standard deviation for speed and direction. They apply simple linear weighting such as (in the case of the speed mean):

\[ \mu_v = \frac{\sum_{i=1}^{N_s} \mu_{v,i} \cdot \gamma_i}{\sum_{i=1}^{N_s} \gamma_i} \] (2.50)

where \( \gamma_i \) is the weight corresponding to the i-th state variable. This mean is limited by the maximum deceleration and the maximum acceleration that a normal human being can achieve.

From the computed speed, they compute now the space travelled by only multiplying the speed times a period of time \( \Delta t \). Using trigonometrical ratios, they compute the concrete pedestrian position as:

\[
x(k + 1) = x(k) + l_k \cdot \cos[\alpha(k + 1)]
\]

\[
y(k + 1) = y(k) + l_k \cdot \sin[\alpha(k + 1)]
\]

where \( l_k \) is the space travelled. Finally, this position will be the input for the next time iteration.

To sum up, it is possible to assure that the authors have worked with a huge realistic point of view, which can make the results also more realistic.

Similarly to [7, 8, 10] (which are also cited in Section 3.3.2), in [11], the author use an array of microphones to localize a robot and also uses a Particle Filter and an extra method: von Mises distribution measurement model.

### 2.3 Other Algorithms to Locate

Moving on to other issues, some other methods used in the world of localization include the Unscented Kalman Filter (UKF) and the Extended Kalman Filter (EKF) [12].

On one hand, the most significant similarity between the Particle Filter and Kalman filters is the use of three steps within its whole position estimation process.

On the other hand, one important difference between the Particle Filter and any Kalman filter is the need of the last one of knowing the initial position of the tracked object or, at
least, having an initialization algorithm which can be used to know that initial position. In the Unscented Kalman Filter, some points are collected from statistical distributions. It also takes sake from its knowledge of the both process noise and measurement noise. These noises are established as those components which make the correct position estimation to vary, in prediction step and correction step.

Going back again to Particle Filter, there is some work driven more to the study of multiple speakers tracking: Particle Filter in [13–15], other algorithms\(^3\) in [16], and a comparison between Kalman Filter and Particle Filter in [17].

\(^3\)Split and merged measurements.
Chapter 3

Mathematical Tools

In this section we talk about the basics of the different mathematical tools we have taken to develop this Master Thesis. Some are easily understandable and others require a more solid mathematical base in order to be understood. For sections 3.1, 3.2: [18–21]

3.1 DFT

The main tool in which the computation of the speaker position is focused is the Discrete Fourier Transform. In analogical communications, the use of the Fourier Transform is much more common but since some years ago, they are becoming more and more obsolete owing to the rise of digital communications. In this type of communications, it is necessary to establish a different system for the frequency domain, which has caused the appearance of the discrete frequencies and whose main base is the already mentioned DFT:

\[ X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi kn/N} \] (3.1)

There is another important characteristic that has to be taken into account: the repetition of the spectrum. It is caused by the periodicity in taking samples from an analogue signal. Since a change of domain is produced, it is necessary to define another parameter: the sampling frequency. It is defined as the inverse of the period:

\[ F_s = \frac{1}{T_s} \] (3.2)
where $F_s$ is the sampling frequency and $T_s$ is the sampling period. The sampling frequency determines how long is the original spectrum, without repetitions, within the whole spectrum. As larger is $F_s$, the larger is the discrete spectrum formed.

### 3.2 FFT

Due to computational reduction needs, a faster tool was invented. It was called Fast Fourier Transform (FFT). It uses different algorithms in order to reduce the computational effort required.

The most common of them is the radix-2 Cooley-Tukey algorithm. It needs only $\frac{N}{2} \log_2 N$ multiplications (ignoring multiplications by 1 and others with the same result) and $N \log_2 N$ complex additions.

It re-expresses the DFT of an arbitrary composite size $N = N_1 N_2$ in terms of smaller DFTs of sizes $N_1$ and $N_2$.

Radix-2 DIT divides a DFT of size $N$ into two interleaved of size $N/2$ with each recursive stage. First, it computes the even indexed inputs and the odd-indexed inputs and then combines the two results. So:

$$X_k = \sum_{m=0}^{N/2-1} x_{2m} e^{-j \frac{2\pi}{N} mk} + \sum_{m=0}^{N/2-1} x_{2m+1} e^{-j \frac{2\pi}{N} (2m+1)k}$$  \hspace{1cm} (3.3)

where the first term is the DFT computation of even-indexed inputs and the second term is the DFT computation of odd-indexed inputs. Coping with the equation it is obtained the following expression:

$$X_k = \sum_{m=0}^{N/2-1} x_{2m} e^{-j \frac{2\pi}{N/2} mk} + e^{-j \frac{2\pi k}{N}} \sum_{m=0}^{N/2-1} x_{2m+1} e^{-j \frac{2\pi}{N/2} mk}$$

$$= E_k + e^{-j \frac{2\pi k}{N}} O_k$$  \hspace{1cm} (3.4)

where the exponential term is known as the twiddle factor and both $E_k$ and $O_k$ have a length of $N/2$, so only $N/2$ outputs are required. Outputs from $N/2$ up to $N$ are identical to outputs from 0 to $N/2 - 1$, which means that $E_{k+N/2} = E_k$ and $O_{k+N/2} = O_k$, due to the periodicity of DFT. Due to that fact, the final expression for FFT can be rewritten into two terms:
\[ X_k = \begin{cases} 
E_k + e^{-j\frac{2\pi}{N}k}O_k & \text{if } 0 \leq k < N/2 \\
E_{k-N/2} - e^{-j\frac{2\pi}{N}(k-N/2)}O_{k-N/2} & \text{if } N/2 \leq k < N 
\end{cases} \tag{3.5} \]

The above process is called radix-2 due to the fact that either \( N_1 \) or \( N_2 \) is 2 and the other one is \( N/2 \).

Nevertheless, in a general case \( N_1 \) and \( N_2 \) might have any other value different from 2 and \( N/2 \) respectively. If \( N_1 \) is the smallest one, the algorithm is called DIT, whereas if \( N_2 \) is the smallest one the algorithm is called DIF. In these cases, a mixed-radix implementation is done.

Firstly, it is done a general Cooley-Tukey factorization:

\[ k = N_2 k_1 + k_2 \tag{3.6} \]
\[ n = N_1 n_2 + n_1 \tag{3.7} \]

where \( k_a \) and \( n_a \) run from 0 to \( N_a - 1 \) for \( a = 1, 2 \).

So:

\[
X_{N_2 k_1 + k_2} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x_{N_1 n_2 + n_1} e^{-j\frac{2\pi}{N_1 N_2} (N_2 k_1 + k_2)(N_1 n_2 + n_1)} \\
= \sum_{n_1=0}^{N_1-1} e^{-j\frac{2\pi}{N_1 N_2} n_1 (N_2 k_1 + k_2)} (\sum_{n_2=0}^{N_2-1} x_{N_1 n_2 + n_1} e^{-j\frac{2\pi}{N_2} n_2 (N_2 k_1 + k_2)}) \\
= \sum_{n_1=0}^{N_1-1} e^{-j\frac{2\pi}{N_1} n_1 k_1} e^{-j\frac{2\pi}{N_1 N_2} n_1 k_2} (\sum_{n_2=0}^{N_2-1} x_{N_1 n_2 + n_1} e^{-j2\pi n_2 e^{-j\frac{2\pi}{N_2} n_2 k_2}}) \tag{3.8} 
\]

The final expression which remains is the following one:

\[
X_k = \sum_{n_1=0}^{N_1-1} e^{-j\frac{2\pi}{N_1 N_2} n_1 k_2} (\sum_{n_2=0}^{N_2-1} x_{N_1 n_2 + n_1} e^{-j\frac{2\pi}{N_2} n_2 k_2}) e^{-j\frac{2\pi}{N_1} n_1 k_1} \tag{3.9} 
\]

in which the first exponential term is again the twiddle factor.
Chapter 3. Mathematical Tools

3.3 Phase Correlation

The phase correlation\footnote{22} is a mathematical tool used in order to compare two similar inputs signals which simply differ to each other in a time delay. Usually, this tool is used in image processing but here we apply the same basics to a different environment, more related with speech processing. The basic concept is the use of the Fast Fourier Transform (FFT) instead of the Discrete Fourier Transform (DFT). The first one is applied in order to reduce the computational cost of a huge number of operations. The DFT takes $\Theta(N^2)$ operations while the FFT takes only $\Theta(N \log N)$ operations.

The first step to be done is the FFT computation for both input signals. Thus, we obtain the spectral representation of the signals. Afterwards, it is necessary to multiply both spectral signals and normalize them. Let $Y$ and $Z$ be the FFTs of the two input signals, so that:

$$ R = \frac{Y \circ Z^*}{|Y \ast Z^*|} $$

where $\circ$ represents the Hadamard product and $\ast$ represents the scalar product between the two signals. It is necessary to take into account that these signals are represented by column vectors and the operator $\ast$ both transposes and conjugates the signal. Finally, the inverse fast Fourier Transform have to be done to achieve the time representation of the resulting spectral signal.

3.3.1 Hadamard Product

The Hadamard Product\footnote{23} is normally used to multiply matrices in a 2-D dimension product. However, in our case, the product required have to be computed in 1-D. We multiply each array component times its corresponding value in the other array and we compute the summation of all resulting multiplications. In consequence, as we are coping with arrays of only one dimension, the result have to be a single number, i.e, an scalar.

However, in general, it differs from scalar product in its dimension, so that the result of a Hadamard Product must be an array of the same dimension than its inputs whereas the scalar product’s result must be, as its name indicates, an scalar. So, they only coincide in the case of a 1-D array.
3.3.2 TDOA

To explain the basics for TDOA we will talk about a generic case (i.e. N receivers). The explanation is extracted from [10]. The TDOA is defined as the peak in the phase correlation\(^1\) between the receivers \(i\) and \(j\):

\[
\Delta T_{ij} = \arg \max_{\tau} R_{ij}(\tau)
\]  \hspace{1cm} (3.11)

Using an array of N receivers, it is possible to compute \(N(N-1)/2\) different phase correlations of which only \(N-1\) of them do not depend on the others: it is only necessary to find out the values of \(\Delta T_{1j}\), the others are easily findable by:

\[
\Delta T_{ij} = \Delta T_{1j} - \Delta T_{1i}
\]  \hspace{1cm} (3.12)

Going deeper, it is also possible to compute the concrete position of the speaker if we know the distance between the microphones\(^2\).

Observing Fig. 3.1, we can observe that \(c\) represents the sound velocity. From the same figure we can extract that:

\[
\cos \phi = \frac{\vec{u} \times \vec{x}_{ij}}{\|\vec{u}\| \|\vec{x}_{ij}\|}
\]  \hspace{1cm} (3.13)

as the vector \(\vec{u}\) is a unit vector that only points to sound source direction, the denominator remains as \(\vec{x}_{ij}\). As:

---

\(^1\)In the original paper, the authors deal with the cross-correlation whereas we cope with phase correlation instead.

\(^2\)In our case we do not know the distance between our two microphones, and actually we do not need it because we are interested in the comparison between different algorithms using our observations (the phase correlation).
\[ \sin \theta = \frac{c \Delta T_{ij}}{\| \vec{x}_{ij} \|} \]  

(3.14)

and \( \sin \theta = \cos \phi \), by isolating, it can be reached that:

\[ \vec{u} \times \vec{x}_{ij} = c \Delta T_{ij} \]  

(3.15)

In the general case, a 3-D position for the sound source is searched, so the vectors \( \vec{u} \) and \( \vec{x}_{ij} \) are necessarily 3-D vectors. Provided that assumption and using an equation system, it is possible to find out the coordinates of the vector which points at sound source position. Depending on the number of microphones it is possible to encounter the position in 1-D (2 microphones), 2-D (4 microphones) or 3-D (8 microphones). In our case we only can locate the speaker in 1-D since we have only 2 microphones [10].

In addition, as we ignore the distance between the receivers, it is only possible to know if the speaker is located closer to one receiver than to the other one by using the TDOA result, but it is not possible to know how far he/she is. Anyway, we are coping with a situation in which the observations were taken from a recording in an indoor environment with a little separation of microphones.

### 3.4 Particle Filter Basics

The particle filter is based on statistical basics, supported by the knowledge of the environment and its belief. As its name says, this alternative non-parametric Bayesian filter is composed by a set of particles that move with the aim of tracking the desired object or person. The explanation of the particle filter is totally extracted from [24], but also supported by explanations from [25] and from [26].

#### 3.4.1 Particle

Each particle represents the concrete belief of a state combined with the associated weight to that particle, which indicates the certainty that this belief is true. The particle is moving according to the knowledge it has about its environment. To represent each particle we define its state. This state is composed for both the concrete particle position and its control. This control, as its name indicates, governs the motion of that certain particle. These values change in each particle filter iteration.
3.4.2 Extensive Basics

Basically, the main idea of the particle filter is to represent the posterior probability (i.e. it belief) using a set of random state samples drawn from this posterior. The belief is commonly represented as:

\[
\text{bel}(x_t) = p(x_t \mid z_{1:t}, u_{1:t})
\]  

(3.16)

in which \(x_t\) represents the state of the particle at time \(t\), whereas \(z_{1:t}\) represents all the observations until time \(t\) and \(u_{1:t}\) all past controls. The posterior for each iteration of the algorithm becomes the prior for the next iteration.

Firstly, a set of particles is randomly distributed over the space state. The number of particles is variable and actually it is a design parameter whose best value have to be encountered. Of course, as it is not fixed, the performance result changes as well as its value. After that, the steps are always put one after the previous one: firstly the motion step followed by the weight step and finishing by the resampling.

Each particle has to be moved from the previous state according its motion model \(^3\): 
\[p(x_t \mid u_t, x_{t-1})\], where \(u_t\) is the control\(^4\) and \(x_{t-1}\) is the state of the particle in the previous time instant.

Then, the importance factor has to be computed. The importance factor is defined as:

\[w_t^{[m]} \propto p(z_t \mid x_t^{[m]})\]  

(3.17)

It takes the value of the observation in whose place the particle is located. As we can see in the formula, each particle has its own weight at a certain time instant. Moreover, a normalization factor is used to make the summation of all weights to be one.

But coming back to the beginning, by using induction and the Bayes rule, the correctness of the particle filter is shown\(^5\):

\[
p(x_t \mid z_{1:t}, u_{1:t}) = \frac{p(z_t \mid x_t, z_{1:t-1}, u_{1:t})p(x_t \mid z_{1:t-1}, u_{1:t})}{p(z_t \mid z_{1:t-1}, u_{1:t})} = \eta \ast p(z_t \mid x_t, z_{1:t-1}, u_{1:t})p(x_t \mid z_{1:t-1}, u_{1:t})
\]  

(3.18)

\(^3\)Our motion model is explained in the chapter Algorithms.

\(^4\)In our case there is no control variable and the state stores both the speed and the position of each particle.

\(^5\)Strictly, this is the demonstration for a Bayes filter but the particle filter is, actually, a Bayesian filter.
where $\eta$ is a constant corresponding to the fraction denominator. This is caused by a particular assumption: our state is complete. A state completeness implies that the past observations or states will not affect future observations or states. So that, $z_t$ does not depend neither on $z_{1:t-1}$ nor on $u_{1:t}$, which implies that the denominator does not vary. Besides, it also affects the first term on the numerator such as follows:

$$p(z_t | x_t, z_{1:t-1}, u_{1:t}) = p(z_t | x_t)$$ (3.19)

and the current observation only depends on the current state. Finally, the belief is:

$$bel(x_t) = \eta p(z_t | x_t)p(x_t | z_{1:t-1}, u_{1:t})$$ (3.20)

Nevertheless, all this is only a first look into the basics of the particle filter. Strictly talking, it is necessary to compute belief of the particles over the entire period of time, which means to compute the following:

$$bel(x_{0:t}) = p(x_{0:t} | u_{1:t}, z_{1:t})$$ (3.21)

However, this belief is computed completely equal than the previous one, so there is no importance change on the final result:

$$= \eta p(z_t | x_t)p(x_t | z_{1:t-1}, u_{1:t})$$
$$= \eta p(z_t | x_t)p(x_t | x_{0:t-1}, z_{1:t-1}, u_{1:t-1})p(x_{0:t-1} | z_{1:t-1}, u_{1:t})$$ (3.22)
$$= \eta p(z_t | x_t)p(x_{0:t-1} | z_{1:t-1}, u_{1:t-1})p(x_t | x_{t-1}, u_t)$$

where all the equation except for the last term is $bel(x_{0:t-1})$, which demonstrates that the belief of the current state depends on the belief of all the past states and on the motion model.

Besides, there are two more concepts that have to be taken into account: the proposal distribution and the target distribution. The proposal distribution is that one from which we are given the samples and the target distribution is that one which we consider we have to reach.

As the weight is defined as the target distribution divided by the proposal distribution, and target distribution is:
Chapter 3. Mathematical Tools

\[ \eta p(z_t \mid x_t)p(x_t \mid x_{t-1}, u_t)p(x_{0:t-1} \mid z_{1:t-1}, u_{1:t-1}) \]  

(3.23)

and the proposal distribution is the following one:

\[ p(x_t \mid x_{t-1}, u_t)p(x_{0:t-1} \mid z_{0:t-1}, u_{0:t-1}) \]  

(3.24)

the final result for the weight is: \( \eta p(z_t \mid x_t) \), where the constant \( \eta \) is used to normalize all the weights to sum the unity. So it is demonstrated that the weight is proportional to the importance sampling, as we show in Eq. 3.17.

The last step in Particle Filter is the resampling step. It consists on changing a set of particles whose some particle have lost their importance for a new one in which all of them have the same weight. Going deeper, we explain that resampling is an optional step and why. As time goes on, more and more particles are erased owing to the random nature of the resampling step, without creation of any new particles. With probability one, \( M \) identical copies of a single particle will survive, the diversity will disappear due to the repetitive resampling.

### 3.5 Mean and median

Some other mathematical tools used in this Master Thesis are the mean and the median of a set of values.

There are some different types of means. The first one is the weighted mean. It is based on taking a set of values, multiplying each one of those values times its associated weight within the whole set of values and finally making the summation of all of them:

\[ \mu_w = \frac{\sum_{i=1}^{n} x_i w_i}{\sum_{i=1}^{n} w_i} \]  

(3.25)

where \( \mu_w \) is the weighted mean, \( x_i \) and \( w_i \) are the \( i \)-th value and its associated weight respectively and \( n \) the total number of values.

A particular case of this type of mean is arithmetic mean, whose all values have the same weight, so instead of multiplying each value times its weight, first the summation of all values is computed and then the division of that result over the number of values is made:
\[ \mu_a = \frac{x_1 + x_2 + \ldots + x_n}{n} \tag{3.26} \]

where \( \mu_a \) is the arithmetic mean and the rest of symbols mean the same than in weighted mean.

There are many other types of means that are not going to be considered owing to their few importance in this Master Thesis.

On the other hand, it is also common the use of the median instead of the mean. The median is used in those cases in which some values among the whole set are far away from the most of them; these values can increase or decrease significantly the value of the mean if those values would not be in the set. The median, instead of using all values, arranges the set into increasing order and takes only the middle-placed value. If most of values have the similar value, it is presumable that the value of the median is going to be around this large amount of similar values.

### 3.6 Percentile

In some cases, due to random variables, it is possible to obtain some failures or unexpected values in some measurements. In this thesis, among the possible ways to compute percentile we have chosen the method of the nearest rank, defined as:

\[ n = \frac{P}{100} N + \frac{1}{2} \tag{3.27} \]

where \( P \) represents the P-th percentile computed, \( N \) the number of ordered values and \( n \) is the nearest rank. The result is rounded to the nearest integer and the value of the percentile will be the value that corresponds to that rank among the ordered values. This tool is used in Table 5.4 [27].
Chapter 4

Algorithms

4.1 TDOA Algorithm

The first algorithm that we implemented was the TDOA algorithm. Within our set of algorithms this is the easiest one. It only implies the computation of the phase correlation between two input signals coming from the two receivers. After this step, we only have to look for the maximum value of the phase correlation. This value will be the Time Difference Of Arrivals.

However, the method is not perfect and it is affected by noise such as the TDOA computed may not be the correct one. In addition, this is not the only problem we encountered while designing the algorithm. Another problem we encountered was the following one: we have two time signals which we had to separate in really small time windows in order to be capable of computing the phase correlation of the same time duration. The smaller are these windows, the larger number of them the total signal will be composed of. Due to the time delay, a time instant can correspond to different time windows within the two input signals. It causes edge effects, and so, the performance of the result gets worse.

4.1.1 Fixing Edge Effects

The best option to reduce the edge effects is based on taking different number of samples for each time window. Changing this number into a larger one reduces the edge effects owing to the smaller number of time windows that we have. Anyway, there is a limit in the number of samples that we can take. If this number is too large, the figure performed becomes totally shapeless. It happens owing to the direct relationship between the
Chapter 4. *Algorithms*

4.1.2 Fixing Noise Problems

The worst noise problems occur when there is no speaker talking, so the maximum value of the phase correlation can come from any space spot, which makes impossible to determine where the speaker is in that concrete moment. After some moments where the trajectory line is not clear, when the speaker speaks again, the line suddenly can appear in any other place.

The first option to fix noise problems is the use of a smoother that weakens strong values in those frequency components where the speech does not reach. There are many windows that can be used. The first one, which is commonly used in Telecommunication Engineering to beat some edge effects, is the Hamming window:

\[
\alpha - \beta \cos \left( \frac{2\pi n}{N - 1} \right) \tag{4.1}
\]

where \( n \) represents a particular sample within the window and \( N \) is the total number of samples. Thanks to the use of these window we achieve a better performance of the TDOA, given that some huge peaks are significantly reduced; and \( \beta \) is always \( 1 - \alpha \).

The values for \( \alpha \) and \( \beta \) are also design parameters and they have to be changed in order to find out the best values. In Figure 4.1 we represent the shape of a Hamming window.

Another possible window is the Gaussian window. Its formula also contains two design parameter, from which we obtain better or worse results:
The two design parameters are $\mu$, the mean, and $\sigma$, the deviation. Depending on when we use the window\(^1\), the effect in the signal will be completely different. The use of a Gaussian window in time domain smooths the components on the edges of the time window whereas in frequency domain it acts as a high pass filter due to the repetition of the spectrum. We detail the explanation in Sec. 5.1.

Anyway, the Gaussian window has to be centered in the middle of the time window, which means that $\mu$ can be expressed as $(N - 1)/2$, where $N$ is the width of the time window. Here, again, $n$ represents a particular sample within the whole window.

The next window used is the Blackman-Harris window, which have 4 design parameters:

\[
a_0 - a_1\cos\left(\frac{2\pi n}{N - 1}\right) + a_2\cos\left(\frac{4\pi n}{N - 1}\right) - a_3\cos\left(\frac{6\pi n}{N - 1}\right)
\]

(4.3)

where either $n$ and $N$ represent the same than in the previous windows.

Finally, the last window used is the Welch window, which actually works similarly to Gaussian window. Its formula is:

\[
1 - \left(\frac{n - \frac{N - 1}{2}}{\frac{N + 1}{2}}\right)^2
\]

(4.4)

with the same meanings for its parameters.

There are many other windows with similar characteristics but these four used are enough representative in order to evaluate algorithm performance.

### 4.1.3 Low Pass Filter

The last way to improve TDOA algorithm directly is the use of a low pass filter after the TDOA algorithm implementation. As we know, the TDOA algorithm is done with time windows, where each one of them is associated with a particular time instant. A low pass filter takes into account a huge part of the previous time instant and a little part of the current instant:

\[
x(i) = \alpha x(i) + (1 - \alpha)x(i - 1)
\]

(4.5)

\(^1\)It is applicable for all windows.
where $\alpha$ is a design parameter which initially is set to 0.01. In Sec. 5.1 we change this parameter in order to find out its best value. By using this method, as we are giving a large importance to the previous instant, the current one is driven to follow the same path that the previous one, so huge changes from the previous time to the current one are slightly avoided. Although the shape representation has improved, a perfect performance is not achieved yet.

4.2 Particle Filter Algorithm: Functional Principle

In this section we explain how we have focused in the thesis but following the same basics extracted from [24]. The particle filter is formed by 3 steps: the motion step, the weight step and the resampling step, although the last one may be not performed; actually, the resampling step can be taken as an improving step, in order to achieve a better performance of the result. But first of all, the initial set of particles have to be sampled into the space state\(^2\). We set the initial particle distribution as a Uniform Distribution, in which the limits of the distribution are the same than for the space state.

4.2.1 Motion Step

In the first step of the algorithm, the movement of the particles is produced. Firstly, it is important to choose an appropriate model for the particles motion. Owing to the lack of awareness of the particles position, the concrete result (where the speaker is) can not be assured. So that, is important to choose a random model for this motion. The easiest model which represents the situation sufficiently well is the following one:

\[
x_k^{[m]} = x_{k-1}^{[m]} + v_k^{[m]} \tag{4.6}
\]

\[
v_k^{[m]} = v_{k-1}^{[m]} + \mathcal{N}(0, \sigma^2) \tag{4.7}
\]

where we can see that the current position depends on the previous position and the velocity of the particle in the current time instant. The current velocity also depends on the previous speed and on a random component which describes the particle ignorance of the position.

\(^2\)We define the space state, in our case, as the 1-D space in which we allocate the particles. Concretely, we set a state space of 1024 samples, i.e., a finite space state.
In order to make it more manageable, we define a matrix which contains both the information of the current position of the particle and its movement in the particular time instant.

\[
x = \begin{bmatrix}
  x^{[1]} & v^{[1]} \\
  \vdots & \vdots \\
  x^{[m]} & v^{[m]} \\
  \vdots & \vdots \\
  x^{[M]} & v^{[M]}
\end{bmatrix}
\quad (4.8)
\]

where \( M \) is the number of samples used. At each particular instant, the matrix is reloaded and overwritten, so we only manage a single matrix with the current values, so we have not to deal with previous values.

### 4.2.2 Weight Step

In the second step each particle receives a certain weight associated with the observation that we have already taken from our receivers (called importance sampling). This importance sampling is computed by:

\[
I_t^{[m]} = p(z_t | x_t^{[m]})
\quad (4.9)
\]

Afterwards, however, this weight have to be related with the previous weight of the same particle. It means that we have to multiply the current importance sampling times the previous weight:

\[
w_t^{[m]} = I_t^{[m]} \cdot w_{t-1}^{[m]}
\quad (4.10)
\]

Finally, a normalization using all weights have to be done in order to make the data to converge:

\[
w_t^{[m]} = \frac{w_t^{[m]}}{\sum_m w_t^{[m]}}
\quad (4.11)
\]

If this normalization is not done, after some iterations, the weight values are likely to diverge either to infinity or to zero, depending on the values of the phase correlation.

---

\(^3\)All mathematical details are explained in the Section 3.4.2, including formulas and demonstrations.
Whether the phase correlation has numbers larger than one, the weights tend to infinity whereas if these numbers are smaller than the unit, the weight values tend to zero.

### 4.2.3 Resampling Step

The last step, as we have mentioned before, is optional. In fact, it is only done if it is necessary. After some iterations of the algorithm, the weights of the particles are degenerated, and some particles receive a huge weight whilst others receive a very small weight. It implies a difficulty for particles with small weight to have the chance of receiving a large weight in next iterations. For this reason, we need to recover the importance of the particles.

To start, we define the following parameter: the effective number of particles. This parameter fixes the limit from which the resampling must be done. It is defined as:

\[
N_{\text{eff}} = \frac{1}{\sum_{k=1}^{P}(w_k^L)^2} \tag{4.12}
\]

where \(w_k^L\) is the associated weight to particle \(L\) in the time instant \(k\) and \(P\) is the total number of particles.

The resampling step is based on substituting a set of particles for a new one and it works as follows\(^4\): all the weights associated to particles are put in a row (or a wheel, it depends on designer imagination) and we depict the new group of particles over the row. The distance between particles is always the same, which makes the method to be easily implemented, as well as the resampling is systematic and it has a linear time complexity.

Right after, these new particles which have been depicted in a concrete weight now are put in their corresponding place into the space state. But what does ”corresponding place” means? The weight in which those particles were depicted was previously associated to an old particle corresponding to the previous set. So that, all these new particles have to be performed in the same place where that original particle was located before the resampling step. Finally, all the particles are given the same weight due to there is no particle more important than the others.

\(^4\)There is more than one possibility of resampling, but the used method is the Stochastic Universal Sampling, due to the low variance that it offers.
4.2.4 Final sampling

Finally, at placing the particles again, it has to be considered the no infinitude of the space state. Due to that fact, more than one single particle can be placed in the same spot. In order to consider the density of the particles placed in the same spot, we simply make the summation of all weights of these particles.

4.3 Heuristic Algorithm

Heuristic algorithms are those ones which include a set of methodological rules that say us how to proceed and what problems we have to avoid when generating solutions or hypothesis. These rules are not based on probabilistic, but in human faculty of seeing what is occurring and generating ways of avoiding some troubles or drawbacks. Besides, once we have dealt with some problems, we also have the capability of not to stumble over the same stone more than twice (proof and error).

Our heuristic algorithm is based on the following rule: a standard person has a speed limit which he/she can reach. So that, it is impossible for he/she to move a certain amount of space. By deduction, as we are coping with TDOAs, it is also impossible that the same person causes a huge change of TDOA, because it is related with the speed of the person.

The computation of the TDOA depends on the phase correlation and its maximum in each time. So, the implementation of this algorithm is related with the different maximum points of the phase correlation. We set a threshold in TDOA representation that is associated with the maximum reachable speed of a person and, if the maximum point of the correlation is located beyond this threshold, we assume that this maximum is not the real one and it is affected by noise. We select the following the maximum (in terms of strength) and we check if this maximum is located outside or inside the threshold. We keep trying until we find a location of the maximum which is found inside this threshold. We make the same procedure over the entire TDOA representation.

This threshold is related with the maximum reachable speed of a standard person. We will assume that, in normal and favourable conditions, a speaker is able to keep talking until he/she reaches a velocity of 10 km/h, which is, roughly, in international system, 3 m/s. We assume that beyond this value the person is not able to talk while moving. So, we have to translate this information in terms of TDOA: first, we have to take into account the amount of time contained in every iteration (each iteration contains an amount of time which actually is the separation between iterations or time instants). These iterations are separated by:
where $S$ represents the amount of time contained in a single iteration, $N_s$ represents the number of samples per time window and $F_s$ represents the sampling frequency. As we assume a maximum speed of 3 m/s, there is a concrete distance reachable within the previous time computed. We contemplate the worst case: it occurs when the speaker is directly going from one microphone to the second one in a direct line: in this case, the maximum velocity has to be taken into account due to that, in this situation, the magnitude of the speed (observed as a vector) is exactly equal to component associated to the direction that joins both microphones. So:

$$R_d = S \cdot |V_{\text{max}}| = 0.023 \, \text{s} \cdot 3 \, \text{m/s} = 0.069 \, \text{m}$$

(4.14)

In the previous equation $R_d$ represents the reachable distance in a single iteration.

We have to translate the previous amount of space into a reachable change of TDOA and provided that the velocity of sound is approximately 343 m/s (in the air, at $20^\circ$, assumption made because our observations were taken in a closed room), the maximum difference of TDOA between two iterations is:

$$\Delta TDOA_{\text{max}} = \frac{0.069 \, \text{m}}{343 \, \text{m/s} \cdot 0.023 \, \text{s}} = 8.746 \cdot 10^{-3}$$

(4.15)

where we can observe that the change in TDOA has no units, due to the fact that we are computing how much the TDOA changes in a time unit.

We assume that any maximum beyond this threshold is deleted, so another new maximum is searched and so on until a correct one is found. To obtain this value in number of samples we simply have to multiply the value obtained times the sampling frequency, which is equivalent to, roughly, 385 samples.

Nevertheless, the assumption that $c = 343 \, \text{m/s}$ is only valid for a temperature of $20^\circ$. The main formula which connects the temperature with the sound speed is the following one:

$$c = \sqrt{\frac{\kappa \cdot R \cdot T}{m}}$$

(4.16)
where $\kappa$ represents the relationship of specific heats, whose value is 1.4 in the air; $R$ is the constant of gases; $T$ is the temperature measured in degrees Kelvin; and finally $m$ is the average molecular weight of the air. The relationship $R/m = 287 \text{ J/kg-K}$.

So that, if we evaluate the typical values in the local environment, Freiburg im Breisgau, setting extreme temperatures both for the lowest and the highest temperatures, what we obtain is depicted in Figure 4.2. Its shape is quite linear, so it is usual to use a linear approximation instead of using such square root function.

In Section 5.3, the effect of a large difference of temperatures is shown.

### 4.4 Back-track tree Algorithm

Differently from the TDOA algorithm, in this one (also taking into account the same observations from the microphones), we extract the 4 strongest maximums from the phase correlation; by doing this, we do not only have 1 possible path to follow but many of them\(^5\).

Fortunately, not all these paths are sufficiently good to be followed. We set a rule of nearness between the maximums in each iteration. It means that for each iteration, we observe at the following iteration and find which is the the nearest placed maximum to the current maximum. So, we select this next maximum as the next point of the path that has to be followed. We continue this way until we have reached the end of the recorded file. The number of possible paths is four owing to the selection of only the

\(^5\)In fact, the number of paths that we have is the number of time windows in which we have separated the entire phase correlation to the 4th power.
nearest maximums, which means that we start with 4 possible paths but each one of them has only one possible combination of points to follow.

It is quite likely that, in some iterations, the maximums taken are placed really close to each other, which actually can be caused for the proximity of some strong echoes. For this reason, we apply the concept of cluster. The cluster limits a region around the maximum computed which we decide that is prohibited for other maximums to be placed. All the values inside this region are set to 0, so they are no longer maximums. It allows to find more distributed maximums in each time iteration. As the main goal of this Master Thesis, we have to find which is the best value for the cluster to use that minimizes the error produced for the back-track-tree algorithm.

The region is delimited by the cluster as:

$$R = 2C + 1$$ (4.17)

where $C$ is the size of the cluster and $R$ the size of the region around the maximum, since we take into account the $C$ values to the right side from the maximum taken as well as the $C$ values to the left side and the value of the maximum itself.

Finally, as we have four paths, each one of them with its particular points, we determine which one of them is that one which provides the smallest error.

We compare all four paths with the ground truth in order to decide which path formed is the best possible, so we delete the weaker ones and only take the strongest one.

**Algorithm 1** Back-track tree algorithm

**Require:** Given a set of $M$ array observations $R = R_1, \ldots, R_M$ which are the correlation between the two inputs for each time iteration $i$, $N_{max}$ the number of maximums to consider and $C$ the cluster size.

```plaintext
for $i = 1$ to $M$ do
    for $k = 1$ to $N_{max}$ do
        $\mu^k_i \leftarrow \arg \max(R_i)$
        for $l = \mu^k_i - C$ to $\mu^k_i + C$ do
            $R_i[l] \rightarrow 0$
        end for
    end for
end for

for $i = 1$ to $M$ do
    for $k = 1$ to $N_{max}$ do
        $\mu^{l+1}_i \leftarrow \arg \min | \mu^{l+1}_i - \mu^k_i |$
    end for
end for
```

selection of less error path
4.5 Interpolation Algorithm

The last algorithm implemented is the interpolation algorithm. In this algorithm, we take advantage of back-track tree algorithm implementation and we also take the 4 strongest maximums from the phase correlation. However, in this case, instead of taking the closest value of the next iteration, if all the maximums are far away (decided with a threshold, which actually is another design parameter), we decide that in this point the value of the maximum is zero. Although, we only choose one single point in a row to be zero, due to the fact that the chosen value to remain different to zero is really wrong, we may be interpolating badly. If we take only one point instead, we do not allow the algorithm to commit serious mistakes at interpolating.

Afterwards, once we have localized the maximums over the whole signal, we search those points that had been set to zero. Now we make an interpolation between the two closest maximums different from zero. The value of current point is affected both by the previous point and the next point, each one of them contributing with the half of its value.

To illustrate it better, we formulate the interpolation mathematically:

$$V_i = \frac{V_p + V_n}{2} \quad (4.18)$$

where $V_i$ is the interpolation value which we are looking for, $V_p$ is the previous non-zero value and $V_n$ is the next non-zero value.

**Algorithm 2 Interpolation algorithm**

Require: Given a certain path $W = W_1, \ldots, W_M$ with $M$ iterations and a threshold $T_{\text{rsh}}$.

for $i = 1$ to $M$ do
  if $W_{i+1}$ is beside $T_{\text{rsh}}$ from $W_i$ then
    $W_{i+1} \rightarrow 0$
  end if
end for

for all points set to zero do
  $a \leftarrow W_{i+1}$
  $b \leftarrow W_{i-1}$
  $W_i = \frac{a+b}{2}$
end for
Chapter 5

Experiments

In this chapter we will focus on the different experiments we have made changing different parameters in order to get the best performance which is achievable. We have computed the errors in some different cases and for different parameter values and we have decided which value is appropriate to be compared with the template we are using\(^1\). Furthermore, we have plotted the graphics comparing our template and each obtained result from each implemented algorithm. Firstly, we show how should be the correct form of the shape according to the ground truth (Fig. 5.1)\(^2\). As we have said before, this shape was extracted from a file proportioned by the chair, and called ground truth.

Like in many projects or thesis, the first experiments are based on trying the functionality and the response of each algorithm. In some results we talk about the mean error of an algorithm. This mean is computed by calculating the relative error in every iteration of the algorithm and then computing the mean of all these relative errors. When we want to talk about the mean or the median of many simulations, we tell it explicitly.

In order to compare the error produced in each algorithm, we compare the performance of the ground truth shown in Fig. 5.1 by computing its relative error for each time instant:

\[
E_r = \frac{V_o - V_d}{V_d} \cdot 100
\]

(5.1)

where \(E_r\) is the relative error, \(V_o\) is the obtained value (for TDOA, for example) and \(V_d\) is the desired value (corresponding to the ground truth). We also multiply this value

---

\(^1\)We have called this template \textit{ground truth}. This ground truth is obtained from which our eye believes that is true.

\(^2\)Provided by the chair.
times 100 in order to have a percentage as a result. Afterwards, we compute the mean of all values of relative error.

After many simulations and changes on the algorithms, we encountered the first results.

### 5.1 First Results

The TDOA algorithm was the first one of being implemented, as we have explained in Section 4.1. As we also explained in such section, we obtain the TDOA value from the phase correlation. A representation of the phase correlation shape for one time window is depicted in Fig. 5.2. The first result sufficiently good to be shown is shown in Fig. 5.3.

Note that a kind of shape can be appreciated in Fig. 5.3, but there is a huge number of tremendous peaks affecting the clear shape. This is caused for the two effects which we describe in Section 4.1: edge effects and noise effects. To solve the second ones, we apply some windows, as we explain in the same section. There is a huge sort of windows that can be applied in order to mend these errors. The first one applied was the Hamming window. Although, as against we had thought at first, the use of a time window does not improve the performance of the signal, but worsens it, as is shown in Fig. 5.4, in which we apply a Hamming window in time domain. This application was made to both input signals before the computation of their FFTs.

The value of mean error over the entire signal is really huge: 50.008%, which in fact is worse than the mean error produced by TDOA without applying this window (38.564%).

Instead of applying a time window, we bet for doing the same in frequency window, in such way that the window works as a filter.
As we explain in Subsection 4.1.2, there are two design parameters to set a Hamming window, although one of them depends on the other one, so actually we only have one single design parameter, either $α$ or $β$. In Fig. 5.6 we represent the application of a Hamming window using concrete values for $α$ and $β$, simply to show how it effects the performance.

An important point that has to be taken into account is the parity of the phase correlation. As we are taking real signals coming from the microphones, their spectrums are pair. So, as we are applying the windows into frequency domain, and due to its parity, we apply a Hamming window at double frequency instead of the normal frequency\(^3\). It causes a moderate increasing importance for the components placed between sample 0 and sample $N/2$ (512) instead of a huge rise around sample $N/2$. In fact, an application of Hamming window without doubling the frequency acts as a high pass filter, which

\(^3\)Which means $4\pi n$ rather than $2\pi n$. 

\[\text{Figure 5.2: Representation of the phase correlation appearance in one time window.}\]

\[\text{Figure 5.3: First representation of TDOA(s) over the entire file (37.8 seconds).}\]
Chapter 5. Experiments

Figure 5.4: Representation of TDOA after applying a Hamming window in time domain.

Figure 5.5: Representation of a Hamming window at double frequency.

actually is not interesting: the sampling frequency is 44100 Hz, so the spectrum of the signal goes from \(-22050\) Hz to \(22050\) Hz. The most important peaks of the correlation come from a low pass spectrum (speech). So, if we apply a high pass filter into such a spectrum, we are giving more importance to an spectrum area where there is no speech information. However, if we double the frequency, the Hamming window is acting as a band pass filter, so it causes a larger impact of medium-band placed components. The representation of such a frequency window is depicted in Fig. 5.5.

After simulating values of \(\beta\) from 0.01 until 1, we figure out the best value for this parameter: with \(\beta=0.47\) we achieve an error of 29.1024\% in the TDOA depiction, which is almost the same performance that in Fig. 5.6, whose value for \(\beta\) is 0.46.

In Fig. 5.6, the clearness of the line is still not good enough, but we can appreciate a reduction on the peaks importance. They do not disappear but their effect is reduced considerably.
Chapter 5. Experiments

Figure 5.6: Representation of TDOA(s) using a Hamming Window with $\alpha=0.54$ and $\beta=0.46$ in frequency domain.

Figure 5.7: Representation of TDOA(s) using a time window of 4096 samples.

On the one hand, after trials by applying different time windows, we change our point of view and set a different number of samples per time window from 1024 samples to 4096. The result is depicted in Fig. 5.7. Nevertheless, there is an important restriction on the number of samples which we can use per time window. The key point is the next one: every time window have the same number of samples and this number is associated with a little time interval. They are directly proportional, since the larger number of samples we take, the larger is the amount of time per window. So, if we keep increasing the number of samples, there would be a moment in which a time window is large enough to be compared with the time depicted in X axis (they have the same order of magnitude). Consequently, the shape is completely destroyed and analysing such a figure becomes senseless.

Moreover, such representation has two important handicaps. Aside from the fact that there is a limited number of samples to take without losing the correct shape, even taking less number of samples than such limit, we lose time resolution. In addition, we can observe the huge magnitude reached now for the peaks, which has clearly increased.
Chapter 5. Experiments

<table>
<thead>
<tr>
<th>Algorithm and improvement</th>
<th>Mean error</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDOA error without using any improvement</td>
<td>38.564</td>
</tr>
<tr>
<td>TDOA error with Hamming window</td>
<td>34.585</td>
</tr>
<tr>
<td>TDOA error with Hamming window at double frequency</td>
<td>29.102</td>
</tr>
<tr>
<td>TDOA error with Gaussian window</td>
<td>34.589</td>
</tr>
<tr>
<td>TDOA error with Blackman-Harris window</td>
<td>54.068</td>
</tr>
<tr>
<td>TDOA error with Welch window</td>
<td>38.565</td>
</tr>
<tr>
<td>TDOA error with Low Pass Filter</td>
<td>38.101</td>
</tr>
<tr>
<td>TDOA error with LPF and Hamming Window at double frequency</td>
<td>28.527</td>
</tr>
</tbody>
</table>

Table 5.1: Errors of different algorithms in %. Hamming windows are depicted with $\beta=0.47$ and LPF with $\alpha=0.01$. Welch window is depicted with parameters $a_0 = 0.35875$, $a_1 = 0.48829$, $a_2 = 0.14128$, $a_3 = 0.01168$. All the windows are applied in frequency domain, not in time domain.

On the other hand, most of the experiments are made using time windows of 1024 samples. The required time computation is smaller because of the less number of operations per phase correlation that the computer has to do. Besides, the previous experiments done before this Thesis were made with 1024 samples, so we will continue with this number in order to be able to compare our experiments with the provided by the tutor.

Another possible way to improve this algorithm is the possibility of adding a low pass filter to the TDOA computation. We have repeated many times the experiments in order to find out which is the best value for the design parameter $\alpha$. What we have encountered may be contradictory: the best value for $\alpha$ is 0.57, which means that actually the filter is no longer a low pass filter, since it implies that the current time instant is given more relevance than the previous one. So, we bet for using a value for $\alpha = 0.01$, which is considerably good in comparison with other values and is still a value which makes sense because it maintains the low pass behavior of the filter.

In Table 5.1 we depict the total errors achieved with different design parameters. Values for Blackman-Harris window are directly extracted from information source and values for Gaussian window are again a design parameter\(^4\). Executing 100 times the program\(^5\) we figure out the best value for the deviation of the Gaussian window, which is the fourth part of the window width, i.e. $1024/4 = 256$. The value for $\alpha$ in the LPF is 0.01 in order to preserve its low pass behavior although the best value encountered is 0.57, as explained in the previous paragraph.

---

\(^4\)Actually, what is a design parameter is only the deviation, owing to the fact that the the mean is always set in the center of the window.

\(^5\)In this case, we set the deviation as a proportion of the weight of the window, which has a width of 1024 samples. So, the proportions come from the whole window until its hundredth part.
Chapter 5. *Experiments*

51

Figure 5.8: Representation of TDOA(s) after applying a Hamming window in frequency domain at double frequency, a LPF with $\alpha = 0.01$ and 81 special filters.

Then, as we can see in the table which contents all errors, the best combination of techniques is that one in which we combine a Hamming window with a Low Pass filter of $\beta = 0.47$.

We have made some experiments with different combinations of windows and a low pass filter but the one which achieves the best result in error is the combination Hamming window + low pass filter.

Nevertheless, this is not the last possible improvement that can be applied over TDOA algorithm. The next technique has been developed due to the appearance of the following question: if a single LPF can improve the signal, it is possible that some filters (which act like LPF does) also can improve it? The answer is affirmative, but changing the configuration of the used filters. Instead of taking a set of low pass filters, we collect a set of filters that give more importance to the current value and take a little part of the previous one\(^6\). The configuration used is the next one: we use a little part of the past value (actually the $\alpha$-th part, with $\alpha = 0.01$) and a large part $(1-\alpha)$ of the current value. It causes that each current value of the TDOA is smoothed by the previous one in a little part. We keep smoothing by adding more and more filters until a certain value with a minimum error is localized.

After making simulations by scanning all possible values in the number of this filters used (from 1 until 200), we have encountered that the ideal value is 81, and the mean error in TDOA for this configuration is 20.591%. Its representation is shown in Fig. 5.8.

The next experiments are no longer carried on with TDOA algorithm, but with Particle Filter algorithm.

\(^6\)In the computation of the error for this method we also use a Hamming window in frequency domain at double frequency.
5.2 Particle Filter Experiments

First of all, it is important to notify the conditions in which these results are taken from the experiments: all the mean errors are computed by comparing the maximum particle in each time instant with the ground truth. It is totally necessary to take into account that our state space is finite, so all particles are also placed in a "sample" or in a "spot", provided that the state space is formed by the 1024 samples of the time windows used to compute the phase correlation. So, it is possible that more than one single particle is located in the same place. To deal with it, we simply sum the multiple particle that are placed there, such a way that, in this concrete spot, the resulting weight in this place is the summation of the weights of the particles. So that, the maximum particle in each time is, in reality, the place where the summation of the weights of particles placed there is larger.

Furthermore, we have designed two other indicator of "how good" is the particle filter aside of the maximum particle, although both of them are actually related with the maximum particle: they are called the average of the particles and the expectation of the particles. The first one is computed as a simple mean\(^7\). The second one is computed as follows:

\[ E_p = \sum_{i=1}^{K} x_i w_i \]  \hspace{1cm} (5.2)

where \( E_p \) is the expectation of the particles in each iteration or time instant, \( K \) is the number of particles used, \( x_i \) is the spot where the particle \( i \)-th is placed and \( w_i \) its corresponding weight. By construction in the implementation of the particle filter, the summation of all weights is 1.

In Fig. 5.9 we show the representation of the maximum particle in front of the ground truth; in Fig. 5.10 we compare the ground truth against the mean of the particles and finally, in Fig. 5.11, we plot the expectation particle also in front of the ground truth. The errors provided for each one of the indicators are depicted in Tab. 5.2. The errors are previously calculated in the domain of the space state, i.e., from 1 to 1024 spots and then the graphical representation is translated into terms of TDOA. This translation is made by translating the number of samples from \([0,1024]\) to \([-512,512]\) and then dividing by the sampling frequency. As is shown in the table, the best indicator in terms of error is the location of the maximum particle.

\(^7\)See details in Section 3.5.
In particle filter is especially important to repeat many times any change on design parameters due to the random behavior of the algorithm, which uses more than a single random variable. For this reason, a single simulation may not be sufficient to determine the quality of the algorithm with those concrete parameters. For instance, with the same values for all design parameters, we have achieved, after 200 simulations, the results depicted in Table 5.3.

To show the random behavior of the algorithm we choose some relevant values like, for example, for the simulation number 28, where is noticeable that the mean error of the
Figure 5.11: Representation of the TDOA followed for the expectation of the particles in a simulation with 1000 particles, resampling percentage = 50% and variance level = 2.

<table>
<thead>
<tr>
<th>Simulation number</th>
<th>Particle filter mean error</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>3.333</td>
</tr>
<tr>
<td>24</td>
<td>3.679</td>
</tr>
<tr>
<td>25</td>
<td>2.947</td>
</tr>
<tr>
<td>26</td>
<td>3.753</td>
</tr>
<tr>
<td>27</td>
<td>7.981</td>
</tr>
<tr>
<td>28</td>
<td>74.299</td>
</tr>
<tr>
<td>29</td>
<td>3.423</td>
</tr>
<tr>
<td>30</td>
<td>3.792</td>
</tr>
<tr>
<td>31</td>
<td>3.521</td>
</tr>
<tr>
<td>32</td>
<td>3.152</td>
</tr>
<tr>
<td>33</td>
<td>14.660</td>
</tr>
<tr>
<td>34</td>
<td>3.495</td>
</tr>
<tr>
<td>35</td>
<td>4.406</td>
</tr>
<tr>
<td>36</td>
<td>3.693</td>
</tr>
<tr>
<td>37</td>
<td>2.991</td>
</tr>
<tr>
<td>38</td>
<td>3.079</td>
</tr>
<tr>
<td>39</td>
<td>4.079</td>
</tr>
<tr>
<td>40</td>
<td>3.578</td>
</tr>
<tr>
<td>41</td>
<td>3.461</td>
</tr>
<tr>
<td>42</td>
<td>3.413</td>
</tr>
<tr>
<td>43</td>
<td>3.328</td>
</tr>
</tbody>
</table>

Table 5.3: This table shows the random behavior of the Particle Filter with 100 particles with mean error in %.

particle filter is 74.299%; or others whose mean error is around 3% or 4%. After all these simulations, we compute the mean of all 200 values, which is a more reliable and real value. In this case is, for example, 5.565165%.

In such a case as the previous one, where in some simulations suffer a large failure, it
Table 5.4: Error percentiles when we use 100 particles and 200 simulations.

<table>
<thead>
<tr>
<th>N-th percentile</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2.896</td>
</tr>
<tr>
<td>10</td>
<td>3.017</td>
</tr>
<tr>
<td>15</td>
<td>3.087</td>
</tr>
<tr>
<td>20</td>
<td>3.141</td>
</tr>
<tr>
<td>25</td>
<td>3.199</td>
</tr>
<tr>
<td>30</td>
<td>3.253</td>
</tr>
<tr>
<td>35</td>
<td>3.284</td>
</tr>
<tr>
<td>40</td>
<td>3.325</td>
</tr>
<tr>
<td>45</td>
<td>3.369</td>
</tr>
<tr>
<td>50</td>
<td>3.423</td>
</tr>
<tr>
<td>55</td>
<td>3.458</td>
</tr>
<tr>
<td>60</td>
<td>3.495</td>
</tr>
<tr>
<td>65</td>
<td>3.544</td>
</tr>
<tr>
<td>70</td>
<td>3.579</td>
</tr>
<tr>
<td>75</td>
<td>3.694</td>
</tr>
<tr>
<td>80</td>
<td>3.822</td>
</tr>
<tr>
<td>85</td>
<td>4.089</td>
</tr>
<tr>
<td>90</td>
<td>5.043</td>
</tr>
<tr>
<td>95</td>
<td>17.491</td>
</tr>
<tr>
<td>98</td>
<td>17.491</td>
</tr>
</tbody>
</table>

makes sense to show how are distributed the values obtained. Thus, we include in Table 5.4 the error percentiles from the 5th until the 95th plus the 98th.

5.2.1 Experiments with Number of Particles

Carrying on with the particle filter experiments, we have checked if the number of particles used is relevant in the time of computing the error.

After running 100 simulations per each number of particles (100, 250, 500, 750 and 1000), the obtained results are depicted in Table 5.5. In this table, all values for CPU time are taking from the beginning of the program until the end of particle filter provided that we only want to compare how it gets worse when we increase the number of particles. We use the maximum number of particles as 1000 due to the space state is formed just for 1024 spots and the use of more particles is actually useless in order to get a better quality, as well as the CPU time increases considerably.

Looking at these results, we can observe a huge difference in the CPU times between simulations with different number of particles as well as the difference in the mean error is also considerable. Although, the relationship between the improvement of the error
Table 5.5: Performance of different number of particles for the particle filter: median error (%) and CPU time (s) with a resampling percentage of 50% after 100 simulations each one.

<table>
<thead>
<tr>
<th>Number of particles</th>
<th>Median error particle filter</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>3.515</td>
<td>7.301</td>
</tr>
<tr>
<td>250</td>
<td>3.423</td>
<td>17.878</td>
</tr>
<tr>
<td>500</td>
<td>3.162</td>
<td>35.627</td>
</tr>
<tr>
<td>750</td>
<td>3.049</td>
<td>52.946</td>
</tr>
<tr>
<td>1000</td>
<td>2.953</td>
<td>70.648</td>
</tr>
</tbody>
</table>

Table 5.6: Comparison of different cases with different number of samples: error improvement between the second number and the first one, CPU time worsening also between the second one and the first one and the relationship between the error improvement and the CPU time worsening.

<table>
<thead>
<tr>
<th>Cases</th>
<th>Error improvement</th>
<th>CPU time worsening</th>
<th>Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>100-250</td>
<td>1.027</td>
<td>2.449</td>
<td>0.419</td>
</tr>
<tr>
<td>100-500</td>
<td>1.112</td>
<td>4.880</td>
<td>0.228</td>
</tr>
<tr>
<td>100-750</td>
<td>1.153</td>
<td>7.252</td>
<td>0.159</td>
</tr>
<tr>
<td>100-1000</td>
<td>1.190</td>
<td>9.676</td>
<td>0.123</td>
</tr>
<tr>
<td>250-500</td>
<td>1.083</td>
<td>1.933</td>
<td>0.560</td>
</tr>
<tr>
<td>250-750</td>
<td>1.123</td>
<td>2.962</td>
<td>0.379</td>
</tr>
<tr>
<td>250-1000</td>
<td>1.159</td>
<td>3.952</td>
<td>0.293</td>
</tr>
<tr>
<td>500-750</td>
<td>1.037</td>
<td>1.487</td>
<td>0.697</td>
</tr>
<tr>
<td>500-1000</td>
<td>1.071</td>
<td>1.983</td>
<td>0.542</td>
</tr>
<tr>
<td>750-1000</td>
<td>1.033</td>
<td>1.334</td>
<td>0.803</td>
</tr>
</tbody>
</table>

and the worsening of the CPU time has to be computed in order to decide which number of particle is more favourable in each situation:

\[
R = \frac{I_e}{W_{CPU_t}} \tag{5.3}
\]

where \( R \) is the relationship, \( I_e \) is the improvement in error and \( W_{CPU_t} \) is the worsening in CPU time.

The relationships between all cases are computed and shown in Table 5.6.

Looking at Table 5.6, we can extract the following idea: the worsening in CPU time is larger than the improvement of the error in all relationships. With this conclusion, we can extract that the better number of particles is 100. Anyway, the use of only 100 particles provide such a large number of simulations with outliers that it is not a reliable number of samples to be used. For this reason, and due to that the most trusty number of particles is 1000, we carry on the next changes on the parameters with this number of particles, although it causes a larger computational cost.
5.2.2 Experiments with Variance Level at Taking Importance Sampling

The next parameter we have simulated is the variance while taking the importance sampling of each particle. In this context, we understand the variance as the number of samples of the observation (phase correlation) around which we take in order to compute a kind of mean:\(^8\):

\[ p(z_t \mid x_t) = \frac{1}{q} \sum_{k=-q/2}^{q/2} A_t(x_t + k) \]  

where \( A_t(i) \) is the value associated to the observation at time \( t \) and \( q \) is the variance level when we take the importance sampling for each particle.

In Tab. 5.7 the values of the mean errors using 100 particles are depicted. As we can observe, except for values 4 and 14 of the variance levels, in which ones, the mean error is totally away from the others, that are around 3% or 4%. There are two points which we can extract: the first one is that with simulations of 100 particles, the reliability in the algorithm is quite low, as the values for variance level equal to 4 and 14 show. The other one is the use of the median instead of the mean, owing to the impact of some outlier values can actually vary completely the value of the mean.

For this reason, in order to obtain more reliable results, we use the median values and simulate with 1000 particles instead, as depicted in Table 5.8. In this case, we only represent in the table all numbers resulting of a 2-power, but in Fig. 5.12 we plot all the values. Therefore, we also show the values of CPU time to demonstrate the effect of taking more values around the particle in order to compute the average of its importance sampling.

5.2.3 Experiments with Resampling Level

In addition, we also have executed 100 simulations with 1000 particles in order to determine which is the best resampling percentage (values between 0.1 and 1). We repeat again that we are working with random variables, so the results will not be exact.

From Table 5.9 we can observe that the CPU time does not depend on the percentage of resampling that we are using. However, we can distinguish a small tendency of worsening by the algorithm when the resampling percentage is too small (\( \leq 30\%) \) or

\(^8\)This equation was provided by the chair.
### Table 5.7: Mean error (%) after 100 simulations with 100 particles and different values for variance level at taking importance sampling.

<table>
<thead>
<tr>
<th>Variance level</th>
<th>Mean Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3.676</td>
</tr>
<tr>
<td>4</td>
<td>21.384</td>
</tr>
<tr>
<td>6</td>
<td>3.504</td>
</tr>
<tr>
<td>8</td>
<td>3.662</td>
</tr>
<tr>
<td>10</td>
<td>3.924</td>
</tr>
<tr>
<td>12</td>
<td>3.383</td>
</tr>
<tr>
<td>14</td>
<td>73.517</td>
</tr>
<tr>
<td>16</td>
<td>4.321</td>
</tr>
<tr>
<td>18</td>
<td>4.652</td>
</tr>
<tr>
<td>20</td>
<td>5.442</td>
</tr>
</tbody>
</table>

### Table 5.8: Median error (%) and CPU time (s) with 1000 particles after 100 simulations each one.

<table>
<thead>
<tr>
<th>Variance level</th>
<th>Median error</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.961</td>
<td>70.063</td>
</tr>
<tr>
<td>2</td>
<td>3.113</td>
<td>70.769</td>
</tr>
<tr>
<td>4</td>
<td>3.415</td>
<td>71.053</td>
</tr>
<tr>
<td>8</td>
<td>3.673</td>
<td>71.593</td>
</tr>
<tr>
<td>16</td>
<td>4.414</td>
<td>72.590</td>
</tr>
<tr>
<td>32</td>
<td>5.847</td>
<td>74.838</td>
</tr>
<tr>
<td>64</td>
<td>9.669</td>
<td>79.206</td>
</tr>
</tbody>
</table>

### Table 5.9: Median error (%) and CPU time (s) after 100 simulations per each resampling percentage with 1000 particles.

<table>
<thead>
<tr>
<th>Resampling percentage</th>
<th>Median error</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>3.239</td>
<td>71.522</td>
</tr>
<tr>
<td>0.2</td>
<td>3.100</td>
<td>70.601</td>
</tr>
<tr>
<td>0.3</td>
<td>3.069</td>
<td>70.605</td>
</tr>
<tr>
<td>0.4</td>
<td>3.023</td>
<td>70.863</td>
</tr>
<tr>
<td>0.5</td>
<td>2.953</td>
<td>70.648</td>
</tr>
<tr>
<td>0.6</td>
<td>2.900</td>
<td>70.630</td>
</tr>
<tr>
<td>0.7</td>
<td>2.984</td>
<td>70.597</td>
</tr>
<tr>
<td>0.8</td>
<td>2.978</td>
<td>70.799</td>
</tr>
<tr>
<td>0.9</td>
<td>2.971</td>
<td>71.531</td>
</tr>
<tr>
<td>1</td>
<td>3.004</td>
<td>71.121</td>
</tr>
</tbody>
</table>
too large ($\geq 90\%$). In conclusion, we can extract that the best performance is obtained between 30% and 90%. While using 1000 particles and computing the median of the error, we can observe that the change in error is not so extremely large as it would be if we have been using 100 particles and computing the mean.

Still in Tab. 5.9, we observe, as well as in Fig. 5.12, it is observable a larger value for a variance level of 2 samples than for the value of no variance level. We only maintained it in order to show how the computational time increases as we are using the computer in other ways although really this value would be smaller if nobody is using the computer.

### 5.2.4 Experiments with Variance in Motion Model

Finally, we also have executed the program with different values for the standard deviation of the velocity associated to each particle in each iteration (corresponding to each time). We set a parameter which we call deviation motion. This parameter is used in the deviation of the motion step of the particles. The deviation of the motion itself is set as a part of the number of particles that forms the space state:

$$\sigma_v = \frac{N_s}{\sigma_0} \quad (5.5)$$

where $\sigma_v$ is the velocity deviation itself, $N_s$ the number of samples (which is always 1024) and $\sigma_0$ our design parameter called deviation motion. We vary the value of $\sigma_0$ in order to decide which one is the value which minimizes the error. In Table 5.10 we depict the results with the different values of $\sigma_0$. Although the values of $\sigma_0$ here are
As we can see in Table 5.10, from values of $\sigma_v$ bigger than 1 sample, the values of the mean error are completely unacceptable. The sharp change visible in this point is also caused by the finite nature of the space state, whose combination with the fact that the velocity in the motion model has memory causes a large change in particles motion since this point. So, the interesting region is depicted in Fig. 5.13.

What we can infer from Fig. 5.13 is that the best values interval for $\sigma_0$ are between 3000 and 6000. Besides this value, the variance is too small, even with such a number of particles used (1000), and the particle filter is no longer able capable of tracking the position of the speaker.

The large number of particles used allows us to keep a good quality with a really small variance on the particles motion instead of needing a large variance. The smaller the number of particles is, the smaller the portion of space state they cover; so that, whether in a time instant the speaker stops talking and he/she starts to speak again after some seconds, a large difference of position is likely to appear. Therefore, having a small number of particles it is needed to work with a large variance of this particles motion.
Figure 5.13: Behavior of the median error of particle filter as a function of its motion variance using 1000 particles, a resampling percentage of 50% and a variance at taking the importance sampling of 2.

In order to be able to cover more part of the space state and so to be capable of still tracking the position location.

5.3 Experiments with Heuristic Algorithm

The measurement observations were taken with an environment temperature of roughly 20°, but in case they were taken in extreme conditions (−20° as bottom and 40° as top), the required threshold from which we do not allow values would change, so it would do the mean of the error computed.

At extreme cold, the sound speed would be 318.93 m/s (≈ 319 m/s), which represents a maximum change on TDOA of:

\[
\Delta TDOA |_{max} = \frac{0.069 \text{ m}}{319 \text{ m/s} \cdot 0.023 \text{ s}} = 9.404 \cdot 10^{-3}
\] (5.6)

and so, in samples is equivalent to, approximately, 415 samples; whereas in the opposite side, at extreme hot, the maximum velocity would be 354.72 m/s (≈ 355 m/s), which means:

\[
\Delta TDOA |_{max} = \frac{0.069 \text{ m}}{355 \text{ m/s} \cdot 0.023 \text{ s}} = 8.451 \cdot 10^{-3}
\] (5.7)

or, equally, 373 samples.

\footnote{We use the Equation 4.15.}
If the measurements had been taken at $-20^\circ$, the mean error computed would be 33.735%, whereas at $40^\circ$ would be 33.795%. Even if temperatures differ so much, there is no large difference from the error computed at a temperature of $20^\circ$, which is, in fact, 33.612%.

The TDOA performance with this method is quite similar to the performance obtained with the previous methods.

### 5.4 Experiments with back-track tree algorithm

In order to improve the quality of the algorithm, we have to set a good value for the cluster, as explained in Section 4.4. In Fig. 5.14 it is shown the behavior of the error when the value of the cluster size changes.

Besides, we show the representation of the back-track tree algorithm when the size of the cluster is 4 samples (so, the region is actually $9 (2C + 1)$) in Fig. 5.15.

As we can see in Fig. 5.14, there is a region where the error is minimized (between $C = 15$ and $C = 40$). After and before this region the error is reduced in comparison with the original TDOA algorithm as well but only in that region we find the minimum errors. However, we could not establish the same exact point of minimum location if the observations were different. This is caused by the large noisy behavior of the observations which we have.

So, if we set a value of the cluster as $C = 38$, as we can see in Fig. 5.15 that provides the minimum error, the result obtained is performed in Fig. 5.16. The mean error provided when this value of $C$ is set is 30.617%. We repeat, this value is only acceptable with the observations we have. With other observations, these values would change.
Figure 5.14: Behavior of the mean error of back-track tree algorithm if we change the value of the cluster.

Figure 5.15: Representation of the back-track tree algorithm performance with a cluster established of 4 samples without using any Hamming window.

Figure 5.16: Representation of the back-track tree algorithm performance with a cluster established of 38 samples and without using any Hamming window.
5.5 Experiments with Interpolation Algorithm

In this algorithm we take profit of the back-track tree algorithm results. Anyway, if we want to interpolate, we have to establish a limit since which we delete values (as explained in Section 4.5). This threshold is a design parameter which we look for by scanning values from 1 until 200. The depiction of the behavior of the error as a function of this limit is shown in Fig. 5.17.

We find out that there are two possible values which provide two similar error values: 12 (mean error = 28.971% and performance shown in Fig. 5.18) and also 28 (which provides a mean error of 28.969%). This concrete numbers are obtained only for the particular set of observations we obtained. So that, if we change the input signals incoming into the microphones, the certain numbers will vary but the behavior will remain similar. Whatever are the observations, although we increase the threshold a lot, there is a point until where all the values are smaller, so the error actually has a top bound.
5.6 Combination of Algorithms and Methods and Final Comparison

In this section we observe the impact of combining some algorithms implemented with some solutions adopted by fixing the noise problems of TDOA algorithm. Here we are comparing the mean or median error and the CPU time charge of all algorithms used.

In Fig. 5.19 we represent the final comparison between the two best achieved algorithms against the ground truth. As it was plausible, the best algorithm in terms of quality is the Particle Filter, whereas in terms of CPU time the Particle Filter, with 1000 particles is quite expensive. On the other hand, if we need quick results without taking care if the algorithm is a bit worse, the best choice is the combination of the interpolation
Table 5.11: Final comparison between all used algorithms. HW: Hamming window at double frequency, LPF: Low pass filter, SF: Special filters. The value of the Particle Filter error corresponds to the median whereas the others correspond to the mean.

<table>
<thead>
<tr>
<th>Algorithm used</th>
<th>Error(%)</th>
<th>CPU time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDOA</td>
<td>38.564</td>
<td>11.965</td>
</tr>
<tr>
<td>TDOA + HW</td>
<td>29.102</td>
<td>11.965</td>
</tr>
<tr>
<td>TDOA + HW + LPF</td>
<td>28.527</td>
<td>12.043</td>
</tr>
<tr>
<td>TDOA + HW + LPF + SF</td>
<td>20.591</td>
<td>12.504</td>
</tr>
<tr>
<td>Heuristic algorithm + HW</td>
<td>24.171</td>
<td>12.012</td>
</tr>
<tr>
<td>Back-track tree algorithm + HW</td>
<td>21.286</td>
<td>12.121</td>
</tr>
<tr>
<td>Interpolation algorithm + HW + SF + LPF</td>
<td>18.017</td>
<td>12.894</td>
</tr>
<tr>
<td>Particle filter (1000 particles)</td>
<td>2.248</td>
<td>70.769</td>
</tr>
</tbody>
</table>

Figure 5.19: Representation of the interpolation algorithm performance with a threshold established of 17 samples and a Hamming window at double frequency.

algorithm and Hamming window at double frequency, plus the set of special filters and a low pass filter with the values indicated in each section for each algorithm.
Chapter 6

Conclusions and Future Work

6.1 Conclusions

It is always important in the technological world to search and find the ideal tools which provides us the best quality and at the same time have a less charge both in computational charge and in an economical way. Because of that, it is also important to work accurately and meticulously in order to compare those tools and which chances they offer to us.

During the development of this Master Thesis, we have coped with many algorithms whose implementation might have more than one possible way to be made, and we have adopted those solutions which we find they are the most intuitive ones rather than trying to avoid a loop that might cause more computational charge. In fact, the computer used has allowed us to make quite large programs without exasperating us owing to large waits.

It is always helpful to find a good compromise between how much time your algorithm need to be executed and how good your algorithm is. Focusing on this discussion branch, many applications drive the programmer to encounter a really good performance without taking care of the duration of its execution. Meanwhile, others actually want a good performance but they give more importance to the fastness of the algorithms.

Centring on how good an algorithm is, the best quality is achieved with particle filter and the next conditions:

- 1000 particles.
- Using the median instead of the mean.
• No variance level at taking importance resampling.

• Importance resampling between 50% and 90%.

• A value of $\sigma_0$ between 3000 and 6000.

On the other hand, a quite good distinguishable shape can be performed with a combination of an interpolation algorithm with some other methods like the addition of a low pass filter to the original result for TDOA algorithm, which reduces the effect of large peaks by giving much more importance to the value on the previous instant of time than to the current one; in addition, we also have added a Hamming window in the frequency domain as a band pass filter, which gives more importance to the right frequency components and finally adding a cascade of special filters acting as a low pass filter does, but giving more importance to the current value than to the previous one as well.

6.2 Future Work

Some future work which can be planned is to think up more algorithms that can be compared. However, it is probably more interesting to implement a multiple speakers tracking. Some work is already done on this way. However, in our case, it would be necessary to have a clearer input signal, which allows us to follow the speakers easily, due to the fact that the current input files are plenty of noise and likely not to track the speakers position correctly.

Also in the case of multiple speakers tracking it is plausible to expect that particle filter is going to be the best on following speakers position. Some results of related work are shown in [13]. Anyway, it would be possible to follow the position of more than a single speaker by using the TDOA algorithm and taking more than one maximum of the correlation at each time instant (the number of maximums taken depends on the number of speakers that are going to be tracked).

Moving on to others issues, focusing on the field of possible concrete applications, in order to compute exact positions, it is common to use more than 2 microphones. As explained in Sec. 1.1, the tracked position and its TDOA are governed by the hyperbola formed for any pair of receivers. The location point is set as that one where hyperbolas coincide. A possible application might be the taking out from a well of someone who has fallen inside: with more than two microphones and knowing the distances between them we can easily compute the certain position of that person.
Bibliography


